Universidade Federal do Pampa

CAMPUS ALEGRETE
CURSO DE CIÊNCIA DA COMPUTAÇÃO
PROJETO DE LINGUAGENS DE PROGRAMAÇÃO

RELATÓRIO
ESTUDO COMPARATIVO ENTRE LINGUAGENS DE PROGRAMAÇÃO

Realizado por:
Bruno Schubert Trindade
Marcos Vinícius Treviso
Wolgan Ens Quepfert

Alegrete, 27 de maio de 2014.

ÍNDICE

1. INTRODUÇÃO

Linguagens de programação servem como meio de comunicação entre computadores e humanos, é através delas que damos “ordens” ao computador. Existem linguagens de alto e baixo nível, as linguagens de baixo nível são interpretadas diretamente pelo hardware do computador, o que resulta em um altíssimo desempenho, por outro lado, são de difícil entendimento.

Já as linguagens de alto nível são de fácil entendimento e precisam de compiladores/interpretadores para traduzir o código para linguagem de máquina, tornando-as mais lentas.

No relatório a seguir será realizada uma breve introdução de 4 linguagens de programação e em seguida uma análise comparatoria de resultados obtidos após a implementação de 2 algoritmos.

Serão abordados através desse relatório conceitos e fundamentos teóricos, bem como uma análise entre 4 linguagens de programação: Python, Haskell, Ruby e Hack.

Este relatório comparará as linguagens de programação nos seguintes quesitos:

- Número de Linhas de Código;
- Tempo de Execução;
- Legibilidade;
- Facilidade de Escrita.

Os resultados obtidos durante a avaliação das linguagens será exposto uma análise geral dos resultados.

2. LINGUAGENS DE PROGRAMAÇÃO

A linguagem de programação **Haskell**, nomeada em homenagem ao lógico Haskell Curry, que teve seu surgimento na década de 90 é uma linguagem funcional, sendo assim excelente para escrever especificações que podem realmente ser executadas (e, portando, testada e depurada) onde encontra-se grande vantagem, uma vez que há uma maior facilidade em efetuar testes. A linguagem Haskell, influenciada pelas linguagens Miranda e ML, é particularmente adequada para aplicações que necessitam ser altamente modificadas e de fácil manutenção. Por ser uma linguagem funcional, tende a ser muito mais concisa que linguagens imperativas, ou seja, seu código é mais curto e mais claro, onde a sua ideia principal é baseada na avaliação de expressões.

Haskell é uma linguagem interpretada, projetada especialmente para lidar com uma ampla gama de aplicações, desde numérica até simbólica. Para este fim, Haskell tem uma sintaxe expressiva, e uma rica variedade de tipos de dados internos, incluindo números inteiros de precisão arbitrária e racionais, bem como o número inteiro mais convencional, de ponto flutuante e tipos booleanos.

A linguagem de programação **Ruby**, é uma linguagem dinâmica com foco na simplicidade e na produtividade. Seu criador Yukihiro “Matz” Matsumoto, uniu partes das suas

linguagens favoritas (Perl, Smalltalk, Eiffel, Ada e Lisp) para formar uma nova linguagem que equilibra a programação funcional com a programação imperativa.

Ruby é também totalmente livre. Não somente livre de custos, mas também livre para utilizar, copiar, modificar e distribuir. É uma linguagem interpretada e orientada a objetos que, como segue a influência da linguagem Smalltalk em atribuir métodos e variáveis de instância a todos os seus tipos. Esta abordagem facilita a utilização do Ruby, uma vez que as regras se aplicam aos objetos aplicam-se a tudo em Ruby.

De forma diferente a muitas linguagens de programação orientadas a objetos, Ruby suporta somente herança simples, propositadamente. Mas existe o conceito de módulos que são coleções de métodos. Ruby não necessita de declarações de variáveis. Usa simples convenções de nomes para denotar o âmbito das variáveis como, por exemplo, @var para uma variável de instância e $var para uma variável global.

Ruby se destaca pela capacidade de tratamento de exceções, tal como Java ou Pyton, de forma a facilitar o tratamento de erros. Também é altamente portável: é desenvolvida principalmente em ambiente GNU/Linux, mas trabalha em muitos tipos de ambientes UNIX, Mac OS X, Windows 95/98/Me/NT/2000/XP, DOS, BeOS, OS/2, etc.

A **Hack** é uma linguagem de programação para HHVM (HipHop Virtual Machine), ou seja, voltada para *server-side*. Foi criada pelo Facebook, onde tem sido utilizada a cerca de um ano, teve o seu código liberado só agora em março de 2014.

Basicamente o que o Hack faz é combinar as vantagens de uma linguagem de programação dinâmica (como o PHP, em que a falha é apontada quando o programa já está rodando e acontece um bug) sem prescindir da segurança na detenção e prevenção de erros ou inconsistências de uma linguagem de programação estática (como Java ou C, onde o erro é apontado antes da execução do programa). A intenção do Hack é ter a sua própria forma de identificar erros, encontrando um meio termo e tem como grande vantagem o aumento na velocidade da programação.

Python é uma linguagem de programação criada por Guido van Rossum em 1991. Os objetivos do projeto da linguagem eram: produtividade e legibilidade. Em outras palavras, Python é uma linguagem que foi criada para produzir código bom e fácil de manter de maneira rápida. Entre as características da linguagem que ressaltam esses objetivos estão:

- baixo uso de caracteres especiais, o que torna a linguagem muito parecida com *pseudo-código executável*;
- o uso de identação para marcar blocos;
- quase nenhum uso de palavras-chave voltadas para a compilação;
- coletor de lixo para gerenciar automaticamente o uso da memória.

Além disso, Python suporta múltiplos paradigmas de programação. A programação procedural pode ser usada para programas simples e rápidos, mas estruturas de dados complexas, como tuplas, listas e dicionários, estão disponíveis para facilitar o desenvolvimento de algoritmos complexos. Grandes projetos podem ser feitos usando técnicas de orientação a objetos, que é completamente suportada em Python (inclusive sobrecarga de operadores e herança múltipla). Um suporte modesto para programação funcional existe, o que torna a linguagem extremamente expressiva: é fácil fazer muita coisa com poucas linhas de comando. Além disso, possui inúmeras capacidades de meta-programação: técnicas simples para alterar

o comportamento de comportamentos da linguagem, permitindo a criação de *linguagens de domínio específico*.

Python tem uma biblioteca padrão imensa, que contém classes, métodos e funções para realizar essencialmente qualquer tarefa, desde acesso a bancos de dados a interfaces gráficas com o usuário. E, logicamente, já que esse é o objetivo deste grupo, existem muitas ferramentas para lidar com dados científicos. Essa característica da linguagem é comumente chamado *baterias inclusas*, significando que tudo que você precisa para rodar um programa está — na maior parte das vezes — presente na instalação básica.

Por fim, e não menos importante, Python é uma linguagem livre e multiplataforma.

3. PROBLEMAS PROPOSTOS

Para a realização deste relatório, foram escolhidas 4 linguagens de programação: Python, Haskell, Ruby e Hack. O Objetivo do mesmo tem por finalidade um estudo comparativo entre as linguagens sobre alguns aspectos, como o número de linhas do código, o tempo de execução, a legibilidade e a facilidade de escrita.

Para tanto, foi desenvolvidos para cada uma delas dois programas, uma implementação de laços encadeados onde foi abordado um exemplo de multiplicação de matrizes, e outro que envolvia o cálculo da função fatorial, a qual foi feito duas versões: uma versão iterativa e outra usando recursividade.

Os códigos abaixo são apenas os definidos nas funções lógicas, deixando de lado as inicializações das estruturas e importações de bibliotecas, ou seja, apenas a aplicação do algoritmo em si. Abaixo segue os códigos referente a cada uma das linguagens de programação:

Python

Fatorial Iterativo:

```
def fatIter(n):
    x = 1
    for i in range(1, n+1):
        x *= i
    return x
```

Fatorial Recursivo:

```
def fatRec(n):
    if n <= 1:
        return 1
    return (n * fatRec(n-1))
```

Multiplicação de Matrizes:

```
def multMatrix(matrix1, matrix2, n):
    mat = [[0 for y in range(n)] for x in range(n)]
    for i in range(n):
            for j in range(n):
                    for k in range(n):
                    mat[i][j] += matrix1[i][k]*matrix2[k][j]
    return mat
```

Haskell

Fatorial Iterativo:

```
factorialIter :: (Integral n) => n -> n
factorialIter n = product [1..n]
```

Fatorial Recursivo:

```
factorialRec :: (Integral n) => n -> n
factorialRec n = if n <= 1
                        then 1
                        else n * factorialRec (n-1)
```

Multiplicação de Matrizes:

```
multMatrix :: Num a => [[a]] -> [[a]] -> [[a]]
multMatrix a b = [[ sum (zipWith (*) ad bc) | bc <- (transpose b)] | ad <- a ]
```

Ruby

Fatorial Iterativo:

```
def fatIter(n)
    fat = 1
    while n > 1
            fat *= n
            n -= 1
    end

    return fat
end
```

Fatorial Recursivo:

```
def fatRec(n)
   if n < 2
         return 1
   else
         return n * fatRec(n-1)
   end
end
```

Multiplicação de Matrizes:

```
def multMatrix(matrix1, matrix2)
   n = matrix1.length
   mat = Array.new(n){Array.new(n){ 0 }}

   0.upto(n-1) do |i|
         0.upto(n-1) do |j|
               0.upto(n-1) do |k|
                     mat[i][j] += matrix1[i][k] * matrix2[k][j]
               end
         end
   end

   return mat
end
```

Hack

Fatorial Iterativo:

```
function fatIter(int $n): int{
   $x = 1;
   for($i=1; $i<=$n; $i++)
         $x *= $i;
   return $x;
}
```

Fatorial Recursivo:

```
function fatRec(int $n): int{
   if($n < 2)
         return 1;
   return $n*fatRec($n-1);
}
```

Multiplicação de Matrizes:

```
function multMatrix(array $matrix1, array $matrix2): array{

   $tam = count($matrix1);
   $mat = array(array());

   for($i=0; $i<$tam; $i++)
         for($j=0; $j<$tam; $j++)
               $mat[$i][$j] = 0;


   for($i=0; $i<$tam; $i++)
         for($j=0; $j<$tam; $j++)
               for($k=0; $k<$tam; $k++)
                     $mat[$i][$j] += $matrix1[$i][$k]*$matrix2[$k][$j];

   return $mat;
}
```

4. RESULTADOS

Com base nos estudos realizados através de documentos de apoio e da análise das quatro linguagens de programação, foram detectados vários aspectos que divergem entre as linguagens, os quais são mostrados a seguir:

- Quanto ao Número de linhas:

Python:

Fatorial Iterativo	Fatorial Recursivo	Multiplicação de Matrizes
4 linhas	3 linhas	6 linhas

Haskell:

Fatorial Iterativo	Fatorial Recursivo	Multiplicação de Matrizes
1 linha	3 linhas	1 linha

Ruby:

Fatorial Iterativo	Fatorial Recursivo	Multiplicação de Matrizes
7 linhas	5 linhas	12 linhas

Hack:

Fatorial Iterativo	Fatorial Recursivo	Multiplicação de Matrizes
4 linhas	3 linhas	13 linhas

É possível analisar que Haskell cumpre sua premissa de encurtar o código e deixa-lo mais claro, onde isso deve-se principalmente as funções built-in da linguagem.

Ruby e Python são quase equivalentes nesse quesito, pois no caso de Ruby, cada iteração tem que ser fechada pelo palavra reservada *end*, fazendo com que o código fique mais longo. Ruby também tem o tipo de dado chamado *Matrix*, onde se for usado, a multiplicação de matrizes é simplesmente a multiplicação de varáveis (caso cabíveis nas restrições algébricas), ou seja, utiliza uma simples linha de código.

A Hack é equivalente a Python e Ruby nos fatoriais, mas na multiplicação de matrizes, ela acaba se tornando maior, devido principalmente ao fato de que a linguagem não permite a inicialização de valores num *array* diretamente durante sua inicialização.

No geral, o comprometimento com a clareza e redução de código é vista em todas as linguagens, uma vez que esse é um objetivo em comum na maioria das linguagens de programação existentes.

- Quanto ao tempo de execução:

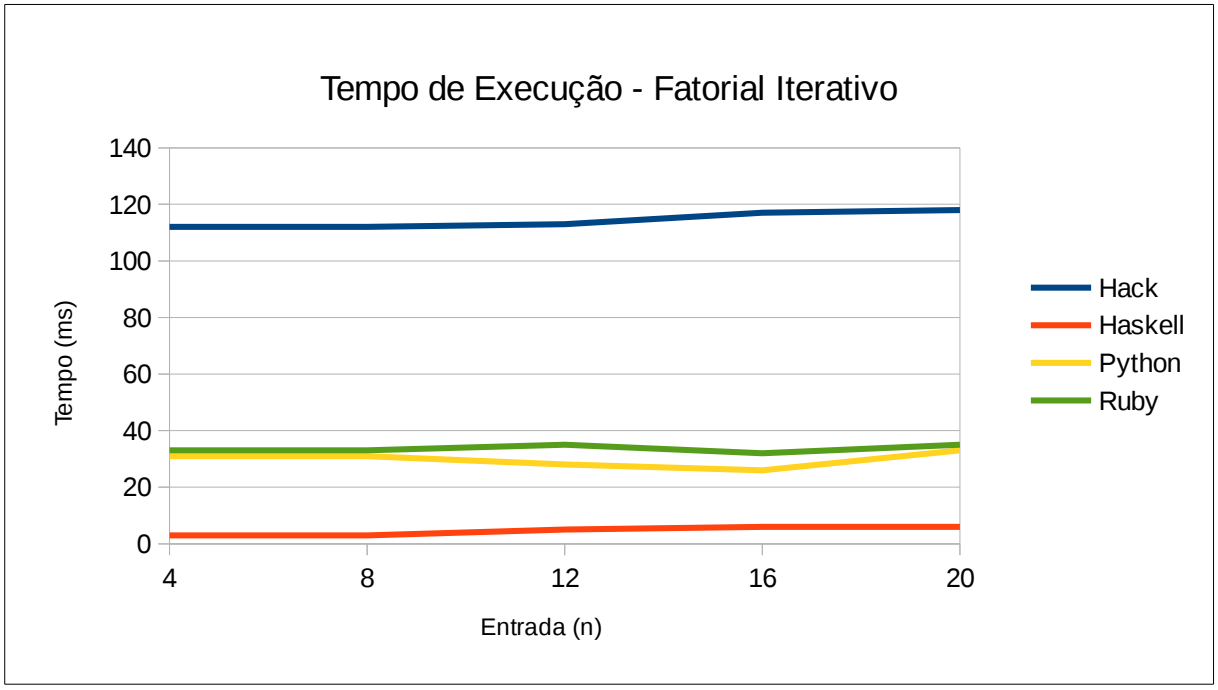

Gráfico 1 - Tempo de execução para fatorial iterativo.

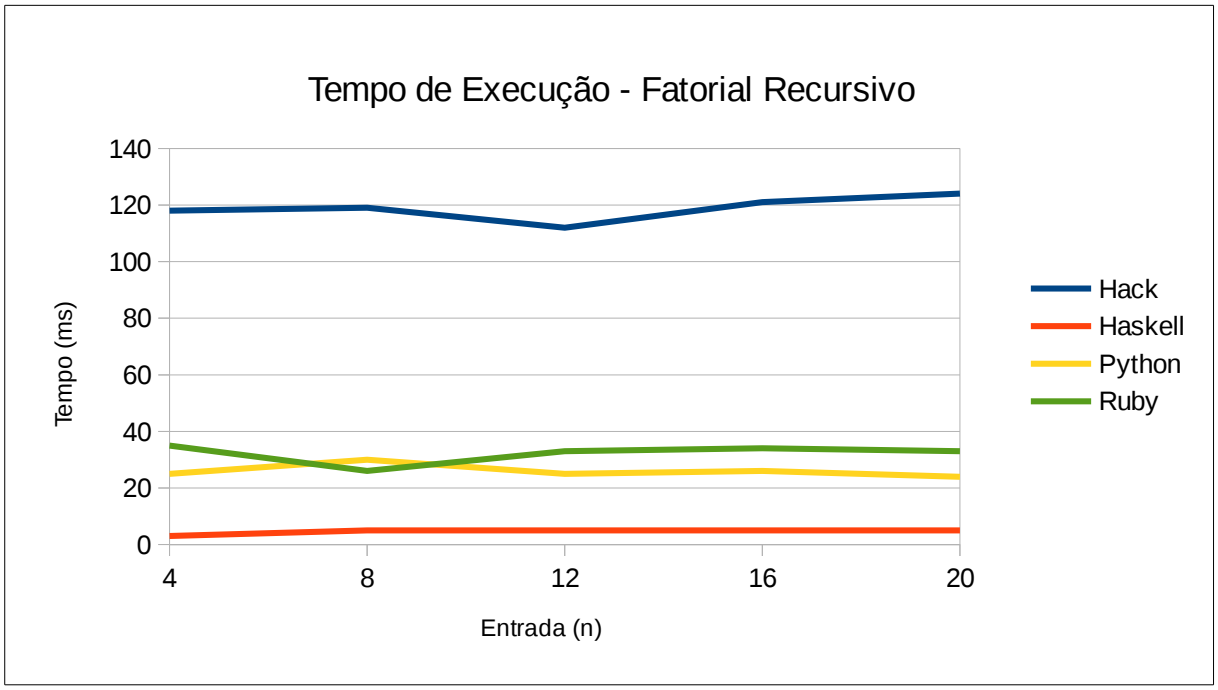

Gráfico 2 - Tempo de execução para fatorial recursivo.

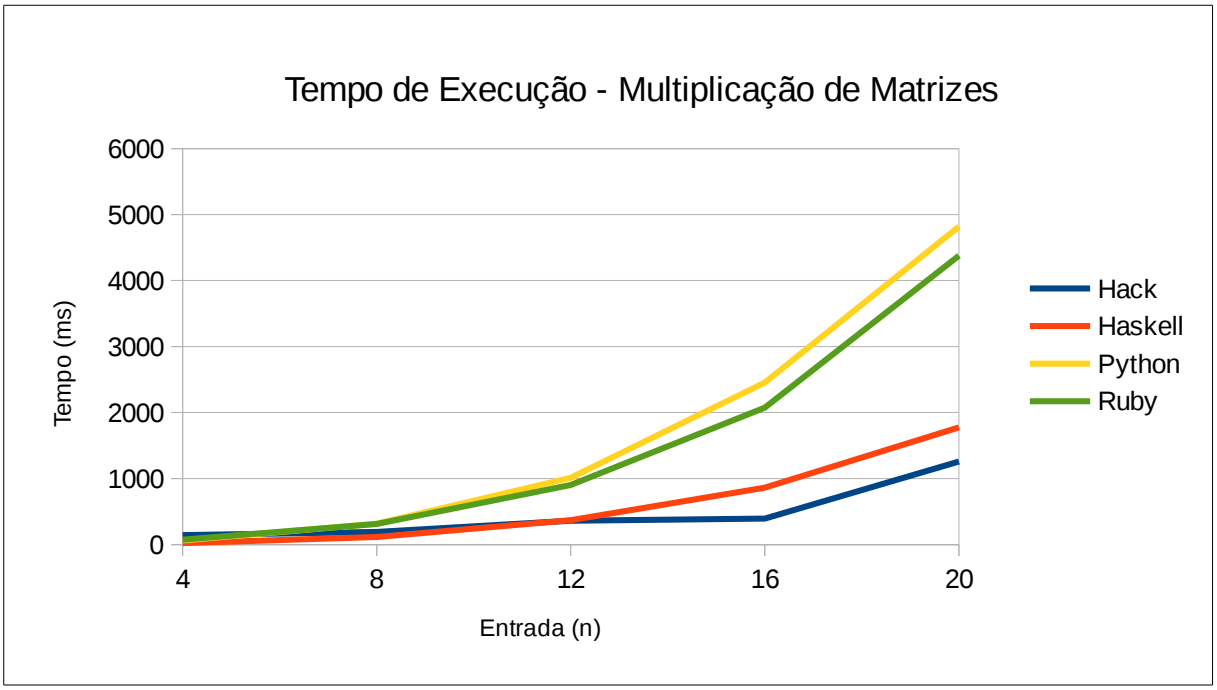

Gráfico 3 - Tempo de execução para multiplicação de matrizes.

É possível observar que os gráficos dos fatoriais estão como o esperado, onde, Hack é a linguagem mais lenta para realizar cálculos matemáticos, Ruby e Python tem uma pequena diferença, e por fim Haskell teve melhor desempenho.

Entretanto, no gráfico de multiplicação de matrizes a linguagem Hack se saiu melhor, isso é devido ao fato de que o tipo de dado *array*, que é herdado diretamente do PHP, consegue lidar muito bem com números grandes, além disso, devemos considerar que a execução não aconteceu em cima de um servidor, e sim diretamente na HHVM, o que caso tivesse acontecido, aumentaria drasticamente o tempo de execução. Resumidamente, para a execução direta através da HHVM o Hack se saiu melhor. Mas é importante considerar que Hack é uma linguagem voltada para a Web, onde na sua originalidade, com certeza seria mais lenta que as demais.

Python e Ruby permaneceram com tempos parecidos, vale ressaltar que foi usado a mesma lógica no algoritmo de multiplicação entre os dois, fazendo com que ficassem com tempo quase linear.

Haskell como o esperado, foi a linguagem que exibiu os menores tempos de execução, uma vez que a linguagem é compilada e adaptativa para esses casos.

O tempo de execução é algo complicado de se calcular para diferentes linguagens, pois para fazer algo coerente é necessário que as implementações possuem a mesma lógica e mesma complexidade. Além disso, é importante analisar o ambiente de execução, uma vez que isso pode intervir diretamente nos resultados.

- Quanto a legibilidade:

Em relação a simplicidade global, de modo geral todas linguagens análisadas são muito simples, a única peculiaridade encontrada foi a diferente forma de realizar a multiplicação de matriz em Ruby, onde isso deve-se ao fato que o tipo de dado *Matriz* é apenas uma classe obtida diretamente das bibliotecas da linguagem. O que pode ser feito em qualquer uma das outras linguagens analisadas.

Quanto a ortogonalidade, Haskell, Python e Ruby se mostraram ótimas, onde todas conseguem ser interpretadas pelo programador de forma relativamente fácil. Haskell principalmente, a multiplicação de matrizes por exemplo, fica evidente o que está acontecendo em apenas uma leitura do código.

Estruturas de controle são fortemente vistas nas linguagens imperativas, nesse caso: Hack, Python e Ruby. Essas linguagens proporcionam um melhor entendimento para com estruturas de controle, facilitando o entendimento do código. Já Haskell, não existem laços de repetições, estruturas repetitivas são feitas através de recursões, o que pode acabar fazendo com que o entendimento seja difícil.

Os tipos de dados e estruturas são legíveis em todas as linguagens apresentadas. Além disso, quase todas possuem suporte a orientação a objetos, fazendo com que classes disponíveis nas bibliotecas da linguagem pode ser usadas livremente para uma maior facilidade de escrita, e consequentemente um maior entendimento do código. Haskell possui uma gama de herança de tipos de dados que permite o programador a interferir diretamente no código no momento de escrita, facilitando o entendimento.

O projeto de sintaxe de Hack é a mais liberal entre as linguagens análisadas, onde toda variavel começa com o caractere $, sendo assim, praticamente qualquer palavra especial pode ser usada como nome de variável. Ruby também possui peculiaridades interessantes para a sintaxe, onde o escopo das variáveis são definidas no momento de escrita do código. Haskell e Python possuem uma sintaxe simples e sem muitas peculiaridades.

- Quanto a facilidade de escrita:

O suporte a abstrações é visto em todas as linguagens, e ajudou bastante na implementação dos códigos, fazendo-os ficarem mais legíveis e também ajudando no momento da escrita. Todos os algoritmos, em todas linguagens, levaram um tempo relativamente baixo para serem implementados , e isso deve-se principalmente a facilidade de escrita de cada linguagem.

A expressividade é vista em todas as linguagens, todas tem formas diferentes de realizar operações e controles, porém todas são de fácil entendimento e fáceis de escreverem. Haskell e Python conseguem reduzir operações para apenas uma linha, usando respectivamente, operadores de restrições de domínio e compressão de listas. Hack possui uma expressividade parecida com a do C, porem devido ao fato de ser tipagem fraca e dinâmica, uma grande quantidade de operações podem ser realizadas de forma única em Hack. Já em Ruby, as operações e controles são também muito simples, parecidas com Python, onde as operações são comuns e os controles são gerados apartir de funções que retornam elementos de uma lista para serem iterados.

5. CONCLUSÃO

Com base nos estudos realizados através dos documentos de apoio e na análise das linguagens, foi possível verificar que linguagens funcionais como Haskell, conseguem ser tão boas como as imperativas, muitas vezes até melhores, como visto nos gráficos de tempo de execução.

Quando é iniciado um novo projeto e deve-se ser escolhido uma linguagem de programação para trabalhar, é difícil fazer a melhor escolha, até porque não existe uma melhor linguagem de programação global, mas sim linguagens que possuem melhores resultados em determinadas áreas avaliadas. Nesse caso, podemos afirmar que segundo o relatório, as linguagens possuem diferentes características decisivas, que ajudam o desenvolvedor a verificar os melhores casos de uso para então aplicá-las em seu projeto, de modo que fique mais produtivo como o determinado.

O objetivo em relação ao trabalho foi atingido, uma vez que os integrantes conseguiram abstrair o conhecimento e entender os diferentes critérios na comparação das linguagens de programação. Onde os algoritmos usados para comparar as linguagens são triviais, no qual então foi possível realizar análises justas com cada linguagem sem perder muito tempo imlementando cada um.

6. BIBLIOGRAFIA

[1] Curso de Circuitos Elétricos – Vol. 1 – 2ª Edição – L. Q. Orsini / Denise Consonni – © 2002 Editora Edgard Blücher LTDA.

[2] Análise de Circuitos – vol. 1 – Teoria e Prática – Allan H. Robbins / Wilhelm C. Miller – © 2010 Cengage Learning.

[3] Rauber, A. Programação Funcional com a Linguagem Haskell, disponibilizado em: http://necc.di.uminho.pt/lib/exe/fetch.php?media=ap:1ano:pf:ap_pf_t_proghaskell.pdf

[4] Haskell Website, https://www.haskell.org/

[5] Ruby Website, https://www.ruby-lang.org/

[6] Hack Website, http://hacklang.org/

[7] PHP Website, https://www.php.net/

[8] Python Website, https://www.python.org/

[9] Guia Haskell, http://haskell.tailorfontela.com.br/

[10] Artigo sobre criação da Hack, http://olhardigital.uol.com.br/pro/noticia/40937/40937

[11] PyScience Brasil, http://pyscience-brasil.wikidot.com/python:python-oq-e-pq