ANAIS DOS TUTORIAIS DA V ESCOLA POTIGUAR DE COMPUTAÇÃO E SUAS APLICAÇÕES

De 05 a 09 de novembro de 2012, no Centro de Eventos do Hotel Praiamar - Natal/RN

Editado por
Sílvio Roberto Fernandes de Araújo (UFERSA)
Marcelo Ferreira Siqueira (UFRN)

Sumário

Prefácio

Sejam bem-vindos à quinta edição da Escola POtiguar de Computação e Suas Aplicações (EPOCA 2012). A EPOCA é um evento regional que tem como objetivo promover a interação e aproximação de estudantes, professores, pesquisadores e profissionais de Computação do Estado do Rio Grande do Norte (RN) e vizinhança.

A EPOCA teve sua primeira edição em 2008 e foi realizada, simultaneamente, em três cidades do RN, Natal, Currais Novos e Mossoró, com o auxílio de vídeo-conferência. A programação do evento incluiu minicursos, palestras, mesa-redonda e mostra de iniciação científica. O evento contou com 250 participantes e teve como tema central "Metrópole Digital e a interação universidade-empresa-sociedade". A EPOCA foi reeditada anualmente, de 2009 a 2011, em Natal ou Mossoró, com aproximadamente as mesmas atividades. Em todas as edições, houve o envolvimento da Universidade Federal do Rio Grande do Norte (UFRN), Universidade Estadual do Rio Grande do Norte (UERN), Instituto Federal do Rio Grande do Norte (IFRN) e Universidade Federal Rural do Semi-Árido (UFERSA).

Em 2012, a EPOCA ocorreu no Praiamar Natal Hotel, em Natal (RN), de 5 a 9 de novembro. O evento consistiu em sessões técnicas, minicursos, palestras e uma mesa-redonda que debateu o tema "empreendedorismo e inovação em tecnologia da informação".

Natal, novembro de 2012
Edgard Correa e Marcio Kreutz
Coordenadores Gerais do EPOCA 2012

Organização

Coordenadores Gerais

Edgard de Faria Correa (Universidade Federal do Rio Grande do Norte – UFRN)
Marcio Eduardo Kreutz (Universidade Federal do Rio Grande do Norte – UFRN)

Coordenadora Local

Monica Magalhães Pereira (Universidade Federal do Rio Grande do Norte – UFRN)

Coordenadores do Comitê de Programa

Silvio Roberto Fernandes de Araújo (Universidade Federal Rural do SemiÁrido – UFERSA)
Marcelo Ferreira Siqueira (Universidade Federal do Rio Grande do Norte – UFRN)

Comitê de Programa

Adilson Lopes (Universidade Federal do Rio Grande do Norte – UFRN)
Alba Lopes (Instituto Federal do Rio Grande do Norte – IFRN)
Araken Santos (Universidade Federal Rural do SemiÁrido – UFERSA)
Bartira Rocha (Universidade do Estado do Rio Grande do Norte – UERN)
Eduardo Aranha (Universidade Federal do Rio Grande do Norte – UFRN)
Gibeon Júnior (Universidade Federal do Rio Grande do Norte – UFRN)
Karla Ramos (Universidade do Estado do Rio Grande do Norte – UERN)
Luciane Fraga (Universidade Federal do Rio Grande do Norte – UFRN)
Marcia Lucena (Instituto Federal do Rio Grande do Norte – IFRN)
Paulo Pagliosa (Universidade Federal de Mato Grosso do Sul – UFMS)
Pedro Velho (Universidade Federal do Rio Grande do Sul – UFRGS)
Tadeu Oliveira (Instituto Federal do Rio Grande do Norte – IFRN)

Revisores

Adilson Lopes (Universidade Federal do Rio Grande do Norte – UFRN)
Alba Lopes (Instituto Federal do Rio Grande do Norte – IFRN)
Araken Santos (Universidade Federal Rural do SemiÁrido – UFERSA)
Bartira Rocha (Universidade do Estado do Rio Grande do Norte – UERN)
Eduardo Aranha (Universidade Federal do Rio Grande do Norte – UFRN)
Gibeon Júnior (Universidade Federal do Rio Grande do Norte – UFRN)
Karla Ramos (Universidade do Estado do Rio Grande do Norte – UERN)
Luciane Fraga (Universidade Federal do Rio Grande do Norte – UFRN)
Marcelo Siqueira (Universidade Federal do Rio Grande do Norte – UFRN)
Marcia Lucena (Instituto Federal do Rio Grande do Norte – IFRN)
Monica Pereira (Universidade Federal do Rio Grande do Norte – UFRN)
Paulo Pagliosa (Universidade Federal de Mato Grosso do Sul – UFMS)
Pedro Velho (Universidade Federal do Rio Grande do Sul – UFRGS)
Tadeu Oliveira (Instituto Federal do Rio Grande do Norte – IFRN)

Introdução ao Desenvolvimento Mobile com Android

Andriê Anderson, Felipe Augusto
andrie.anderson1@gmail.com, felipe.aga@gmail.com

Resumo: Este artigo apresenta um estudo sobre a tecnologia de dispositivos móveis Android, bem como seus principais componentes e interfaces com o usuário. Este artigo também apresenta um estudo de caso, a fim de demonstrar e esclarecer o funcionamento dos componentes ensinados.
Abstract: *This paper presents a study about the technology of Android mobile devices, as well its main components and interfaces with the users. This paper also presents a case study which demonstrates and clarifies the functionality of the components taught.*

1 Introdução

O Android é um sistema operacional baseado no Linux para dispositivos móveis, conta com diversas aplicações já instaladas e um ambiente de desenvolvimento poderoso e flexível.

A plataforma conta com várias características, dentre elas:

Código Fonte Open Source
Baseado no Kernel do Linux 2.6
SGDB nativo (SQLite)
Gráficos 3D baseado na especificação 1.0 da OpenGL ES.

A arquitetura do Android é dividida em camadas, onde cada parte é responsável por gerenciar seus respectivos processos. Elas são: Camada de Aplicações, Camada de Bibliotecas, Camada de Runtime e Camada de Kernel.

Camada de Aplicações: É onde se encontra todas as aplicações que são executadas sobre o sistema operacional, como navegador, mapas, calculadora, etc.

Camada de Biblioteca: É a camada que contém as bibliotecas C/C++ que são utilizadas pelo sistema. Bibliotecas de multimídia, visualização de camadas 2D e 3D, funções de acesso a banco de dados (SQLite), dentre outras.

Camada de Runtime: É nessa camada que se instancia a máquina virtual Dalvik, responsável por executar as aplicações Android.

Camada de Kernel: Uma parte importante utilizada do Linux no Android é o controle de processo, gerenciamento de threads, memória, protocolo de rede, modelo de driver e segurança.

2 Desenvolvimento Mobile para Android

No mundo mobile existem diversos sistemas operacionais, como Palm OS, Symbian, Windows Phone, iOS, Blackberry OS, e outros. Então, por que utilizar Android? Por que desenvolver aplicativos para esta plataforma?

O sistema operacional Android vem conquistando o mundo. Em 2009, cerca de 6,8 milhões de aparelhos utilizavam o sistema, e a estimativa é que em 2014 esse número aumente para 259 milhões de aparelhos.

Atualmente, o mercado para a plataforma Android está em uma ascenção meteórica. Celulares, Tablets e até TVs estão utilizando a plataforma do Google. Com isto, programar nesta plataforma se torna cada vez mais indispensável.

Neste artigo será ensinado os componentes básicos, mas essenciais, para criação de aplicações Android. Será dado o necessário para que cada um consiga levar adiante os estudos na tecnologia, com uma base sólida. Para ter um maior acompanhamento do que se está fazendo, e facilitar o entendimento, depois de cada seção de código, o mesmo será explicado em detalhes.

3 Interfaces do Usuário (UI)

- **Vistas**
  - **TextView** - Texto apresentável na tela.
  - **EditText** - Campo de texto editável.
  - **ListView** - Exibe uma lista de itens em uma lista verticalmente rolante.
  - **Checkbox** - Caixa de seleção onde é possível selecionar/des-selecionar opções.
  - **Button** - Botão de ação.
  - **ImageButton** - Similar ao Button, com exceção que exibe uma imagem.
  - **RadionButton** - Semelhante ao CheckBox, mas aqui, só pode selecionar uma única opção.
  - **Spinner** - Lista *drop-down* que permite o usuário selecionar uma opção de um conjunto.
  - **Datapicker** - Permite a seleção de uma data.
- **Layouts** - É a estrutura visual para a interface do usuário. Pode ser declarada de duas formas: no XML, ou instanciada em tempo de execução.

- **LinearLayout** - É uma visão que alinha todos os "objetos" em uma única direção: vertical ou horizontal. Defini-se a direção com o atributo *Android:orientation*.
- **AbsolutLayout** – É um layout que permite especificar a localização exata de cada objeto na tela.
- **TableLayout** – Agrupa os objetos em linhas e colunas. Para informar uma nova linha, usa-se o elemento <*TableRow*>. A quantidade de colunas é definida pela quantidade de objetos inseridos em uma linda.
- **RelativeLayout** - É uma visão que alinha os "objetos" em posições relativas. A posição do objeto é relacionada à posição do elemento irmão.
- **ScrollView** – Permite que o usuário role a tela através de uma lista de vistas que ocupam mais espaço que o display do dispositivo.

- **Componentes**
  - **Activity** - É uma classe responsável por gerenciar uma UI (interface do usuário). Quando uma aplicação é iniciada, é uma Activity que é lançada.
  - **Intent** - É a "ligação" que permite que uma *activity* chame outra, para que trabalhem juntas, dando a impressão que eles pertencem ao mesmo aplicativo.

4 Ciclo de vida de uma Activity

Como explicado, atividades representam classes com elementos a serem executados assim que forem chamados. Cada atividade possui um ciclo de vida que varia desde a sua criação até o momento de término da aplicação.

Os métodos do ciclo de vida são:

onCreate() - É criado quando uma *activity* (atividade) é iniciada.

onStart() - A *activity* está ficando visível para o usuário.

onResume() - Neste momento, a aplicação já está visível para o usuário.

onPause() - Se ocorrer algum evento, este método salvará o estado da aplicação.

onStop() - É chamado para encerrar uma *activity*, a aplicação não estará mais visível para o usuário.

onRestat() - Chamado quando uma *activity* foi parada temporariamente e está sendo iniciada outra vez.

onDestroy() - A *activity* é destruída e é liberado os recursos de memória.

5 Aplicação: Gerenciador de Gastos Pessoais

Para começar a desenvolver o aplicativo, o ambiente deverá está instalado e configurado.

Veja como instalá-lo em: http://developer.Android.com/sdk/installing/index.html.

Detalhes:

O sistema contará com três telas onde serão explicados cada componente utilizado na aplicação, e mostrado o ciclo de vida de toda aplicação implementando os métodos citados acima com o *LogCat*.

Inicialmente, será implementado a tela de Menu, que conterá dois botões. O primeiro abrirá a tela de cadastro, e o outro abrirá a tela de listar despesas cadastradas. Nesse passo será apresentado o evento de *onClick*, e como realizar a navegação entre telas (Figura 01).

Em seguida, será desenvolvida a tela de cadastro que contará com a maior quantidade dos componentes a serem explicados (Figura 02).

Por fim, será desenvolvida a tela que listará as despesas cadastradas, mostrando como utilizar o componente *ListView* (Figura 03).

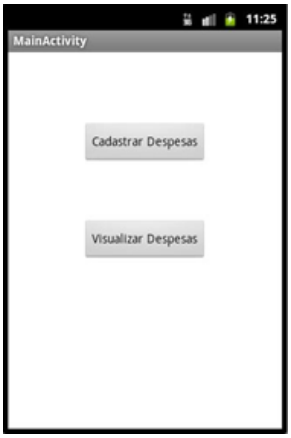

Figura 01

Figura 02

Figura 03

6 Como funciona:

Abra o Eclipse e crie um novo projeto Android: *File > New Application > Android Project*. Crie uma nova *Activity* chamada *Menu*. O eclipse irá gerar o arquivo Menu.xml e a classe Menu.java extendendo de *Activity*.

Abaixo segue a implementação da classe Menu (Figura 04).

```
11 public class MainActivity extends Activity {
12
13     private Button btCadastro, btBuscar;
14
15     @Override
16     public void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18
19         setContentView(R.layout.activity_main);
20
21         this.btCadastro = (Button) findViewById(R.id.button1);
22         this.btBuscar = (Button) findViewById(R.id.button2);
23
24         this.btCadastro.setOnClickListener(new OnClickListener() {
25
26             @Override
27             public void onClick(View arg0) {
28                 startActivity(new Intent(MainActivity.this, Cadastro.class));
29             }
30         });
```

Figura 04

Na linha 21 e 22 é obtida a referência dos botões de Cadastro e Visualização, através do método *findViewById*, passando o *id* deles, que são definidos no Menu.xml. Em seguida é criado o evento de clique e a navegação entre telas, declarando uma nova *Intent* passando como parâmetro a classe atual, e a classe de destino (linha 23 à 30). A *Intent* será inicializada no método *startActivity* passando a *activity* criada.

Agora crie uma nova *Activity* chamada Cadastro. Esta tela será responsável pela maior parte das regras de negócio da aplicação. Nela, o usuário poderá cadastrar uma nova despesa (Figura 05). No arquivo Cadastro.xml insira os componentes mostrados na Figura 02. Abra a classe Cadastro e siga os mesmos passos para obter as referências dos campos da tela de cadastro.

```
59             @Override
60             public void onClick(View v) {
61
62                 // Recupera campos
63                 descricao = edDescricao.getText().toString();
64
65                 if (!edValor.getText().toString().equals("")) {
66                     try {
67                         valor = Double
68                                 .parseDouble(edValor.getText().toString());
69                     } catch (Exception exception) {
70                         Toast msg = Toast.makeText(Cadastro.this,
71                                 "Utilize . em vez de ,", Toast.LENGTH_LONG);
```

```
73                      msg.show();
74                  }
75
76              }
77
78              categoria = spCategoria.getSelectedItem().toString();
79              parcelas = Integer.parseInt(spParcelas.getSelectedItem()
80                      .toString());
81              pago = "";
82
83              dia = String.valueOf(dpData.getDayOfMonth());
84              mes = String.valueOf(dpData.getMonth());
85              ano = String.valueOf(dpData.getYear());
86
87              if (cbPago.isChecked() == true) {
88                  pago = "Sim";
89              } else {
90                  pago = "Não";
91              }
92
93              // data = dpData.getYear();
94              // cadastra despesa
95              if (descricao.equals("") && valor == 0.0) {
96                  Toast msg = Toast.makeText(Cadastro.this,
97                          "Insira pelomenos uma informação.",
98                          Toast.LENGTH_LONG);
99
100                 msg.show();
101             } else {
102                 Despesas d = new Despesas();
103                 d.cadastrarDespesa(new DespesasBean(descricao, categoria,
104                         dia, mes, ano, pago, valor, parcelas));
105
```

Figura 05

```
134 private void preencherSpinnerCategorias() {
135     final Spinner combo = (Spinner) findViewById(R.id.spinner1);
136     ArrayAdapter<String> adaptador = new ArrayAdapter<String>(this,
137             android.R.layout.simple_spinner_item, Categoria);
138     adaptador.setDropDownViewResource(android.R.layout.simple_spinner_item);
139     combo.setAdapter(adaptador);
140 }
```

Figura 06

Para preencher os *Spinners*, será chamado o método *preencherSpinnerCategoria* e *preencherSpinnerParcelas*, onde será criado um *ArrayAdapter* do tipo *String*, onde se adiciona o *Spinner* e um vetor de *Strings*, que contém as categorias de despesas, que neste caso foi chamado de *Categoria*, e o adiciona como adaptador da referência do *Spinner*, que foi chamado de *combo* (Figura 06).

Na linha 102 é instanciada a classe *Despes*a, e utilizando o método *cadastrarDespesa* passando como parâmetro um *DespesaBean* (nessa classe contém os

atributos da despesa com seus devidos *getters* e *setters*). Após cadastrar a despesa, será exibida uma mensagem utilizando o método, e será retornado para a tela de Menu.

Por fim, será criada uma nova *activity* chamada Resultado. Nesta tela será exibido todas as despesas cadastradas (Figura 07).

```
13 public class Resultado extends Activity {
14
15     @Override
16     public void onCreate(Bundle bundle) {
17         super.onCreate(bundle);
18         setContentView(R.layout.activity_resultado);
19
20         Despesas d = new Despesas();
21
22         ArrayList<DespesasBean> despesas = d.listarDesepesas();
23
24         ArrayAdapter<DespesasBean> ad = new ArrayAdapter<DespesasBean>(this,
25                 android.R.layout.simple_list_item_2, android.R.id.text1,
26                 despesas);
27
28         ListView lista = (ListView) findViewById(R.id.listView1);
29         lista.setAdapter(ad);
30     }
31 }
```

Figura 07

No código acima é instanciado um objeto da classe *Despesas*, e criado um *ArrayList* do tipo *DespesasBean* que receberá uma lista de despesas. Em seguida, é criado um *ArrayAdapter* do tipo *DespesasBean* passando como parâmetro a classe atual, o formato de exibição da lista (nativo do Android), o modo de apresentação do texto (nativo do Android), e o *ArrayList* de despesas. É recuperado a instância da *ListView* e em seguida adicionado o *ArrayAdapter* como seu adaptador.

Um detalhe importante a ser notado, é que para que cada uma destas *activitys* possam ser iniciadas, elas devem ser instanciadas no *AndroidManifest.xml*. O *AndroidManifest.xml* é o principal arquivo do projeto Android, e fica na sua raiz. Ele é responsável por descrever o projeto; dar todas as permissões de acesso aos recursos, como Internet, GPS, e outros; mapear as *activitys*, ou seja, fazer com que o Android as reconheça; dentre outras coisas. Então, adicione as *activitys* criadas no arquivo como na Figura 08.

```
<activity
    android:name=".MainActivity"
    android:label="@string/title_activity_main" >
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".Cadastro"
    android:label="@string/title_activity_cadastro" >
</activity>
<activity
     android:name=".Resultado"
     android:label="@string/title_activity_cadastro">
</activity>
```

Figura 08

No *intent-filter* é declarada a forma que *activity* é iniciada. A ação *MAIN* indica que a *activity MainActivity* é o ponto inicial da aplicação, e a categoria *LAUNCHER* indica que a *activity* estará disponível para o usuário na tela inicial (Figura 08).

7 Conclusão

Este artigo apresentou uma introdução ao desenvolvimento mobile com Android, conceitual e prática, e também um projeto real para demonstrar sua utilidade. O Android é uma ferramenta poderosa para desenvolvimento *mobile*, bastando apenas que se tenha uma fonte de referências para dúvidas, neste caso, o Android disponibiliza toda a documentação em seu site, contendo até mesmo exemplos.

O Android é uma tecnologia que tem evoluído durante os últimos anos, e alguns analistas arriscam dizer que em 2016, irá superar até o mais famoso sistema operacional, o Windows (GARTNER).

8 Referências

Google. **Android components**. Disponível em: <http://developer.Android.com/guide/components/index.html>. Acesso em: 30/10/2012.

LECHETA, Ricardo R. **Google Android:** aprenda a criar aplicações para dispositivos móveis com o Android SDK**.** 2 ed. Novatec, 2010.

LEE, Wei-Meng. **Introdução ao desenvolvimento de aplicativos para o Android.** Ciência Moderna, 2011.

Desenvolvendo plug-in do *Moodle* em forma de módulo

Roberto D. Costa[1,2], Rommel W. Lima[1], Thiago Reis da Silva[1], Dimas K. Fernandes[3]

[1]Programa de Pós-Graduação em Ciência da Computação – PPgCC
Universidade do Estado do Rio Grande do Norte – UERN
Universidade Federal Rural do Semiárido – UFERSA
Laboratório de Redes e Sistemas Distribuídos – LORDI

[2]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte – IFRN
Campus de Educação a Distância

[3]Universidade Potiguar – UNP
Curso de Sistemas de Informação

douglas.costa@ifrn.edu.br, rommelwladimir@uern.br,
trsilva.si@gmail.com, k45t1b@gmail.com

Abstract. *To develop custom plugins in virtual learning environments, in this case module for Virtual Learning Environment (VLE) Moodle requires knowledge beyond the programming language. For besides being a platform, Moodle has features own development by the same being also a framework. Such features are called Guidelines for development, which is a set of rules which the programmer must follow the creation of modules / plugins. This short course will address what these policies are and how important to follow them to the community, and provide a quick passage through the API and a demonstration case.*

Resumo. *Para desenvolver plugins sob medida em ambientes virtuais de aprendizado, neste caso módulo para o Ambiente Virtual de Aprendizagem (AVA) Moodle, requer conhecimentos que vão além da linguagem de programação. Pois além de ser uma plataforma, o Moodle possui características de desenvolvimento próprias por o mesmo se tratar também de um framework. Tais características são chamadas Diretrizes de desenvolvimento, que é um conjunto de regras a qual o programador deve seguir na criação dos módulos/plugins. Este minicurso abordará que diretivas são essas e qual a importância em segui-las para a comunidade, além de proporcionar uma passagem rápida pela API e uma demonstração de caso.*

1. Introdução

Existe uma infinidade de ferramentas que solucionam inúmeras necessidades encontrados durante a utilização do Ambiente Virtual de Aprendizagem (AVA) Moodle.

Porém tais soluções não podem ser compartilhadas pelos demais utilizadores da comunidade Moodle, devido estes não estarem de acordo com as diretivas impostas pela equipe que administra a plataforma Moodle.

Uma das primeiras orientações que é feita para as pessoas que desejam iniciar suas experiências no desenvolvimento para o "Moodle" são as diretivas de codificação, ela estabelece entre muitas coisas como escrever códigos para o Moodle.

O Moodle é escrito em php e por ser uma linguagem de script e é extremamente fácil de programar. "ele ser extremamente simples para um iniciante, mas oferece muitos recursos para o programador profissional".

Portanto, ao final do curso os participantes estarão aptos a desenvolver ferramentas para o Moodle, que podem ser disponibilizadas para a comunidade de desenvolvedores, servindo assim como contribuição ao aperfeiçoamento da plataforma.

Para isso, iremos dividir este trabalho em oito tópicos, começando por esta introdução, seguido pelos tópicos dois e três onde conceituaremos o ambiente virtual de aprendizagem Moodle e seu contexto histórico, logo após, nos tópicos quatro e cinco, destacaremos as características das versões 1.9 e a 2.0, fazendo um breve comparativos sobre estas, depois, no tópico seis, demonstraremos as bibliotecas de códigos disponíveis para os desenvolvedores juntamente com o tópico sete no qual veremos exemplos de usos destas e fecharemos no tópico oito com a demonstração dos APIs - Interfaces de Programação de Aplicativos - disponíveis para uso no Moodle e concluindo com as considerações finais.

2. O que é o Moodle

O Moodle é um acrônimo para "*Modular Object-Oriented Dynamic Learning Environment*" [Moodle 2012] que em português poderia ser traduzido como "Ambiente de aprendizado dinâmico e modular orientado a objetos".

Segundo Martin Dougiamas seu criador, ele é "um ambiente virtual de aprendizagem que trabalha com uma perspectiva dinâmica da aprendizagem em que a pedagogia socioconstrutivista e as ações colaborativas ocupam lugar de destaque" [Silva 2011].

É nesse contexto que o Moodle foi criado, com o objetivo é permitir que o processo de ensino-aprendizagem ocorra, não apenas pela interatividade proporcionada pela ferramenta, mas, principalmente, pela interação entre as partes desse processo, privilegiando assim a construção, autoria e produção de conhecimento e a aprendizagem significativa do aluno.

3. Contexto histórico do Moodle

A História do Moodle que encontra-se disponível no site oficial moodle.org [Moodle 2012], conta que o projeto Moodle teve início na década de 1990s na "Curtin University of Technology" – Austrália, quando Martin Dougiamas frustrado pelo manuseio de outros AVAs lança em agosto de 2002 a primeira versão do AVA Moodle para a comunidade, contando com ferramentas, e sendo atualizada gradativamente e motivado pelo seu pensamento: "Para mim é crucial que esta plataforma seja fácil de usar - de fato, deveria ser tão intuitiva quanto possível".

3.1. Moodle 1.0 - agosto/2002 ao moodle 1.3 maio/2004:

Não houve mudanças bruscas nesse período.

3.2. Moodle 1.4 - agosto/2004:

A partir desta versão, duas características merecem destaque: a compatibilidade com PHP 5, melhorando o seu desempenho e a renovação do módulo Materiais que agora torna mais fácil a tarefa de adicionar novos recursos e enviar dados a eles.

3.3. Moodle 1.5 – Junho/2005:

Nesta versão foi possível a criação de novos temas ao sistema**,** como os de cascata, temas de usuários, temas de curso, com controle muito refinado de cada página no Moodle via CSS. Outro ponto importante desta versão foi o recurso de integração de mensagens para comunicação direta entre todos os usuários do ambiente, feito agora, em tempo real através de notificações via janelas *pop-up* e cópias enviadas através de e-mail, bloqueio de mensagens, histórico de mensagens e a presença do editor WYSIWYG.

3.4. Moodle 1.6 – Junho/2006:

Unicode foi escolhido como padrão de linguagem. Meu Moodle Uma interface de painel que permite uma visão geral de cada usuário de todos os seus cursos etc.

3.5. Moodle 1.7 – Novembro/2006:

Permissões com base em capacidades permitem todos os tipos de papéis a serem criados e atribuídos em todos os contextos em torno Moodle. Isso cria uma flexibilidade. Muito mais nas permissões que você pode conceder às pessoas.

3.6. Moodle 1.8 – Março/2007:

Customização dos perfis de usuários

3.7. Moodle 1.9 – Março/2008:

Grupos e Agrupamentos, Melhorias no Banco de questões

3.8. Moodle 2.x – Novembro/2010:

Suporte a Portfólio, Conclusão do curso e Pré-requisitos - Os professores podem especificar um Curso de conclusão para todos os alunos. Enguiner de nova questão. Suporte para dispositivos móveis. Avançados métodos de grades, incluindo Rubricas. Maior integração com ferramentas externas. Cursos de agora podem optar por mostrar seções em páginas individuais. Agora você pode arrastar os arquivos diretos para a página do curso e eles serão adicionados como recursos.

4. Características das versões 1.9 e 2.x

A medida que o Moodle se espalhou e a comunidade Moodle no mundo cresceu. Mais sugestões e comentários foram recebidos de uma gama mais ampla de pessoas em diversas situações de ensino, trazendo para Moodle novas mobilidades com uma única finalidade, deixar o ambiente cada vez mais amigável.

Com isso, muitas melhorias foram desenvolvidas principalmente nas versões mais atuais usadas, ainda pela grande maioria dos profissionais que atuam no desenvolvimento e administração deste AVA.

4.1. Versão 1.9

Uma das inovações desta versão foi a criação de um **Gradebook.** Com esta ferramenta, em comparação as outras versões, ganhou velocidade juntamente com flexibilidade. Com ele é possível gerar relatórios, importações e exportações.

Existe uma série de relatórios já formatados para alunos, estudantes que funciona identicamente a uma planilha eletrônica. Possui a capacidade de edição manual, cálculos, agregações, de ponderação, de bloqueio, ou notas textuais e assim por diante.

Outro ponto foi a **Escalabilidade e melhorias de desempenho.** Foi feita uma revisão completa da funções voltada para correção e escalabilidade. Grandes sites com os de cursos milhares e cursos para usuários agora podem ser carregados rapidamente e se terem comportamento melhor no tráfego pesado, graças ao código retrabalhado para essas funções.

Além disso ocorreu um incentivo adicional para sites usando PHP pré-compiladores e melhorias significativas no código de acesso de banco de dados para todos os sistemas de gerenciamento de bancos de dados. Muitas outras partes do Moodle foram otimizados para lidar melhor com um grande número de cursos e alunos. O desempenho geral foi visivelmente aumentado.

Outro item foi a criação do **Banco de questões** que permite compartilhamento de perguntas por todo o site, selecionado por categoria de curso, por um curso único, ou ser privado de um único curso, além disso, essa versão trouxe um melhor gerenciamento sobre os que podem fazer pergunta.

Finalizando, a criação de **Grupos e Agrupamentos** que dá suporte para agrupamentos com capacidade de controlar a visibilidade de bloco com papéis definidos.

4.1. Versão 2.x

Um excelente mudança neste versão é que professores cadastrados em algum site podem publicar seus cursos completos para hubs comunitários, para download.

Suporte a repositório: Sem dúvidas uma das mudanças que podem ser vista é o suporte a repositório, o mesmo sofreu uma grande mudança tanto na interface quanto na sua funcionalidade.

Figura 1 – interface do repositório

Mudanças também no selecionador de arquivo, agora apresenta uma forma padrão para acessar o banco de arquivos sistema de repositório.

Através de uma interface AJAX que se parece com um padrão aberto de diálogo em aplicações desktop.

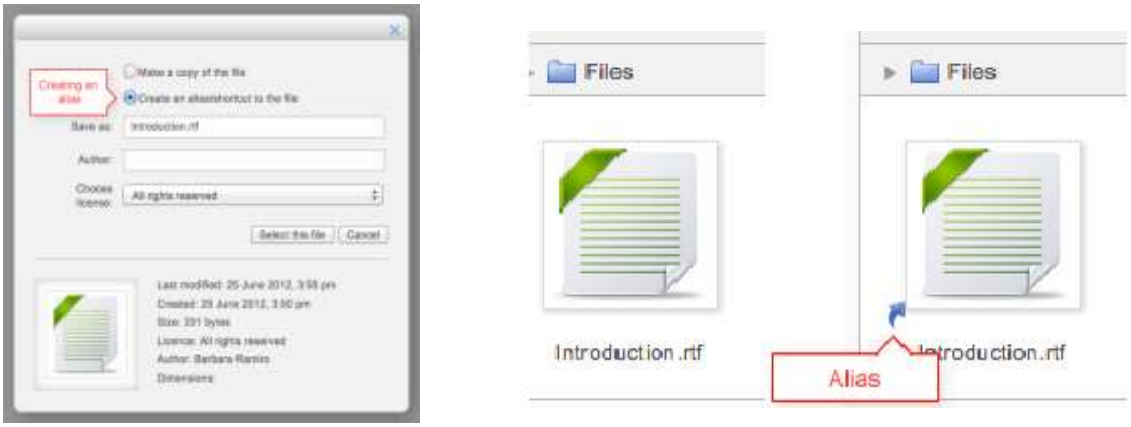

Figura 2 – repositório de arquivos

Além disso, existe a **possibilidade de Integração** com serviços cloud como Amazon S3, Box.net, Flickr, Google Docs, Picasa, servidores WebDAV, a Wikimedia, Youtube. Como também há possibilidade de enviar arquivos do desktop ou especificando uma URL.

Outro ponto importante é a **possibilidade de reutilização do mesmo**: diminuindo a duplicação de arquivos (por exemplo, um arquivo de vídeo grande usado em dois cursos diferentes) armazenando-os apenas uma vez, economizando espaço em disco.

Além do mais, esta versão armazena **metadados sobre cada arquivo** (autor, data, licença, etc) e em que o arquivo é usado. Todas estas ifnromações são armazenadas no banco de dados.

E concluindo, o item **Navegação**: Standard (Navegação) são blocos em cada página que mostra links contextuais, permitindo-lhe saltar rapidamente para outro lugar.

Configurações padrão para blocos em cada página mostra as configurações contextuais, bem como configurações para qualquer outra coisa que você tem permissões para acessar.

5. Diretrizes de desenvolvimento

Existe uma infinidade de benefício que o moodle trás consigo visando ajudar aos programadores na realização de suas tarefas de desenvolver.

5.1 Arquitetura do moodle

O Moodle tenta executar no maior número possível de plataformas, para um número maior possível de pessoas, mantendo-se fácil de instalar, utilizar, atualizar e integrar com outros sistemas.

5.2 Banco de dados

O Moodle tem uma camada poderosa de abstração de banco de dados, chamado XMLDB . Isso permite que o trabalho mesmo código Moodle em MySQL, PostgreSQL, Oracle e outros.

5.3 Plug-ins

O "M" de Moodle representa a palavra "modular", que é a maneira mais simples para adicionar novas funcionalidades para o Moodle. Existe uma infinidade de tipos padronizados que podem se adaptar às sua necessidade, caso algum não esteja disponível, pode-se usar tipo "local".

5.4 Segurança

O moodle tem um controle rigoroso quanto a segurança ou vulnerabilidades e diminuí-las é um papel essencial. A segurança da aplicação web depende do uso pretendido e os recursos disponíveis para cada tipo de usuário. Ex: Administradores, professores, estudantes, convidados.

5.5 XHTML e CSS

O Moodle produz rigoroso, bem-formado código XHTML, compatível com todas as diretrizes de acessibilidade comuns (como WAG W3C), isso ajuda a consistência entre os navegadores agrada melhor aos designers de tema. Além desses, existem outros formatos compatíveis como JavaScript, Internacionalização, acessibilidade, atuação, estilo de codificação, eventos, usabilidade.

6. Biblioteca de código Moodle

A Plataforma Moodle tem uma excelente biblioteca. Trata-se de um conjunto de funções que compõe o corpo do sistema. Essas funções são utilizadas para implementar os módulos que compõem a Plataforma.

O trabalho de desenvolvimento e customização do Moodle se torna muito mais rápido e fácil com o uso dessa biblioteca.

Toda biblioteca fica organizada na pasta *lib* no endereço raiz da instalação do Moodle. Para torná-la disponível basta incluir o arquivo "*config.php*"

7. API (*Application Programming Interface)* Moodle

O Moodle conta com uma série de APIs (Interface de Programação de Aplicativos**)** para facilitar aos desenvolvedores a desenvolverem suas próprias funcionalidades, e são essenciais para desenvolver *plugins*. Existe uma infinidade de APIs disponíveis no qual apresentaremos de uma forma geral:

Form API (API de formulário) – Formulários no Moodle são criados usando a Form API. API de formulário suporta todos os elementos html (rádio box, textfield etc).

Passo 1: Crie uma classe e ela terá que estender da classe *moodleform* salve-a como "**simplehtml_form.php**".

```
require_once("$CFG->libdir/formslib.php");
 class simplehtml_form extends moodleform {
   //Adicionando elementos no Formulário
   function definition() {
```

```
        global $CFG;
         $mform =& $this->_form; // Don't forget the underscore!
        $mform->addElement('text', 'email', get_string('email'));
// Add elements to your form
        $mform->setType('email', PARAM_NOTAGS);
 //Set type of element
        $mform->setDefault('email', 'Please enter email');        //Default value

    }
//Adicione a validação
    function validation($data, $files) {
        return array();
    }
}
```

Passo 2: Instancie o *form* (neste caso o *simplehtml_form.php*) em sua página.

```
require_once('DIRETÓRIO/simplehtml_form.php');

//instancie simplehtml_form
$mform = new simplehtml_form();
//Formulário aparece aquie
if ($mform->is_cancelled()) {
    //Caso execute a opção cancela
} else if ($fromform = $mform->get_data()) {
  //Em caso de validação dos dados. $mform->get_data() retorna com os dados.
} else { // caso não  seja validado o formulário é repreenchido
  $mform->set_data($toform);
  // depois é renderizado
  $mform->display();
}
```

Data manipulation API – é uma API que possui funções voltadas para a manipulação do banco de dados do Moodle, afim de recuperar ou modificar o conteúdo.

Todas as funções da chama nesta página são métodos públicos do objeto "$DB global", de modo que você terá que "importar" suas funções (não é necessário em scripts globais) com um simples:

```
global $DB;
```

Recuperando Registro - Todos os parâmetros $table nas funções são destinadas a ser o nome da tabela , sem prefixos.

```
$user = $DB->get_record_sql('SELECT * FROM {user} WHERE id = ?', array(1));
```

Todos os parâmetros $conditions nas funções são arrays de elementos fieldname=>fieldvalue.

```
$user = $DB->get_record('user', array('firstname'=>'Martin', 'lastname'=>'Dougiamas'));
```

Todos os parâmetros $params nas funções são matrizes de valores usados para preencher espaços reservados em instruções SQL. Podem ser usados tanto o ponto de interrogação quanto espaços reservados nomeados. Sepre que usar parâmetros nomeados os mesmos devem ser único.

```
/// Question mark placeholders:
  $DB->get_record_sql('SELECT * FROM {user} WHERE firstname = ? AND
lastname = ?',array('Martin', 'Dougiamas'));
 /// Named placeholders:
  $DB->get_record_sql('SELECT * FROM {user} WHERE firstname = :firstname
AND lastname = :lastname', array('firstname'=>'Martin', 'lastname'=>'Dougiamas'));
```

Recuperar um array de registros desordenados

```
$DB->get_records_list($table, $field, array $values, $sort='', $fields='*', $limitfrom='',
$limitnum='')
```

Inserirndo registros

```
$objeto = new stdClass();
$objeto ->name        = 'overview';
$objeto ->displayorder = '10000';
$DB->insert_record('quiz_report', $objeto, false);
```

Excluindo Registro

```
$ DB -> delete_records_select ( $tabela ,  $array('id'=> 3 );
```

Atualizando um registro

```
$objeto = new stdClass();
$objeto ->id        = '1';
$objeto ->name        = 'overview';
$objeto ->displayorder = '10000';
$DB->update_record($table, $ objeto, $bulk=false);
```

Além dessas existem uma série de APIS que não foram aperesentadas aqui, como: Access API, File API, Logging API, Navigation API, Page API, Output API, String API e outras.

8. Exemplos de uso

Para executar o exemplo foi utilizado o notpad++ ferramenta de desenvolvimento que auxilia na organização do código. Você pode baixar o mesmo no site *Portable Apps* (figura 3) através do link: http://portableapps.com/apps/development/notepadpp_portable.

Após baixar o mesmo, execute o arquivo e escolha a linguagem, na próxima tela avance, e escolha o local onde irá instalar (figura 4).

Figura 3 – página do Portable Apps

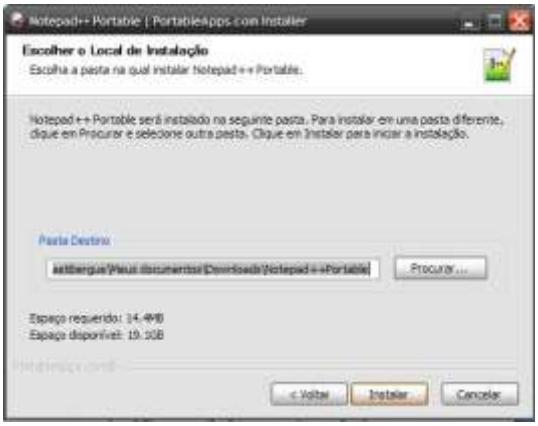

Figura 4 – Tela de início da instalação

Pronto! Agora é só executar e finalizar a instalação. No diretório no qual o Moodle está instalado, crie um novo diretório com o nome "teste". Neste diretório crie dois arquivos: *índex.php* e *simplehtml_form.php* e por ultimo realize as seguintes alterações no arquivo: *simplehtml_form.php*

```
<?php
require_once(dirname(__FILE__) . '/../config.php');
require_once("$CFG->libdir/formslib.php");
 class simplehtml_form extends moodleform {
   //Add elements to form
   function definition() {
        $mform =& $this->_form; // Don't forget the underscore!
        $mform->addElement('text', 'email', get_string('email'));
// Add elements to your form
       $mform->setType('email', PARAM_NOTAGS);                //Set type of element
       $mform->setDefault('email', 'Entre com seu E-mail');        //Default value
$this->add_action_buttons($cancel = true);
   }
//Custom validation should be added here
   function validation($data, $files) {
return array();
   }
}
```

No arquivo *index.php* escreva o seguinte código.

```
require_once(dirname(__FILE__) . '/../config.php');
//include simplehtml_form.php
require_once($CFG->dirroot .'/teste/simplehtml_form.php');
require_login();
$PAGE->set_url('/teste/index.php', $params);
$PAGE->set_pagelayout('mydashboard');
$PAGE->blocks->add_region('content');
$PAGE->set_title('Tituo teste');
//Instantiate simplehtml_form
echo $OUTPUT->header();
$mform = new simplehtml_form();
 //processe o form
if ($mform->is_cancelled()) {
    //caso clique no botão cancelar
} else if ($fromform = $mform->get_data()) {
// caso tenha enviado algum valor
} else {
// Rederisa o form
  $mform->set_data($toform);
  $mform->display();
}
echo $OUTPUT->footer();
```

Para testar deve acessar a seguinte url: http://localhost/moodle/teste/ se estiver de acordo com a figura 5 está tudo certo.

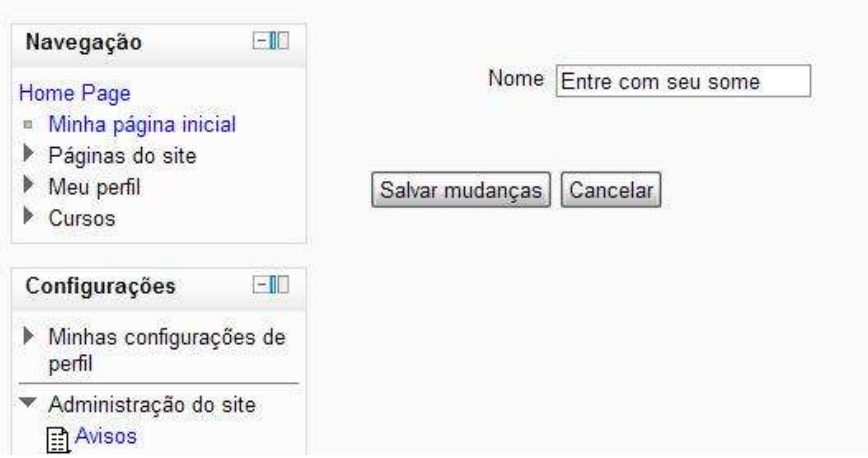

Figura 5 – Localhost Moodle

8.1. Execute o código SQL para criar a tabela “teste”

```
CREATE TABLE `teste` (
`id` INT(10) NOT NULL AUTO_INCREMENT,
`nome` VARCHAR(100) NULL,
PRIMARY KEY (`id`)
)
```

8.2. Inserido e recuperando registros

Refaça a função *validation* no arquivo *simplehtml_form.ph*p para que fique igual ao quadro abaixo:

```
function validation($data, $files) {
global $DB;
$objeto = new stdClass();
$objeto ->nome = $data['nome'];
$DB->insert_record('teste', $objeto, false);
// caso ocorra algum erro verificar o prefixo da tabela.
//ALTER TABLE `mdl_teste` RENAME TO `teste`;
      return array();
}
```

Modifique também o arquivo *índex.php* para recuperar o primeiro valor inserido na tabela.

```
global $DB;
 //In this case you process validated data. $mform->get_data() returns data posted in form.
$user = $DB->get_record('teste', array('id'=>1));
echo $user->nome;
```

9. Conclusões

Diante do tutorial exposto, espera-se que os desenvolvedores que cria plugins para o ambiente Moodle, possam desenvolver ferramentas para este ambiente de forma mais robusta, contribuindo cada vez mais para o aperfeiçoamento desta plataforma e ajudando a outros desenvolvedores a realizarem suas tarefas através da publicação de seus trabalhos na comunidade.

Referências

SILVA; R. S. **Moodle para autores e tutores**. 2ª Ed. rev. e ampl. São Paulo: Novatec Editora, 2011.

MOODLE – "**Modular Object-Oriented Dynamic Learning Environment**" 2012. Disponível em: http://moodle.org. Acesso em: Nov. de 2012.

Mary; C. "**Moodle 2.0 First Look**", Editora: Packt Publishing Ltd; 2010

Myrick; J. C., "**Moodle 1.9 Testing and Assessment**", Editora: Packt Publishing Ltd; 2010

Projeto de Sistemas Embarcados usando a Plataforma de Desenvolvimento Arduíno

Katielle D. Oliveira[1], Jorge F.M.C. Silva[1], Thiago O. Rodrigues[1], José W. M. Menezes[1]

[1] Departamento de Telemática – Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
CEP 60.040-531 – Fortaleza – CE – Brasil

{katielledantas,jf.engtelecom,thiagoliveira08,josewallymm}@gmail.com

Abstract. *This paper aims to show participants how to develop embedded systems projects and will have the Development Platform Arduíno as proof of concept. These themes will be discussed and demonstrated so theoretical and practical, including the validation of the models presented in class.*

Resumo. *Este trabalho visa mostrar aos participantes como desenvolver projetos de Sistemas Embarcados e terá a Plataforma de Desenvolvimento Arduíno como prova de conceito. Esses temas serão abordados e demonstrados de maneira teórica e prática, inclusive com a validação dos modelos apresentados em sala.*

1. INTRODUÇÃO

Nos últimos anos, a área de Sistemas Embarcados tem avançado de forma muito dinâmica. Esse dinamismo dá-se por diversos aspectos, ora pela convergência das redes de telecomunicações, ora pela convergência digital, ora pela evolução da microeletrônica e, por conseguinte, dos produtos advindos desse meio (sensores e atuadores), além da contínua redução dos custos proporcionada pelos avanços tecnológicos e pela popularização dos microprocessadores, da evolução das linguagens de programação e da própria forma de programar esses sistemas.

Os sistemas embarcados estão cada vez mais presentes no nosso cotidiano, brinquedos, telefones, máquinas de lavar, televisões, automóveis, caixas de banco eletrônicos, entre outros, são todos sistemas processados, onde algum tipo de informação é manipulada.

Logo, um treinamento objetivo e dinâmico para alunos/profissionais é importante para o desenvolvimento desses sistemas, porque eles irão realizar tarefas nas quais é exigido o conhecimento de ferramentas de programação e de componentes eletrônicos. Dessa forma, a motivação máxima do minicurso é dar embasamento teórico e prático sobre Sistemas Embarcados utilizando a Plataforma de Desenvolvimento Arduíno.

2. SISTEMAS EMBARCADOS

Um sistema embarcado é um sistema computacional que faz parte de um sistema maior e implementa alguns dos requerimentos deste sistema (IEEE, 1990).

Além dessa característica, Peter Marwedel identificou outras características de um sistema embarcado, tais como (MARWEDEL, 2003):

• São projetados para realizar uma função ou uma gama de funções e não para serem programados pelo usuário final, como os computadores pessoais. O usuário, pode alterar ou configurar a maneira como o sistema se comporta, porém não pode alterar a função que este realiza.

• Normalmente, interagem com o ambiente em que se encontram, coletando dados de sensores e modificando o ambiente utilizando atuadores.

• Sistemas embarcados devem ser confiáveis. Muitos destes sistemas realizam funções críticas, onde falhas podem causar catástrofes. A principal razão para que estes sistemas sejam a prova de falhas, é que eles interagem com o meio, causando impactos a este.

• Sistemas embarcados possuem outras métricas de eficiência, além das já conhecidas por projetistas de sistemas desktop e servidores, como: consumo de energia, tamanho de código, execução eficiente, peso, custo.

• Sistemas embarcados possuem interfaces dedicadas, como botões, leds e chaves. Por isso dificilmente o usuário reconhece a informação sendo transmitida ou processada dentro deles.

• Muitos sistemas embarcados são híbridos, pois são compostos por partes analógicas e partes digitais. As partes analógicas utilizam sinais contínuos em valores de tempo contínuos, e as partes digitais usam sinais discretos no tempo discreto.

3. ARDUÍNO

O projeto Arduíno iniciou-se em Ivrea, Itália, em 2005, como dispositivo para controle de construções acadêmicas interativas com um modelo diferente dos outros protótipos, até o momento, disponíveis no mercado. Seus idealizadores foram Massimo Banzi, David Cuartielles, Tom Igoe, Gianluca Martino e David Mellis e colocaram este nome em referência a um bar local.

A palavra Arduíno se refere a três ferramentas separadas, que agrupadas criam um conjunto de ferramentas que nos referimos como Plataforma de Desenvolvimento Arduíno. Primeiro, o microcontrolador, que existe em várias formas, de grande a pequeno esquemático, disponível livremente para qualquer pessoa, com os conhecimentos necessários, montar com investimento inicial baixo. Segundo, a linguagem e o compilador, que cria o código para este microcontrolador, assim como a linguagem *Processing*, que simplifica muitas das tarefas dos designers e desenvolvedores, quando o desafio é trabalhar com *hardware* e interação física. Finalmente, há o ambiente de programação do Arduíno, que como o Ambiente Integrado de Desenvolvimento (IDE) *Processing*, é uma IDE de código aberto simples construído em Java. Portanto, o porquê da palavra Arduíno ter vários significados, é muito específico ao se referir a um aspecto particular do ambiente, tais como a linguagem Arduíno. Quando simplesmente se referem a Arduíno, está referenciando ao meio ambiente como um todo (NOBLE, 2009).

Aliado a isso, a filosofia do ambiente é pautada para o fácil uso, ou seja, para iniciantes que não têm nenhuma experiência em software ou eletrônica. Pois com ele

pode-se construir objetos que respondem ao controle de luzes/LED's, som, toque e movimento. Além disso, a plataforma foi usada para criar uma variedade incrível de coisas, incluindo instrumentos musicais, robôs, esculturas de luz, jogos, móveis interativos e até roupas interativas (MARGOLIS, 2011).

Para aproveitar, ao máximo, os recursos da Plataforma de Desenvolvimento Arduíno é necessário estudá-la em duas partes: o *hardware* e o *software*.

3.1. HARDWARE

O Arduíno possui várias versões, usaremos como base o o Arduíno Diecimila, ilustrado na Figura 1.

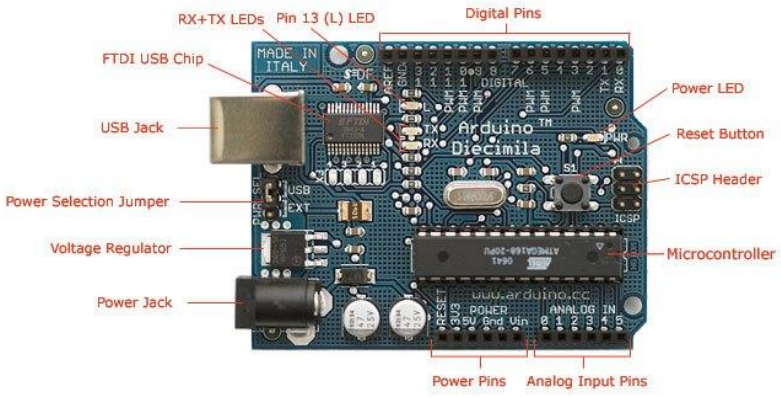

Figura 1. Arduíno Diecimila.

Como podemos observar, o Arduíno possui:

- Três (3) pinos de **GND** (Ground);
- Um (01) pino de alimentação de 3.3V (3V3) e um de 5V;
- Um pino denominado Vin, que possibilita o uso da tensão colocada à entrada (Pwr), antes de passar pelo controlador de tensão, sendo esta funcionalidade programável por software;
- Um botão de reset, que como o próprio nome indica, reinicia o Arduíno, ou seja, executa o programa a partir do início novamente, através da aplicação de um sinal de entrada. Voltando a executar o bloco de instruções da função setup;
- 14 portas digitais (0 ao 13), com a possibilidade de em seis destas (5,6,9,10,11) usar PWM – Pulse Width Modulation. Esta potencialidade é muito importante, pois através da variação da largura do impulso pode-se "simular" tensões entre 0 e 5V;
- 6 portas analógicas, possuindo um conversor analógico digital de 10 bits.

3.2. SOFTWARE

Na figura abaixo é apresentado o ambiente de desenvolvimento do Arduíno. Para carregar um programa é necessário, apenas elaborá-lo e fazer o upload, o programa é compilado e enviado para o Arduíno.

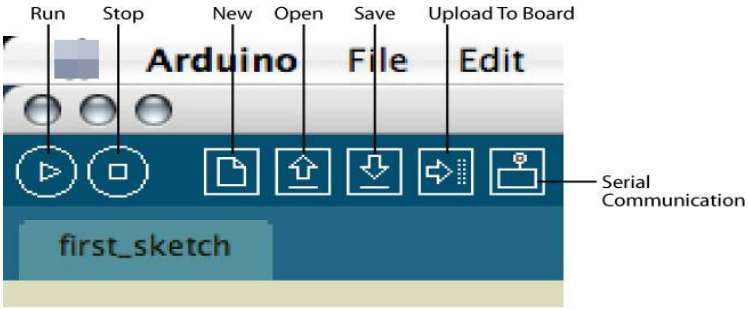

Figura 02. Detalhamento do IDE do Arduíno.

O IDE do Arduíno é uma plataforma de aplicação implementada em Java e multiplataforma. É designado para programação introdutória para quem já desenvolve trabalhos com esse padrão e para facilitar a familiarização de novos usuários do software de desenvolvimento. O ambiente, também, inclui um código editor com: realce de sintaxe, brace matching, identação automática e capacidade de compilação e uploading de programas na placa. Isso faz com que não seja necessário um editor de arquivos ou programas com linhas de comandos.

4. LINGUAGEM DE PROGRAMAÇÃO

A linguagem de programação do Arduíno é estruturada em dois blocos de funções que carregam outros blocos de funções escritas em linguagem C/C++. O primeiro bloco de funções forma a função setup(), onde é iniciado cada opção de inicialização, e o segundo forma a função loop(), função iniciada, após a execução da função setup, para fazer um loop sucessivo e permitir que o programa faça ações pré-estabelecidas.

4.1. PRINCIPAIS FUNÇÕES

4.1.2. Funções digitais

- pinMode (pin, mode) – estabelece a direção do fluxo de informação em qualquer pino digital. Os parâmetros da função são:

- pin : define qual pino será usado;

- mode: define se o pino será entrada (INPUT) ou saída(OUTPUT).

- digitalWrite (pin, valor) – envia um nível lógico para o pino digital. Os seus parâmetros são:

- pin: define qual pino será usado;

- valor: define em qual o estado lógico o pino deverá permanecer, HIGH ou LOW.

- digitalRead (pin) – lê o valor de um pino digital especificado.

4.1.3. Funções analógicas

- analogRead(pin) – lê o nível analógico do pino indicado e, após a conversão para o seu equivalente em bits, o guarda em uma variável determinada pelo programador.

- analogWrite(pin, valor) – atribui um valor analógico (PWM) ao pino, possui como parâmetro:

- pin: define o pino a ser utilizado;

- valor: determina o valor da amplitude da onda de PWM, que deve ficar entre os valores 0 e 255.

4.1.4. Funções de tempo

- millis() - retorna o número de milissegundos desde que a placa Arduíno começou a executar o programa atual.

- delay() - interrompe o programa para a quantidade de tempo, em milissegundos, especificado como parâmetro.

4.1.5. Funções matemáticas

- abs() – retorna o valor absoluto do número real usado como parâmetro.

- map(valor, min1, max1, min2, max2) – converte uma faixa de valores para outra faixa. Os parâmetros da função são:

- valor: é a variável que será convertida;

- min1e max1: são os valores mínimo e máximo da variável;

- min2 e max2: são os novos valores mínimo e máximo da variável.

4.1.6. Interrupção externa

- attachInterrupt(pino, função, modo) – trata-se de uma rotina de interrupção. Sempre que ocorrer uma interrupção por hardware no pino digital 2 ou 3, uma outra função criada pelo programador será chamada. A função possui como parâmetros:

- pino: define em qual pino ocorrerá a interrupção;

- função: determinar a função que será chamda;

- modo: informa como a interrupção será disparada(LOW, CHANGE, RISING, FALLING).

- detachInterrupt(interrupts) – desliga a interrupção indicada pelo parâmetro “interrupts”.

4.1.7. Interrupção

- noInterrupts() – desabilita interrupções.

- interrupts() – rehabilita a interrupção desativada pela função noInterrupts().

4.1.8. Comunicação

- Serial.begin(taxa) – habilita a porta serial e fixa a taxa de transmissão e recepção em bits por segundo entre o computador e o Arduíno.

- Serial.print(valor, formato) – envia para a porta serial um caracter ASCII. O segundo parâmetro "formato" é opcional e especifica com quantas casas decimais ou com que base numérica vai ser o número transmitido.

- Serial.println(valor, formato) – semelhante a anterior, porém ao final da transmissão retorna para o início da próxima linha.

- Serial.read() – lê a entrada de dados seriais.

- Serial.available() – obtem o número de bytes (caracteres) disponíveis para a leitura da porta serial.

5. PRÁTICAS

5.1. TERMÔMETRO DIGITAL

Após a etapa de conhecimento das funções básicas do Arduíno, veremos como controlar as portas digitais, analógicas e comunicação serial. Para isso, temos que ver alguns conceitos, uma vez que a eletrônica analógica tem suas peculiaridades. Conceitos como acionamento de carga, conversão analógico-digital e comunicação serial, são necessários para esta prática.

Para esta prática não podemos definir apenas dois estágios de tensão, ou seja, não podemos nos basear apenas nos níveis lógicos para o pleno funcionamento desse sistema. Portanto, precisamos de outra forma de manipular tais dados ao longo do projeto. A seguir temos o exemplo do Sensor LM35.

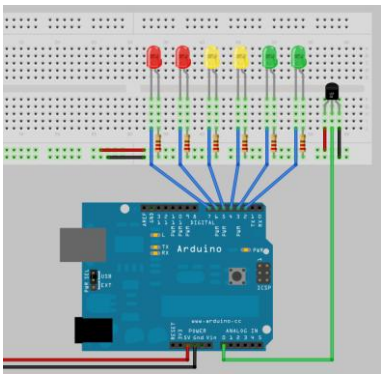

Figura 03. Termômetro digital usando LM35.

5.2. CONTROLE DE ILUMINAÇÃO

Após o exemplo anterior, veremos como controlar a iluminação de um ambiente por meio da modulação por largura de pulso, PWM em inglês. O conceito de PWM parte da resolução da equação do valor médio de um sinal, logo através desse resultado pode-se manter certa quantidade de energia sobre determinada carga ao longo de um período. Essa técnica é amplamente usada para controle de iluminação, velocidade de motores de corrente contínua, dentre outra aplicações em Sistemas Embarcados.

Portanto, esta prática consistirá no uso de um potenciômetro, um resistor variável, linear para variar o sinal que irá ascender o led.

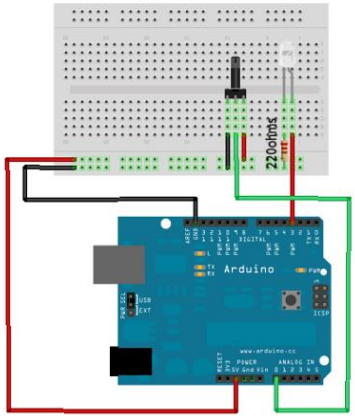

Figura 04. Controle de iluminação por PWM.

6. CONCLUSÃO

Após a participação no minicurso, alunos e profissionais da área de Sistemas Embarcados e, também, as demais pessoas que possuem interesse na área estarão aptos a começar a desenvolver esses sistemas utilizando uma ferramenta de fácil uso e "open-source".

REFERÊNCIAS

Arduíno. **Suporte de hardware e software para Arduíno.** Disponível em: < http://*Arduíno*.cc/en/Reference/HomePage > Acessado em: 08 de setembro de 2012.

Atmel. **Documentação do microprocessador Atmega328.** Disponível em: < http://www.atmel.com/devices/ATMEGA328.aspx> Acessado em: 09 de setembro de 2012.

Evans, Brian. ***Beginning Arduíno Programming: Writing code for the most popular microcontrololler board in the world.*** New York: Technology in Action, 2011.

Faludi, Robert. ***Building Wireless Sensor Network.*** USA: O'Reilly, 2011.

IEEE. ***"IEEE Standard Glossary of Software Engineering Terminology"***, Version 610.12-1990, Standards Coordinating Committee of the IEEE Computer Society, pp. 30, USA, 1990.

Margolis, Michael. ***Arduíno Cookbook.*** USA: O'Reilly, 2011.

Noble, Joshua. ***Programming Interactivity.*** USA: O'Reilly, 2009.

Oliveira, A. S. Andrade, F. S. **Sistemas Embarcados Hardware e Firmware na Prática.** São Paulo, Editora Érica: 2006.

Ordonez, E. D. M. Penteado, C. G. Silva, A. C. R. **Microcontroladores e FPGAs Aplicações em Automação.** São Paulo, Novatec Editora: 2006.

Oualline, Steve. ***Pratical C.*** 3ª Ed. USA: O'Reilly, 1997.

Oxer, Jonathan; Blemings, Hugh. ***Pratical Arduíno: cool projects for Open Source hardware.*** New York: Technology in Action, 2009.

Marwedel, Peter; ***Embedded System Design***. Dortmund: Kluwer Academic Publishers, 2003.

Timmis, Harold. ***Pratical Arduíno Engineering.*** New York: Technology in Action, 2011.

Warren, J. D; Adams, Josh; Molle, Harald. ***Arduíno Robotics.*** New York: Technology in Action, 2011.

Projetos de Placas de Circuito Impresso com utilização de *Computer-Aided Design* (CAD)

Franklin D. Silva[1], Laíza E. B. Alves[1], Jorge F. M. C. Silva[2], Antonio H. S. Lira[1] José W. M. Menezes[2]

[1]Departamento de Indústria – Instituto Federal de Educação, Ciência e Tecnologia do Ceará (IFCE)
CEP 60.040-531 – Fortaleza – CE - Brasil

[2]Departamento de Telemática
Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Fortaleza, CE - Brasil

{franklin.dias.silva,laizaeveline,jf.engtelecom,
9.2elet.hugo.lira,josewallymm}@gmail.com

Abstract. *This minicouse aims to show participants how to fabricate printed circuit boards (PCB) by means of specialized software for this purpose. Throughout the presentation is intended to introduce some CAD (Computer-Aided Design) tools that assist from the project to manufacturing these plates in a practical, dynamic and objective way. Thus, the authors develop a simple board as illustrative model to show such tools presented during the minicourse.*

Resumo. *Este minicurso visa mostrar aos participantes como confeccionar Placas de Circuito Impresso (PCI) por meio de softwares especializados neste propósito. Ao longo da apresentação pretende-se apresentar algumas ferramentas CAD (Computer-Aided Design), que auxiliam desde o projeto a manufatura dessas placas de maneira prática, dinâmica e objetiva. Dessa forma, os autores desenvolverão uma placa simples como modelo ilustrativo que conterá tais ferramentas apresentadas no decorrer do minicurso.*

1. Introdução

O mundo, nas últimas décadas, tem apresentado grandes avanços tecnológicos associados à informática, robótica, automação e eletrônica, de modo geral. Tais avanços estão, normalmente, atrelados a sua aplicabilidade, *performance* e *design*.

Ao observar um aparelho celular ou um PC (*Personal Computer*), por exemplo, pode-se perceber a presença de arranjos complexos de circuitos eletrônicos casados com interfaces gráficas e de funcionalidades diversas que interagem com o usuário. Tais dispositivos, sejam computacionais ou embarcados, usam uma lógica baseada em sinais elétricos – a eletrônica.

O *hardware*, a parte física de um dispositivo, é resultado do arranjo lógico-elétrico de componentes eletrônicos. Deste ponto de vista, é mister garantir que tais componentes estejam associados de maneira lógica e espacial ao corpo do dispositivo. Para tanto, existem inúmeros *softwares* que auxiliam, de modo preciso e profissional, o projeto de *hardwares*, desde o dimensionamento, simulação e *design* destes.

Tendo em vista os elevados padrões de qualidade em circuito eletrônicos na atualidade, é proposto o uso de uma ferramenta CAD (*Computer-Aided Design*) aplicada ao projeto e fabricação de PCI (*Placas de Circuito Impresso*) – o *Altium Designer*®.

Figura 1. Abertura do CAD *Altium Designer Summer 09*® para Projetos de PCI

A abordagem leva em conta modalidades de fabricação, termos e especificações técnicas, além de sugestões práticas para projeto e confecção de PCI.

2. Apresentação

O *Altium Designer* é um software que traz consigo ferramentas necessárias para criar um ambiente completo para desenvolvimento de produtos eletrônicos, em um único aplicativo.

As ferramentas oferecidas pelo software vão desde esquemáticos, simulações de circuitos, analisador de sinais, desenvolvimento de sistemas embarcados baseados em FPGA e *design* (ou roteamento) de PCI.

3. Plataforma *Altium*

A plataforma *Altium* é baseada na idéia de projeto.

Existem seis modalidades de projetos no *Altium*: *PCB projects* (projetos de PCI), *FPGA projects*, *Core Projects*, *Embedded Projects*, *Script Projects* e *Integrated Library*.

Cada *PCB Project*, por exemplo, agrega arquivos como *SCH sheet* (esquemático), o *PCB* (*Printed-Circuit Board*), o *netlist* (lista de conexões), e *libraries* (bibliotecas) que estejam incorporadas ao projeto.

Através do painel *Projects*, é possível acessar todos os documentos relacionados ao projeto; da janela *Workspace*, é possível abrir e conectar vários arquivos de projetos diferentes.

Este material apresenta algumas IDE (*Integrated Development Environment*) e ferramentas para projetos e desenvolvimento de PCI no *Altium – PCB Project*.

3.1. *PCB Project*: Criação do projeto

A primeira etapa para o desenvolvimento de uma PCI é a criação do projeto.

O *file* (arquivo) gerado, de extensão **.PrjPcb* (*Project Printed Circuit Board*), associa outros arquivos com as informações e detalhes do projeto: *SCH sheets*, o *PCB file*, as *libraries*, por exemplo.

Todos os arquivos do projeto apresentam *interfaces* e ferramentas diferentes para auxiliar a desenvolvimento do projeto de PCI. Ora, percebem-se ferramentas de desenho, ora ferramentas de *debug* (depuração de erros).

3.2. *SCH Sheets*: Folhas de esquemáticos

Criado o projeto, o esquema do circuito eletrônico deve ser representado numa *SCH sheet* (folha de esquemático), levando em conta as ligações elétricas e arranjos entre os componentes. Para tanto, deve ser adicionado ao projeto, o arquivo de extensão **.SchDoc* (*Schematic Document*).

O ambiente do esquemático (Figura 2) serve de referência para o *layout* da PCI - todas as ligações serão fielmente transferidas para este. Desta forma, devem ser tomados cuidados referentes à correta representação do circuito.

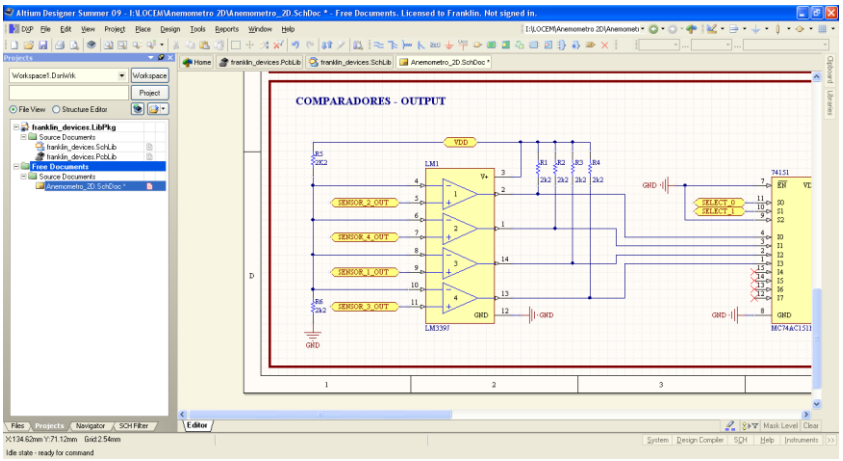

Figura 2. *SCH*: Interface de esquemáticos

Este ambiente possui ferramentas elétricas (*wire*, *junction*, *labels* e *ports*), de desenho (*text*, *lines* e *polygons*), de inserção de componentes e de listagem de materiais e conexões. Os comandos podem ser realizados com interação do mouse usando os ícones das ferramentas, ou ainda por atalhos usando o teclado.

3.3. *PCB File*: Organização do *layout* da PCI

O arquivo de PCB representa a visão final da PCI – *layout*.

O arquivo criado, de extensão **.PcbDoc* (*Printed Circuit Board Document*), é um ambiente de desenvolvimento e organização físico-espacial dos componentes eletrônicos, que leva em conta suas ligações, real espaçamento e posicionamento (*placement*) destes. Por isso, assim como em outros CADs, este ambiente baseia-se na idéia de camadas (*layers*), onde as informações são distintas de acordo com sua camada.

As ligações elétricas entre os componentes, feitas através de trilhas, são fruto da técnica, experiência e criatividade do usuário do *software*. Desta forma, são requeridos alguns conhecimentos de eletrônica como: grandezas elétricas (eg. rigidez dielétrica, corrente, indutância), características e natureza de sinais (eg. freqüência e picos, digital ou analógico), encapsulamentos (*packages*) e métodos de fabricação de PCI (eg. transferência térmica com corrosão em ácido, fresagem, corrosão a *laser*).

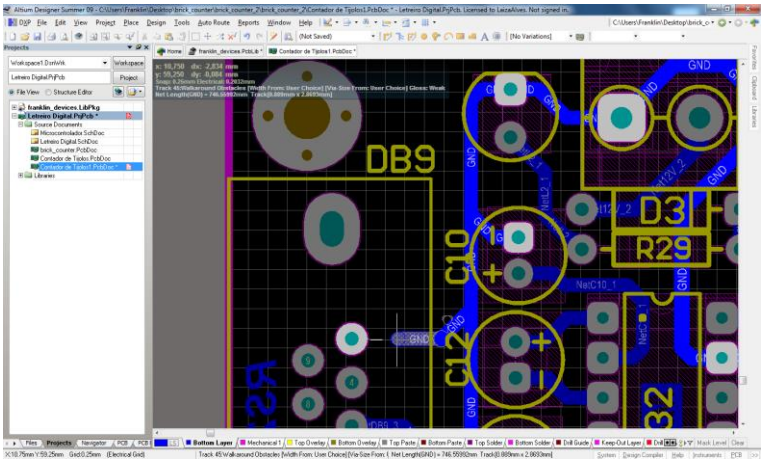

Figura 3. *PCB*: Interface de roteamento das trilhas de PCI

No ambiente de roteamento do PCB, encontram-se ferramentas CAD (*layers*, *2D/3D View*) e elétricas (*pads*, *vias*, *route*, *polygon pour, placement*).

A qualidade do produto eletrônico, dessa forma, é resultado de cuidados e observância de detalhes do projeto (eg. método de confecção, número de camadas, limites físicos da placa e modelo de trilhas).

Por meio do *software* CAD, ainda é possível gerar arquivos tipo CAM (*Computer-Aided Manufacturing*) compatíveis com fresadoras, ou mesmo simples imagens para estampa do *layout* para fabricação de PCI.

3.4. *Libraries*: Desenvolimento de componentes (SCH e PCB)

Dependendo da necessidade, os componentes podem ser criados ou modificados através de ferramentas e suporte apropriados.

Ambos *SCH* e *PCB Libraries* são incorporadas com ferramentas que discriminam as características elétricas, pinagem, encapsulamento de cada componente – seja um simples resistor ou mesmo um CI (Circuito Integrado).

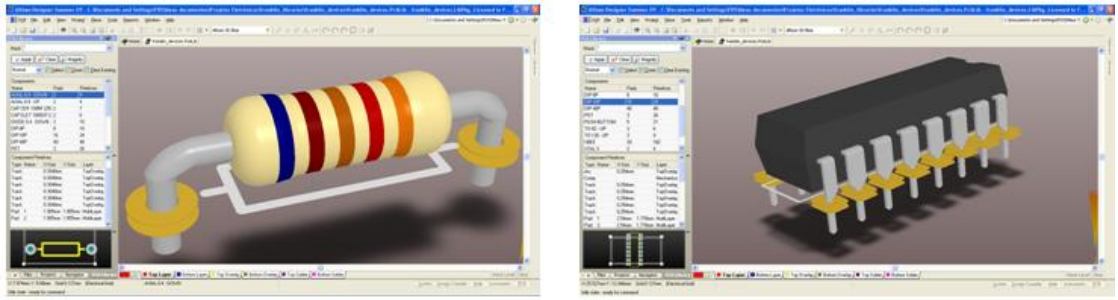

Figura 4. *PCB Libraries*: Desenvolvimento de componentes

As vistas 3D são, naturalmente, apreciadas por prever uma visão realista do protótipo.

Tais configurações proporcionam uma ótima percepção do *design* da PCI (dimensões, espaçamentos, simetria e aparência, por exemplo) além de incrível profissionalismo na confecção de protótipos eletrônicos (Figura 5).

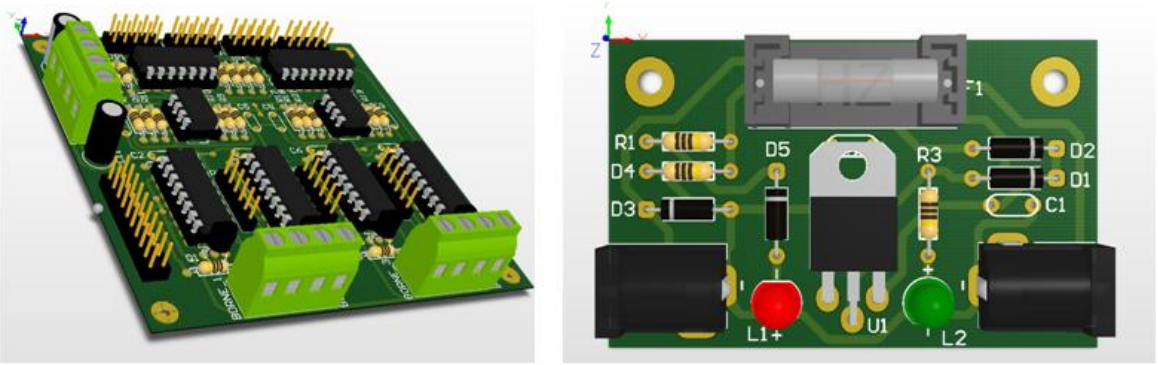

Figura 5. *3D Views*: Detalhes físicos da PCI

4. Conclusão

Ao final deste minicurso é esperado que os alunos tenham um conhecimento sobre a importância das Placas Circuitos Impressos (PCI) bem como seus diversos processos de manufatura. Além disso, a apresentação das ferramentas do tipo CAD usados no decorrer do minicurso servirão de base para aqueles que desejam iniciar trabalhos nessa área.

Referências

Altium Designer. **"Training: Manual and downloads".** Disponível em: <http://www.altium.com/training/en/manuals-and-downloads.cfm> Acessado em: 15 de outubro de 2012.

Kugler, M. (2004). **Projetos de Placas de Circuito Impresso.** Disponível em: <http://www.mauriciokugler.com/publications/pcb_mauricio_kugler.pdf> Acessado em: 12 de setembro de 2012.

Alphapress PCI. **Cálculo de Corrente em Trilhas.** Disponível em: <http://www.alfapress.com/site/index.php?option=com_wrapper&Itemid=38> Acessado em: 1 de outubro de 2012.

3D Content Central. **3D Views.** Disponível em: <http://www.3dcontentcentral.com/default.aspx> Acessado em: 20 de setembro cd 2012.

Desenvolvimento de jogos multiplataforma através do MOAI-SDK

André M. C. Campos[1], Matheus A. Gadelha[1], José F. S. Neto[1]

[1]Departamento de Informática e Matemática Aplicada – DIMAp
Universidade Federal do Rio Grande do Norte – UFRN
Campus Lagoa Nova, Cx.Postal 1524, 59078-970, Natal-RN

andre@dimap.ufrn.br, matheusabrantesgadelha@gmail.com, netolcc06@gmail.com

Abstract. *The proposed course hereby aims to present an alternative tool for developing games targeting several platforms. The tool, named MOAI, uses an abstraction layer to provide game interoperability and defines specific implementations for each targeted platform, which is called host. The later can be modified or created by development teams in order to achieve the particular needs of each game.*

Resumo. *O minicurso aqui proposto visa apresentar uma ferramenta alternativa ao desenvolvimento de jogos que tenham como alvo diferentes plataformas. A ferramenta, denominada MOAI, utiliza uma camada de abstração para prover a interoperabilidade do jogo e define implementações específicas para cada plataforma-alvo, a qual ela denomina de host. Estes podem ser alterados e/ou criados por equipes de desenvolvimento a fim de atender as necessidades particulares de cada jogo.*

1. Introdução

Há praticamente uma década que o mercado de jogos vem crescendo num ritmo mais acelerado que os demais setores da economia mundial. Nos últimos anos, com o aparecimento de novos dispositivos (*tablets* e celulares) e seus respectivos sistemas operacionais, o mercado, que antes era categorizado em linhas de desenvolvimento já bem consolidadas (por exemplo, jogos para desktop, para web, para consoles, etc.), precisou adaptar-se a essas novas tecnologias. Porém, como o custo para desenvolver um jogo é diretamente proporcional ao número de plataformas que ele se endereça, criar um jogo a ser distribuído nas presentes tecnologias passou a necessitar de um investimento cada vez maior.

Se, por um lado, a proliferação de sistemas permite que uma aplicação, no caso um jogo, atinja um número maior de usuários, por outro, ela requer um investimento maior na equipe de desenvolvimento e na de testes. Muitas vezes, esse investimento não

compensa o custo associado à adaptação do jogo às particularidades de cada plataforma. Por exemplo, a empresa Rovio, autora do *best-seller* Angry Birds, jogo inicialmente desenvolvido para *iPhone*, precisou re-escrevê-lo completamente para outras plataformas. Isso não seria possível se o jogo não tivesse sucesso entre o público da primeira plataforma escolhida. Em outras palavras, o alcance do segundo público só foi possível devido ao sucesso no primeiro, que permitiu um retorno de investimento necessário a um novo desenvolvimento do mesmo jogo.

Como a Rovio, até pouco tempo muitas empresas especializavam-se em jogos voltados a uma única plataforma em detrimento a várias outras opções no mercado. Como dito, a exclusividade da escolha era um reflexo do custo associado ao desenvolvimento de um jogo para diferentes plataformas. Com o rápido crescimento da plataforma Android, muitas empresas precisaram rever suas prioridades. Elas precisaram escolher entre focar numa plataforma específica ou investir para abrir o leque de opções. Porém, projetar um jogo tendo como alvo inicial várias plataformas é um risco grande para a maioria das empresas de jogos. De fato, é difícil prever se um jogo vai "cair no gosto" do público-alvo. Assim, um investimento em larga escala é normalmente realizado apenas para os títulos e/ou estilos de jogos bem consolidados, onde o risco de retorno é mínimo.

Uma forma de contornar esse problema é desenvolver jogos utilizando-se de ferramentas que permitam que um mesmo código possa ser executado em diferentes plataformas. Essa é uma abordagem antiga, na qual várias linguagens de programação se basearam, por exemplo, Java. Porém, a necessidade de gerenciamento de recursos de baixo nível presente na maioria dos jogos (resposta em tempo-real e uso de recursos nativos dos dispositivos) fez com que Java e sua tecnologia MIDlet não se tornassem a opção preferida das empresas de jogos nos dispositivos móveis mais recentes. Uma segunda opção seria utilizar a portabilidade natural das páginas web. De fato, a evolução da interatividade da web, com o HTML5 e suas respectivas tecnologias associadas, também é um natural candidato à resolução do problema de desenvolvimento multiplataforma. Porém, o tempo de resposta das aplicações em HTML5 é, por enquanto, consideravelmente menor em relação às aplicações desenvolvidas em código nativo. Enquanto o tempo de resposta de aplicações HTML5 não for equiparado às aplicações em código nativo, empresas de jogos vão continuar preferindo esta em detrimento daquela. Por fim, uma terceira alternativa tem ganhado recentemente o cenário de ferramentas multiplataformas, que é a opção de escrever o código em uma linguagem intermediária e esta é traduzida para uma linguagem de código nativo das diferentes plataformas-alvo. Assim, o mesmo código, num nível de abstração maior, é transformado em níveis de abstração menores e específicos para as plataformas-alvo, e com um mínimo de perda de eficiência no tempo de resposta ou em uso de recursos.

Para facilitar o desenvolvimento de seus jogos multiplataforma, a empresa Ziplines desenvolveu uma alternativa similar à terceira opção de ferramenta mencionada acima. Esta ferramenta, inicialmente construída para desenvolvimento *in-house*, foi

recentemente (meados de 2011) disponibilizada de forma *Open-Source* sob o codinome MOAI. Porém, por ser uma ferramenta inicialmente construída por uma equipe pequena de desenvolvedores, dentro de um ambiente fechado (empresa Ziplines) e com o um processo de desenvolvimento já bem definido, a documentação do produto MOAI não foi uma prioridade no momento que ela foi disponibilizada para a comunidade *Open-Source*. O número de tutoriais e de recursos didáticos sobre a sua utilização ainda é escasso. Assim, apesar do potencial da ferramenta, principalmente para as equipes que desejam desenvolver um jogo para diferentes plataformas com um investimento equivalente ao de se desenvolver para uma única plataforma, a curva de aprendizado do MOAI pode ser grande. Isso pode tornar, mais uma vez, o retorno duvidoso.

Assim, este documento visa prover material complementar à documentação oficial do MOAI de forma que equipes de desenvolvimento de jogos possam ingressar mais facilmente no uso da ferramenta. O retorno esperado do minicurso é divulgar o MOAI como alternativa viável à construção de jogos multiplataforma e apresentar os conceitos e passos iniciais no uso da ferramenta.

O texto é organizado em 6 seções, descritas a seguir:

1. INTRODUÇÃO (atual seção)
2. CONCEITOS INICIAIS: Uma vez que a criação de jogos não envolve exclusivamente o uso de tecnologias, esta seção inicia apresentando o processo de desenvolvimento de um jogo. Em seguida, como o uso do MOAI requer a programação em linguagem Lua, faremos um rápido tutorial sobre esta linguagem. Por fim, daremos uma visão geral sobre o MOAI e seus principais elementos.
3. INTRODUÇÃO AO MOAI: Esta seção apresenta, de forma resumida, os conceitos e elementos presentes no MOAI. A seção explica, de forma geral, como definir a área de visualização, seu sistema de coordenadas, como elementos visuais são inseridos no jogo, animações são criadas e eventos do jogador são capturados.
4. PRIMEIRO JOGO EM MOAI: Nessa seção, será apresentado um jogo simples, desenvolvido com o propósito de ilustrar os principais conceitos do MOAI. O foco do texto encontra-se nas particularidades existentes no jogo e como elas foram implementadas.
5. ESTUDO DE CASO: Nessa fase, apresentaremos um caso de estudo do desenvolvimento de um jogo usando o MOAI e comercializado na Google Play. Este jogo foi desenvolvido por dois dos autores do presente minicurso. Iremos, portanto, tratar dos aspectos práticos do desenvolvimento de um jogo completo através do MOAI, ressaltando as dificuldades encontradas e dicas para quem está ingressando na área e/ou na ferramenta. Serão abordadas principalmente questões sobre como organizar a parte gráfica do jogo e a criação de menus.
6. FERRAMENTAS E DEPLOYMENT: Nesta última etapa, apresentaremos o processo de *deployment* do jogo no Android e no iOS, mostrando que um jogo

desenvolvido no MOAI é, de fato, multiplataforma. Apresentaremos também algumas ferramentas úteis no desenvolvimento antes de tecermos os comentários finais do curso.

2. Conceitos iniciais

2.1. Processo de desenvolvimento de um jogo

Por onde começar?

Antes de iniciar o desenvolvimento um jogo qualquer, é essencial que algumas questões sejam respondidas e um processo seja definido. Não basta apenas idealizar o jogo e "começar a programar". Esse é o tipo de atitude que mais leva a não concretização da ideia, seja pela falta constante de um escopo delineado para o jogo, seja pela não estruturação de uma metodologia de desenvolvimento (concepção, desenvolvimento, testes, etc). Apesar do foco do curso ser a apresentação de uma ferramenta, esta seção visa apresentar o mínimo de informações e dicas para direcionar o participante a utilizar a ferramenta da melhor forma, ou seja usando-a dentro de um processo de desenvolvimento.

Antes de iniciar o desenvolvimento de um jogo, espera-se que a equipe (ou o idealizador do jogo), tenha claro em sua mente respostas para as questões a seguir. Em função de suas respostas, o jogo idealizado pode ser completamente inviável, sendo necessário remodelá-lo.

1. O que quero com o jogo? É unicamente para divertir ou há algum objetivo secundário (educacional, de marketing, etc.)?
2. Qual o público alvo? Qual a faixa etária? Gênero preferencial (se houver)? etc.
3. Haverá mercado para o jogo em questão?
4. Com quem eu posso contar no desenvolvimento? e quanto tempo tenho para que meu jogo seja viável comercialmente?

A primeira questão é essencial para guiar todo o processo de concepção do jogo. Se o jogo visa, além de divertir o jogador, transmitir uma informação específica, treinar uma habilidade ou avaliar um conhecimento, o *design* do jogo precisa refletir esse objetivo. Idealmente, esse objetivo não deveria estar explícito no jogo, pois, pra o jogador, um jogo precisa unicamente divertir (e não mais do que isso).

O público alvo define também muitos dos elementos do *design* do jogo. Um jogo voltado a crianças de 5 a 7 anos de idade possuirá, certamente, um visual, estilo e "jogabilidade" diferente de outro voltado a adolescentes, e este também será diferente se considerarmos, por exemplo, um público alvo é predominantemente feminino ou masculino. Antes de iniciar a concepção de um novo jogo, muitas das empresas realizam um estudo de mercado com o objetivo de levantar os potenciais compradores do jogo. Questões como: qual o tipo de filme que o público-alvo prefere? se lê, quais tipos de livros? quanto tempo tem para "gastar" no jogo? qual o estilo visual preferido

(roupas, cores, etc)?, ajudam a caracterizar o público-alvo, permitindo tratar questões do *design* do jogo mais facilmente.

A terceira questão, abordando o mercado, não influencia diretamente no *design* do jogo, mas na viabilidade financeira do projeto. O custo de desenvolvimento, da concepção ao lançamento no mercado, deve ser compensado pelo retorno esperado. Um plano de negócio bem elaborado sobre o jogo, abordando modelos de pagamento (mensalidade, compra de itens, compra de mídia, etc.), é vital para garantir o retorno sobre o investimento. Vale ressaltar que o plano de negócio será fortemente influenciado pelo público-alvo.

A última questão refere-se à capacidade da equipe envolvida no desenvolvimento em realizar a empreitada em um tempo viável, seja sob o aspecto financeiro (abordado no parágrafo anterior), seja tecnológico. Este último é importante devido à rápida evolução tecnológica na área de jogos. O que pode ser uma novidade e diferencial atrativo para a comercialização do jogo no momento de sua concepção, pode ser algo usual se o jogo tardar-se no seu lançamento.

Quais passos a serem seguidos?

Uma vez que as questões inicias estejam respondidas, passamos agora para a concretização da ideia inicial. Cada equipe ou empresa possui seu próprio mecanismo, que depende primordialmente da experiência acumulada da equipe. Podemos, entretanto, generalizar esse processo de acordo com a Figura 1. Nela, a etapa logo após a idealização do jogo é sua conceitualização. Nesta etapa, iremos moldar os elementos abstratos da ideia na forma de um documento de *Game Design*. Em função deste, poderemos estabelecer um plano de ações, com um cronograma associado de maneira a termos uma previsão (mesmo que mínima) sobre quando e em que ordem os componentes do jogo serão desenvolvidos, quando haverá versões lançadas do jogo e quais as funcionalidades de cada versão. Após o planejamento inicial, haverá espaço para duas equipes trabalharem concomitantemente: uma endereçando os requisitos audiovisuais do jogo, como as texturas, modelos a serem renderizados, música e efeitos sonoros, e outra endereçando os componentes de *software* do jogo. Essas duas equipes precisam estar interagindo constantemente, seja através de testes de desempenho ou robustez dos elementos audiovisuais sobre as ferramentas utilizadas (desenvolvidas *in-loco* ou terceirizadas, como no caso do MOAI) ou através da composição de protótipos sobre cenários (ou níveis do jogo). Quando um protótipo (caso de uso) estiver adequado, novos componentes são integrados e testes mais elaborados devem ser realizados.

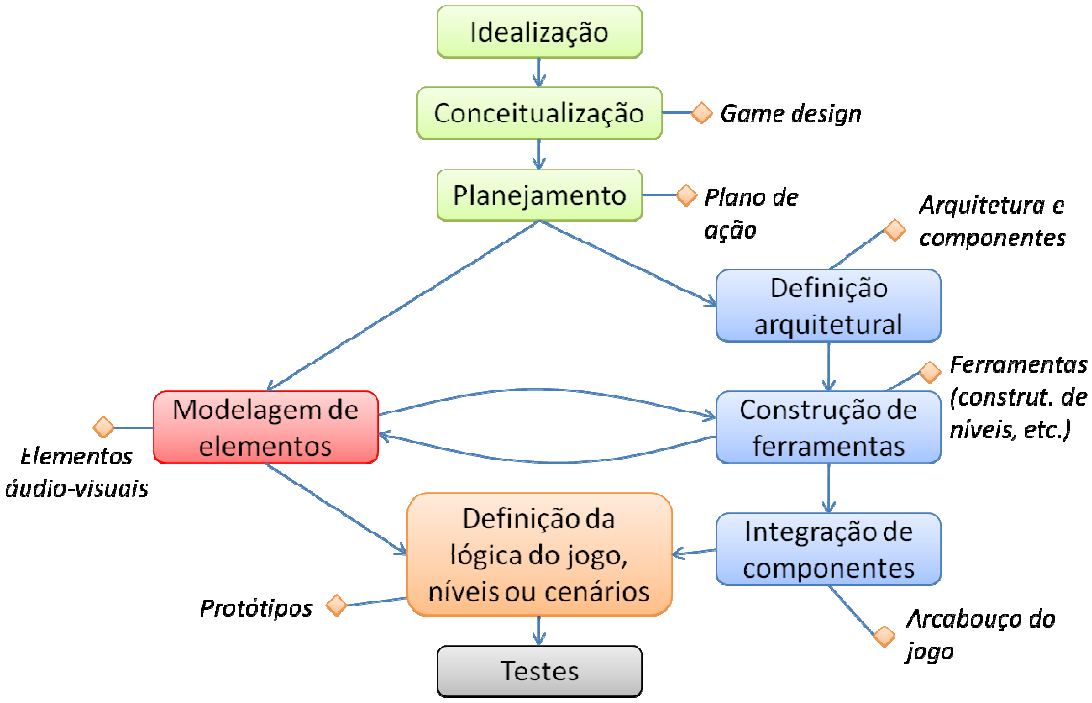

Figura 1. Etapas do desenvolvimento de um jogo (em geral).

Apesar de extremamente importante, o detalhamento desse processo e principalmente da elaboração do documento de *Game Design* foge do escopo do presente curso. Deixamos, assim, que cada equipe possa adequá-lo às suas necessidades.

2.2 Introdução à linguagem Lua

Antes de introduzirmos a ferramenta MOAI, apresentaremos um breve resumo sobre a linguagem Lua a fim de provermos subsídios para o entendimento dos capítulos seguintes. O leitor que já conhece a linguagem poderá pular esta seção e reiniciar a leitura a partir da seção seguinte.

Lua foi inicialmente criada para ser uma linguagem de configuração ou extensão. Ou seja, ela foi projetada para ser acoplada a aplicações desenvolvidas em outras linguagens, mais especificamente em C, de forma a permitir que esta aplicação pudesse ser configurada através de um *script*. O *script* seria interpretado por uma máquina virtual embutida na própria aplicação. Isso permitiria facilmente remodelar partes de uma aplicação, como os elementos de interface por exemplo, pelos próprios usuários da aplicação, uma vez que o processo de compilação não era mais necessário.

Apesar de ter sido criada para configurações de interfaces de usuário, foi na área de jogos, mais especificamente na definição da lógica do jogo, que ela obteve uma maior visibilidade. A grande utilização de Lua na área de jogos (cerca de 80% dos jogos que usam *scripts* de configuração, utilizam Lua para esta finalidade [REF]) ocorreu devido a algumas características da linguagem:

- É uma linguagem pequena (com poucas construções sintáticas) e eficiente (em tempo de execução e espaço de armazenamento de sua máquina virtual).
- Apesar de pequena, possui mecanismos encontrados em linguagens mais complexas como gerenciamento automático de memória e tratamento de erros.
- É uma linguagem simples e fácil de aprender (foi concebida para ser usada por não-programadores)
- Apesar da simplicidade, é flexível o suficiente para criar representações de diferentes paradigmas de programação (estrutural, funcional, orientado a objeto, etc).
- É interpretada e possui tipagem dinâmica, facilitando a prototipagem da lógica dos jogos.
- Possui uma forma eficiente de integração com aplicações em linguagem C, sendo esta uma das linguagens mais utilizadas na construção de motores de jogos.

Variáveis e tipos

Como na maioria das linguagens *scripts*, Lua possui um sistema de tipagem dinâmica, o que significa que os tipos de valores são verificados durante a execução do programa. Assim, as variáveis são criadas dinamicamente e podem assumir um valor de qualquer tipo. Os tipos pré-definidos em Lua são:

- boolean: pode assumir apenas os valores true e false;
- number: representa qualquer valor numérico, inteiro ou ponto flutuante;
- string: sequência não mutável de caracteres;
- table: dicionário armazenando pares (chave, valor da chave);
- function: referência para uma função;
- usardata: referência para dados externos (definidos em C);
- thread: corotinas (rotinas com pontos de parada e retorno)
- nil: representa a falta de dado associado (significa “nenhum valor”).

As variáveis são alocadas dinamicamente no momento da primeira atribuição e podem ser modificadas ao longo do programa. O tipo da variável assume, portanto, o tipo do valor que ela armazena. Pode haver atribuições múltiplas, onde uma sequência de valores é atribuída a uma sequência de variáveis. Assim, os seguintes exemplos de atribuições são válidos.

```
a = true       -- boolean
b, c = 5, 7.2  -- atribuição múltipla (b = 5, c = 7.2)
d = b + c      -- 'd' é do tipo number
a = "abc"      -- 'a' passa a ser agora uma string
```

```
pessoa = {     -- table é um dicionário de dados com...
  nome  = "fulano",  -- ...pares (chave-valor)
  idade = 25
}
inc = function(x)    -- o valor de 'inc' é uma função
  return x+1
end
```

É possível perceber, pela última atribuição, que as funções são tratadas da mesma forma que qualquer outro valor. Assim, elas podem ser atribuídas a variáveis, que passam a ter uma referência para elas. Para facilitar, as funções podem ser, entretanto, escritas da forma que é normalmente encontrada em outras linguagens.

```
function inc(x)   -- equivalente à atribuição anterior
  return x+1
end
```

Os parâmetros de uma função são normalmente passados por valor, exceto se forem tabelas ou funções (funções, como outro valor qualquer, podem ser passadas como parâmetros para outra função). As variáveis que recebem os parâmetros são consideradas locais, dentro da função. Para definir outras variáveis locais, é necessário usar a palavra-chave local. Da mesma forma que na atribuição, funções podem retornar vários valores. O exemplo a seguir ilustra essas afirmações.

```
a,b = 1,2
function m(a,b)
  local c = a + b
  a,b = c*a, c*b
  return a,b
end
c,d = m(a,b)
print(a, b, c, d)  -- imprime 1, 2, 3, 6
```

Sobre os valores de uma variável, pode-se aplicar os operadores normalmente encontrados em outras linguagens (+ - * / < > <= >= ==). A diferença encontra-se no operador de diferença, que é ~=.

Controle de fluxo

As expressões de controle de fluxo são as normalmente encontradas em outras linguagens, exceto pela sintaxe mais simplificada de Lua. Os comandos de controle de fluxo seguem as seguintes sintaxes (exemplos):

```
if expr then bloco_de_comandos end
if expr then bloco_1 else bloco_2 end
```

```
if expr then bloco_1 elseif expr_2 then bloco_2 end
while expr do bloco end
repeat bloco until expr
```

Além dos loops utilizando while e repeat, há duas formas de for. A primeira é numérica, com um contador incrementando ou decrementando. A segunda é através de um iterador sobre dados. No primeiro exemplo do código a seguir, a variável var assume valores de inicio a fim, com passos de incr (opcional). No segundo exemplo, line assume, uma a uma, todos as linhas de um arquivo lido.

```
for var = inicio, fim [,incr] do
   bloco de comandos
end

for line in io.lines() do
   bloco de comandos
end
```

Tabelas e metatabelas

As tabelas são a única forma de estruturar dados em Lua. Elas implementam um dicionário de dados armazenando pares de (chaves, valores) através de uma combinação eficiente de arranjos e *hash*. Assim, as chaves numéricas são armazenadas como arranjos (iniciando a partir de 1) e as chaves não-numéricas são armazenadas como uma *hash*. Os valores armazenados nas tabelas não possuem um tipo pré-definido. Pode-se, então, armazenar valores de qualquer tipo, desde valores numéricos a funções e, inclusive, outras tabelas. O mesmo vale para as chaves. Ou seja, qualquer tipo de dado pode ser uma chave, inclusive tabelas e funções. O exemplo a seguir ilustra esses conceitos, mostrando também que há diferentes formas de criar e de acessar os dados de uma tabela.

```
local t = {}  -- cria uma tabela vazia
t[1] = 4      -- o 1o elemento da tabela é o número 4
t[2] = "x"    -- e o 2o elemento é a string "x"
t["x"] = 5    -- o elemento de chave "x" é o número 5

q = {4, "x", x = 5} -- cria uma tabela igual a anterior

t.c = 0      -- equivale a escrever t["c"]= 0
t[t[2]] = 4  -- como t[2] = "x", então faz t["x"]= 4

local enemy = {
  strength = 640,  -- chave "strength" tem valor 640
  agility = 480,
```

```
  behavior = {  -- elemento da tabela que é outra tabela
       attack = function() … end,-- os valores são funções
      escape = function() … end,
  },
}
```

Apesar de ser a única forma de estruturar dados em Lua, as tabelas permitem a construção de estruturas mais complexas como listas, filas, pilhas, árvores, etc. Por exemplo, uma lista simplesmente encadeada pode ser implementada como ilustrado no exemplo a seguir. A função addList() retorna uma nova tabela com 2 atributos: um para armazenar o valor do nó, outro com referência para a tabela passada (cabeça da lista).

```
function addList( l, v )
  return { next = l, value = v }
end
list = {}                -- cria uma lista (tabela) vazia
list = addList(list, 7)  -- insere um valor na lista
```

Lua é uma linguagem procedural e, portanto, não possui construções sintáticas que dêem suporte a Orientação-Objeto (OO). Porém, como uma tabela pode ter como atributos tanto dados quanto funções, uma tabela pode representar objetos. Para facilitar a ligação com OO, foi definida uma forma especial de especificar e chamar funções que simulam ser "métodos" de tabelas. Essa forma utiliza o operador ":" nada mais é do que um *alias* para acessar a função da tabela passando a própria tabela como primeiro parâmetro. Na definição de uma função usando o operador ":", o parâmetro que recebe a tabela é definido automaticamente através da variável "self". O código a seguir ilustra esse conceito. Nele, a tabela Rect é criada com dois atributos, mas a função draw() lhe é adicionada usando o operador ":". Assim, implicitamente é criado o parâmetro self, que pode ser usado no corpo da função. Da mesma forma, quando a função é chamada usando o operador ":", a própria tabela é passada como parâmetro.

```
Rect = {
  width = 10,
  height = 10,
}

function Rect:draw()  -- equivalente a Rect.draw(self)
  print(self.width, self.height)
end

Rect:draw()   -- equivalente a Rect.draw(Rect)
```

Mecanismos de herança e encapsulamento podem ser simulados através do uso de metatabelas. Estas alteram o comportamento padrão dos operadores que atuam sobre

uma tabela, conforme ilustra a exemplo a seguir. Nesse exemplo, foi definido um construtor para Rect. Esse construtor irá criar novas tabelas baseadas na tabela Rect. Para isso, define-se Rect (que será recebida como o parâmetro self) como a metatabela da nova tabela criada obj (que será a "instância da classe") e muda o operador de acesso (__index) da metatabela apontando-o para si mesmo (Rect). Isso fará com que as tentativas de acesso a atributos não definidos em obj sejam repassados para quem __index está apontando, ou seja para a tabela Rect. Assim, a nova instância criada (obj) terá as funções definidas na "classe" Rect.

```
function Rect:new()
  local obj = {}     -- obj será a nova instância de Rect
  setmetatable(obj, self) -- Rect é a metatabela de obj
  self.__index = self     -- muda o operador de acesso
  return obj              -- retorna a nova instância
end

r1 = Rect:new()
r2 = Rect:new()
r1:draw()
r2:draw()
```

Este é um exemplo simples, apenas para ilustrar a possibilidade de programar segundo o paradigma OO em Lua, mesmo sendo esta uma linguagem procedural. Outros mecanismos, como herança, encapsulamento, polimorfismo, etc. podem ser simulados alterando o comportamento dos operadores, normalmente realizado através das metatabelas.

3. INTRODUÇÃO A MOAI-SDK[1]

A necessidade do mercado fez surgir inúmeras ferramentas de desenvolvimento de jogos multiplataformas, cada uma com suas vantagens e desvantagens, que dependem do tipo de jogo a ser desenvolvido, do tipo de suporte pretendido, do nível de maturidade da equipe desenvolvedora, entre outros. Assim, fazer uma escolha sobre a ferramenta mais adequada depende de vários fatores, mas principalmente do jogo pretendido e do *know-how* da equipe desenvolvedora.

Há, por exemplo, ferramentas voltadas ao desenvolvimento de jogos em *Desktop* que são extremamente simples, fáceis de usar, apresentando uma curva de aprendizado relativamente pequena em comparação às demais. Porém, essa simplicidade trás normalmente consigo uma limitação na utilização de recursos, seja em razão do modelo arquitetural da ferramenta, seja em função da abstração realizada para acessar os recursos das plataformas-alvo. Por outro lado, há ferramentas extremamente versáteis,

[1] O MOAI-SDK não engloba o MOAI-Cloud, que é um serviço pago provido pela ZipLines para facilitar o armazenamento de jogos online. Não estamos abordando o serviço.

que permitem explorar diretamente os recursos nativos das plataformas-alvo. Entretanto, esta versatilidade promove, geralmente, um custo alto na curva de aprendizado, podendo não compensar no ciclo de desenvolvimento de um jogo.

Aparentemente, o ideal seria encontrar um equilíbrio entre essas duas opções. Porém, esta solução também não é necessariamente a mais adequada, uma vez que a mesma pode ser complexa demais para equipes iniciantes e limitada demais para equipes experientes. Uma solução mais adequada seria uma ferramenta híbrida, que tivesse uma camada para simplificar o uso da mesma, por exemplo através de uma linguagem *script* fácil de aprender e uma API fácil de utilizar, mas que, se necessário, os desenvolvedores mais experientes pudessem acessar recursos específicos de uma plataforma sem os limites impostos pela camada acima.

A ferramenta MOAI foi construída nesse paradigma de solução híbrida. Ela possui uma API específica para o "modo simples", voltada à programação na linguagem Lua. Esta camada tem como objetivo abstrair o uso dos vários motores utilizados pela MOAI (o gráfico, feito sobre OpenGL, o motor físico, usando o Box2D, um motor de particulas e o motor de áudio, usando o MOAIUntz). Porém, se necessário, o desenvolvedor pode acessar diretamente os motores descritos através de uma API em C++. Além disso, se a limitação for referente ao uso de um recurso específico de uma plataforma, o desenvolvedor poderá igualmente acessar o *host*-alvo permitindo manipulação do recurso. Isso é realizado na linguagem de programação nativa do host, por exemplo Java se o host for Android ou Objetive-C se for iOS. A Figura 2 ilustra a abordagem adotada.

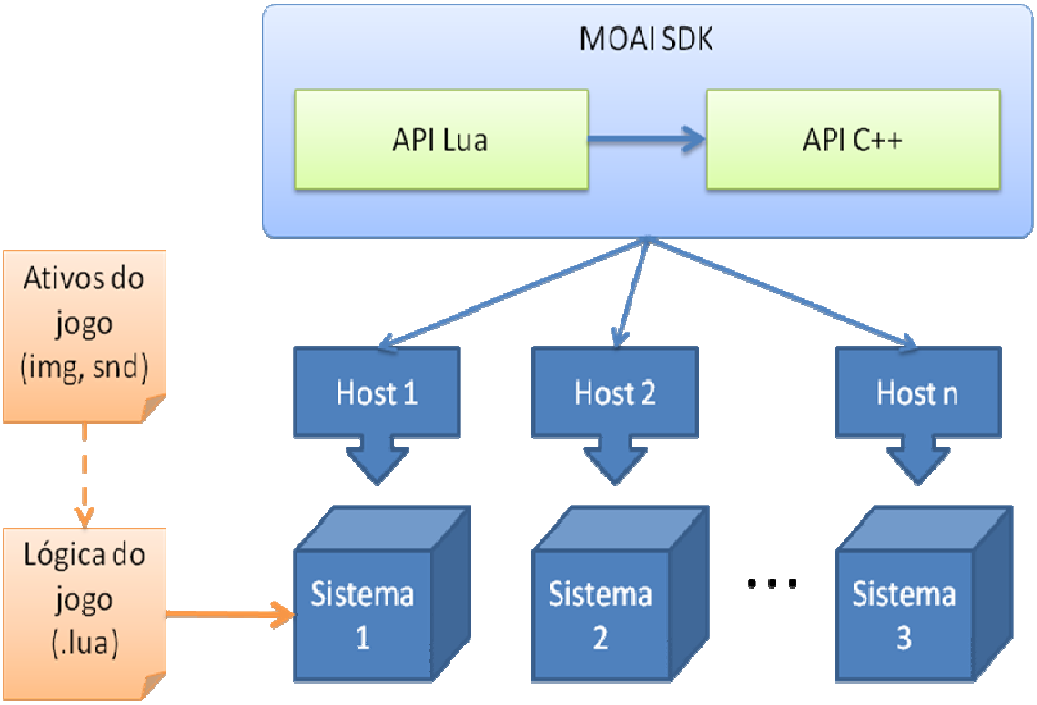

Figura 2. Abordagem multiplataforma adotada no MOAI.

A vantagem do MOAI em relação às demais ferramentas é a capacidade de se criar *hosts* para diferentes plataformas. No SDK disponibilizado por padrão, há *hosts* para Android, iOS, Windows, MacOS, bem como para ser executado como aplicativo nativo do Chrome (e assim, ser também aplicação web). Porém, em função da necessidade de uma equipe desenvolvedora, é possível criar novos *hosts* endereçando outras plataformas e/ou incluindo acesso a recursos externos, como, por exemplo, o uso do Kinect. Sendo um projeto *Open-Source*, a flexibilidade da abordagem é, portanto, o seu ponto forte.

Apesar das possibilidades mencionadas, por limitações de tempo e espaço, iremos abordar neste curso apenas a API acessível através de código em Lua. De fato, esta é a principal forma de uso do MOAI, permitindo que os desenvolvedores se concentrem na lógica do jogo, independente de plataforma.

3.1 Conceitos iniciais

O MOAI é um motor de jogos organizado em forma de componentes. Pode-se organizar os elementos de um jogo de forma hierárquica, fazendo que haja dependência de transformações sobre os elementos (elementos de hierarquia mais alta transmitem suas transformações aos seus sub-elementos). Há igualmente componentes específicos para tratar ações sobre atributos de um elemento do jogo, podendo alterar, por exemplo, cor, posição, rotação, seja de forma direta ou gradativa, entre outros.

De forma resumida, os principais componentes do MOAI são:

- **Props**: São os elementos visuais do jogo. Eles possuem atributos pré-definidos, como posição, cor, tamanho, entre outros, e podem ser hierarquizados. Os valores dos atributos podem ser definidos na própria instância do elemento, mas também como função de outros atributos (por exemplo, a cor mudar em função da posição do personagem), seja da própria instância ou de outras instâncias de Prop;
- **Decks**: São os modelos visuais sobre os quais os Props se baseiam. Por exemplo, imagens únicas ou imagens organizadas em tiles podem servir como base visual para os Props. Essa separação Deck-Prop é útil para que um Deck (modelo) possa ser utilizado por vários elementos do jogo (Prop). Assim, podemos ter, por exemplo, vários "vilões" com o mesmo aspecto visual.
- **Layer**: Definem uma área de visualização onde Props são inseridos. Um jogo pode ter vários layers, cada um com um sistema de coordenada próprio (*viewport*). Assim, é possível definir diferentes janelas e HUDs (*Head-ups Display*) no jogo.
- **Actions**: Permitem a criação de atualizações no jogo, definidas numa árvore de dependência. As actions são acionadas pelo motor MOAI de forma que simulem um "paralelismo" das diferentes ações que ocorrem no jogo.

O processo de renderização no MOAI é independente da atualização do "mundo" (simulação do jogo). Assim, há uma taxa de atualização dos frame de visualização e outra de atualização dos elementos do jogo. Esta última pode ser configurada para ocorrer em uma frequência específica, fazendo com que a velocidade das modificações sobre os elementos do jogo não ocorra de maneira diferente se este for executado em plataformas com processadores de velocidades diferentes (o passo de simulação é feito em um tempo fixo).

Como mencionado, o MOAI organiza as Actions através de uma árvore de dependência. Quando uma Action é criada, ela é inserida na árvore de Actions do MOAI (na raiz ou como filho de outra Action) e é atualizada a cada passo de simulação. A maioria das Actions repassa a atualização aos seus filhos, mas ela pode também modificar a chamada alterando, por exemplo, o tempo simulado de forma a criar, por exemplo, efeitos como o de câmera lenta. Se uma Action é removida, ela e todos seus filhos são removidos da atualização do jogo.

Além da árvore de Actions, há também uma estrutura fundamental no MOAI, que é o grafo de dependências, cujo objetivo é estabelecer a ordem na qual os elementos do jogo são atualizados. Se um elemento depende de outro (seus atributos são dependentes), ele será atualizado apenas após a atualização deste. A estrutura é, portanto, ordenada topologicamente antes de iniciar o processo de atualização.

As ativações da árvore de Actions e do grafo de dependência ocorrem em momentos diferentes. Primeiro, as Actions são ativadas e, se um atributo de uma Prop que possui dependentes é modificado, esta alteração não é propagada no grafo de dependência. Somente após a ativação de todas as Actions da árvore é que as dependências são resolvidas. Assim, se tivermos um jogo com quatro Props, dois carros e dois competidores, estes com dependência de posição dos primeiros, e duas Actions, uma para controlar cada carro, as Actions iriam atualizar os carros e, somente após a posição dos dois serem atualizadas, é que a posição dos competidores seriam ajustadas em função delas. Tudo isso ocorre em um único passo de simulação, logo o jogador não percebe como a relação de dependência ocorre.

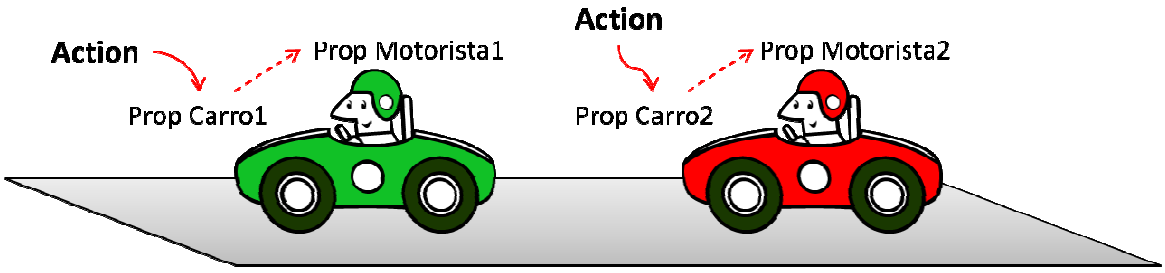

Figura 3. Somente após as Actions atuarem sobre os Props, que estes irão resolver interdependências.

3.2 Ambiente de desenvolvimento

O primeiro passo para começar a desenvolver seu jogo é preparar o ambiente, baixando as ferramentas adequadas e configurando, se necessário, variáveis de

ambiente. Para baixar o MOAI SDK, é necessário registra-se no site do MOAI[2] ou acessar o repositório do código-fonte no GitHub[3]. Basta, então, descompactar o arquivo e os exemplos presentes na pasta 'samples' podem ser executados (via .bat). Para editar seu próprio código, qualquer editor de texto (preferencialmente um reconheça código em Lua) pode realizar essa tarefa, do Vim ao Eclipse.

Para facilitar, disponibilizamos um ambiente pré-configurado[4] para o Windows com o editor ZeroBrane[5] e bibliotecas que consideramos essenciais no desenvolvimento de um jogo: LDocs para a documentação e Busted para realização de testes unitários. Porém o uso destas ferramentas não faz parte do escopo do presente curso e, portanto, não serão tratadas aqui. Para que o ambiente funcione, é necessário que a variável de ambiente MOAI_BIN seja configurada para a pasta bin do MOAI-SDK baixado e a variável MOAI_CONFIG para a pasta onde o arquivo de configurações do MOAI config.lua encontra-se (normalmente em 'samples/config').

3.3 Preparando a visualização

O primeiro passo na criação de um jogo é definir sua área de apresentação. Em dispositivos como tablets e smartphones, o jogo tomará toda a tela do dispositivo. Porém, no caso de *hosts* em Desktop, é necessário criar uma janela. Isto é feito através do componente *singleton* MOAISim, que é responsável por gerenciar os passos da simulação (jogo). No exemplo abaixo, definimos uma janela com título "Invasion" de tamanho 480x320 e com cerca de 30 atualizações por segundo (vale ressaltar que esta é a frequência de atualização da simulação e não da renderização).

```
MOAISim.openWindow("Invasion", 480, 320)
MOAISim.setStep (1/30)
```

É necessário também definir o elemento sobre o qual os elementos visuais são inseridos. Isso é feito através do objeto Layer. O *layer* define uma área de visualização, que possui seu próprio sistema de coordenadas, definido através de um *viewport*. Este estabelece uma relação entre as dimensões do sistema de coordenada nativo do dispositivo (em termos de pixels) e as dimensões do mundo virtual do jogo. O código a seguir exemplifica a criação de um *viewport* criando a relação exposta na Figura 4.

```
viewport = MOAIViewport.new()
viewport:setSize(480,320)   -- área em pixels do device
viewport:setScale(1000,500) -- dimensões do mundo virtual
```

[2] http://getmoai.com/moai-sdk.html
[3] https://github.com/moai/moai-dev
[4] http://goo.gl/jpEqX
[5] http://studio.zerobrane.com/

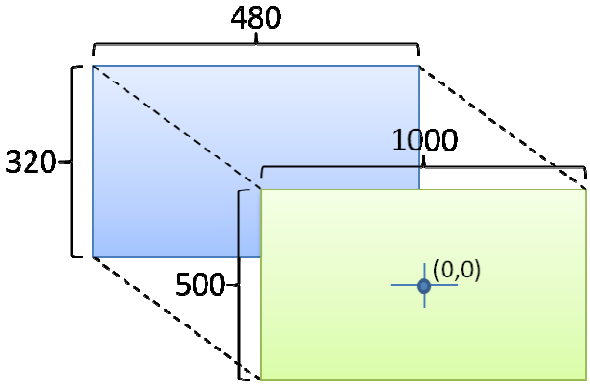

Figura 4. Relação entre as coordenadas do dispositivo e as do *viewport*.

Por padrão, o sistema de coordenadas criado pelo *viewport*, ao contrário de muitos sistemas gráficos computacionais, possui sua origem (0,0) no centro da área definida e a ordenada (eixo Y) não é invertida (cresce para cima). Porém, pode-se modificar esse padrão usando uma escala invertida ou alterando o deslocamento (*offset*) do *viewport*. Pode-se igualmente definir uma rotação para o *viewport*, permitindo assim uma rotação da câmera de projeção.

Com o *viewport* criado, precisamos criar a área de projeção e inclui-la no renderizador do MOAI (MOAIRenderMgr). Para isso, é necessário passar uma tabela com os layers a serem projetados. A ordem que eles são definidos na tabela estabelece a ordem de profundidade de projeção (*z-buffer*) de cada layer. No exemplo abaixo, a tabela passada a setRenderTable() possui um único elemento, que é o layer criado.

```
layer = MOAILayer2D.new()
layer:setViewport(viewport)
MOAIRenderMgr.setRenderTable {layer}
```

3.4 Inserindo imagens e animações

Como mencionado anteriormente, antes de apresentar qualquer elemento do jogo (Prop), é necessário definirmos o modelo visual em que ele se baseia (Deck). Existem várias tipos de Decks (gfxQuad2D, gfxQuadDeck2D, gfxQuadListDeck2D, scriptDeck, stretchPatch, tileDeck2D e tilemap), cada um com um propósito específico. A título de introdução, iremos apresentar apenas um deles, gfxQuad2D, cujo objetivo é apresentar uma imagem bitmap simples.

```
model = MOAIGfxQuad2D.new()
model:setTexture ("character.png")
model:setRect (-30, -10, 30, 60)
character = MOAIProp2D.new()
character:setDeck (model)
character:setLoc (300, 200)
layer:insertProp (character)
```

No código anterior, um modelo é criado para usar a imagem "character.png". Esta imagem possui um tamanho em pixels que não necessariamente corresponde às dimensões que ela deve ser apresentada no mundo virtual. Faz-se, portanto, necessário estabelecer as dimensões que a imagem terá no jogo. Para isso, informa-se através de setRect() as coordenadas mínimas e máximas da região envoltória (*bounding-box*) do modelo. Esta região será preenchida pela imagem quando um Prop usá-lo. A função setRect() define também, de forma implícita, o ponto de origem do modelo. A origem está associada à posição do Prop no jogo. Ou seja, no código anterior, ao posicionar o Prop character na posição (300,200) do mundo (setLoc()), o ponto (0,0) de model passa a ocupar essa posição, conforme ilustra a Figura 5.

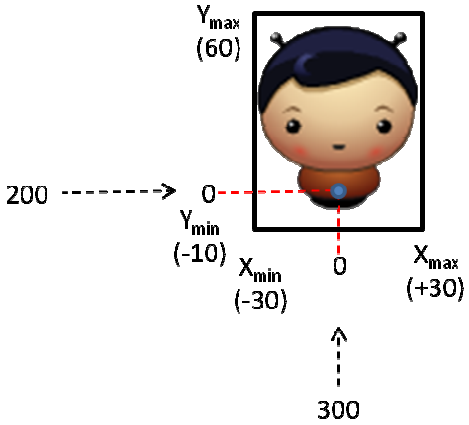

Figura 5. Posicionamento da imagem a chamada Rect(-30, -10, 30, 60).

Além de setLoc(), há outras possíveis transformações sobre os Props, como setRot(), para definir uma rotação, e setScl(), para definir uma escala. As transformações podem ser igualmente hierarquizadas. Nesse sentido, as transformações dos Props que não se encontram no topo da hierarquia são sempre relativas ao Prop ou transformação do nível acima. Por exemplo, no código a seguir o prop1 ficaria 10 unidades à esquerda de root e prop2 10 à direita, ocupando, portanto, em X as posições 20 e 40 respectivamente.

```
root = MOAITransform.new()
root:setLoc(30,0)

prop1 = MOAIProp2D.new()
prop2 = MOAIProp2D.new()

prop1:setParent(root)
prop2:setParent(root)

prop1:setLoc(-10,0)
prop2:setLoc(10,0)
```

Da mesma forma que podemos modificar atributos de Props com as transformações citadas, podemos igualmente criar Actions para modificar esses atributos gradativamente ao longo de um tempo predefinido. O código a seguir ilustra esse uso criando Actions para deslocar prop de 200 unidades em X e 100 em Y, rotacioná-lo de 90 graus e mudar sua escala em 3 vezes em ambos os eixos, ao longo de 2 segundos em todas as operações.

```
prop:moveLoc(200,100, 2)
prop:moveRot(90, 2)
prop:moveScl(3, 3, 2)
```

Vale ressaltar que, mesmo que as operações estejam em sequência, por serem Actions, as transformações ocorreram em paralelo. Como citado anteriormente, todas as Actions são atualizadas a cada passo de simulação em função do tempo decorrido. Por padrão, os valores iniciais (atuais) e finais (definidos na chamada da função) são interpolados segundo o método SMOOTH, que suaviza os transformações nos valores iniciais e finais. Porém, pode-se igualmente utilizar outros métodos, como EASE_IN, EASE_OUT, FLAT, LINEAR, entre outros), como exemplificado no código a seguir.

```
prop:moveLoc(200,100, 2, MOAIEaseType.LINEAR)
```

Além das Actions predefinidas (move, moveLoc, movePiv, moveRot, moveScl, moveColor, seek, seekLoc, seekPiv, seekRot, seekScl, seekColor), o MOAI provê mecanismos para definirmos nossa própria forma de transição ou forma de interpolação, seja sobre atributos predefinidos dos Props ou sobre atributos específicos do jogo sendo desenvolvido. O exemplo a seguir ilustra essa ideia definindo uma função que cria uma única Action (MOAIEaseDriver é subclasse de Action) para modificar a coordenada X de dois Props. Os atributos podem ser específicos do jogo, permitindo, por exemplo, que um personagem aumente sua força (atributo) ao longo de um tempo.

```
function mirror(prop1, prop2, delta, time)
  ease = MOAIEaseDriver.new()
  ease:reserveLinks(2)
  ease:setLink(1, prop1, MOAIProp2D.ATTR_X_LOC, delta)
  ease:setLink(2, prop2, MOAIProp2D.ATTR_X_LOC, delta)
  ease:setLength(time)
  ease:start()
end
```

Apesar das Actions serem executadas em paralelo, muitas vezes queremos que elas ocorram de forma sequencial, uma após a outra. Existem duas formas de sequenciar Actions: uma através de *callbacks* (a serem chamadas quando uma Action termina) e a outra através de corotinas. O código a seguir ilustra a primeira forma criando uma

Máquina de Estado onde, após a rotação do Prop de 180 graus em 1 segundo (Action criada em `acao1()`), a função `acao2()` é chamada, criando uma nova Action, que ao terminar chama `acao3()` para criar a última Action. Dessa forma, temos um sequenciamento de ações.

```
function acao1()
    local action = prop:moveRot (180, 1)
    action:setListener (MOAIAction.EVENT_STOP, acao2)
end
function acao2()
    local action = prop:moveLoc (64, 0, 2)
    action:setListener (MOAIAction.EVENT_STOP, acao3)
end
function acao3()
    prop:moveScl (-0.5, -0.5, 1)
end
acao1()
```

A segunda forma é através de corotinas de Lua. O MOAI possui uma classe, MOAICoroutine, que empacota a biblioteca de corotinas de Lua, inserindo-a no sistema de classes do MOAI e acrescenta funções para facilitar o desenvolvedor. Esta solução dá uma maior flexibilidade uma vez que é permitido testar e realizar as transições de estado em situações específicas, que não dependem das predefinidas pelo MOAI. Por exemplo, o código a seguir cria uma sequência de duas Actions, uma deslocando prop de 10 unidades em cada coordenada em 1 segundo, e a segunda deslocando de 20 unidades em 2 segundos. O código fica em loop após a criação da primeira Action até que ela termine, passando então a criar a segunda Action. Esta fica também fica em loop até terminar. Porém, dentro do loop, testamos a situação em que o ponto (20,30) encontra-se dentro da região envoltória (*bounding-box*) do Prop. Quando isso ocorre, terminamos a Action e, consequentemente, o loop.

```
function actSequence()
  local act = prop:moveLoc (10, 10, 1)
  while act:isBusy() do
    coroutine:yield()
  end

  act = prop:moveLoc (20, 20, 2)
  while act:isBusy() do
    coroutine:yield()
    if prop:inside(20,30) then
      act:stop()
    end
  end
end
```

```
co = MOAICoroutine.new()
co:run(actSequence)
```

Essa forma de sequenciar Actions através de corotinas facilita a criação de scripts de jogo com o foco no comportamento dos personagens. Podemos, assim, definir a lógica como no exemplo a seguir.

```
npc:moveTo(door)
npc:waitWhile(door:isClose)
npc:changeWeapon(knife)
target = closestObj(npc)
while not npc:reach(target) do
  if npc:inFrontOfObstacle() then
    npc:jump()
  end
  if npc:distanceTo(target) < 10 then
    npc:fire(target)
  end
end
```

3.4 Capturando a entrada do usuário

Não se faz jogo sem a "intromissão" do jogador. A captura de eventos do jogador é, portanto, parte fundamental de qualquer jogo. Porém, gerenciar a captura de eventos em jogos que pretendem ser multiplataforma não é tarefa simples, uma vez que eles dependem do dispositivo sobre o qual o jogo será executado. Não adianta, por exemplo, elaborarmos um jogo que usa acelerômetro se ele for executado num *Desktop*.

O uso de Lua como linguagem script permite, entretanto, amenizar alguns dos problemas de ferramentas multiplataforma. Nessa abordagem, ao invés de verificar sobre qual plataforma o jogo está sendo executado, de forma a inferir quais recursos podemos utilizar, testamos diretamente se os recursos estão presentes (provavelmente não sendo nem necessário inferir sobre qual plataforma o jogo se encontra). Por exemplo, para usar o teclado, basta testar se há teclado, para usar acelerômetro, testa-se se há acelerômetro, e assim por diante, sem nos preocuparmos a plataforma utilizada. Na maioria dos jogos voltados tanto a tablets quanto a desktops, o principal teste realizado é: se houver mouse captura o evento de mouse e se não houver captura o evento de toque. Os dados são, então, passados para a mecânica do jogo, que não se preocupa em como os dados foram capturados.

Há, no momento, 9 sensores predefinidos em MOAI (`Button`, `Joystick`, `Keyboard`, `Pointer`, `Wheel`, `Touch`, `Location`, `Motion`, `Compass`), mas novos sensores podem ser adicionados (inserindo um sensor para câmera, por exemplo). Porém, para isso, é necessário alterar o código do *host* das plataformas-alvo.

A captura de eventos é realizada através de *callbacks*. O exemplo a seguir ilustra a captura do evento de teclado, primeiro testando se o dispositivo dispõe de teclado. Se houver, registra uma função (no caso, anônima) a ser chamada quando o evento ocorrer. Nos eventos de teclado, os parâmetros passados são o código da tecla pressionada e um valor booleano informando se a tecla foi pressionada ou solta.

```
if MOAIInputMgr.device.keyboard then
  MOAIInputMgr.device.keyboard:setCallback (
    function ( key, down )
      if down and key == 32 then
        cannon:fire()
      end
    end
  )
end
```

Uma alternativa para os eventos de teclado é verificar se uma determinada tecla encontra-se pressionada. Essa técnica é mais adequada quando o jogo deve reagir de uma forma enquanto o evento for verdadeiro e de outra, caso contrário. Um exemplo para isso encontra-se no código a seguir.

```
if MOAIInputMgr.device.keyboard:keyIsDown("a") then
  cannon:turnLeft()
end

if MOAIInputMgr.device.keyboard:keyIsDown("d") then
  cannon:turnRight()
end
```

Os eventos de mouse são capturados através de sensores de movimento de mouse (pointer) e os respectivos botões (`mouseLeft`, `mouseMiddle`, `mouseRight` e `wheel`). No exemplo a seguir, se existir o sensor de mouse, é adicionado uma *callback* para tratar o evento de movimento do mouse sobre a área de visualização e outra para tratar do evento de clique com o botão esquerdo. O que a primeira *callback* faz é unicamente armazenar a posição do mouse no sistema de coordenadas do mundo virtual (os parâmetros recebidos na função encontram-se no sistema de coordenadas do dispositivo e o método `wndToWorld`() transforma-os baseado nos valores do *viewport* do layer). A segunda *callback* verifica se o botão foi pressionado e armazena na variável `pick` o primeiro Prop (na ordem inversa de renderização) cuja região envoltória engloba o ponto passado. Para fazer esse cálculo, é necessário usar um objeto, `partition`, cujo objetivo é otimizar consultas espaciais (por padrão, ele utiliza listas simples, mas, se necessário, pode-se definir estruturas de busca mais sofisticadas como *quadtrees*).

```
function mouseMove ( x, y )
  wx, wy = layer:wndToWorld (x, y)
end

function mouseClick ( down )
  if down then
    pick = partition:propForPoint (wx, wy)
  end
end

partition = MOAIPartition.new()
layer:setPartition (partition)

if MOAIInputMgr.device.pointer then
  MOAIInputMgr.device.pointer:setCallback (mouseMove)
  MOAIInputMgr.device.mouseLeft:setCallback (mouseClick)
end
```

É fácil perceber que a captura dos eventos não ocorre sobre os elementos do jogo. Ou seja, não há como registrar, por exemplo, um clique sobre um Prop. O clique é capturado de forma global e, depois, verificado se o mesmo ocorreu sobre um Prop ou outro. Se, por um lado, esse mecanismo é adequado a eventos que relacionam vários objetos do jogo (uso de acelerômetro, por exemplo), por outro, ele é inadequado para eventos característicos de certos tipos de objetos (botões, por exemplo). Se o desenvolvedor do jogo desejar registrar eventos diretamente sobre os elementos visuais, ele pode, entretanto, desenvolver uma camada de abstração, capturando os eventos e repassando às *callbacks* individuais de cada objeto registrado.

4. Primeiro jogo no MOAI

Como primeiro exemplo de jogo em MOAI, apresentaremos os passos necessários para desenvolver um jogo simples chamado "Invasion". Este jogo é um clone de jogos existentes desde a década de 80. Nele, o jogador controla um canhão cujo objetivo é não deixar que paraquedistas alcancem sua base. Estes, por sua vez, vão caindo em posições aleatórias na tela e, quando alcançam o chão, dirigem-se à base do canhão. Depois que um certo número de paraquedistas alcançam a base, estes explodem o canhão e o jogador perde. A figura 6 apresenta o jogo sendo executado.

O jogo foi organizado em três arquivos, um para gerenciar o loop principal do jogo, outro para gerenciar as propriedades e ações dos paraquedistas e, por fim, outro para o canhão. O primeiro (main.lua) possui as inicializações de janela e de eventos apresentadas na seção anterior. A novidade reside, entretanto, na definição de um loop para o jogo.

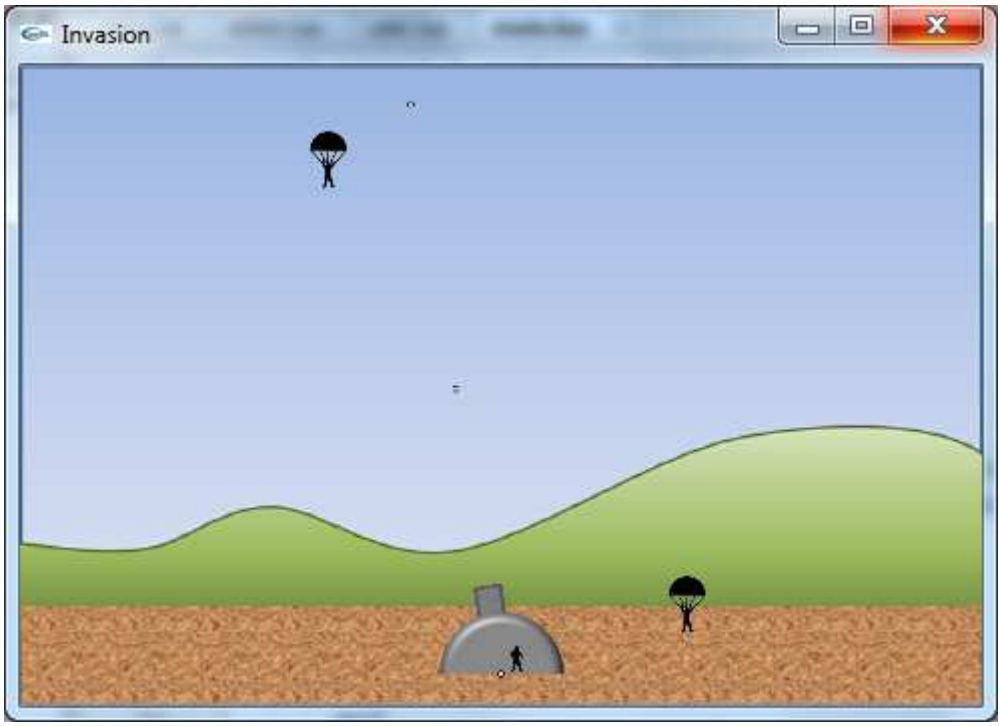

Figura 6. Tela do jogo "invasion", desenvolvido para apresentar os recursos do MOAI.

Criamos o loop principal basicamente para testar as teclas pressionadas (para mover o canhão para esquerda, direita ou atirar) e para criar um novo paraquedista a cada intervalo de tempo predefinido, uma vez que sabemos que a atualização ocorre em intervalos de tempo fixo. Poderíamos, entretanto, se passar do loop principal se registrássemos *callbacks* de eventos de teclado e se definíssemos uma Action específica para a criação de novos paraquedistas. Ambas as soluções são viáveis. Escolhemos a primeira para exemplificar como um loop de jogo pode ser criado no MOAI. Isso porque, diferentemente de outros motores de jogos, o MOAI não possui uma construção específica para definir o loop do jogo, nem uma *callback* padrão a ser chamada a cada passo de simulação. O que existe são as Actions. O gerenciador de simulação do MOAI atualiza cada Action a cada passo e, portanto, para definir um loop de jogo, basta criar uma Action com esse propósito, normalmente usando a abordagem de corotinas.

```
mainThread = MOAICoroutine.new()
mainThread:run(
  function()
    while true do
      coroutine.yield()
      -- gerencia aqui o for necessário
    end
  end
)
```

No arquivo de gerenciamento dos paraquedistas (parachutist.lua), definimos um paraquedista como um objeto que possui apenas o construtor e o método `die`(), que será chamado quando uma bala o atingir. No construtor, após a inicialização do Prop, definimos duas Actions sequenciadas: uma para ele cair (do céu) e a outra, ao término da primeira, para se deslocar à base do canhão. Como não há testes a serem realizados nesta sequência (serão as balas que testarão se atingiram o paraquedista), optamos por definir esta sequência registrando uma *callback* para o término da ação. No código a seguir, o paraquedista é inicialmente posicionado em uma coordenada X aleatória e em Y na posição máxima (`SKY`). Depois, é criada uma Action para deslocá-lo em Y de `-SKY` (ou seja, indo até a valor 0) em 10 segundos. Em seguida, é registrada uma *callback* para quando esta Action terminar. O que a função *callback* faz é criar uma nova Action para deslocar o paraquedista, agora, para o centro do jogo (onde o canhão se encontra) em 1 segundo, usando uma interpolação linear.

```
parac:setLoc ( math.random ( MIN_X, MAX_X ), SKY )
parac.fallDown = parac:moveLoc ( 0, -SKY, 10 )

parac.fallDown:setListener ( MOAIAction.EVENT_STOP,
  function()
    parac:seekLoc ( 0, 0, 1, MOAIEaseType.LINEAR )
  end
)
```

No arquivo de gerenciamento do canhão (cannon.lua), definimos o canhão como um objeto composto hierarquicamente por dois outros: a base e cano. Assim, posicionando o canhão, estamos também definindo a posição de seus componentes, enquanto podemos rotacionar apenas o cano sem alterar os demais elementos. O canhão possui os seguintes métodos: `turnLeft`(), `turnRight`(), `fire`() e `explode`(). Enquanto os métodos `turnLeft`(), `turnRight`() e `explode`() basicamente alteram as propriedades do canhão (não fizemos nenhum efeito especial para a explosão do canhão), o método `fire`() aciona uma Action responsável pelo disparo da bala do canhão. Esta Action deve estar ativa enquanto não tocar um paraquedista. Assim, criamo-la através de corotinas a fim de testar esta situação específica durante sua execução.

```
function bulletAction (bullet, maxX, maxY)
  local moving = bullet:moveLoc (maxX, maxY, 1,
                                 MOAIEaseType.LINEAR)
  while moving:isBusy() do
    coroutine.yield()
    local bx, by = bullet:getLoc()
    for parac in pairs (parachutists) do
      if parac:inside (bx, by) then
        parac:die()
```

```
        moving:stop()
      end
    end
  end
  bullet.fire:stop()
  layer:removeProp (bullet)
end

function cannon:fire()
  local maxX = -MAX * math.sin (math.rad (self.angle))
  local maxY =  MAX * math.cos (math.rad (self.angle))

  local bullet = MOAIProp2D.new()
  bullet:setDeck (bulletModel)
  layer:insertProp (bullet)

  bullet.fire = MOAICoroutine.new()
  bullet.fire:run (bulletAction, bullet, maxX, maxY)
end
```

O código do método `fire`() inicialmente calcula os valores máximos em X e em Y da bala (além desses limites, não há razão da bala existir no jogo) em função do ângulo do canhão e cria um novo Prop para a bala de canhão. Em seguida, ele cria e inicia uma Action para realizar o deslocamento da bala. A chamada do método run() da corotina passa como parâmetro a função que servirá de *thread (*`bulletAction`()) e seus parâmetros: a Prop recém-criada (bullet) e as coordenadas máximas calculadas. A função `bulletAction`(), por sua vez, cria uma Action para deslocar a bala até as coordenadas passadas ao longo de 1 segundo seguindo uma interpolação linear. Se, durante este percurso (ou seja, enquanto a ação moving estiver ocupada), a posição da bala encontrar-se dentro da *bounding-box* de algum dos paraquedistas, o paraquedista é então eliminado (método `die`()). Ao término do loop, seja porque a bala chegou às coordenadas máximas ou porque tocou em um paraquedista, `fire` é removido da árvore de Actions e a bala do layer.

5. Bluft: estudo de caso de um Jogo de estratégia baseado em tiles

Determinados elementos de um jogo desenvolvido em MOAI são imprescindíveis e estarão presentes em qualquer projeto elaborado com esta ferramenta. Elementos como *Decks*, *Props* e *Layers* são cruciais em qualquer abordagem, visto que não há outra maneira de renderizar elementos com MOAI sem utilizá-los. Contudo, existem diferentes abordagens para utilizar essas e outras estruturas definidas por MOAI. Cada uma dessas estratégias deve se adequar a diferentes aspectos do *Game Design*. Com o intuito de revelar alguns desses mecanismos, iremos apresentar um estudo de caso baseado em um jogo completo, comercializado no Google Play, desenvolvido por dois

autores deste texto. Com o objetivo de contextualizar algumas decisões de projeto, iremos descrever brevemente o jogo desenvolvido, denominado *Bluft*.

Bluft é um jogo de estratégia para plataformas Android e iOS. Neste jogo, o usuário terá que salvar a maior quantidade de criaturas (blufts) possível. Ao salvar todas as criaturas, o jogador será desafiado em um novo estágio. O jogador atuará no papel de um herói representado por um bluft mais velho, munido com explosivos (ver Figura 7). Os explosivos são divididos em três categorias:

- **Verde**: Utilizado para atrair os blufts para aquele local;
- **Vermelho**: Utilizado para destruir objetos do cenário;
- **Azul:** Utilizado para criar obstáculos no cenário.

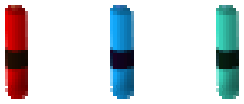

Figura 7. Tipos de bombas presentes no jogo.

Os blufts são salvos quanto atingem uma marca no canto inferior direito de cada cenário. Cada estágio possui uma quantidade limitada para cada tipo de bomba, assim, o jogador deve usar uma estratégia que economize a quantidade de bombas utilizadas. Em alguns estágios, surgem inimigos (robôs) que tem o poder exterminar os blufts. Quando um bluft é morto, o jogador perde e o cenário tem que ser jogado novamente, até que o usuário seja capaz de salvar todos os blufts. O detalhe interessante é que os robôs não podem matar o herói (jogador) e também não podem ser destruídos. Ou seja, o jogador tem que elaborar uma estratégia para contornar a existência do robô em uma determinada posição do cenário. Uma cena do jogo com todos os elementos descritos pode ser observada na Figura 8.

5.1 Arquitetura de Suporte a Renderização

Os elementos gráficos do bluft foram organizados em folhas de sprites (*Sprite Sheets*). De maneira ideal, os elementos deveriam ser agrupados visando criar um conjunto de imagens que otimizasse o uso de memória, gerando imagens cujas dimensões sejam potências de dois. Imagens que não possuem tais dimensões são tratadas para ocuparem tal espaço, fazendo com que memória seja desperdiçada, pois um conteúdo qualquer é adicionado a imagem, fazendo com que esta tenha uma dimensão em potência de dois, obrigatoriamente. Esse recurso não altera a imagem exibida no jogo, mas faz com que uma memória a mais seja gasta para “preencher” o espaço. Contudo, devido ao pouco tempo disponível para o desenvolvimento de Bluft, as folhas de sprites foram separadas de acordo com os elementos que estas representavam. O herói, os blufts e os robôs, por exemplo, são representados em folhas separadas. O mesmo acontece para todos os elementos do cenário. As imagens foram armazenadas no formato png com 8 bits. A Figura 9 ilustra uma folha de sprites utilizada para a animação de um bluft, enquanto a Figura 10 apresenta a folha utilizada para compor o cenário.

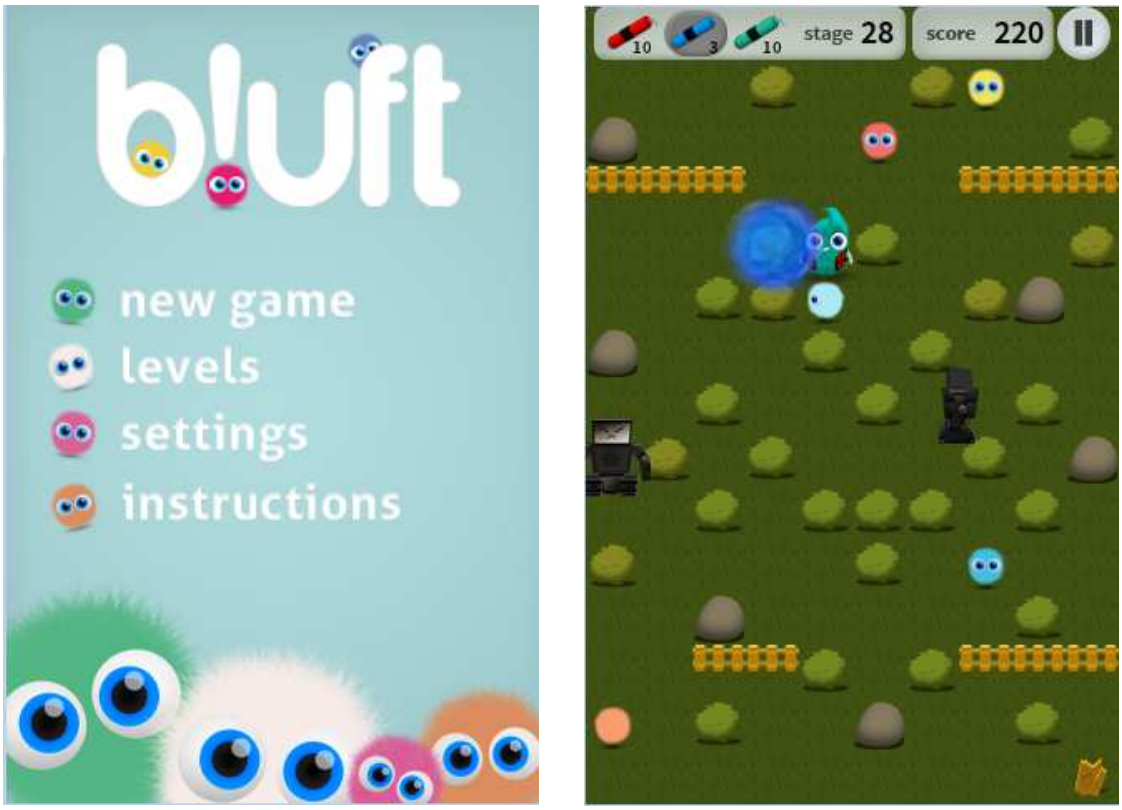

Figura 8. Screenshot da tela de entrada e durante o jogo Bluft

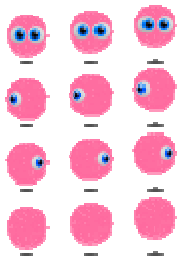

Figura 9. SpriteSheet do Bluft

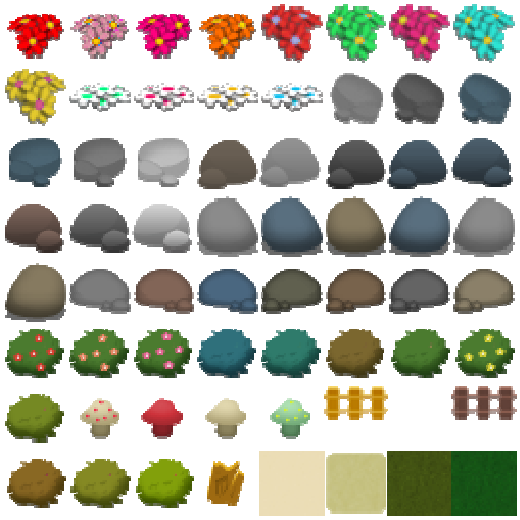

Figura 10. SpriteSheet dos demais elementos visuais do jogo.

Para representar a folha de sprites, foi criada uma classe denominada *SpriteSheet*. Essa classe encapsula um MOAITileDeck2D e outros atributos necessários para definir a estrutura da folha como o tamanho de cada elemento da folha e quantidade de linhas e colunas que aquela folha tem. A folha de sprites que representa o bluft, por exemplo (Figura 9) possui 4 linhas e 3 colunas, com cada elemento medindo 32 pixels de altura e largura, totalizando uma imagem 96 x 128 pixels.

```
function SpriteSheet:new ( aImgPath, aSize, aGridSize )
   local fields = {}
   fields.moaiDeck = MOAITileDeck2D.new ()
   fields.size    = aSize
   fields.gridSize = aGridSize

   fields.moaiDeck:setTexture( aImgPath )
   fields.moaiDeck:setSize( aGridSize.x, aGridSize.y )
   fields.moaiDeck:setRect( -aSize.x/2, -aSize.y/2,
      aSize.x/2, aSize.y/2 )

   return setmetatable ( fields, SpriteSheet_mt )
end
```

Conforme pode ser observado no construtor do SpriteSheet, utilizamos os métodos `setTexture()`, `setSize()` e `setRect()` para definir os atributos da nossa folha de sprites. setTexture define a imagem que utilizaremos como folha, `setSize()` define a quantidade de linhas e colunas que a folha tem e `setRect()` define o tamanho de cada elemento da folha.

Entretanto, a folha de sprites apresentada ainda não pode ser renderizada, visto que esta represta um deck e não uma prop. Assim, devemos criar uma entidade que encapsule uma prop e apresente métodos para facilitar sua manipulação. Para desempenhar tal tarefa foi criada a classe Tile.

```
function Tile:new ( aDeck, aPosition, aIndex )
   local fields  = {}

   fields.deck   = aDeck
   fields.position = aPosition
   fields.prop   = MOAIProp2D.new ()

   fields.prop:setDeck ( aDeck )
   fields.prop:setIndex ( aIndex )
   fields.prop:setLoc ( aPosition.x, aPosition.y )

   return setmetatable ( fields, Tile_mt )
end
```

Além de encapsular uma Prop, o tile armazena outros atributos como a posição e o índice. O índice corresponde a um valor inteiro que representa aquele tile em uma folha de sprites. Levando em consideração que os tiles estão intimamente relacionados com as folhas de sprites, foi pertinente criar um método para acessar os tiles a partir de uma folha de sprites:

```
function SpriteSheet:getTile ( aIndex )
   return Tile:new ( self.moaiDeck, {x=0,y=0}, aIndex )
end
```

Os elementos apresentados são suficientes para representar qualquer gráfico do jogo. É possível fazer objeto se moverem, por exemplo, alterando a posição de um determinado tile gradualmente, a cada iteração to loop principal. Entretanto, nossa pequena API ainda não oferece suporte a animações feitas a partir da mudança de tiles em uma folha de sprites. Por exemplo: a movimentação de um bluft é composta pela mudança de quadros em uma linha da sua folha de sprites. Assim, além de tile que representa este elemento sofrer uma mudança de posição, o índice que representa esse tile na folha de sprites também precisa ser modificado. Para representar essa funcionalidade, foi criada uma classe denominada TiledAnimation. Essa classe é inicializada a partir de uma folha de sprites, uma lista de índices que representam os quadros da animação e uma lista de números que representa o tempo que cada quadro deverá permanecer ativo na animação. O método `play()` dessa classe é responsável por tocar a animação definida e merece atenção especial.

Esse método recebe um argumento que representa a quantidade de tempo (em segundos) que se passou desde a última interação. Portanto, a animação é executada incrementando um contador (`self.frameTimeElapsed`) que representa o tempo que o frame atual já foi exibido. Quando esse tempo é maior do que tempo armazenado na lista que define a animação, o frame atual passa a ser o próximo frame.

Essas estruturas compõem os elementos de suporte a renderização e são utilizadas para todos os elementos visuais do jogo.

5.2 Menus e Telas de Apresentação

Para a criação de menus e da tela de apresentação foram criadas 2 classes: Menu e MenuElement. A ideia por trás dessas estruturas é estabelecer uma hierarquia lógica entre as telas do jogo, bem como guardar todos os recursos necessários para visualização e tratamento lógico dos callbacks do game, como o que deve acontecer caso um determinado botão seja pressionado.

A ideia básica por trás da implementação de Menus é conter imagens que serão exibidas para o usuário, guardando e tratando-os como elementos lógicos individuais. Para isso, a classe Menu possui um vetor `elements` que guarda os MenuElements. Cada menu é identificado por um `name` e possui uma referência `previousMenu` para

o menu anterior a ele (caso deseje-se voltar para o menu anterior), um booleano `active` indicando se o menu está ativo, uma referência `nextMenu` para o menu para o qual o game será encaminhado após o tratamento de input, o tempo em que o menu ficará ativo caso ele não seja infinito (ou seja, ele independe de algum input para levar ao próximo menu) e uma `nextAction`, que indica se o menu será trocado ou não.

```
function Menu:new ( aPreviousMenu, aName, aSpriteSheet)
  local fields = {}

  fields.name              = aName
  fields.infinite          = false
  fields.elements          = {}
  fields.previousMenu = aPreviousMenu
  fields.active            = false
  fields.time              = 0
  fields.maxTime      = 0
  fields.nextAction        = ' '
  fields.nextMenu          = nil
  fields.bg                = aSpriteSheet:getTile ( 1 )

  return setmetatable ( fields, Menu_mt)
end
```

As imagens e a posição dos elementos desenhados no menu estão contidos nos MenuElements. Assim, a lógica de mudança de menus é estabelecida no código cliente, tratada no Menu e computada no MenuElement. Tomando como exemplo o menu principal do Bluft, temos um background composto do desenho dos blufts e do logo principal do game. Os nomes “new game”, “level”, “settings” e “instructions” simbolizam MenuElements, contendo um texto e uma posição relativa ao background, bem como referências para cada menu para o qual o jogo deve ser direcionado caso haja um clique sobre uma das opções.

Alguns dos campos da classe MenuElement têm o mesmo significado dos homônimos da classe Menu. Contudo, em MenuElement é preciso destacar a presença de atributos booleanos que identificam o MenuElement como texto (`text`), botão (`button`) ou animação (`animation`). O MenuElement deve ser indicado como animação quando o Menu ao qual ele pertencer for finito. Dessa forma, durante a função `draw()`, é possível realizar os preparativos necessários para a correta renderização dos MenuElements.

Segue a seguir a estruturação básica da classe MenuElement.

```
function MenuElement:new (aPosition, aAction,
      aNextmenu, aSize, aTiledAnimation)

  local fields = {}
  fields.position     = aPosition
```

```
    fields.button       = false
    fields.text         = false
    fields.animation    = false
    fields.font         = nil
    fields.action       = aAction
    fields.nextMenu     = aNextMenu
    fields.tiledAnimation = aTiledAnimation
    fields.pressed= false
    fields.string       = ' '
    fields.size         = aSize
    fields.textProp     = nil

    return setmetatable ( fields, MenuElement_mt)
  end
```

Existem duas funções primordiais presentes na MenuElement: as funções `draw()` e `update()`. A função `update()` recebe como parâmetro a posição aClickPosition da última operação de input efetuada pelo usuário, calculando se o clique realizado pertencia à área correspondente ao MenuElement. Caso sim, a variável booleana pressed é atribuída como verdadeira, gerando um tratamento lógico no Menu ao qual o MenuElement pertence.

```
function MenuElement:update( aClickPosition )
  if not aClickPosition.release then
    self.pressed = false
    return false
  end

  self.pressed=
    (aClickPosition.x > self.position.x - self.size.x/2) and
    (aClickPosition.x < self.position.x + self.size.x/2) and
    (aClickPosition.y > self.position.y - self.size.y/2) and
    (aClickPosition.y < self.position.y + self.size.y/2)
end
```

Para conhecer o estado atual da variável `pressed` de cada MenuElement, o Menu dispõe da função `getAction()` da classe MenuElement. Caso o botão ou texto tenha sido pressionado, o Menu receberá c action do MenuElement correspondente.

```
  function MenuElement:getAction()
    if self.pressed == true then
      return self.action
    else
      return 'nothing'
    end
```

```
    end
```

Dessa forma, a atualização do menu resume-se a um tratamento lógico da atualização de todos os seus MenuElements:

```
function Menu:updateElements ( aClickPosition, aElapsedTime )
  for k,v in pairs ( self.elements ) do
    v:update(aClickPosition)

    if(v:getAction () ~= 'nothing') then
      self.nextAction = v:getAction()
      if(v:getAction () == 'goTo') then
        self.nextMenu = v.nextMenu
        return
      end
    end
  end

  if self.infinite then
    local allNothing = true
    for k,v in pairs ( self.elements ) do
      v:update( aClickPosition, aElapsedTime )
      allNothing = allNothing and v:getAction() == 'nothing'
    end

    if allNothing then
      self.nextAction = 'nothing'
      self.nextMenu = nil
    end
  end
end
```

Já o método `draw()` dos MenuElements consideram com qual tipo de MenuElement estamos tratando, dedicando atenção especial aos textos:

```
function MenuElement:draw ( aLayer )
  if(self.button == true) or (self.animation == true) then
    self.tiledAnimation:putOnLayer ( aLayer )
    self.tiledAnimation:setPosition ( self.position )
  end

  if(self.text == true) then
    self.textProp:setFont( self.font )
    self.textProp:setRect(
      self.position.x – self.size.x/2,
      self.position.y - self.size.y/2,
```

```
        self.position.x + self.size.x/2,
        self.position.y + self.size.y/2
      )
      self.textProp:setAlignment ( MOAITextBox.CENTER_JUSTIFY )

      self.textProp:setString ( self.string )
      aLayer:insertProp ( self.textProp )
    end
  end
```

Analogamente ao `update()` dos MenuElements, a renderização de um Menu invoca o método `draw()` dos MenuElements pertencentes a ele.

5.3 Som

Desde os primórdios dos games, efeitos sonoros e trilha sonora têm sido importantes na popularização e sucesso de alguns títulos. Desta forma, é essencial que a MOAI disponibilize recursos para carregamento e execução de recursos sonoros. Felizmente, esses recursos estão disponíveis e podem ser facilmente incorporados ao game em desenvolvimento. Para isso, a MOAI faz uso de uma engine chamada MOAIUntzSystem.

Primeiramente é necessário inicializar a MOAIUntzSystem, simplesmente chamando a função `initialize()`.

```
MOAIUntzSystem.initialize()
```

Com a MOAIUntzSystem inicializada, é possível escolher um volume global para sua aplicação através da função `setVolume()`. O valor desse volume varia entre 0 (mínimo) e 1 (máximo).

```
MOAIUntzSystem.setVolume(1)
```

A partir desse momento, já é possível criar sons específicos para a aplicação. No jogo Bluft, temos como elementos de som: música em loop do menu principal; a voz dos blufts; a voz do robô; efeito sonoro para entrada em um menu; efeito sonoro para saída de um menu. Para a criação de cada um desses sons, utiliza-se a classe MOAIUntzSound. Tal classe contém um método `load()` que receberá como parâmetro o nome do arquivo de som a ser tocado no game.

Em Bluft, utilizamos o formato WAV, uma vez que, assim como o formato OGG, tem sido reproduzido com sucesso tanto em dispositivos Android quanto iOS. A seguir, um trecho de código de Bluft que exemplifica a utilização da classe MOAIUntzSound e a função `load()`.

```
bgMenuSound = MOAIUntzSound.new ()
bgMenuSound:load ('sounds/menu.wav')
```

Mesmo que já tenha sido definido um valor para o volume global da aplicação, é possível utilizar mais uma vez a função `setVolume()` para definir o volume do efeito sonoro, dessa vez a partir do próprio elemento criado a partir de MOAIUntzSound. Dessa forma, seria possível adicionar a seguinte linha de código ao exemplo acima:

```
bgMenuSound:load ('sounds/menu.wav')
```

Também podemos definir se queremos que nosso som execute em loop ou apenas uma vez. Através da função `setLooping()`, é possível definir se o som executará repetidamente (`setLooping(true)`) ou apenas uma única vez (`setLooping(false)`).

Abaixo, o trecho de código correspondente à inicialização dos sons do jogo Bluft.

```
if MOAIUntzSystem ~= nil then
  MOAIUntzSystem.initialize()
  MOAIUntzSystem.setVolume(1)

  bgMenuSound = MOAIUntzSound.new ()
  bgMenuSound:load ('sounds/menu.wav')
  bgMenuSound:setVolume ( 1 )
  bgMenuSound:setLooping ( true )

  -- ...
end
```

Uma vez armazenados na memória, podemos manipular facilmente sua execução usando as funções `play()`, `pause()` e `stop()`. A função `play()` determina a execução do som. A função `pause()` determina que o som seja interrompido naquele determinado instante, fazendo com que, caso seja utilizada a função `play()` novamente sobre ele, o som volte a ser tocado no trecho exato em que foi pausado. Já a função `stop()` interrompe a execução do som de forma que, caso seja utilizada a função `play()` novamente sobre ele, o som volte a ser executado a partir do início. Dessa forma, eventualmente, no jogo Bluft, seria possível realizar as seguintes operações sobre qualquer um dos sons criados anteriormente.

```
menuInSound:play()
menuInSound:pause()
menuInSound:play()
menuInSound:stop()
```

6. Ferramentas e *Deployment*

Uma das grandes vantagens de desenvolver um jogo em MOAI é poder usar o mesmo código para diferentes plataformas. Como mencionado anteriormente, MOAI disponibiliza hosts para Android, iOS e Chrome. A seguir, abordaremos algumas particularidades do *Deployment* para Android e iOS.

6.1 Deployment no Android

Inicialmente, é necessário que todo o aparato padrão para desenvolver um aplicativo Android esteja configurado. Isso compreende a instalação do SDK Android[6] e o plugin ADT para o Eclipse. Mais informações podem ser obtidos no site.

Para efetuar gerar o host MOAI do Android, devemos modificar e executar alguns scripts que já estão disponíveis no sdk padrão. A pasta do host Android está localizada dentro da pasta hosts no diretório ant. Dentro desse diretório, devemos modifica o arquivo settings-local.sh, de maneira a preencher os campos com a informação referente ao jogo. No campo android_sdk_root definimos o caminho para o sdk Android. No campo src_dirs definimos a pasta que contem os arquivos do jogo (scripts, imagens, sons, etc). No campo run dizemos o nome do script principal a ser executado. Após isso, geramos o host executando os scripts sdk-setup.sh e run-host.sh. O host, com todos os scripts e arquivos do jogo, foi criado dentro da pasta build. A pasta gerada pode ser aberta como um projeto do eclipse e o aplicativo pode ser gerado como um aplicativo Android convencional.

6.2 Deployment no iOS

O *deploy* em um dispositivo iOS requer um computador com MacOS e o programa XCode instalado. O processo é bem mais simples do que o efetuado para Android, pois o host já está pronto para ser utilizado. Basta acessar a pasta xcode do diretório hosts e abrir o arquivo moai.xcodeproj. Com o programa aberto, devemos modificar o arquivo moai-target dentro da pasta build, colocando o caminho da pasta que contem os arquivos do jogo na primeira linha. Ao executar o host pela primeira vez (clicando no botão play), os arquivos são automaticamente copiados para a pasta e o jogo é executado.

6.3 Ferramentas úteis

No desenvolvimento de Bluft, dois softwares se mostraram incrivelmente úteis: o Audacity e o Tiled Map Editor.

O Audacity é um software gratuito de gravação e edição de som bastante simples de se utilizar, contando inclusive com muitos tutoriais e fóruns dedicados a ele.

[6] http://developer.android.com

Está disponível para os sistemas Windows, Mac OS X, GNU/Linux. Para maiores informações, consultar o site oficial da ferramenta[7]

Tiled é um editor gratuito de mapas de propósito geral construído de forma flexível para que possa ser incorporado a qualquer game, desenvolvido em C++ usando o framework de aplicação Qt. No desenvolvimento de Bluft, foi utilizado para organizar cada camada de sprites de todos os cenários, mostrando-se uma ferramenta de fácil utilização. Para maiores informações, consultar o site oficial da ferramenta[8].

7. Comentários finais

O mercado de dispositivos móveis é um dos que mais cresce na economia mundial e, dentro desse segmento, são os jogos as aplicações mais utilizadas. Para dar uma noção desse crescimento, vale destacar que a renda obtida por jogos para dispositivos móveis já ultrapassou gigantes da indústria de jogos como Nintendo e Sony [Woolcock 2012]. De acordo com a empresa de pesquisa de mercado Nielsen, cerca de 64% dos usuários de dispositivos móveis (tablets e smartphones) nos EUA jogam ao menos uma vez por dia [Nielsen 2012].

Associado ao grande público-alvo do mercado de jogos, encontra-se também a facilidade de uma pequena empresa local em disponibilizar seu produto a nível mundial através das centrais de aplicativos (Apple Store, Google Play, Amazon Store, etc.). Assim, incentivar a criação de empresas locais na produção de jogos casuais é fundamental para criamos de uma cadeia de exportadores de *software* e, por conseguinte, fomentar a indústria tecnológica local.

O presente curso teve como intuito contribuir nessa área, dando subsídios para que grupos e pequenas empresas locais possam começar a enveredar no desenvolvimento de jogos casuais através do MOAI. Muitos dos elementos do SDK não foram, entretanto, abordados neste curso, uma vez que a ferramenta é extensa e um curso de 4 horas não dá, certamente, para abordar todos seus elementos, seja em largura ou em profundidade.

Referências

Nielsen (2012) “Play Before Work: Games Most Popular Mobile App Category in US”. Nielsen Wire Reports. (resumo disponível em http://blog.nielsen.com/nielsenwire/online_mobile/games-most-popular-mobile-app-category/. Último acesso em setembro de 2012).

Woolcock, K. (2012) “Is Mobile Gaming Facebook’s Achilles Heel?”. Rev. *Time – Business*. Janeiro de 2012 (disponível em http://business.time.com/2012/01/19/why-mobile-gaming-changes-everything-or-is-mobile-gaming-facebooks-achilles-heel/. Último acesso em setembro de 2012).

[7] http://audacity.sourceforge.net.
[8] http://www.mapeditor.org/

Introdução à Linguagem de Descrição de Hardware – VHDL

Maria Aline P. dos Santos[1], Eliselma V. dos Santos[2], Aparecida L. de Medeiros[2]

[1]Universidade do Estado do Rio Grande do Norte (UERN)
Av. Airton Senna, 4241, Neópolis | Natal/RN – Brasil

[2]Universidade Federal do Rio Grande do Norte (UFRN)
Campus Universitário, Lagoa Nova | Natal/RN – Brasil

aline18278, eliselmavieiradossantos, aparecidalpb }@gmail.com

Abstract. *This minicourse are focused on introducing Hardware Description Language with emphasis on High Speed Integrated Circuit (VHDL- Very High Speed Integrated Circuit Hardware Description Language) in order to show how VHDL the can contribute to the development of digital systems as well how to implement and simulate basic components.*

Resumo. *Este minicurso tem como foco introduzir a Linguagem de Descrição de Hardware com ênfase em Circuito Integrado de Alta Velocidade (VHDL - Very High Speed Integrated Circuit Hardware Description Language), a fim de mostrar como a VHDL pode colaborar para o desenvolvimento de sistemas digitais, bem como implementar e simular componentes básicos.*

1. Introdução

Devido ao aumento da complexidade dos circuitos integrados atuais, os projetos são desenvolvidos utilizando linguagens de descrição de hardware, em inglês, (*Hardware Description Language* – HDL) e os circuitos são gerados automaticamente a partir das descrições em alto nível de abstração. Existem diversas HDLs, as quais são: VHDL, AHDL (*Altera Hardware DescriptionLanguage*), *Verilog*, *Handel-C*, *Abel*.

VHDLé uma linguagem para descrição de *hardware* com ênfase em circuitos integrados de alta velocidade. Esta linguagem foi criada visando à simulação, modelagem e documentação, tal que recebeu a possibilidade de síntese, com o objetivo de automatizar o projeto do circuito. A VHDL teve suas definições postas em domínio público e inicialmente foi usada para fins militares e ganhou popularidade fora deste ambiente, quando a IEEE (*Institute of Electrial and Electronics Engineering*) estabeleceu padrões para tornar a linguagem universal em 1987. A partir desta padronização, houve novas alterações e aprimoramentos e a linguagem sofreu algumas modificações, e um novo padrão foi lançado em 1993. Com a VHDL, pode-se especificar um circuito a partir de seu comportamento ou de sua estrutura, em vários níveis e podendo ser implementado em um dispositivo programável, como o FPGA (*Field Programmable Gate Array*).

Atualmente a ferramenta mais usada para implementação de VHDL é o *Quartus*, ferramenta da *Altera*. Nas versões anteriores do *Quartus* era possível realizar a simulação, porém com o desenvolver de cada versão chegou-se a conclusão de que as versões seguintes não ofereceriam mais tal recurso, e foi desacoplada da versão atual. Assim, passou-se a utilizar a ferramenta *ModelSim* da *Mentor Graphics*, que possui uma

versão para a *Altera*, a qual permite fazer a descrição, compilação e realizar simulações funcionais e temporais.

Este minicurso tem como objetivo introduzir conceitos básicos de sistemas digitais, circuitos lógicos e descrição de *hardware,* assim como a implementação e simulação de componentes básicos em VHDL.

2. Motivação

Nos dias atuais, com o grande avanço das tecnologias e processos de fabricação, os circuitos integrados podem conter milhões de transistores, o teste destes transistores configura-se como o segundo maior custo do chip, só perdendo para o custo de fabricação, pois é necessário garantir que o circuito integrado não possui erros e que o mesmo está em condições de uso pelo consumidor final [Chen, 2003a]. O custo do projeto pode ser reduzido se a etapa de teste for considerada nos estágios iniciais do projeto e consequentemente, em níveis de abstração mais elevados, como em linguagens de descrição de hardware [Chen, 2003b] [Dey, 1998].

O projeto de um circuito integrado geralmente pode utilizar técnicas diferentes de projeto e construir códigos VHDL de uma forma diferente entre elas, mas mesmo assim o circuito resultante precisa atender as especificações de: um correto funcionamento da função a ser realizada, restrição de área a ser ocupada, frequência de ocupação. A linguagem VHDL permite que estas especificações sejam realizadas, de acordo com a necessidade do projeto, sendo assim, buscamos difundir a linguagem para que possamos despertar o desejo de trabalhar com projetos de *hardware* aumentando a demanda de profissionais nesta área.

3. Publico Alvo

Alunos de graduação, técnicos na área de computação, autodidatas e demais interessados em Linguagem de Descrição de *Hardware* com conhecimento em Lógica Computacional e Técnicas de Circuitos Digitais.

4. Grau de Interesse para o EPOCA

A importância da Linguagem de Descrição de *Hardware* atualmente é alta, pois a mesma garante que a produção do *hardware* será sem erros. Esta descrição é uma etapa muito importante para o desenvolvimento tecnológico, pois evita gerar erros na produção de um determinado *hardware,* prevenindo o desperdício da matéria prima usada para desenvolver o mesmo.

Desde a sua criação em 2008 a EPOCA passou a ter minicursos, palestras e discussões de diversas áreas da Computação, há cada edição, o evento passou a possuir conteúdos relacionados com a área de *hardware* e minicursos relacionados à HDL. A grande importância a qual gera interesse para a Escola Potiguar de Computação e suas Aplicações está em propagar as HDLs de forma geral, tendo em vista que o mercado de trabalho precisa de mão de obra qualificada para projetar *hardwares* específicos.

É significante a busca de profissionais nesta área, e, a atual proposta do minicurso é dar uma base aos participantes em VHDL para que seja despertado o interesse pela Descrição de *Hardware*.

5. Cronograma do Minicurso

- Introdução
- Linguagem de Descrição de Hardware - HDL
- VHDL
  - ✓ Projeto
  - ✓ Estrutura
  - ✓ Semântica
  - ✓ Arquitetura
  - ✓ Máquina de Estados
- Software
  - ✓ ModelSim
- Exemplos
  - ✓ Implementações

Tempo estimado: 4 horas

6. Requisitos para Apresentação

Para a realização do minicurso será necessário um projetor para a exposição da parte teórica, um laboratório contendo computadores com o sistema operacional Linux ou Windows, o quais devem conter o software *ModelSim* (disponível em, https://www.altera.com/download/software/modelsim-starter). Porém, se a disponibilização do laboratório for inviável, faremos um minicurso teórico, com ilustrações de exemplos práticos. Para esta parte teórica, irá ser feito uma apresentação de todo conteúdo que será abordado no minicurso, e a prática, os participantes irão se familiarizar com a ferramenta e desenvolver alguns projetos. Podendo também participar do minicurso participantes que possuam *notebook* ou *netbook*.

7. Biografia (curta) dos Autores

Aparecida Lopes de Medeiros, bacharel em Ciência da Computação pela Universidade do Estado do Rio Grande do Norte (UERN). Aluna de mestrado da Universidade Federal do Rio Grande do Norte (UFRN), na área de Sistemas e Computação, com ênfase em Sistema Distribuído e Integrado, fazendo parte do Laboratório de Sistema em Chip (LASIC), realizando pesquisas na área de Sistema e Redes em Chip, Sistemas Digitais. Currículo Lattes: http://buscatextual.cnpq.br/buscatextual/visualizacv.do?id=K4488710T1.

Eliselma Vieira dos Santos, bacharel em Ciência da Computação pela Universidade do Estado do Rio Grande do Norte (UERN). Aluna de mestrado da Universidade Federal do Rio Grande do Norte (UFRN), na área de Sistemas e Computação, com ênfase em Sistema Distribuído e Integrado, fazendo parte do Laboratório de Sistema em Chip (LASIC). Realiza pesquisas nas áreas de Redes em

Chip, Síntese de Alto Nível e Sistemas Digitais Complexos. Currículo Lattes: http://buscatextual.cnpq.br/buscatextual/visualizacv.do?id=K4488719U0.

Maria Aline Pereira dos Santos é graduanda em Ciência da Computação pela Universidade do Estado do Rio Grande do Norte (UERN) e integrante do Grupo de Sistemas Embarcados e de Tempo Real – GSET. Currículo Lattes: http://buscatextual.cnpq.br/buscatextual/visualizacv.do?id=K4365342J8.

Referências

CHEN, C.-I.H. *Behavioral Test Generation / Fault Simulation.IEEE Potentials*, [S.1], v.22, n.1, p. 27-32, Feb. 2003.

CHEN, X.; HSIAO, M. *Energy-Efficint Logic BIST Based on State Correlation Analysis.In: IEEE VLSI TEST SYMPOSIUM, VTS, 2003. Proceedings… Napa Valley*, CA, *USA:IEEE Computer Society*, 2003b. p. 267-272.

DEY, S.; RAGHUNATHAN, A.; ROY, R. *Considering Testability During High-Level Design (Embedded Tutorial). In*: *ASIA AND SOUTH PACIFIC DESIGN AUTOMATION CONFERENCE, ASP-DAC*, 1998. *Proceedings…Yokohama, Japan*:[s.n], 1998. P. 205-210.

D'AMORE, R. Descrição e síntese de circuitos. 2ª ed. 2012.Editora: LTC.

HAMBLEN, J. O; HALL, T. S; FURMAN, M. D. *Rapid Prototyping of digital Systems Quartus*.2ª ed. Editora: Springer.

MESQUITA, DANIEL. VHDL. 2011. Disponível em: <http://teaching.danielmesquita.com/disciplinas/2011_1/gsi0132011_1/material-deapoio>.

TOROK, D. L; CAPELATTI, E. A. Praticando VHDL. 1ª ed. 2010. Editora: Novo Hamburgo: Freevale.

AMORY, ALEXANDRE. Validação de VHDL: técnicas e ferramentas. Disponível em:<https://corfu.pucrs.br/tikiwiki/tutorials/test_bench.ppt>.

Introdução a Plataformas de Computação em Nuvem – Uma Abordagem Prática

Frederico Lopes[1], André Almeida[2,3], Thais Batista[3], Everton Cavalcante[3], Renato Gondim[3], Thomas Diniz[1], Arthur Cássio[3], Thiago Cesar[3]

[1]Escola de Ciência e Tecnologia – Universidade Federal do Rio Grande do Norte (UFRN), Natal – RN – Brasil

[2]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte (IFRN), Parnamirim – RN – Brasil

[3]Departamento de Informática e Matemática Aplicada (DIMAP)
Universidade Federal do Rio Grande do Norte (UFRN), Natal – RN – Brasil

{fred.lopes,thaisbatista, evertonranielly, renatuga, thomasfdsdiniz, arthurecassio, lord.sena}@gmail.com, andre.almeida@ifrn.edu.br

Abstract. *This paper describes a short-course that provides a theoretical and practical perspective on cloud computing platforms usage. By using a practical approach we demonstrate the use of two public cloud computing platform, Amazon Web Services (AWS) and Google Application Engine (GAE) as well as the steps needed to setup a private cloud computing infrastructure, OpenStack.*

Resumo. *Este artigo descreve a proposta de Minicurso com o objetivo de fornecer uma base teórico-prática sobre o uso de plataformas de nuvem. Demonstramos, através de abordagem prática, o uso de uma plataforma de nuvem pública, a Amazon Web Services (AWS) e Google Application Engine (GAE) bem como todo processo necessário para montagem de uma infraestrutura de nuvem privada, OpenStack.*

1. Introdução

A computação em nuvem é um novo paradigma que visa prover **serviços** computacionais acessados via internet e sob demanda, onde usuários pagam apenas pelos serviços que utilizam. Essa ideia de computação como um utilitário segue o mesmo modelo de outros serviços essenciais de utilidade pública, tais como, eletricidade, água, esgoto e telefone, que são entregues à população de modo bastante transparente e o usuário paga apenas pelo que efetivamente consome. Aplicando a mesma lógica de negócio na informática, esse modelo possibilita que serviços de TI sejam oferecidos aos usuários sob demanda, onde estes pagam apenas pelo que utilizaram do serviço.

O modelo de *Computação em Nuvem* provê recursos dinamicamente escaláveis e normalmente virtualizados através de **serviços** sobre a Internet. Nesse modelo, desenvolvedores de serviços para Internet não precisam mais investir muito capital em hardware e em recursos humanos para manter o serviço em operação. Além disso,

empresas que precisam de um alto poder de processamento podem usar a infra-estrutura fornecida por provedores de serviços em nuvem, ao invés de investor recursos em manter um parque computacional que demanda máquinas potentes e equipe técnica especializada.

Do ponto de vista do usuário, infra-estruturas de computação em nuvem disponibilizam sistemas operacionais que possibilitam que os usuários, utilizando qualquer dispositivo conectado a Internet, possam ter acesso aos serviços, arquivos, informações e programas situados "na nuvem". Assim, os dispositivos utilizados pelos usuários não necessitam de grandes recursos computacionais, aproximando-se de simples terminais (Armbrust et al. 2009),.

Atualmente há uma variedade de ambientes de nuvem disponíveis no mercado que dão suporte ao desenvolvimento de aplicações. Amazon Web Services (AWS)[1], Google App Engine (GAE)[2], Microsoft Azure Platform[3] e OpenStack[4] são alguns exemplos de ambientes de desenvolvimento de aplicações em nuvem. Esses diferentes ambientes seguem diferentes tipos de modelos de provisão de serviços e cada um fornece uma gama de serviços que se diferenciam em diversos aspectos tais como preço, facilidade de uso e desempenho.

Dado esse cenário de diversidade de ambientes de nuvem, um dos principais desafios em termos de desenvolvimento de aplicações diz respeito à implementação, implantação e migração de aplicações na nuvem, que são tarefas desafiadoras no sentido da maior complexidade inerente à heterogeneidade desses ambientes (Galán et al. 2009). Tais ambientes não são implementados utilizando padrões comuns; cada um possui sua própria API, ferramentas de desenvolvimento, mecanismos de virtualização, características de governança, tecnologias de implantação e gerenciamento de recursos, etc. Consequentemente, os usuários tendem a optar por implementar suas aplicações em um ambiente específico, tornando-se altamente dependentes de um único provedor de nuvem, problema conhecido como *cloud lock-in* (Armbrust et al. 2009), sendo impossibilitados de aproveitar as melhores características de diferentes ambientes. Além disso, essa dependência implica que, em caso de indisponibilidade de um serviço do ambiente, a aplicação não tem a flexibilidade de recorrer ao serviço equivalente em outro ambiente.

Dado esse cenário diversificado, esse minicurso tem como objetivo mostrar aspectos de implementação e implantação de uma aplicação usando três importantes ambientes de nuvem: AWS, GAE e OpenStack. Esse documento está estruturado da seguinte forma. A Seção 2 introduz os conceitos básicos relacionados com a Computação em Nuvem. A Seção 3 apresenta os serviços e detalhes de uso de três plataformas de computação em nuvem: Amazon Web Services (AWS), Google App Engine (GAE) e OpenStack. Por fim, a Seção 4 contém as conclusões.

[1] Amazon Web Services (AWS) – http://aws.amazon.com

[2] Google App Engine (GAE) – http://code.google.com/appengine/

[3] Microsoft Windows Azure – http://www.windowsazure.com/

[4] OpenStack Cloud Software – http://openstack.org/

2. Conceitos Básicos

Computação em Nuvem é "*um tipo de sistema paralelo e distribuído que consiste de uma coleção de computadores conectados e virtualizados que são dinamicamente provisionados e apresentados como um ou mais recursos computacionais*" (Buyya *et al.*, 2008). Porém, o termo *Computação em Nuvem* refere-se não somente à estrutura de *hardware* e de *software* dos *datacenters*, mas também aos serviços providos por estes (Armbrust *et al.*, 2009; Zhang *et al.*, 2010). Do ponto de vista dos serviços oferecidos, a Computação em Nuvem pode ser conceituada como um conjunto de serviços disponibilizados via rede, provendo escalabilidade, personalização, garantias de QoS, infraestruturas sob demanda e de baixo custo, os quais podem ser acessados de um modo simples e pervasivo (Wang *et al.*, 2010).

Um conceito de computação em nuvem bastante aceito pela comunidade é o do *National Institute of Standards and Technology* (NIST), apresentado por P. Mell and T. Grance (2011), que afirma que a computação em nuvem é um "*modelo computacional para acesso conveniente, sob demanda e de qualquer localização, a uma rede compartilhada de recursos computacionais (redes, servidores, armazenamento, aplicativos e serviços) que possam ser prontamente disponibilizados como serviços e acessados com um esforço mínimo de gestão ou de interação com o provedor de serviços".*

Para atender a todas essas características, tipicamente, plataformas de Computação em Nuvem disponibilizam interfaces que possibilitam que os usuários, utilizando qualquer dispositivo conectado à Internet, possam ter acesso aos serviços, arquivos, informações e programas situados "na nuvem". Ou seja, plataformas de Computação em Nuvem são provedores de serviços as quais viabilizam a ideia da *computação utilitária*, que significa computação ao alcance de todos, onde usuários pagam pelos serviços computacionais que utilizam.

As subseções a seguir apresentam conceitos e fundamentos inerentes ao contexto de Computação em Nuvem.

2.1 Nuvens públicas, privadas, híbridas e comunitárias

O termo *nuvem* tem sido usado historicamente como metáfora para a Internet, que, em alguns diagramas, era comumente representada como a figura de uma nuvem (Rittinghouse e Ransome, 2010). Na visão de Vaquero *et al.* (2009) e Cearley *et al.* (2009), as *nuvens* seriam grandes repositórios de recursos (*hardware*, *software*, *plataformas*, *serviços* etc.) virtualizados e de fácil acesso, recursos esses que podem ser configurados e reconfigurados dinamicamente de maneira a se ajustar a cargas variadas, otimizando a utilização desses mesmos recursos. As nuvens podem ser organizadas basicamente em quatro modelos em termos de localização física e distribuição, a saber, *nuvem pública*, *nuvem privada* e, *nuvem híbrida*, discutidos a seguir e ilustrados na Figura 1, adaptada de Sosinsky (2011).

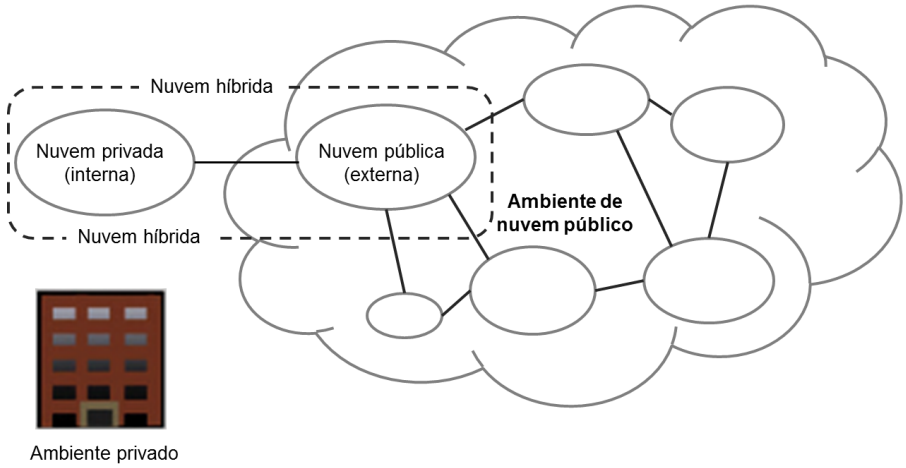

Figura 1. Modelos de organização em nuvens.

Nuvem pública. Em termos simples, serviços de nuvem pública são caracterizados pela sua disponibilidade para clientes por um provedor de serviços através da Internet. O fato de uma nuvem ser pública não significa que os dados do usuário estejam publicamente visíveis; os fornecedores de serviços tipicamente proveem mecanismos de controle de acesso aos seus usuários. No caso das nuvens públicas, estas são tipicamente exploradas utilizando-se um modelo do tipo pagamento por uso, onde os fornecedores de infraestrutura oferecem garantias no formato de SLAs (*Service-Level Agreements*) (Keller e Ludwig, 2003; Buyya *et al.*, 2008; Bose *et al.*, 2011) personalizados, que precisam ser claros, concisos e possuir suporte baseado em penalidades caso não sejam cumpridos. Nesse acordo entre provedores de serviços e clientes, devido à natureza dinâmica dos ambientes de Computação em Nuvem, atributos de QoS, como disponibilidade, tempo de resposta, etc., precisam ser continuamente monitorados, além de outros vários fatores (*e.g.* confiabilidade) também precisarem ser considerados (Patel *et al.*, 2009).

Nuvem privada. Geralmente o termo é utilizado para referir-se aos centros de dados (*datacenters*) internos de uma empresa ou outro tipo de organização que não estão disponíveis publicamente (Armbrust *et al.*, 2009), oferecendo muitos dos benefícios de um ambiente de Computação em Nuvem pública, como a elasticidade e o modelo baseado em serviços. A diferença entre uma nuvem pública e uma nuvem privada é que em um ambiente de serviços baseados em nuvem privada, dados e processos são gerenciados internamente a uma organização sem restrições de banda de rede, exposição à segurança e requisitos legais que o uso de serviços de nuvem pública pode requerer. Além disso, serviços de nuvem privada oferecem ao provedor e ao usuário um maior controle da infraestrutura de nuvem.

Nuvem híbrida. Uma nuvem híbrida toma forma quando uma nuvem privada é suplementada com características presentes em nuvens públicas (*e.g.* a disponibilidade de serviços via Internet de maneira pública), prevendo assim uma utilização mista e

integrada desses dois modelos de nuvem. Em nuvens híbridas, por exemplo, informações de negócio que sejam não críticas e operações de processamento podem ser mantidas em nível de nuvem pública, enquanto serviços e dados de negócio que sejam considerados críticos podem ser mantidos sob o controle dos usuários no escopo de nuvem privada. Além disso, como o paradigma de Computação em Nuvem emprega um modelo de computação sob demanda, serviços de nuvens públicas podem ser utilizados como alternativa a sistemas locais de contexto privado quando a demanda computacional superar a disponibilidade de recursos disponíveis localmente na nuvem privada da organização.

Além desses três modelos supracitados, existem ainda as *nuvens comunitárias*. Uma *nuvem comunitária* é compartilhada por várias organizações que partilham interesses como a missão, requisitos de segurança, políticas, entre outros e decidem partilhar parte das suas infraestruturas e/ou serviços. Nuvens comunitárias podem ser administradas pelas próprias organizações ou por um terceiro. Um exemplo interessante de nuvem comunitária seria uma nuvem do governo federal que possa suprir necessidades de vários ministérios, poupando esses órgãos de cada um ter que gerenciar e manter a sua própria nuvem.

2.2 Modelo arquitetural da Computação em Nuvem

Na Computação em Nuvem, tudo é um serviço, representado em inglês como *Everything as a Service – X*aaS, onde *X* pode ser *software*, plataformas, infraestruturas, etc. (Rimal *et al.*, 2009). Conceitualmente, as plataformas de Computação em Nuvem proveem suas funcionalidades como serviços, categorizadas como IaaS – *Infrastructure as a Service* (*estrutura* como um serviço), PaaS – *Platform as a Service* (plataforma como um serviço) e SaaS – *Software as a Service* (*software* como um serviço). Assim, serviços baseados em uso de *infraestrutura*, *plataforma* e *serviço*, respectivamente, são oferecidos aos usuários, como mostra a Figura 2. Cada uma dessas categorias de serviços é detalhada a seguir.

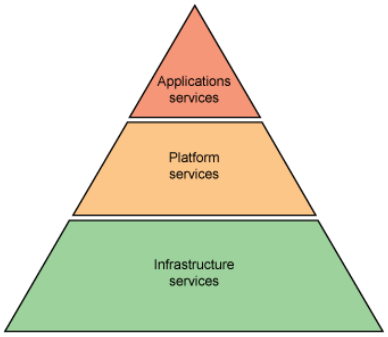

Figura 2. Camadas do modelo arquitetural da computação em nuvem (Fonte: www.ibm.com)

Infrastructure as a Service (IaaS). Essa camada inclui serviços como servidores, sistemas de armazenamento, máquinas virtuais e outros sistemas que são agrupados e padronizados a fim de serem disponibilizados pela rede. É válido ressaltar que são os prestadores de infraestrutura que, através da virtualização, oferecem esses serviços por demanda aos prestadores de serviços (VAQUERO et al. 2009). Nesse modelo a nuvem disponibiliza ao consumidor recursos computacionais e privilégios administrativos sobre eles. Dessa forma, o usuário detém o controle dos recursos, podendo controlar o ambiente, incluindo instalação de software e aplicações. Amazon Elastic Compute Cloud (Amazon EC2 - http://aws.amazon.com/ec2/), Eucalyptus e OpenStack são bons exemplos de desse modelo.

Platform as a Service (PaaS). Camada que encapsula uma camada de *software* e disponibiliza serviços como linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo provedor. Estes serviços, por sua vez, servem de plataforma para que serviços de mais alto nível possam ser desenvolvidos de modo que os consumidores utilizam o ambiente da nuvem como host para suas aplicações, onde a nuvem é apresentada como arcabouço de desenvolvimento para as aplicações. O PaaS é oferecido na camada de plataforma por prestadores de serviços, e os seus usuários também são prestadores de serviços (nesse caso, desenvolvedores). Por exemplo, desenvolvedores podem construir essa plataforma considerando a integração de um sistema operacional, de ambientes de desenvolvimento ou ainda de *softwares* de aplicação. Esse modo, os desenvolvedores que usarão serviços PaaS veem os serviços dessa camada como uma Interface de Programação de Aplicativos (*Application Programming Interface – API*). Eles irão interagir com a plataforma através da API sem ter a preocupação de instalar software localmente, gerenciar e escalar os recursos, o que torna o processo de desenvolvimento de aplicações mais rápido e simples. Entretanto, os usuários de serviços desta camada não detém controle da infraestrutura sobre a qual os serviços PaaS estão sendo oferecidos. Isso porque a camada de infraestrutura permanece transparente aos prestadores de serviços que utilizam o PaaS. Além disso, as aplicações desenvolvidas utilizando serviços PaaS são dependentes das plataformas envolvidas. Google App Engine (GAE) e Amazon Web Services (AWS) são exemplos de PaaS.

Software as a Service (SaaS). Camada que representa os serviços de mais alto nível de abstração disponibilizados em uma nuvem. Esses serviços dizem respeito a aplicações completas que são oferecidas aos usuários, em vez de oferecer recursos computacionais como nas camadas supracitadas. Um SaaS é disponibilizado por prestadores de serviços na camada de aplicação. Ele roda inteiramente na nuvem e pode ser considerada uma alternativa a rodar um programa em uma máquina local. No modelo de software como serviço, o consumidor utiliza uma aplicação, mas não tem gerência nem o controle dos hosts onde a aplicação é disponibilizada, isso é, esses clientes não precisam instalar firmwares, sistemas operacionais, etc. Dropbox, Google Calendar, Gmail e Salesforce.com são exemplos de SaaS.

2.3 Características da Computação em Nuvem

As subseções a seguir apresentam as principais características inerentes à Computação em Nuvem, elencadas nos trabalhos de Breitman (2010), Zhang *et al.* (2010) e pelo NIST. Algumas dessas características advêm de pesquisas anteriores em virtualização, computação distribuída, computação utilitária e, mais recentemente, serviços Web (Vouk, 2008); entretanto, um ponto importante que diferencia a Computação em Nuvem desses modelos anteriores de computação é justamente a sua natureza de direcionamento a serviços (Hu e Klein, 2009; Jinnan e Sheng, 2010).

2.3.1 Virtualização de recursos

A virtualização pode ser definida como a abstração de recursos lógicos dos seus recursos físicos subjacentes a fim de prover agilidade, flexibilidade, redução de custos e aumento do valor de negócio (Rimal *et al.*, 2009), mascarando a natureza física e as fronteiras desses recursos de seus usuários. Essa virtualização é conseguida a partir do uso de tecnologias já estabelecidas, como máquinas virtuais, virtualização de memória, de armazenamento e de rede, desatrelando os serviços de infraestrutura dos recursos físicos (*hardware*, rede). Essa abstração permite maior flexibilidade no modo com que os recursos são combinados e disponibilizados.

Em uma arquitetura baseada em serviços, características do consumidor do serviço são abstraídas do provedor do mesmo através de interfaces de serviços bem definidas, interfaces essas que ocultam os detalhes de implementação e possibilitam trocas automatizadas entre provedores e consumidores de serviços. Nesse modelo, serviços ganham um nível a mais de abstração, ou seja, passam a ser desenhados, para servir a necessidades específicas dos consumidores, através de se ater a detalhes de como a tecnologia funciona. Entretanto, sempre devem ser consideradas falhas e paradas não programadas, de maneira que são necessários mecanismos de tolerância/adaptação a falhas.

2.3.2 Independência de localização dos recursos

Na Computação em Nuvem, os usuários, utilizando qualquer dispositivo conectado à Internet, podem ter acesso aos serviços, arquivos, informações e programas na nuvem. Assim, a nuvem aparenta ser o único ponto de acesso para todas as necessidades de computação dos usuários.

Com o processamento em ambientes de Computação em Nuvem, uma gama de aplicações que fazem uso intensivo de recursos de infraestrutura (*e.g.*, processamento, armazenamento) passa a contar com a possibilidade de acesso ubíquo, através de uma grande variedade de dispositivos (*desktops*, *laptops*, *smartphones*, PDAs etc.), independentemente do dispositivo e da localização. Como grande parte do

processamento é realizada na nuvem, esses dispositivos podem ser simples e desprovidos de grandes recursos computacionais, aproximando-se de simples terminais.

2.3.3 Elasticidade

Talvez esta seja a característica mais inovadora do modelo de Computação em Nuvem, visto que ela pode propiciar benefícios ausentes nas tecnologias atuais. A *elasticidade* – que por vezes é confundida com escalabilidade[5] – seria a capacidade de adequação a variações de demanda, *i.e.*, a capacidade de expansão e retração voluntária e controlada como resposta a um estímulo, de maneira que grandes quantidades de recursos podem ser providos e desprovidos em tempo de execução, dinamicamente, de acordo com a demanda. Por exemplo, uma loja *on-line* necessita, em dias normais, de *x* unidades de recursos, porém há certos períodos do ano nos quais há um aumento considerável das vendas (*e.g.* nas semanas que antecedem as festas de fim de ano), fazendo com que essa loja virtual precise triplicar a quantidade de recursos por apenas alguns dias. Com a elasticidade dos recursos, pode ser provida uma maior quantidade de recursos em caso de aumento no uso dos mesmos e, quando não estiverem mais sendo utilizados, serem desprovidos.

Para o usuário, a nuvem aparenta ser infinita e ele pode adquirir tanto poder computacional quanto ele precise, além de não incorrer em custos desnecessários com recursos ociosos ou subutilizados. A elasticidade na Computação em Nuvem não é medida em termos de número de servidores, por exemplo, mas sim em termos da facilidade com a qual os serviços são fornecidos de maneira a atender às demandas dos consumidores (Cearley, 2009).

2.3.4 Serviços sob demanda

Um consumidor pode unilateralmente dispor de capacidades de computação, tais como tempo de servidor, armazenamento de dados e utilização da rede, conforme necessário, automaticamente, sem a necessidade de interação humana com os prestadores de serviço. Ele deve ser capaz de provisionar os recursos automaticamente e de acordo com sua necessidade.

2.3.5 Amplo acesso a rede

Recursos são disponibilizados através da rede e acessados por meio de mecanismos-padrão que promovam o uso por diferentes dispositivos com qualquer capacidade de processamento (por exemplo, *PC*, *tablet*, *smartphone*, etc).

[5] A escalabilidade é a habilidade de satisfazer um requisito de aumento da capacidade de trabalho pela adição proporcional da quantidade de recursos. Uma arquitetura dita escalável é construída tipicamente com base em uma infraestrutura básica passível de repetição e/ou modificação cujo crescimento pode ser alcançado simplesmente com a adição repetida do mesmo conjunto básico. Dessa forma, usualmente, não há a preocupação com a remoção de recursos nem se os recursos estão sendo plenamente utilizados, visto que os recursos já adquiridos são custo consolidado.

2.3.6 Agrupamento de recursos

Os recursos de computação do provedor são agrupados para atender múltiplos consumidores através de um modelo multi-inquilino, com diferentes recursos físicos e virtuais atribuídos dinamicamente de acordo com a demanda do consumidor. Há um senso de independência de localização em que o cliente geralmente não tem controle ou conhecimento sobre a localização exata dos recursos disponibilizados, mas pode ser capaz de especificar um local em um nível maior de abstração (por exemplo, estado, país, ou *datacenter* específico).

2.3.7 Serviço mensurado

Sistemas em nuvem são capazes de automaticamente controlar e otimizar o uso dos recursos, aproveitando uma capacidade de medição em algum nível de abstração apropriado de acordo com o tipo de serviço (por exemplo, contas de armazenamento, processamento e largura de banda). A utilização dos recursos pode ser monitorada e controlada de modo transparente tanto para o provedor quanto para o consumidor do serviço utilizado.

2.3.8 Modelo de pagamento baseado no consumo

Dentro do paradigma de Computação em Nuvem, os consumidores de serviços e recursos computacionais necessitam pagar aos provedores apenas quando e pelo que utilizarem de tais serviços. A grande vantagem do modelo é permitir a contratação de novos recursos na medida em que estes se tornem necessários e a liberação (finalização dos contratos) dos mesmos quando desnecessários, fazendo com que não seja preciso se fazer grandes investimentos em infraestrutura e manutenção e um planejamento para provisão de recursos em longo prazo, havendo assim uma grande redução em custos de investimento que são convertidos em custo de operação.

2.3.9 Vantagens do paradigma de Computação em Nuvem

Mediante as características enumeradas nas subseções anteriores, podem-se elencar algumas vantagens, dentre muitas outras, da utilização da Computação em Nuvem:

- Considerável redução de custos, dado que não há a necessidade de investimento em TI tanto em termos de *hardware* quanto de *software*; mesmo se necessárias altas capacidades de processamento e armazenamento podem ser adquiridas da nuvem, além do fato de que *softwares* podem ser utilizados sem estarem instalados no próprio dispositivo computacional. Em adição, têm-se as reduções de custos com manutenção, pessoal, espaço físico e energia.
- Independência de sistema operacional e *hardware*.
- Facilidade de gerenciamento, de acesso aos dados e às aplicações, uma vez que estes são centralizados na nuvem.

- Facilidade de compartilhamento de dados e trabalhos colaborativos.
- Maior controle de custos, uma vez que os serviços são pagos por utilização e não pela licença dos mesmos, o que acaba por também facilitar os contratos.
- Elasticidade, visto que os recursos de TI são virtualmente infinitos.
- Possibilidade de os usuários concentrarem-se na informação e na lógica de negócio, e não na infraestrutura computacional, devido à abstração que é realizada.
- A Computação em Nuvem possibilita uma grande democratização, visto que as barreiras financeiras para entrada no mercado são menores. Assim, praticamente qualquer um com uma ideia, conhecimentos técnicos e uma conexão à Internet pode usar a Computação em Nuvem para montar seu próprio negócio e competir com grandes empresas, visto que não são necessários investimentos em infraestrutura, proveem-se ferramentas para desenvolvimento, tem-se facilidade para publicação e distribuição e as barreiras geográficas são inexistentes.

Do ponto de vista de negócio, Breitman (2010) aponta dois direcionadores principais para adoção de Computação em Nuvem, válido para a adoção de novas tecnologias de um modo geral: (i) redução de custos e (ii) aumento de capacidade. No modelo de Computação em Nuvem, a redução de custos tem um caráter evolucionário baseado principalmente no pagamento por uso, nos casos de utilização de um provedor de nuvem pública, e na virtualização dos recursos, no caso de uso da adoção de ambiente de nuvem privada. Já o aumento da capacidade proporcionado por esse tipo de ambiente, principalmente por suas características de elasticidade e acesso aos recursos via Internet, tem um caráter mais revolucionário. O uso de ambientes de Computação em Nuvem viabiliza o surgimento de novos serviços ou aplicações que se beneficiem dessas características (elasticidade no provimento de recursos), independência de localização e dos dispositivos utilizados para acesso, que não ficam limitados a computadores apenas.

3. Plataformas de computação em nuvem

Essa seção apresenta três plataformas de computação em nuvem. Duas delas são plataformas proprietárias (AWS e GAE), e a terceira plataforma permite a instalação de nuvens próprias do cliente, ou seja, nuvem privada (OpenStack).

3.1 Amazon Web Services

O *Amazon Web Services* (AWS) (http://aws.amazon.com/), serviços de nuvem providos pela Amazon, são utilizados largamente por empresas de vários tamanhos e domínios e oferecem poder computacional, facilidades de armazenamento e várias outras funcionalidades que permitem que empresas implantem aplicações e serviços a um baixo custo, com grande flexibilidade, escalabilidade e confiabilidade.

Figura 3. Página principal do Amazon Web Services (AWS) – http://aws.amazon.com/pt

3.1.1 Principais serviços

Essa subseção apresenta os principais serviços da plataforma Amazon Web Services (AWS), sendo eles: (i) Amazon Elastic Compute Cloud (EC2), Amazon Simple Storage Service (S3), Amazon Relational Database Service (RDS), Amazon SimpleDB.

3.1.1.1 Amazon Elastic Compute Cloud (EC2)

Um dos principais serviços oferecidos no portfolio AWS é o *Amazon EC2* (http://aws.amazon.com/ec2/), um serviço que oferece capacidade de computação redimensionável (i.e. elástica) na nuvem. Esse serviço apresenta-se como um verdadeiro ambiente de computação virtual, permitindo aos usuários, através de uma interface Web simples, criar, usar e gerenciar máquinas virtuais com sistemas operacionais Windows e Linux, ou mesmo iniciar tais máquinas de acordo com as necessidades das aplicações. Como acontece na Computação em Nuvem, o usuário paga apenas pelos recursos consumidos, por instância-horas e/ou transferência de dados (cobrado por *gigabyte* de dados transferidos).

No Amazon EC2, tem-se AMIs (*Amazon Machine Images*), que funcionam como uma espécie de *template* e contêm uma pré-configuração de *software* (*e.g.* sistema operacional e aplicações), a partir das quais se podem criar *instâncias* (máquinas virtuais), que são cópias executáveis da AMI, como ilustra a Figura 4. Essas instâncias, que podem ser múltiplas e inclusive de diferentes tipos, são executadas até que sejam paradas ou finalizadas pelo usuário; se uma instância porventura falhar, pode-se criar uma nova a partir da AMI selecionada.

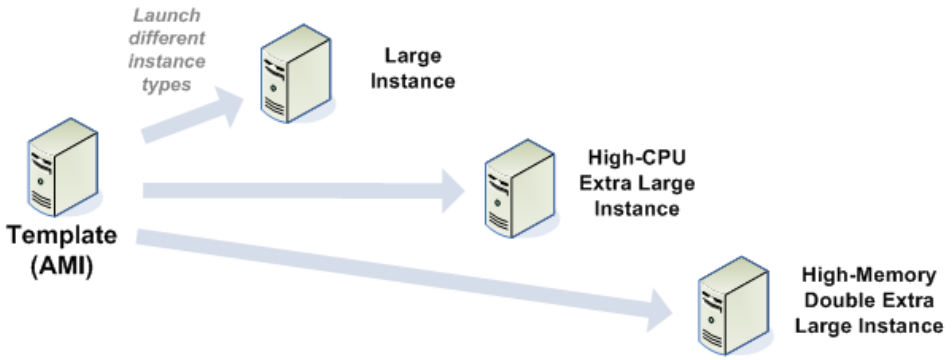

Figura 4. Criação de diferentes instâncias a partir de uma AMI no Amazon EC2.

O SLA (*Service Level Agreement*)[6] do Amazon EC2, disponível em http://aws.amazon.com/ec2-sla/, estabelece que a disponibilidade do serviço é de 99,95% por ano, sendo provido ao cliente um crédito de 10% do valor de sua conta caso essa disponibilidade não seja verificada.

3.1.1.2 Amazon Simple Storage Service (S3)

Amazon S3 (http://aws.amazon.com/s3/) é um serviço que provê uma infraestrutura de armazenamento para lidar com grandes quantidades de dados. Esse serviço provê uma interface Web simples que pode ser utilizada para armazenar e recuperar qualquer quantidade de dados a partir de qualquer lugar da Web. No Amazon S3, cada *objeto* (i.e. dados e respectivos metadados), cujo tamanho pode ir de 1B a 5TB, é armazenado em um *bucket*, que é um *container* para objetos armazenados no Amazon S3 o qual pode ser recuperado de maneira unívoca através de uma *chave* de acesso.

O SLA do Amazon EC2, disponível em http://aws.amazon.com/s3-sla/, estabelece que a disponibilidade do serviço (*uptime*) é de 99,9% por mês, sendo provido ao cliente um crédito de 10% do valor de sua conta caso essa disponibilidade seja maior ou igual a 99% porém inferior aos 99,9% estabelecidos por mês, ou de 25% caso seja inferior a 99%.

3.1.1.3 Amazon Relational Database Service (RDS)

Amazon RDS (http://aws.amazon.com/rds/) é um serviço PaaS que implementa um banco de dados relacional em nuvem, além de possibilitar a configuração e operação do banco. O Amazon RDS é projetado para desenvolvedores ou empresas que necessitam de todos os recursos e capacidades de um banco de dados relacional ou que desejam migrar aplicações existentes e ferramentas que utilizam um banco de dados desse tipo para a nuvem. Assim, como o Amazon RDS provê acesso às capacidades de um banco

[6] SLA (*Service Level Agreement*) é uma espécie de contrato firmado entre o provedor de um serviço e o cliente do mesmo, sendo descrito o serviço, suas metas, papeis e garantias de qualidade de serviço (Verma, 1999).

de dados MySQL ou Oracle, códigos e aplicações, bem como ferramentas utilizadas para esses tipos de bases de dados, podem ser utilizados de maneira simples no Amazon RDS.

3.1.1.4 Amazon SimpleDB

Amazon SimpleDB (http://aws.amazon.com/simpledb/) é um serviço que, como o próprio nome diz, implementa um banco de dados simples, no sentido em que oferece um número de funcionalidades menor se comparado a bancos de dados relacionais, mas que, entretanto, pode ser considerado suficiente para várias aplicações que empregam dados e não necessitam manipular relacionamentos entre tais dados, como em aplicações de *log*.

No *Amazon SimpleDB* utiliza-se o conceito de *domínios*, que corresponde ao conceito de tabelas existente em bancos de dados relacionais. Esse serviço provê uma API simples para se fazer operações de acesso e armazenamento em bancos de dados, além de indexar dados automaticamente, facilitando assim as tarefas relacionadas ao gerenciamento de dados. Entretanto, como esse serviço é bem limitado em termos de funções, para aplicações que dependem de desempenho e de sistemas com bases de dados relacionais comerciais (além de tipicamente um administrador de banco de dados), o Amazon RDS seria a melhor escolha para a implantação desse tipo de banco de dados.

3.1.1.5 Amazon Elastic Beanstalk

O *Amazon Elastic Beanstalk* (http://aws.amazon.com/elasticbeanstalk/) é o serviço que provê implantação e gerenciamento de aplicações na nuvem de forma direta, isto é, sem a necessidade de outros serviços de infraestrutura já configurados. Ele aloca e fornece recursos computacionais, balanceamento de carga e monitoramento de status da aplicação. Atualmente, tem suporte a *Java*, Python *PHP* e *.NET*, utilizando os servidores *Tomcat*, *Apache* e *Microsoft IIS 7.5*. Para implantar-se uma aplicação através do *Elastic Beanstalk* basta efetuar o upload dela em um arquivo empacotado (por exemplo, arquivo *War*) ou indicar seu repositório *Git*. O armazenamento da aplicação empacotada e dos arquivos de log é realizado pelo Amazon S3.

3.1.2 Ferramentas de apoio ao desenvolvimento

O *AWS Toolkit for Eclipse* (http://aws.amazon.com/eclipse/) é um *plug-in open-source* para o IDE Eclipse[7]que tem por objetivo tornar fácil o desenvolvimento, implantação (*deployment*) e depuração de aplicações desenvolvidas na linguagem de programação Java utilizando os serviços AWS. Esse *toolkit* inclui o chamado AWS Explorer, que permite ao usuário interagir com os serviços AWS através do IDE, e também possui

[7] Eclipse – http://www.eclipse.org/

suporte para o AWS Elastic Beanstalk, para que o usuário possa implantar a aplicação desenvolvida na nuvem da Amazon.

O *AWS Toolkit for Visual Studio* (http://aws.amazon.com/visualstudio/), similar ao Eclipse *plug-in*, é uma extensão para o IDE Microsoft Visual Studio, que tem por objetivo tornar fácil o desenvolvimento, depuração e implantação de aplicações utilizando a plataforma .NET e os serviços AWS. Esse *toolkit* também inclui o AWS Explorer e possui suporte para o AWS CloudFormation, permitindo a implantação de aplicações .NET na nuvem da Amazon.

3.1.3 Implantando uma Aplicação na Amazon

Para demonstrar a utilização de diversos serviços da Amazon realizaremos o processo de implantação do gerenciador de conteúdos *Joomla*[8], utilizando duas abordagens: (i) realizando o processo de configuração de maneira manual e (ii) utilizando o serviço *Amazon Elastic BeanStalk.* Para utilizar os serviços da amazona faz-se necessário realizar o cadastramento na plataforma, seguindo os passos a serem descritos na próxima seção.

3.1.3.1 Criando uma Conta no AWS

Para criar uma conta para utilização dos serviços da Amazon faz-se necessário possuir um cartão de crédito internacional. O sistema de faturamento da Amazon é baseado no pague o quanto consumir, porém para segurança da própria empresa, o usuário deverá fornecer o número do cartão de crédito para fins de faturamento. Para iniciar o cadastramento o usuário deverá a acessar http://aws.amazon.com/pt/, selecionar a opção cadastrar-se e informar dados como e-mail, nome, telefone e endereço, além do cartão de crédito internacional. Ao final do processo de cadastramento o usuário irá receber uma ligação para que seja informado o código de segurança para finalizar o processo de cadastramento. A ativação da conta pode ocorrer de maneira instantânea ou demorar algumas horas.

Após a ativação da conta o conjunto de serviços oferecidos pela Amazon torna-se disponível para o usuário. Com objetivo de permitir que novos usuários integrem-se a sua base e permitir que os mesmos possam utilizar os serviços sem compromisso, a Amazon estabeleceu a faixa *free* que consiste em cotas para utilização de uma série de serviços da Amazon, sem que sejam realizados cobranças no cartão. Cada serviço define os limites, como também os preços e técnicas de cobrança praticada por cada serviço. Para maiores informações sobre as cotas da faixa gratuita, acesse http://aws.amazon.com/pt/free/.

Ao acessar o sistema de Amazon, o usuário é levado ao painel de controle dos serviços Amazon, onde podemos acessar a configuração de cada serviço disponível. Na figura 5 observamos o painel de controle com link de acesso para cada um dos serviços disponibilizados pela plataforma.

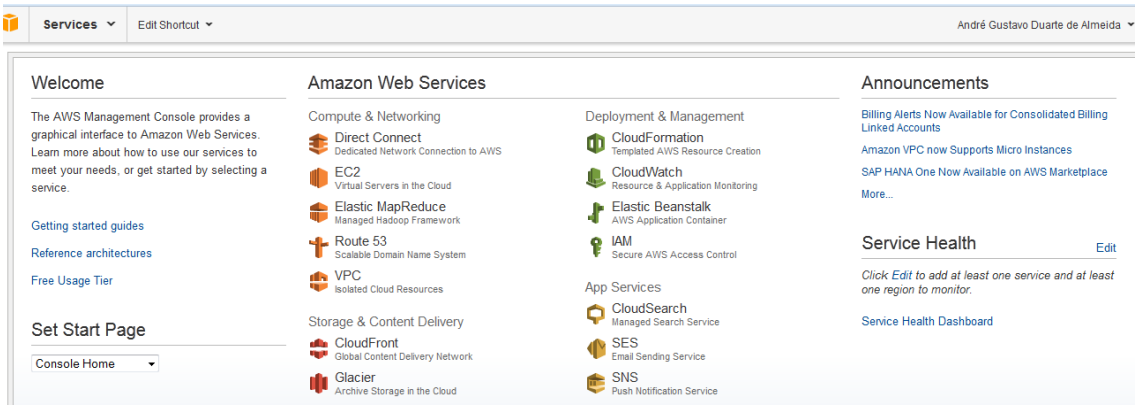

Figura 5. Painel de Controle do AWS

3.1.3.2 Implantando o Joomla – Abordagem Manual

Apesar de fornecer um mecanismo para implantação automática de aplicações, tal facilidade possui custo adicional, além de não permitir ao usuário responsável pelo processo de implantação configurar e gerenciar as máquinas virtuais/banco de dados utilizados pela aplicação. Nesta seção descreveremos o passo a passo necessário para configurar o ambiente para implantação uma aplicação PHP com acesso a banco de dados MySQL, do ponto de vista de um gerente de redes/administrador de sistemas. Na seção 3.1.3.3 descreveremos como o desenvolvedor pode realizar o processo de implantação de maneira mais transparente.

Amazon EC2 é o serviço de computação da Amazon. Como visto na seção 3.1.1 podemos criar uma máquina virtual do zero ou utilizar uma imagem, que consiste de uma máquina com pré-configuração de software estabelecida. Ao criar uma instância do EC2 devemos utilizar uma das opções de configuração de hardware(memória RAM, processador, etc) disponibilizadas pela Amazon. Para criar a instância que irá hospedar nosso gerenciador de conteúdo, consultamos o *Amazon MarketPlace* que é um repositório de imagens, para os mais diversos fins, variando de imagens preparadas para hospedar de aplicações web a servidores de banco de dados e assim por diante. As imagens existem nas mais diferentes configurações, sistemas operacionais e preços. Custos associados à licença de software não estão incluídos dentro do da faixa gratuita, sendo necessário, portanto atenção na escolha da imagem a ser utilizada. Para o nosso estudo de caso iremos utilizar a imagem fornecida pela própria Amazon, que consiste em uma distribuição Linux da própria Amazon, que vem apenas com os softwares básicos de uma distribuição Linux. A outro configuração realizada diz respeito a região de implantação da instância. A região é localização física da instância. Tal configuração irá impactar em questões como preço, disponibilidade, latência dentre outras características associadas ao acesso a instância. Recentemente a Amazon lançou a região São Paulo, que diminui a latência para instâncias no Brasil, porém a preços maiores. Dependendo da aplicação ainda não vale a pena utilizar a região São Paulo.

Antes de criar efetivamente a instância, precisamos criar um par de chaves de acesso. Esse par de chaves permite o acesso remoto a máquina, para que possamos realizar de maneira adequada o gerenciamento. Para criar um par de chaves, devemos acessar o painel de controle da Amazon e selecionar a opção Amazon EC2 → *Key Pairs*. Para criar uma nova chave basta acionar o botão *Create Key Pair*. Devemos então informar o

nome do par de chaves, ao concluir o processo será gerado um arquivo contendo a chave pública. O referido arquivo deve ser salvo em lugar seguro e realizado o backup, pois para efeitos de segurança não é possível recuperar o arquivo, depois de baixado. Uma vez criado o par de chaves podemos passar a criação da instância.

Para criar a instância devemos selecionar a imagem desejada, informar a configuração de hardware escolhida e o par de chaves associado à máquina virtual. Na figura 6 vemos a tela de resumo das configurações escolhidas. Ao concluir as configurações devemos pressionar o botão *Launch with 1-Click* que o processo de criação e registro da nossa máquina virtual terá início.

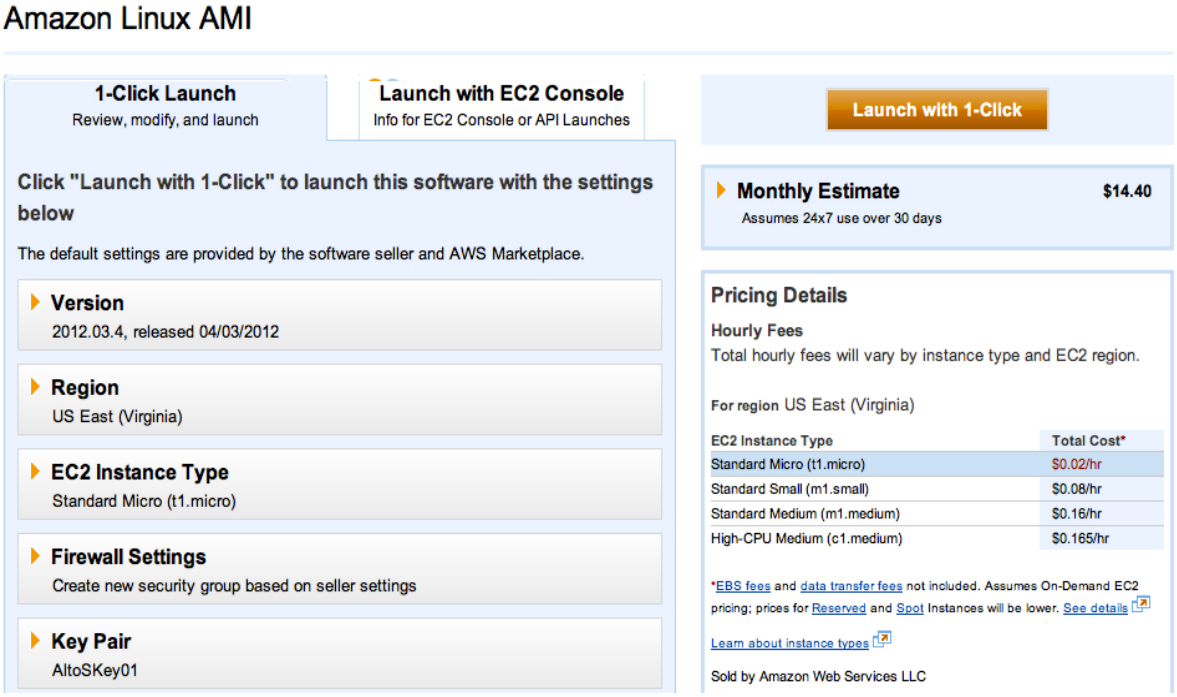

Figura 6. Configuração de Criação de uma Instância

Uma vez criada uma instância, a mesma estará disponível dentro do gerenciamento do EC2, na opção *Instances*. Através dessa página podemos iniciar uma instância, reiniciar, parar, excluir ou mesmo. Além disso, temos acesso ao DNS público da instância, que será usado para acesso à mesma. Por padrão é criado um usuário Unix chamado *ec2-user*, que está associado à chave pública criada, esses usuários variam de acordo com o tipo de imagem (consequentemente Sistema Operacional) selecionado, por isso é importante verificar a documentação da imagem para verificar as informações relacionadas ao acesso.

Uma vez criada à instância precisamos realizar o processo de configuração interna da mesma, ou seja, instalar o servidor web (Apache), interpretador PHP, bem como copiar os arquivos necessários para instalação do Joomla. Caso o usuário possua um sistema operacional em sua máquina baseado em Unix, podemos utilizar ssh para realizar a conexão com a instância. Caso seja uma máquina Windows, podemos fazer uso de ferramentas para conexão ssh, tais como *Putty*. Para fins de demonstração de comandos iremos utilizar o processo através do sistema operacional Linux, usando uma máquina com distribuição Linux, Ubuntu 12.01. O usuário deve abrir o terminal e entrar na pasta

onde o arquivo contendo a chave pública para acesso a instância foi salvo. Chaves públicas devem ter acesso restrito, para fins de segurança. No Linux utilizamos o comando chmod para configurar as permissões. Na figura 7 temos os comandos necessários para conectar e instalar o software necessário para preparar a instância para iniciarmos o processo de instalação do Joomla. Na linha 1 atribuimos a permissão 400 para o arquivo da chave pública, na linha 2 realizamos uma conexão ssh com a instância, usando seu DNS público e a chave de acesso. Nas linhas 3 e 4 realizamos a instalação dos pacotes httpd(Servidor Web) e do interpretador do PHP. Na linha 5 reiniciamos o servidor web.

```
1. chmod 400 <chave>.pem
2. ssh –i <chave>.pem ec2-user@<dnspublico>
3. sudo yum httpd
4. sudo yum php php-mysql php-xml
5. sudo apachectl restart
```

Figura 7. Conectando e instalando serviços em uma instância

Uma vez instalados o serviços devemos baixar a versão mais recente do Joomla e copiar para pasta de documentos html do servidor web, que nessa configuração fica na pasta */var/www/html*. Uma vez copiado o arquivo podemos proceder com a instalação do Joomla, porém faz-se necessário configurar o serviço de banco de dados do RDS para armazenar os dados da aplicação.

Como visto anteriormente, o RDS é um serviço de banco de dados relacional, que permite a criação de instâncias com dedicação exclusiva para banco de dados relacionais. Para o Joomla iremos utilizar uma instância do RDS com o banco de dados MySQL. Para iniciar o processo de configuração, devemos acessar a opção RDS no painel de controle. Acessado o painel de controle, antes de criar um banco de dados precisamos criar um grupo de segurança, que irá definir as políticas de acesso ao banco de dados. As permissões são feitas de duas formas: (i)através de faixas de IP, onde as máquinas com IP dentro da faixa especificada, tem acesso liberado ao banco de dados e (ii) associando um grupo de segurança do EC2, para que as máquinas(instâncias) do EC2 tenham acesso ao banco de dados. Como nossa aplicação executa dentro de uma máquina EC2 essa permissão é suficiente. Caso seja necessário realizar alguma operação de administração do banco de dados, deveremos utilizar a permissão de faixa de IP, para permitir que ferramentas de gerenciamento de banco de dados se conectem ao nosso servidor.

Estabelecido o grupo de segurança, iremos agora criar uma instância com banco de dados MySQL. Dentro do gerenciamento do RDS, selecione *DB Instances-> Launch DB Instance*, será então aberto um assistente para configuração do banco de dados. Serão requisitas as informações a respeito do nome do banco de dados; tamanho inicial alocado, porta de acesso, grupo de segurança e usuário e senha de acesso ao banco de dados. Na figura 8 temos um fragmento das telas do assistente de criação do banco de dados. Ao final do processo a instância será criada e a mesma possuirá um endereço de DNS público, que devemos utilizar no processo de instalação do Joomla.

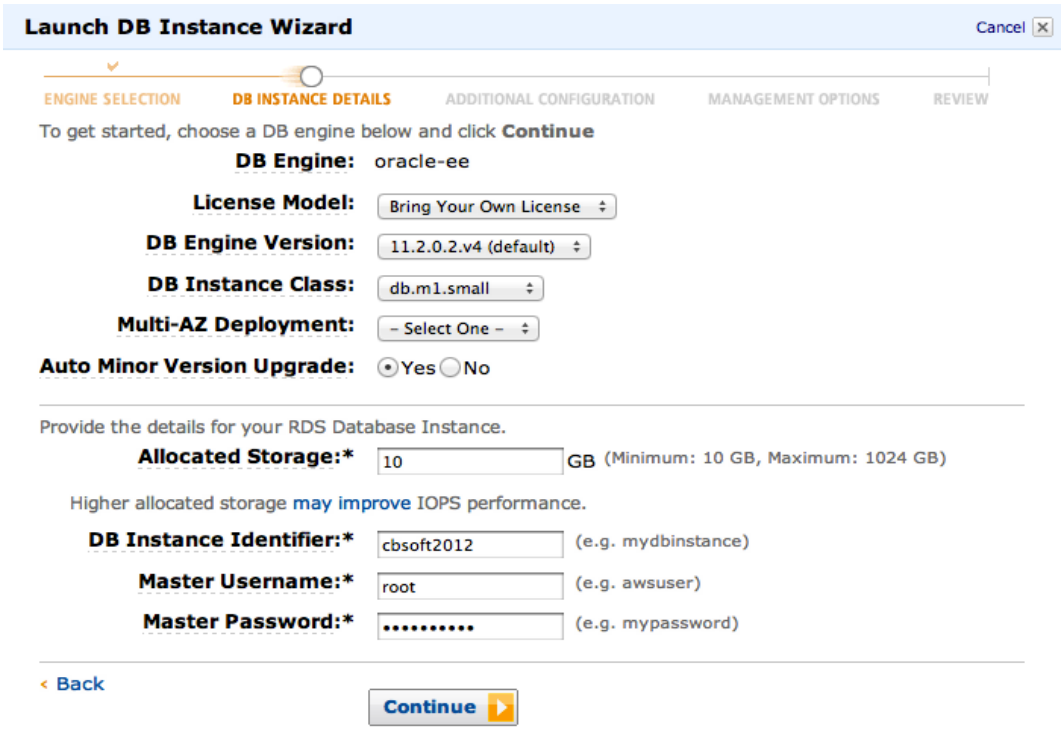

Figura 8. Configurando acesso ao Banco de Dados

Para concluir a instalação do Joomla, devemos acessar a máquina através do endereço <dnspublico>/joomla, e no momento de configurar o banco de dados, informar o dns do banco de dados, usuário e senha para que seja criada a base de dados inicial do Joomla.

3.1.3.3 Implantando o Joomla – Abordagem Automática – BeanStalk

O Amazon Elastic Beanstalk (http://aws.amazon.com/elasticbeanstalk/) é o serviço que provê implantação e gerenciamento de aplicações na nuvem de forma direta, isto é, sem a necessidade de outros serviços de infraestrutura já configurados. Ele aloca e fornece recursos computacionais, balanceamento de carga e monitoramento de status da aplicação. Atualmente, tem suporte a *Java*, *PHP* e *.NET*, utilizando os servidores *Tomcat*, *Apache* e *Microsoft IIS 7.5*. Para implantar-se uma aplicação através do Elastic Beanstalk basta efetuar o upload dela em um arquivo empacotado (por exemplo, arquivo *War*) ou indicar seu repositório *Git*. O armazenamento da aplicação empacotada e dos arquivos de log é realizado pelo Amazon S3.

Uma vez que o desenvolvedor possua o arquivo empacotado, basta acessar https://console.aws.amazon.com/elasticbeanstalk/ a fim de efetuar a upload de sua aplicação. Dando inicio ao upload da aplicação é exibido um assistente como nas figuras 9 e 10.

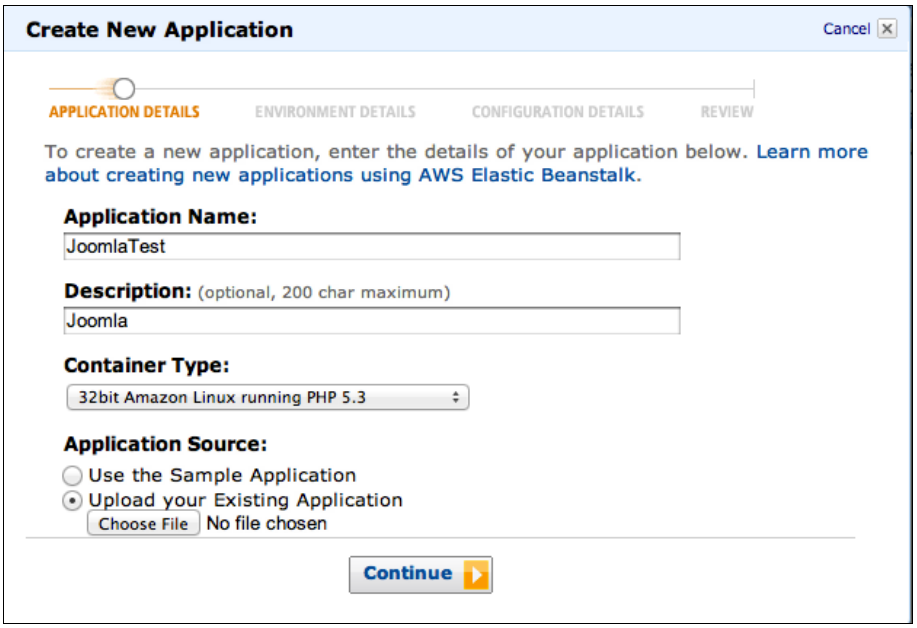

Figura 9. Upload da aplicação

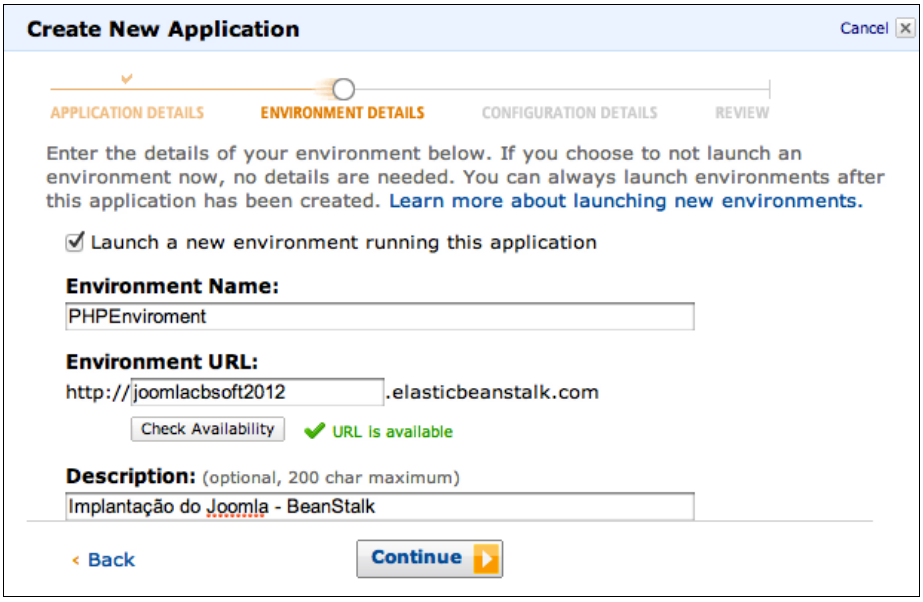

Figura 10. Configuração do Ambiente

Como visto na figura 9, o desenvolvedor informa o "Applicaton Name" e a "*Description*", bem como seleciona o "*Container Type*" e opção de "Upload your Existing Application". Tomando-se, por exemplo, uma aplicação em PHP é possível selecionar o "32bit Amazon Linux running PHP 5.3". O passo seguinte é configurar o ambiente de execução e informar a URL, conforme figura 10. Logo após, se define o tipo de instância que a aplicação irá rodar e a chave de acesso, caso se deseje acessar a instância. Ao fim do assistente, a instância EC2 é carregada, tornando-se possível o acesso à aplicação implantada.

3.1.4 Utilizando o Amazon S3

Amazon S3 (http://aws.amazon.com/s3/) é um serviço que provê uma infraestrutura de armazenamento para lidar com grandes quantidades de dados na internet. Ele fornece uma interface Web simples que pode ser utilizada para armazenar e recuperar qualquer quantidade de dados a partir de qualquer lugar da Web. No Amazon S3 cada *objeto* (i.e. dados e respectivos metadados), cujo tamanho pode ir de 1B a 5TB, é armazenado em um *bucket*.

O bucket é um *container* para os objetos armazenados no Amazon S3. Nele definem-se regras de acesso para upload ou download dos objetos. Existem 03 (três) formas de gerenciamento dos buckets no S3: via interface web (https://console.aws.amazon.com/s3), via cliente desktop S3 (por exemplo: http://s3browser.com/) e via Application Programming Interface (API). Utilizando-se da API o desenvolvedor pode ter acesso a ele, de maneira unívoca, através de uma *chave* de acesso. Como exemplo apresenta-se a criação via web em seguida.

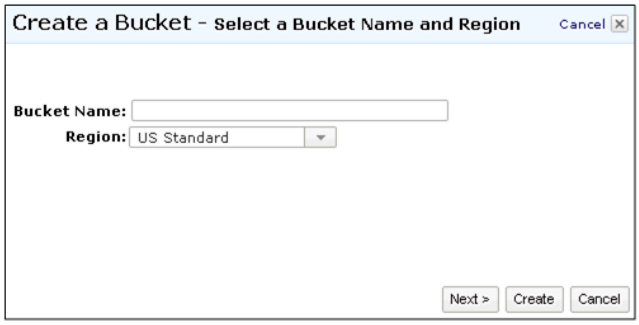

Figura 11. Criando um Bucket

Utilizando a interface Web pode-se criar um bucket clicando em "*Create Bucket*". Logo após, será exibido uma caixa de dialogo como na figura 1. Basta informar o "*Bucket Name*" e clicar em "*Create*". Logo após será exibido um tela como na figura 12.

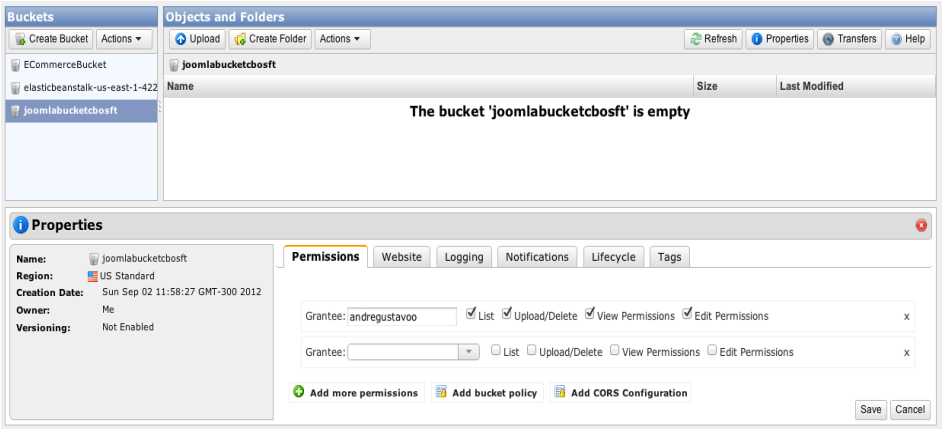

Figura 12. Buckets Painel

Como visto na figura 12, é possível explorar os containers criados na secção "*Buckets*". Na secção "Objects and Folders" gerencia-se diretórios e objetos (arquivos). Na secção "Properties" configura-se permissões, log e outras propriedades do *bucket*. O bucket pode ser utilizado como um site estático, essa funcionalidade é configurada na aba

"Website". Com o *bucket* criado é possível incluir objetos, clicando-se em "Upload". Isso faz com que seja exibida uma tela como na figura 13.

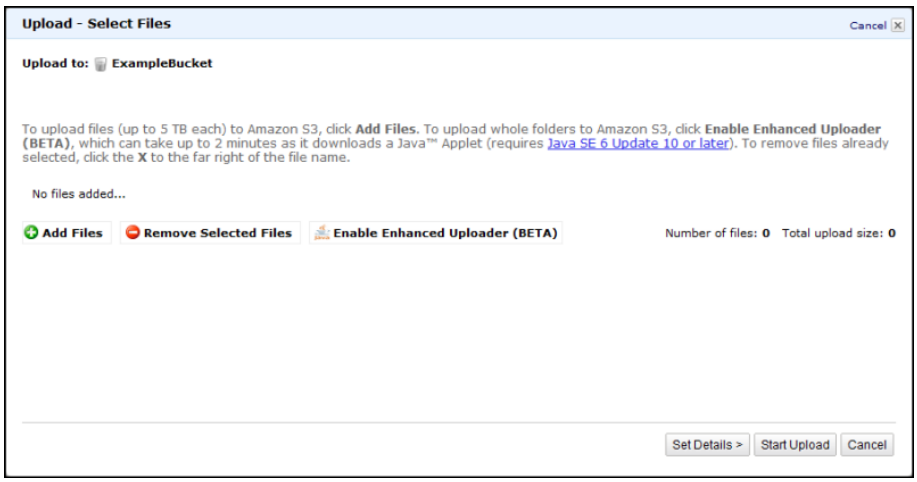

Figura 13. Upload de Objetos

Clicando em "*Add Files*", adicionam-se os arquivos a serem enviados. Uma vez que todos os arquivos estejam selecionados clica-se em "*Start Upload*".

Qualquer operação (armazenamento, solicitações e transferência de dados) sobre o serviço Amazon S3 tem um custo associado. É importante que os clientes desses serviços estejam cientes desses valores, por isso a Amazon disponibiliza em http://aws.amazon.com/pt/s3/pricing/ uma tabela com tais valores.

O SLA do Amazon S3, disponível em http://aws.amazon.com/s3-sla/, estabelece que a disponibilidade do serviço (*uptime*) corresponde a 99,9% por mês. Caso essa disponibilidade seja maior ou igual a 99%, porém inferior aos 99,9% ao mês, o cliente recebe um crédito de 10% do valor de sua conta; se ela for inferior aos 99%, o crédito passa a ser de 25%.

3.2 Google App Engine

O *Google App Engine* (GAE)[9] é classificado como *Platform-as-a-Service* (PaaS), priorizando o suporte a hospedagem de aplicações web. É um serviço fornecido pela Google que permite aos desenvolvedores criar aplicações que podem rodar na sua infra estrutura. A virtualização e elasticidade são praticamente imperceptíveis. Isto ocorre porque todo o gerenciamento de virtualização e a elasticidade são feitos de forma automática, de acordo com o número de requisições recebidas por uma aplicação. O GAE utiliza o *Jetty servlet container* para hospedar aplicações como um mecanismo de interação com o servidor web, oferecendo suporte a API Java Servlet na versão 2.4. Quando o GAE recebe uma solicitação, ele determina qual classe de *servlet* deve ser chamada através de um arquivo de configuração XML (web.xml) conhecido como descritor de implantação

O GAE pode ser descrito por meio de três partes: **Ambiente de execução**, **datastore** e **serviços escalonáveis**.

3.2.1 Ambiente de Execução

O ambiente de execução determina o ciclo de execução de uma aplicação implantada no GAE. Assim, quando o ambiente de execução recebe uma requisição HTTP, o primeiro passo é

identificar a aplicação alvo daquela requisição através do nome de domínio. De posse do nome da aplicação requisitada, o ambiente de execução seleciona um servidor para atender tal requisição com base em dados de uso de CPU, memória, etc. Depois de selecionar um servidor, a aplicação alvo é iniciada no servidor escolhido e recebe a requisição HTTP encaminhada inicialmente. Finalmente, a aplicação trata a requisição, retorna uma resposta para o cliente e é finalizada. Esta sequencia de passos faz com que uma aplicação implantada no GAE tenha seu tempo de existência limitado a duração de um processamento de requisição. Este curto tempo de vida de uma aplicação faz com que não seja possível manter estado entre requisições, o que permite distribuir o trafico de requisições entre vários servidores.

Como várias aplicações podem estar executando em um mesmo servidor, o ambiente de execução utiliza o conceito de *sandbox* para restringir os recursos utilizados por cada aplicação. O sandbox restringe a utilização dos seguintes recursos:

- Gravar no sistema de arquivos. As aplicações devem utilizar o armazenamento de dados do GAE para armazenar dados persistentes.
- Abrir um soquete ou acessar diretamente outro host.
- Gerar um sub-processo ou linha. Uma solicitação da web a uma aplicação deve ser manipulada em um único processo, dentro de no máximo 30 segundos. Os processos que ultrapassarem este tempo são encerrados.
- Fazer outros tipos de chamadas do sistema.

O GAE fornece ambientes de execução para aplicações escritas em Java 6.0 e Python 2.5.

3.2.2 Datastore

Talvez o recurso mais importante que o GAE oferece é seu serviço de armazenamento de dados. Com sua infraestrutura gigantesca – a mesma do mecanismo de busca da Google – seu sistema de banco de dados BigTable e seu Google File Systems, o GAE proporciona um armazenamento dimensionável e robusto, com mecanismos de consulta e transações atômicas. Esse serviço de persistência utiliza uma arquitetura distribuída.

Os dados podem ser distribuídos nessa estrutura de forma otimizada, para isso basta descrever o relacionamento entre os objetos de dados (entidades), como mostrado na Figura 14, e definir índices para as consultas. O GAE para Java inclui suporte a duas grandes interfaces para bancos de dados: JDO (Objetos de dados Java) e JPA (API persistente Java).

```
18 @PersistenceCapable(identityType = IdentityType.APPLICATION)
19 public class MediaObject {
20   @PrimaryKey
21   @Persistent(valueStrategy = IdGeneratorStrategy.IDENTITY)
22   private Key key;
23
24   @Persistent
25   private User owner;
26
27   @Persistent
28   private BlobKey blob;
29
30   @Persistent
31   private Date creation;
32
```

Figura 12. Anotações referentes à persistência de dados utilizando o mecanismo JDO.

3.2.3 Serviços Escalonáveis

A relação do armazenamento de dados com o ambiente de tempo de execução é o de um serviço. A aplicação usa uma API para acessar um sistema separado que gerencia todas as suas necessidades de dimensionamento próprio separadamente do ambiente de execução. O GAE inclui vários outros serviços auto escaláveis úteis para aplicações web.

3.2.3.1 Memcache

O serviço de Memcache é utilizado para acelerar consultas comuns no armazenamento de dados. Por exemplo, se muitas solicitações fizerem a mesma consulta com os mesmos parâmetros e não for necessário exibir imediatamente no site as alterações nos resultados, a aplicação poderá armazenar os resultados em cache no Memcache. As solicitações subsequentes podem consultar o cache de memória e executar a consulta no armazenamento de dados apenas se não houver resultados ou se eles tiverem expirado. Dados da sessão, preferências do usuário e quaisquer outras consultas executadas na maioria das páginas de um site são boas candidatas para o armazenamento em cache. Para interagir com o cache, usamos uma implementação da interface *net.sf.jsr107.Cache*. obtemos então uma instância do Cache utilizando um *CacheFactory*, através de um método estático no *CacheManager* como mostrado na Figura 15.

```
32  public ShardedCounter(String counterName) {
33    this.counterName = counterName;
34    cache = null;
35    try {
36      cache = CacheManager.getInstance().getCacheFactory().createCache(
37          Collections.emptyMap());
38    } catch (CacheException e) {
39    }
40  }
```

Figura 13. Utilizando a API Memcache para obtenção de uma instância do Cache.

3.2.3.2 URL Fetch

Devido às restrições do sandbox, as aplicações do GAE não podem se comunicar diretamente com outras aplicações. Para que isso seja feito, o GAE disponibilizou o serviço de URL Fetch, onde através dele, sua aplicação faz uma requisição ao Google para que ele autorize a abertura de uma porta na sua aplicação e assim ela interaja com outra aplicação web ou web service como mostrado na Figura 16.

```
60      String strDictionaryServiceCall =
61              "http://services.aonaware.com/DictService/DictService.asmx/Define?word=";
62
63      strDictionaryServiceCall += strWord;
64
65      URL url = new URL(strDictionaryServiceCall);
66      BufferedReader reader = new BufferedReader(new InputStreamReader(url.openStream()));
67      StringBuffer response = new StringBuffer();
68      String line;
```

Figura 14. Obtendo um serviço web através da API URL Fetch.

Uma aplicação pode usar o serviço URL Fetch para emitir solicitações HTTP ou HTTPS e receber respostas. Além disso o serviço dispõe de opções para personalizar suas requisições através da implementação de métodos da classe FetchOptions como por exemplo:

- *allowTruncate*: esta função permite o truncamento de respostas muito grandes.

- *doNotFollowRedirects*: é chamando quando não se pretende permitir redirecionamento de requisições.
- *validateCertificate*, se a requisição for do tipo HTTPS, este método permite que seja validado o certificado SSL da aplicação requisitada.

O serviço URL Fetch usa a infraestrutura de rede do Google para proporcionar eficiência e escalabilidade.

3.2.3.3 Serviço de E-mail

As aplicações do GAE podem enviar mensagens de e-mail em nome dos administradores da aplicação e em nome de usuários com Contas do Google. As aplicações podem receber e-mails em vários endereços. Elas enviam mensagens usando o serviço de E-mail e recebem mensagens na forma de solicitações HTTP iniciadas pelo GAE e postadas para a aplicação.

Para enviar uma mensagem de e-mail, a aplicação prepara um objeto *MimeMessage* e o envia com o método *send()* na classe *Transport*. A mensagem é criada usando um objeto de Sessão *JavaMail*. A Sessão e o Transporte trabalham em conjunto com o serviço de E-mail do GAE sem qualquer configuração adicional como mostrado na Figura 17.

```
34 try {
35     Message msg = new MimeMessage(session);
36     msg.setFrom(new InternetAddress("lord.sena@gmail.com", "lord.sena"));
37     msg.addRecipient(Message.RecipientType.TO,
38             new InternetAddress(dest, "Mr. User"));
39     msg.setSubject("assunto");
40     msg.setText(msgBody);
41     Transport.send(msg);
42
```

Figura 17. Utilizando a API Serviço de E-mail.

3.2.3.4 Mensagens Instantâneas

Uma aplicação do GAE pode enviar e receber mensagens instantâneas para qualquer serviço de mensagens instantâneas compatível com XMPP, como o Google Talk. Uma aplicação pode enviar e receber mensagens de bate-papo, enviar convites de bate-papo e solicitar informações de status. Mensagens XMPP de entrada são processadas por manipuladores de solicitação, semelhantes a solicitações da web.

Alguns usos possíveis de mensagens instantâneas incluem participantes de bate-papo automatizados ("bots de bate-papo"), notificações instantâneas e interfaces de bate-papo para serviços. Um cliente avançado com uma conexão a um servidor XMPP (como o Google Talk) pode usar XMPP para interagir com uma aplicação do GAE em tempo real, inclusive para receber mensagens iniciado pela aplicação. Observe que esse tipo de cliente usando o Google Talk deve usar a senha do usuário para fazer uma conexão XMPP e não pode usar um cookie das Contas do Google.

Atualmente, uma aplicação não pode participar de bate-papos de grupo. Uma aplicação só pode receber mensagens dos tipos "bate-papo" e "normal". Uma aplicação pode enviar mensagens de qualquer tipo definido em RFC 3921.

Para ativar o serviço XMPP para uma aplicação Java, editamos o arquivo *appengine-web.xml* como mostrado na Figura 18.

```
16  <inbound-services>
17    <service>xmpp_message</service>
18  </inbound-services>
```

Figura 18. Ativando o serviço de Mensagens Instantâneas.

3.2.3.5 Task Queue

Com a API Task Queue, aplicações podem desempenhar trabalho fora do escopo de uma solicitação web. Se uma aplicação precisar executar algum trabalho em segundo plano, pode usar a API Task Queue para organizar esse trabalho em unidades pequenas e discretas, chamadas Tarefas. A aplicação, então, insere essas tarefas em uma ou mais filas. O GAE detecta novas tarefas automaticamente e as executa quando os recursos do sistema permitem.

Para enfileirar uma tarefa, é necessário obter uma *Queue* usando o *QueueFactory* e, em seguida, chamar o método *add()*. É possível obter uma fila nomeada especificada no arquivo queue.xml usando o método *getQueue()* da fábrica ou obter a fila padrão usando o método *getDefaultQueue()*. É possível chamar o método *add()* de *Queue* com uma instância *TaskOptions* (produzida por *TaskOptions.Builder*) como mostrado na Figura 19.

```
34    QueueFactory.getDefaultQueue().add(
35        TaskOptions.Builder.withUrl("/workers/simplecounter")
36          .param("name", "thecounter")
37          .param("delta", "1"));
38
39    resp.sendRedirect("/simplecounter.jsp");
```

Figura 19. Adicionando uma tarefa a uma fila com opções.

3.2.3.6 Blobstore

A API do Blobstore permite que sua aplicação exiba objetos de dados, chamados blobs, que são muito maiores que o tamanho permitido para objetos no serviço do Armazenamento de dados. Blobs são criados fazendo upload de um arquivo através de uma solicitação HTTP. Em geral, suas aplicações farão isso apresentando um formulário com um campo de upload de arquivo para o usuário. Quando o formulário é enviado, o Blobstore cria um blob a partir do conteúdo do arquivo e retorna uma referência opaca ao blob, chamada de chave blob, que você pode usar mais tarde para exibir o blob (no caso da Figura 20 a chave do blob está representada pela variável *blobkey*). A aplicação pode exibir o valor do blob completo em resposta a uma solicitação do usuário ou pode ler o valor diretamente usando uma interface de fluxo contínuo semelhante a arquivo.

Blobs são úteis para exibir arquivos grandes, como arquivos de imagem ou vídeo, e para permitir que usuários façam upload de arquivos de dados grandes.

```
53    BlobInfoFactory blobInfoFactory = new BlobInfoFactory();
54    BlobInfo blobInfo = blobInfoFactory.loadBlobInfo(blobKey);
55
56    String contentType = blobInfo.getContentType();
57    long size = blobInfo.getSize();
58    Date creation = blobInfo.getCreation();
59    String fileName = blobInfo.getFilename();
60
61    String title = req.getParameter("title");
62    String description = req.getParameter("description");
63    boolean isShared = "public".equalsIgnoreCase(req.getParameter("share"));
64
65    try {
66      MediaObject mediaObj = new MediaObject(user, blobKey, creation,
67        contentType, fileName, size, title, description, isShared);
68      PMF.get().getPersistenceManager().makePersistent(mediaObj);
```

Figura 20. Criando um arquivo blob utilizando a API Blobstore.

3.2.3.7 Images

O GAE oferece um recurso de manipulação de dados de imagens através de um serviço de imagens dedicado. O serviço de imagens permite redimensionar, girar, inverter e recortar imagens. Também é possível aperfeiçoar fotografias através de um algoritmo predefinido.

O serviço *Images* pode aceitar dados de imagem diretamente da aplicação ou pode usar um valor do *Blobstore*. Quando a fonte é o *Blobstore*, o tamanho da imagem a ser transformada pode ser igual ao tamanho máximo de um valor do Blobstore. Entretanto, a imagem transformada será retornada diretamente para a aplicação e, por isso, não poderá ser maior que 1 megabyte. Isso pode ser útil para criar imagens de miniatura de fotos enviadas para o Blobstore por usuários.

Para transformar uma imagem do *Blobstore*, é preciso criar um objeto *Image* chamando o método estático *ImageServiceFactory.makeImageFromBlob()*, passando a ele um valor *blobstore.BlobKey* como mostrado na Figura 21. O resto da API funciona normalmente. O método *applyTransform()* retorna o resultado das transformações ou lança um *ImageServiceFailureException* se o resultado for maior do que o tamanho máximo de 1 MB.

```
61      String rotation = req.getParameter("rotate");
62      if (rotation != null && !"".equals(rotation) && !"null".equals(rotation)) {
63        int degrees = Integer.parseInt(rotation);
64
65        ImagesService imagesService = ImagesServiceFactory.getImagesService();
66        Image image = ImagesServiceFactory.makeImageFromBlob(blobKey);
67        Transform rotate = ImagesServiceFactory.makeRotate(degrees);
68        Image newImage = imagesService.applyTransform(rotate, image);
69        byte[] imgbyte = newImage.getImageData();
70
71        resp.setContentType(result.getContentType());
72        resp.getOutputStream().write(imgbyte);
73        return;
74      }
75      blobstoreService.serve(blobKey, resp);
```

Figura 21. Transformando uma imagem com a API Images.

3.2.3.8 User Service

As aplicações do GAE podem autenticar usuários através de um destes três métodos: Contas do Google, contas em seus próprios domínios do Google Apps ou identificadores OpenID. Uma aplicação pode detectar se o usuário atual fez login e pode redirecioná-lo para a página de login adequada para que ele possa fazer login ou, se a aplicação usar autenticação das Contas do Google, criar uma nova conta. Enquanto um usuário estiver conectado, a aplicação pode acessar o endereço de e-mail do usuário (ou o identificador OpenID se sua aplicação estiver usando OpenID). A aplicação também pode detectar se o usuário atual é um administrador, facilitando a implementação de áreas da aplicação restritas a administradores.

Uma vez que identificadores OpenID são fornecidos por um grande número de websites e serviços populares, incluindo o Google, oferecer suporte a OpenID é uma excelente maneira de integrar sua aplicação ao Google App Marketplace e torná-lo amplamente acessível para usuários.

É possível testar se o usuário está conectado e obter seu endereço de e-mail ou identificador do OpenID usando a API servlet padrão, com o método *getUserPrincipal()* do objeto da solicitação. É possível usar a API de serviço do Usuário para gerar URLs de login e logoff.

A API de serviço do Usuário pode retornar as informações atuais do usuário como um objeto *User*. Os objetos do usuário podem ser armazenados na forma de valor de propriedade no armazenamento de dados.

É possível verificar se o usuário fez login com uma Conta do Google. No entanto, caso o usuário não esteja logado, ele será redirecionado para a tela de login das Contas do Google. O método *userService.createLoginURL()*, como mostrado na Figura 22, retornará a URL da tela de login.

O recurso de login é capaz de redirecionar o usuário de volta para a aplicação pela URL através do método *createLoginURL()* que, nesse caso, é a URL da página atual.

```
25    UserService userService = UserServiceFactory.getUserService();
26    User user = userService.getCurrentUser();
27
28    String authURL = (user != null) ? userService.createLogoutURL("/")
29      : userService.createLoginURL("/");
30
31    PersistenceManager pm = PMF.get().getPersistenceManager();
32
33    Query query = pm.newQuery(MediaObject.class, "owner == userParam");
34    query.declareImports("import com.google.appengine.api.users.User");
35    query.declareParameters("User userParam");
```

Figura 22. Utilizando a API User Service para se autenticar como usuário Google.

3.2.4 Implantação

Para fazer a implantação no GAE, é necessário que a aplicação tenha um registro de ID, fornecido quando se cria uma aplicação usando o Console de administração do GAE. Depois de registrado o ID, este é enviado para o GAE usando o *plug-in* do Eclipse ou uma ferramenta de linha de comando do SDK.

3.3 OpenStack

O OpenStack é uma coleção de projeto de software código aberto (opensouce) licenciado sob a Licença Apache na sua versão 2.0 (License Apache version 2.0) e fornecido ao público pela OpenStackTM Compute. Qualquer organização pode utilizar a nuvem OpenStack tanto para nuvem computação quanto para armazenamento, desde que respeite a licença estabelecida.

O OpenStack surgiu de um projeto de desenvolvimento colaborativo entre a NASA (National Aeronautics and Space Administration) que contribuiu com a parte do projeto Cloud Files e a Rackspace Hosting, grande provedor de hospedagem e de serviços de nuvem, o qual contribuiu com a plataforma Nebula. O objetivo era ter no final uma plataforma de nuvem de código aberto que pudesse permitir ao adquirente a possibilidade de implantar nuvens privadas e/ou públicas em sua organização de forma simples e escalável. A comunidade de colaboradores do projeto OpenStack cresce a cada dia e incorpora empresas de renome como AT&T, Canonical, Cisco, Citrix, Dell, HP, IBM, Intel, Nebula, Rackspace, Red Hat, Inc entre outras.Atualmente, participam da comunidade de tecnólogos, desenvolvedores e pesquisadores do OpenStack cerca de 6024 pessoas em 87 países.

Em julho de 2010, a Rackspace e a NASA anunciaram a criação do projeto OpenStack. Em outubro do mesmo ano a primeira versão do OpenStack foi disponibilizada com o nome de Austin. A Tabela 1 mostra a versão e a data de disponibilização do OpenStack.

Tabela 1 – OpenStack - Versão e data de disponibilização

Nome da versão	Data da disponibilização
Austin	21 de outubro de 2010
Bexar	3 de fevereiro de 2011
Cactus	15 de abril de 2011

Diablo	22 de setembro de 2011
Essex	5 de abril de 2012
Folson	27 de setembro de 2012

3.3.1 Visão geral do OpenStack

A Figura 23 apresenta uma visão geral do Openstack. No OpenStack, o administrador da nuvem interage com a plataforma partir de uma interface web de gerenciamento, Openstack Dashboard, de modo que através de chamadas de API, ele acessa os serviços de processamento (Compute), de armazenamendo de objetos e imagens (Storage) conectados e disponibilizados através do serviço de Networking.

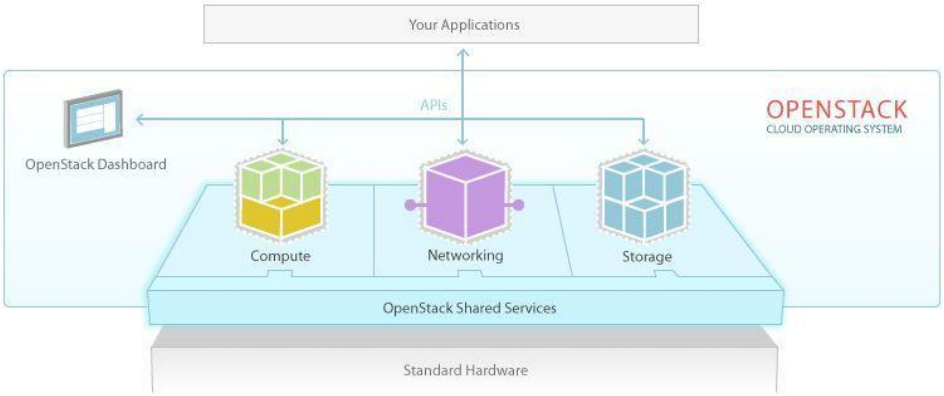

Figura 23 – Visão geral de uma aplicação no OpenStack

Fonte: https://www.openstack.org/software/

3.3.2 – Principais serviços

Os principais serviços do OpenStack são: (i) Computação, (ii) Dashboard, (iii) Identidade, (iv) Serviço de imagem, (v) Rede e (vi) Object Storage.

Computação (Nova Compute). fornece servidores virtuais sob demanda. É um controlador estrutural de computação que gerencia todas as atividades necessárias para suportar o ciclo de vida de instâncias dentro da nuvem OpenStack. Recebem os pedidos de gerenciamento pelo AMQP e as operações correspondentes são executadas através de algoritmos de escalonamento. Empresas como o Mercado livre e a NASA utilizam este componente internamente e outras empresas como a HP e a Rackspace fornecem serviços comerciais de computação construídas usando Nova. E sua Api nativa propõe compatibilidade com os encontrados na EC2 da Amazon e a Api S3.

Dashboard (Horizon). provê para o usuário uma interface web para todos os serviços OpenStack. Com esta interface web, o usuário pode realizar a maioria das operações em sua nuvem, como o lançamento de uma instância, a atribuição de endereços IP e definir controles de acesso, ou seja, é um front-end web para os outros serviços OpenStack.

Identidade (Keystone). fornece uma política de acesso, autenticação e autorização, para todos os serviços OpenStack. Ele também oferece um catálogo de serviços dentro de uma nuvem OpenStack.

Imagem (Glance). concede um catálogo e um repositório de imagens de disco virtual. Estas imagens de disco são mais comumente usados em OpenStack Compute.

Objeto Store (Swift). prover o armazenamento de objetos, permitindo o usuário armazenar ou recuperar arquivos, mas não é permitido montar diretórios como em servidor de arquivos. Algumas empresas como Rackspace e Internap se baseiam no swift para fornecer serviço de armazenamento comercial; e outras empresas o utilizam internamente para armazenar seus dados.

3.3.3 Virtualização.

O controlador de computação virtual nova suporta várias tecnologias de virtualização como por exemplo: KVM, Xen, VMWare, LXC, entre outros. O controle dessas ferramentas de virtualização é realizada a partir da biblioteca *libvirt* que é uma API de código aberto Linux para gerenciar os servidores virtualizados.

3.3.4 Ambiente de implantação da plataforma de nuvem OpenStack.

Antes de começar a instalar e configurar a nuvem, é necessário conhecer e escolher a arquitetura de instalação desejada. A cloud Openstack pode-se dividir em quatro tipos de Nodes:

a) **Nó de processamento.** responsável por armazenar e hospedar as VM´s.

b) **Nó de Controle.** Responsável por gerenciar e direcionar as chamadas de API´s aos serviços corretos.

c) **Nó de Rede.** Responsável por gerencias os IP´s da nuvem e criar vlans.

d) **Nó de armazenamento.** Responsável por armazenar objetos e imagens do Glance.

Pode-se ainda planejar uma instalação a partir das abordagens Single-host ou Multi-host.

a) **Single-Host.** Todos os Nós anteriormente citados (com exceção do Nó de armazenamento) são simulados em uma única maquina física.

b) **Multi-Host.** Uma maquina é responsável por ser Nó de controle e Rede, e as demais ligadas a elas são maquinas de processamento, e outras maquinas podem prover o serviço de armazenamento. Esta é uma arquitetura mais distribuída e é a que utilizamos neste minicurso.

Outra escolha que deve ser realizada é se o ambiente será virtualizado ou não. No ambiente físico é necessário ter duas interfaces de rede. A primeira interface faz a comunicação com a rede privada e a segunda interface fica responsável pela comunicação com a rede pública. O Sistema Operacional instalado tem acesso direto ao hardware e o OpenStack é implantado neste S.O. Já o ambiente virtualizado pode ser divido em duas categorias. A primeira seria o ambiente Nativo, onde o OpenStack é implantado em cima de um gerenciador de máquinas virtuais (hypervisor). A segunda seria o ambiente hibrido onde o OpenStack ficaria na terceira camada (S.O. Hospedeiro, ferramenta de virtualização (VirtualBox, VMWare, outros) e OpenStack). Nestas abordagens é necessário criar interfaces virtuais, sendo que na primeira duas interfaces (uma para a rede privada e a outra para a rede pública) e na segunda abordagem seria interessante criar três interfaces (uma realizando *NAT* com a máquina hospedeira, uma para a rede privada e a última para a rede pública).

3.3.4.1 Instalação

A implantação da plataforma de nuvem OpenStack pode ser iniciada com a configuração da rede, definindo a interface de comunicação com a rede pública e a interface de comunicação com a rede privada. Em seguida, inicia-se a instalação dos softwares básicos necessários. As etapas seguintes serão para o ambiente *single host*, ou seja, todos os serviços serão executados em um único servidor. Desse modo será necessário a instalar um SGBD (Sistema de Gerenciamento de Banco de Dados), pacotes de suporte ao python como python-dateutil, python-mysqldb, python-memcache; cliente de acesso VNC, módulo que usa a api EC2 como interface com os serviços OpenStack, softwares de serviços a virtualização, volume e outros. Em seguida inicia-se a instalação dos serviços do OpenStack como nova-api, nova-objectstore, nova-scheduler, nova-network, nova-compute, glance, glance-api, glance-client, glance-common, glance-registry, python-glance, keystone, python-keystone, python-keystoneclient, openstack-dashboard. As tabelas a seguir apontam os pacotes que precisam ser instalados e configurados para o bom funcionamento da plataforma de nuvem OpenStack e explicam a sua função nesta. A Tabela 2 mostra os pacotes básicos necessários, a Tabela 3 apresenta os pacotes do serviço nova, a Tabela 4 exibe os pacotes do serviço glance e por fim a Tabela 5 exposição dos pacotes do serviço keystone.

Tabela 2 - Pacotes básicos

Pacote	***Função***
mysql (server, client),	Persistir dados do Openstack
novnc	Cliente VNC usado no Dashboard
memcached,pyton-memcached, python-dateutil, python-mysqldb	Pacotes para suporte ao Pyton. O Openstack foi escrito em Pyton.
qemu	Emulador usado caso não tenha um hypervisor instalado.
euca2ools	Modulo que usa a api EC2 como interface com os serviços Openstack.
libvirt-bin, tgt, apache2, libapache2-mod-wsgi, ntp	Softwares de suporte a serviços gerais*
open-iscsi, open-iscsi-utils	Auxiliam na criação de volumes

Tabela 3 - Pacotes do serviço nova

Pacote	***Função***
nova-api	*Interface de interação com a nuvem*
nova-scheduler	*Escalonador de mensagens*
nova-objectstore	*Prover Compatibilidade com api S3(img)*
nova-network	*Responsável por controlar a rede*
nova-compute	*Gerenciar ciclo de vida das instancias*
nova-consoleauth, nova-console	*Necessário para configurar VNC*

	(noVNC)
nova-volume	*Gerenciar os volumes*
nova-common, python-nova e python-novaclient	*Necessário para instalação dos demais scripts*

Tabela 4 - Pacotes do serviço glance

Pacote	***Função***
Glance	*Gerencia imagens*
glance-api	*api de interação com o glance*
glance-common, glance-registry e python-glance	*Pacotes de suporte ao serviço glance*

Tabela 5 - Pacotes do serviço keystone

Pacote	***Função***
keystone	*Autenticação e políticas de acesso*
python-keystone python-keystoneclient	*Pacotes de configuração com o Python*

3.3.4.2 Exemplo de configuração de arquivo do OpenStack

O exemplo de configuração abaixo consiste no arquivo nova.conf no Ubuntu que se encontra localizado no /etc/nova/nova.conf. As informações a ser adicionadas este arquivo são: endereço IP do servidor de computação, endereço IP do S3, tipo de virtualização, endereço IP do servidor mysql, configurações de rede(interna e externa) e endereço IP do servidor de imagem (glance).

```
--verbose
--daemonize
--dhcpbridge_flagfile=/etc/nova/nova.conf
--dhcpbridge=/usr/bin/nova-dhcpbridge
--force_dhcp_release
--logdir=/var/log/nova
--state_path=/var/lib/nova
--lock_path=/var/lock/nova
--libvirt_type=qemu
--libvirt_use_virtio_for_bridges
--connection_type=libvirt
--sql_connection=mysql://nova:openstack@localhost/nova
```

```
--s3_host= localhost
--rabbit_host=localhost
--ec2_host=localhost
--ec2_dmz_host=localhost
--fixed_range=192.168.1.0/24
--network_size=256
--num_networks=1
--public_interface=eth1
--image_service=nova.image.glance.GlanceImageService
--glance_api_servers=localhost:9292
--auto_assign_floating_ip
--iscsi_helper=tgtadm
--root_helper=sudo nova-rootwrap
```

3.3.4.3 Iniciando uma instância através do dashboard

São necessários alguns passos para se iniciar uma instancia de uma máquina virtual no *OpenStack*, sendo esses passos:

a)**Acessar a interface de gerenciamento.** Uma vez instalado o Openstack e devidamente configurado, a nuvem já está pronta para entrar em produção. Para isso, veremos os passos necessários para instanciar uma Maquina Virtual Ubuntu 10.04 LTS através do dashboard Horizon. O acesso ao Horizon se dá por padrão através da porta HTTP padrão (80) utilizando um browser (*http:///[IP_SERVIDOR]*). Após acessar esse endereço, a tela da Figura 24 é apresentada.

Figura 24: Tela de login do Openstack DashBoard.

b) **Criando chave de permissão.** Em seguida, antes de instanciar a VM, é preciso criar uma chave de permissão, a KeyPair. Para isso basta acessar o menu keypair. Será

feito o download de um arquivo criptografado de usando na criação da instancia, conforme Figura 25.

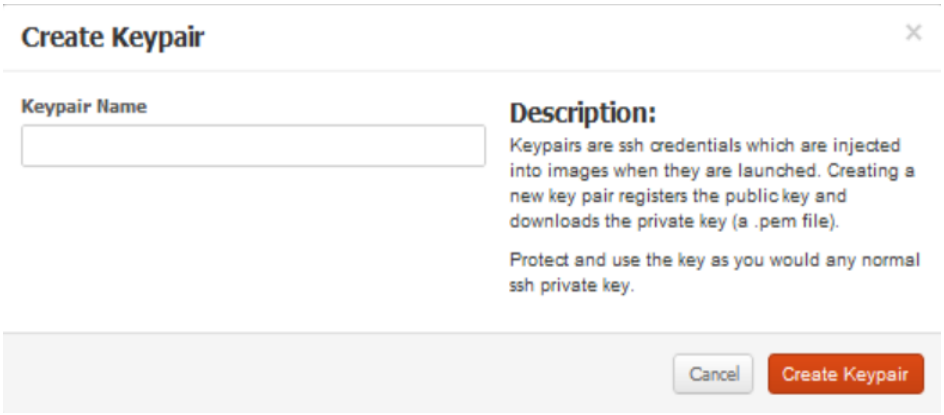

Figura 25: Criando chave de permissão.

c) **Iniciando uma instância.** Uma vez que o *Keypair foi criado,* deve-se selecionar o meu *Images,* selecionar a imagem do Sistema Operacional que a VM deverá ter e clicar em *Launch,* passando a chave de acesso conforme a Figura 26.

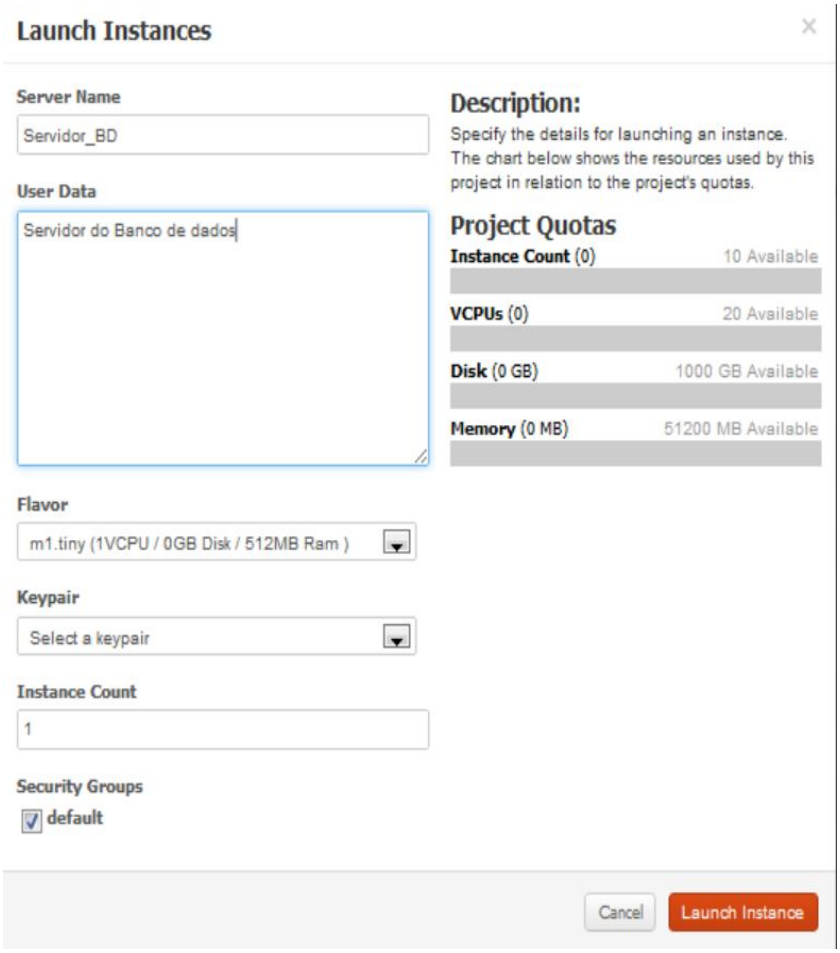

Figura 26: Instanciando VM

d) **Acessando a instancia via SSH.** Já com a Instância iniciada, é necessário verificar se o estado "Rodando" (Running) conforme a figura X, basta olhar qual IP está associado a ela (Conforme Figura 27), e acessá-la passando o *keypair* usando na sua criação, via SSH, através de qualquer software que de suporte a isso conforme a Figura 28.

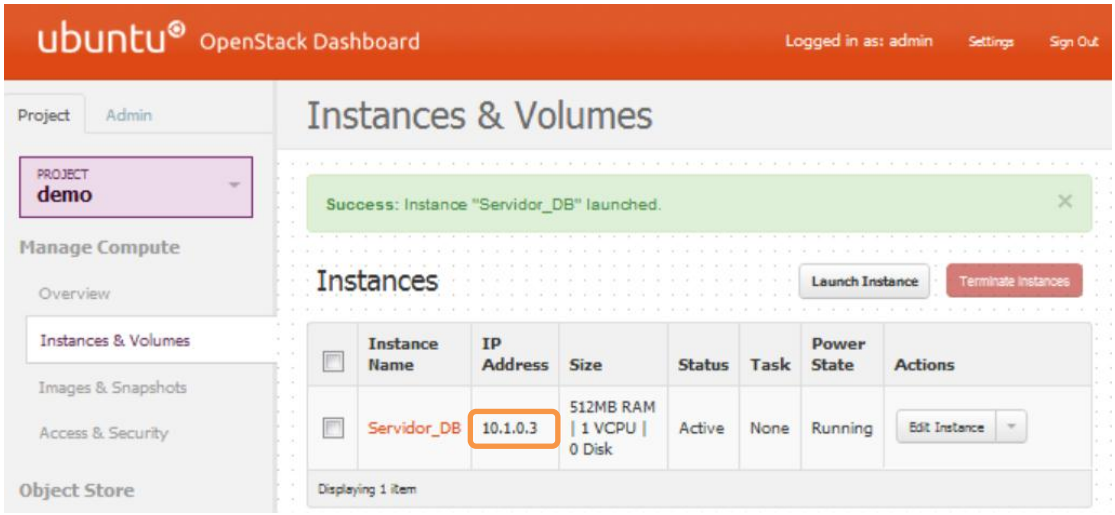

Figura 27: Instância ativa na nuvem.

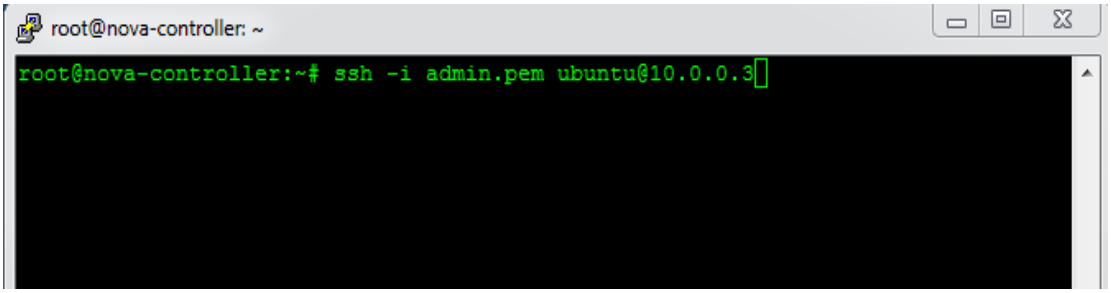

Figura 28: Acessando a instancia via SSH.

e) **Configurando acesso public.** O Openstack dá ao usuário a possibilidade de níveis de acesso da nuvem através de IP´s públicos e privados. Ao iniciar uma instancia, é atribuído a ela um IP privado. Para que a instancia possa ser visível, é necessário associá-la a um IP publico. Para isso deve-se acessar o Menu *Access & Security* e na opção *Floating IP´s* escolher qual instancia terá seu IP privado associado a um externo e selecionar *Allocate IP* seguido de *Associate IP* conforme a as figuras 29 e 30. Feito isto, temos uma instância Openstack em produção em uma Nuvem.

Figura 29 – Alocando IP externo.

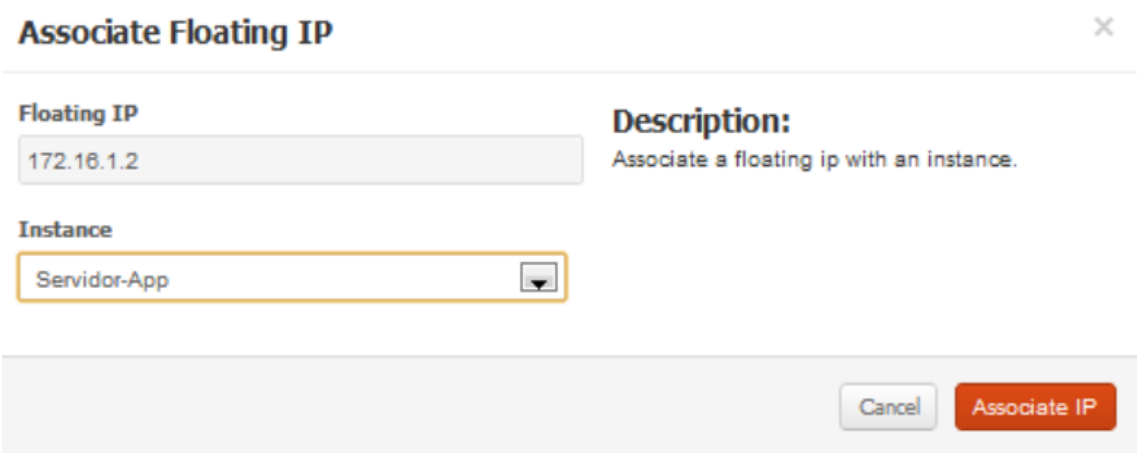

Figura 30 – Associando IP externo à Instancia.

4. Conclusão

A adoção da computação em nuvem permite que o usuário não se preocupe com o SO e hardware utilizados, além de permitir que o usuário acesse os dados e recursos computacionais independente de sua localização. Mais do que isso, permite ainda a diminuição de custos uma vez que possibilita o uso de grandes servidores de terceiros, de modo que o cliente não precisa adquirir tais servidores, nem contratar técnicos qualificados para manter os mesmos. Outra característica que permitir a diminuição de custos é a não exigência de que o cliente adquira licença integral de software.

Esse minicurso apresentou os conceitos básicos da computação em nuvem e ainda detalhes de uso de três diferentes plataformas que provêem essa tecnologia, sendo elas: Amazon Web Services (AWS), Google App Engine (GAE) e OpenStack.

Referências

Armbrust, Michael; Fox, Armando; Griffith, Rean; Joseph, Anthony D.; Katz, Randy H.; et al. (2009) Above the clouds: A Berkley view of Cloud Computing – Technical report. Reliable Adaptive Distributed Systems Laboratory, University of California at Berkley, USA.

Bose, Sumit; Pasala, Anjaneyulu; Ramanujam, Dheepak; Murthy, Sridhar; Malaiyandisamy, Ganesan (2011) SLA management in Cloud Computing: A service provider's perspective. In: Buyya, Rajkumar; Broberg, James; Goscinski, Andrzej (eds.) Cloud Computing: Principles and paradigms. New Jersey, USA: John Wiley & Sons, pp.413–436.

Breitman, Karin (2010) Computação em Nuvem. In: Meira Jr., Wagner; Carvalho, André Carlos Ponce de Leon Ferreira de (org.) Atualizações em Informática 2010. Rio de Janeiro, Brasil: Editora da Pontifícia Universidade Católica do Rio de Janeiro / Porto Alegre, Rio Grande do Sul, Brasil: Sociedade Brasileira de Computação, pp.11–50.

Buyya, Rajkumar; Yeo, Chee Sin; Venugopal, Srikumar (2008) Market-oriented Cloud Computing: Vision, hype and reality for delivering IT services as computing utilities. *In*: HPCC 2008 – 10th IEEE International Conference on High Performance Computing and Communications, 2008, Dalian, China. *Proceedings of*... Washington, D.C, USA: IEEE Computer Society, pp.5–13.

Cearley, David W. (2009) The Cloud Computing scenario – Technical report. Gartner Group.

Cearley, David W. et al. (2009) Hype cycle for application development – Technical report. Gartner Group.

Galán, F. et al. (2009) "Service specification in cloud environments based on extensions to open standards". Proc. of the Fourth Int. ICST Conf. on Communication System Software and Middleware (COMSWARE 2009). New York, NY, USA: ACM.

Hu, Ji; Klein, Andreas (2009) A benchmark of transparent data encryption for migration of Web applications in the clouds. In: DASC 2009 – Eighth IEEE International Conference on Dependable, Autonomic and Security Computing, 2009, Chengdu, China. Proceedings of… Washington, DC, USA: IEEE Computer Society, pp.735–740.

Jinnan, Yang; Sheng, Wu (2010) Studies on application of Cloud Computing techniques in GIS. In: IITA-GRS 2010 – 2nd IITA International Conference on Geoscience and Remote Sensing, 2010, Qingdao, China. Proceeding of… [s.l.] IEEE, pp.44–51.

Keller, Alexander; Ludwig, Heiko (2003) The WSLA Framework: Specifying and monitoring service level agreements for Web Services. Journal of Networks and Systems Management 11(1), pp.57–81.

libvirt. Disponível em: <http://libvirt.org/> Acesso 27 out. 2012.

Mell, P. and Grace, T. (2011) The NIST definition of Cloud Computing. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

OpenStack (2012). OpenStack Guide. http://www.openstack.org

Patel, Pankesh; Ranabahu, Ajith; Sheth, Amit (2009) Service-Level Agreement in Cloud Computing. In: OOPSLA 2009 – 24th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, 2009, Orlando, USA.

Rackspace. Disponível em: <http://www.rackspace.com/> Acesso 27 out. 2012.

Rimal, Bhaskar Prasad; Choi, Eumni; Lumb, Ian (2009) A taxonomy and survey of Cloud Computing systems. In: NCM 2009 – 5th International Joint Conference on INC, IMS and IDC, 2009, Seoul, Korea. Proceedings of… Washington, DC, USA: IEEE Computer Society, pp.44–51.

Rittinghouse, John W.; Randsome, James F. (2010) Cloud Computing: Implementation, management and security. USA: CRC Press.

Sosinsky, Barrie (2011) Cloud Computing Bible. Indianapolis, USA: Wiley Publishing, Inc.

Vaquero, Luis M.; Rodero-Merino, Luis; Caceres, Juan; Lindnet, Maik (2009) A break in the clouds: Towards a cloud definition. ACM SIGCOMM Computer Communication Review 39(1), pp.50–55.

Verma, Dinesh (1999) *Supporting Service Level Agreements on IP networks* [*s.l.*, *s.n.*] Macmillan Technical Publishing.

Vouk, Mladen A. (2008) Cloud Computing: Issues, research and implementations. Journal of Computing and Information Technology 16(4), pp.235–246.

Wang, Lizhe; Von Laszewski, Gregor; Kunze, Marcel; Tao, Jie (2010) Cloud Computing: A perspective study. *New Generation Computing* 28(2), pp.137–146.

Zhang, Qi; Cheng, Li; Boutaba, Raouf (2010) Cloud Computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications* 1(1), pp.7–18of computing. In *Advances in Computer Science*, pages 555–566. Publishing Press.

Bios

Frederico Lopes é mestre em Sistemas e Computação e doutor em Ciência da Computação pela UFRN, tendo realizado parte de seu doutorado no IST/UTL (Lisboa, Portugal). É professor adjunto da Universidade Federal do Rio Grande do Norte (UFRN). Atualmente, Fred é gerente do AltoStratus, projeto que envolve pesquisadores da UFRN, UFRJ, UFPE, UNICAMP, PUC-Rio, UFRGS, UFABC e UNIFOR, e com o objetivo de propor, especificar, implementar, implantar e avaliar técnicas e mecanismos de middleware para ambiente de nuvens computacionais híbridas e heterogêneas. Tem experiência na área de Ciência da Computação, com ênfase em Sistemas Distribuídos, atuando principalmente com os seguintes temas: computação ubíqua, middleware, sistemas sensíveis ao contexto e computação em nuvem. Link para Curriculum Lattes http://lattes.cnpq.br/9177823996895375

André Almeida é professor do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte. É aluno de doutorado do PPGSC, onde desenvolve sua tese que tem como tema "Adaptação Dinâmica de Aplicações Baseadas em Nuvem". André participa do AltoStratus, projeto que envolve pesquisadores da UFRN, UFRJ, UFPE, UNICAMP, PUC-Rio, UFRGS, UFABC e UNIFOR, e tem como objetivo de propor, especificar, implementar, implantar e avaliar técnicas e mecanismos de middleware para ambiente de nuvens computacionais híbridas e heterogêneas. Tem experiência na área de Ciência da Computação, com ênfase em Sistemas Distribuídos, atuando principalmente com os seguintes temas: middleware, e computação em nuvem Link para Curriculum Lattes: http://lattes.cnpq.br/1800962258138555

Thais Vasconcelos Batista possui mestrado em Informática (1994), doutorado em Informática (2000), ambos pela PUC-Rio e pós-doutorado realizado na Lancaster University – UK (2005). Atualmente é professora associada do Departamento de Informática e Matemática Aplicada (DIMAp) da Universidade Federal do Rio Grande do Norte (UFRN) e bolsista de produtividade nível 2 do CNPq. Suas pesquisas na área de Sistemas Distribuídos e Engenharia de Software envolvem middleware, computação ubíqua, computação em nuvem, desenvolvimento orientado a aspectos, arquitetura de software, entre outros. Atualmente coordena o projeto AltoStratus, para desenvolvimento de infra-estrutura de aplicações em nuvem, financiado pela Rede

Nacional de Pesquisa (RNP). Esse projeto conta com o apoio da Amazon para uso, no projeto, da infraestrutura de nuvem EC2. Link para Curriculum Lattes: http://lattes.cnpq.br/5521922960404236

Everton Ranielly de Sousa Cavalcante é aluno de Doutorado em Ciência da Computação na UFRN – Universidade Federal do Rio Grande do Norte, possui Mestrado em Sistemas e Computação (2012) e Bacharelado em Ciência da Computação (2010) ambos pela UFRN, e é Técnico em Desenvolvimento de Sistemas para Internet (2008) pelo IFRN – Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte. Possui experiência na área de Ciência da Computação com ênfases em sistemas distribuídos, Engenharia de Software, desenvolvimento Web, linguagens de programação e algoritmos experimentais, atuando principalmente nos seguintes temas: middleware, Computação em Nuvem, Computação Ubíqua, linguagens de descrição arquitetural e linhas de produto de software. Link para Curriculum Lattes: http://lattes.cnpq.br/5065548216266121

Renato Gondim Renato Gondim Sarmento concluinte do curso de Ciência da Computação na UFRN– Universidade Federal do Rio Grande do Norte, foi bolsista da Gerência de Redes do Departamento de Informática e Matemática Aplicada/ UFRN e atualmente faz parte do grupo de residência em software na Superintendência de Informática – SInfo. Possui experiência em sistemas distribuídos, engenharia de software e linguagens de programação. Atualmente, trabalhando com computação em nuvem e teste de software. Link para Curriculum Lattes: http://lattes.cnpq.br/0362105301851454

Thomas Diniz é graduando do curso de Ciência da Computação pela Universidade Estadual do Rio Grande do Norte (2008) e graduando do curso de Ciências e Tecnologia da Universidade Federal do Rio Grande do Norte (2009). É monitor da disciplina Informática Fundamental do Projeto "Monitoria na EC&T: o plano integrado dos três primeiros semestres" do curso Ciências e Tecnologia da UFRN. Link para Curriculum Lattes: http://lattes.cnpq.br/3931868566871340

Arthur Cassio é aluno especial do programa de Pós-Graduação em Sistemas e Computação da Universidade Federal do Rio Grande do Norte, UFRN. Graduou-se em Tecnologia em Desenvolvimento de Software pelo Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte, IFRN. Possui experiência nas áreas de sistemas web, arquitetura orientada a serviços e computação em nuvem. Link para Curriculum Lattes: http://lattes.cnpq.br/0936695072170254

Thiago Cesar é aluno no curso Engenharia de Software pela Universidade Federal do Rio Grande do Norte. Possui experiência na área da Computação com ênfases em programação distribuída, modelagem de software, desenvolvimento Web, atuando principalmente no seguinte tema: Computação em Nuvem. Link para Curriculum Lattes: http://lattes.cnpq.br/3981406236417839