UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIAS
COORDENAÇÃO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ÍTALO ANDRÉ DA SILVA SIMÕES

AMBIENTE VISUAL PARA SIMULAÇÃO BASEADA EM EVENTOS DISCRETOS: Um estudo de caso com a ferramenta SimPy.

São Luis
2019

UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIAS
COORDENAÇÃO DE CIÊNCIA DA COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ÍTALO ANDRÉ DA SILVA SIMÕES

AMBIENTE VISUAL PARA SIMULAÇÃO BASEADA EM EVENTOS DISCRETOS: Um estudo de caso com a ferramenta SimPy.

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Dr. Mário Antônio Meireles Teixeira

São Luis
2019

Simões, Ítalo
AMBIENTE VISUAL PARA SIMULAÇÃO BASEADA EM EVENTOS DISCRETOS: Um estudo de caso com a ferramenta SimPy./ ÍTALO ANDRÉ DA SILVA SIMÕES. – São Luis, 2019.
44 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Mário Antônio Meireles Teixeira

Monografia – UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIAS
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO, 2019.

1. Python. 2. Simpy. 3. simulação. 4. interface 5. modelagem

ÍTALO ANDRÉ DA SILVA SIMÕES

AMBIENTE VISUAL PARA SIMULAÇÃO BASEADA EM EVENTOS DISCRETOS: Um estudo de caso com a ferramenta SimPy.

Monografia apresentada ao curso de Ciência da Computação da Universidade Federal do Maranhão, como parte dos requisitos necessários para obtenção do grau de Bacharel em Ciência da Computação.

Data da Defesa:
Conceito:

Banca Examinadora

Prof. Dr. Mário Antônio Meireles Teixeira
Universidade Federal do Maranhão
Orientador

Tiago Bonini Borchartt
Universidade Federal do Maranhão
Professor

Carlos Eduardo Portela Serra de Castro
Universidade Federal do Maranhão
Professor

São Luis
2019

Este trabalho é dedicado a toda a família, amigos e professores da Universidade Federal do Maranhão que de alguma forma contribuíram em minha graduação.

AGRADECIMENTOS

Agradeço primeiramente a Deus, por ter me permitido chegar a esse momento. Agradeço ao meu orientador que acreditou em mim e me deu total suporte para conclusão deste trabalho. Agradeço também a minha família e amigos por toda a ajuda, motivação e paciência que tiveram. Por fim, agradeço à uma pessoa que foi crucial nesta caminhada, Karla, sem ela esse momento não seria possível.

RESUMO

Simulação de sistemas é um método computacional que desempenha as funções matemáticas para simular o comportamento de vários tipos de operações, processos ou sistemas do mundo real. É um estudo que pode ser utilizado para analisar o comportamento do cérebro. Os meios de implementação de um simulador, porém, atualmente, dependem do conhecimento de uma área e sua utilização para aprendizado é restrita. Para estudantes, algumas ferramentas podem ser utilizadas facilmente. A linguagem Python oferece um ambiente para a simulação de eventos, simplificando a utilização, capaz de otimizar e simplificar a programação de simuladores, algo que facilita bastante o ensino do tema. O principal objetivo é contribuir para o aprendizado dos principais conceitos de desenvolvimento através da construção de um software que seja intuitivo e que demonstre como principal meta a funcionalidade do ambiente de simulação. Para a validação do software será implementado um ambiente web com utilização de métodos “arraste e solte” que consiga montar modelos genéricos de simuladores utilizando o SimPy.

Palavras-chave: simulação, python, simpy, ambiente visual.

ABSTRACT

System simulation is a computational method that performs mathematical functions to simulate the behavior of various types of real-world operations, processes, or systems. It is a study that can be used to analyze the behavior of the brain. For Schriber (1974) "Simulation implies the modeling of a process or system, so that the model imitates the responses of the real system in a succession of events that occur over time." The means of implementing a simulator, however, currently depends on the knowledge of an area and its use for learning is restricted. For students, some tools can be used easily. The Python language provides an environment for simulating events, simplifying the use, able to optimize and simplify the programming of simulators, something that greatly facilitates teaching the theme. The main objective is to contribute to the learning of the main concepts of development through the construction of software that is intuitive and that demonstrates as main the functionality of the simulation environment. For the validation of the software was implemented a webspace using methods of "drag and drop" that can assemble generic models of simulators using SimPy.

Keywords: simulation. python. simpy, visual tool.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

AOL	America Online
CSS	Cascading Style Sheets
HTTP	Hypertext Transfer Protocol
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
UFMA	Universidade Federal do Maranhão
WEB	World Wide Web

SUMÁRIO

1 INTRODUÇÃO

Ambientes de desenvolvimento de modelos de simulação voltados a eventos discretos são ferramentas computacionais com ampla variedade de utilização e são facilitadores no processo de análise de problemas que envolvem logística, padrões de comportamento, avaliação de desempenho e vários outros tipos de eventos que podem ser discretizados. Foram criados com o intuito de gerar modelos que simulem eventos com base em dados coletados para que, no mundo real, com base nos resultados obtidos, possam ser criados ou alterados eventos ou ambientes nos quais o modelo se baseia.

Por se tratar de uma área da computação com teor bastante técnico, ainda é muito difícil a implementação de modelos mais complexos e também requer bastante conhecimento prévio de programação e teorias matemáticas. Isso é um dos principais fatores que limitam a difusão desse tipo de conhecimento e tornam bastante custosa a criação de modelos de simulação.

No mercado, há mais de sessenta simuladores comerciais, de acordo com uma pesquisa do Instituto de Pesquisa Operacional e Ciências da Administração - INFORMS. Há também ambientes não comerciais, implementados pela comunidade acadêmica para uso livre, principalmente voltados para o aprendizado. Mesmo com a disponibilidade de ambientes e bibliotecas voltados à simulação de eventos discretos, o aprendizado nessa área é dificultado pela falta de interfaces gráficas que tornem mais intuitivo o aprendizado e que possibilitem o usuário criar modelos sem a necessidade de conhecimentos de programação, facilitando assim o processo de modelagem. No âmbito educacional, ferramentas com esse tipo de interface mais amigável tornam menos complicado o ensino teórico e prático desse tema, melhorando o aprendizado tornando-o mais dinâmico.

Nesse contexto, surge a necessidade de criar ferramentas que auxiliem o aprendizado e que sejam mais intuitivas, fornecendo aos usuários, a capacidade de criar modelos funcionais de maneira fácil e rápida e que possam ser efetivamente utilizados por eles para o aprendizado.

Python vem ganhando força entre os programadores por ser uma linguagem de fácil compreensão, muito por conta de sua sintaxe bem definida, em que a indentação é o meio de criação de blocos de código. Isso ajuda a diminuir o uso de caracteres especiais como ->, ::, :=, , , etc. Nela podemos contar com a biblioteca SimPy para a simulação de eventos discretos que possui elementos muito poderosos e de fácil implementação e compreensão na hora de criar modelos.

Por ser bastante popular e por possuir esse ambiente, essa linguagem foi escolhida para a criação de uma ferramenta que torne seu uso simples e que auxilie no aprendizado. Podendo ser usada por docentes para o ensino de simulação de eventos discretos.

1.1 Motivação

Com o aumento da capacidade de utilização de ferramentas voltadas para o desenvolvimento de sistemas de modelagem de simulação de eventos. torna-se importante a difusão do conhecimento acerca dessa área da computação, pois cada vez mais faz necessário o uso de modelos que simulem eventos do mundo real, para agilizar e reduzir possíveis erros e até criar novas perspectivas no que tange o desenvolvimento de um projeto, além de economizar insumos na produção do mesmo.

Muitos tipos de projetos podem ser contemplados com a difusão desse tipo de conhecimento, visto que ao simular os eventos, pode-se dimensionar o uso e fazer alterações importantes no decorrer do desenvolvimento. Como no caso da implantação de novas rotas de trânsito, organização do atendimento de vários tipos de estabelecimentos, tais como bancos, lojas e até da análise de investimentos no mercado financeiro.

Tendo em vista a necessidade de criar ferramentas que auxiliem o aprendizado dessa área, que carece de maneiras simplificadas de meio de utilização, aliada à popularidade da linguagem Python e da disponibilidade do ambiente propício à essa criação, surgiu a oportunidade de implementação de um facilitador para esse fim.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo deste trabalho é desenvolver uma ferramenta de modelagem de simulação de eventos discretos que utilize o ambiente de simulação da linguagem Python, o SimPy, e possua uma interface intuitiva utilizando uma página da web.

1.2.2 Objetivos Específicos

- Estudar técnicas de simulação.
- Criar uma interface amigável e intuitiva para usuário menos experientes.

- Abstrair a necessidade de programação na criação de modelos de eventos discretos..

1.3 Organização do Trabalho

O capítulo 2 apresenta o conceito de simulação de eventos discretos, Python e a biblioteca SimPy.

O capítulo 3 explica o conceito de aplicações web, também aborda as linguagens HTML, javascript e as ferramentas principais utilizadas para o desenvolvimento da interface da aplicação.

O capítulo 4 demonstra a arquitetura utilizada no desenvolvimento do protótipo.

O capítulo 5 mostra casos de teste realizados para a validação da ferramenta.

O capítulo 6 possui a conclusão do trabalho.

2 SIMULAÇÃO DE EVENTOS DISCRETOS

O termo “simulação” possui muitos significados, dentre eles, segundo o dicionário Michaelis, podemos destacar:

1 Ação ou efeito de simular.

2 Reprodução do funcionamento de um processo através de funcionamento de outro.

3 Falta de correspondência com a verdade; disfarce, dissimulação, fingimento.

4 Caráter do que é hipócrita, falso, impostor.

Para este trabalho, iremos além do sentido apenas da palavra em si, conheceremos significados mais técnicos que expliquem o sentido de simular eventos discretos.

Como pudemos ver anteriormente, a simulação se divide basicamente em dois tipos: simulação computacional e não-computacional. Para nós, a computacional é a mais importante. Este tipo refere-se à qualquer simulação feita com a utilização de um computador, e é criada com o objetivo de gerar uma estimativa sobre o comportamento de determinado evento, estrutura ou processo baseado nos dados de utilização inseridos no modelo durante o teste. Vale ressaltar que há uma diferença entre emulação e simulação. já que a primeira diz respeito a, em termos gerais, imitar o funcionamento de um dispositivo ou sistema em cada aspecto de seu funcionamento, para que esse novo dispositivo possa realizar os mesmos processos em uma arquitetura diferente, já na simulação, há a preocupação principalmente com os resultados obtidos pelo dispositivo ou sistema real, discretizando os processos e eventos e focando-se em reproduzir suas entradas e saídas para que os resultados possam corresponder aos do objeto real.

Para entendermos o funcionamento de um modelo de simulação, devemos reproduzir os resultados de cada evento, para que possa ser feita uma avaliação fiel de todos os aspectos que possam influenciar no comportamento do mesmo. Como são eventos discretos, o importante são as entradas e resultados obtidos, logo, não é necessário preocupar-se com o contexto específico do evento. Para exemplificar como um simulador de evento discretos funciona, podemos utilizar o exemplo de um posto de gasolina. O elemento carro é gerado para que possa ser inserido na fila de atendimento da bomba de combustível, esse atendimento é definido como um processo, que demora um tempo definido pelo programador e mantém uma fila, após esse atendimento o carro é liberado e se encaminha para a saída do posto. Esse

processo, desde a chegada do carro até sua saída, pode ser modelado e modificado para que sejam inseridos outros elementos, como uma nova bomba de combustível, uma parada para calibragem de pneus ou ir ao banheiro. A figura 1 mostra como ocorre esse processo.

Figura 1 – Bomba de combustível simples.

A figura 2 demonstra uma expansão do exemplo da bomba com mais elementos adicionados ao processo.

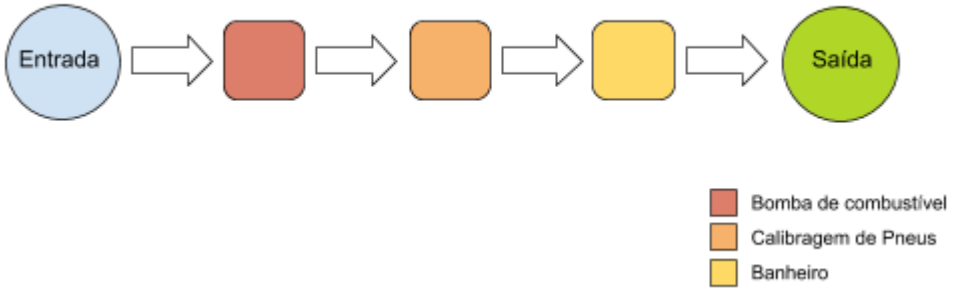

Figura 2 – Posto de combustível.

2.1 Python

Python é uma linguagem livre, de alto nível, orientada a objeto, interpretada, dinâmica, robusta, multiplataforma que foi lançada em 1991 por Guido van Rossum, podendo ser usada para programar para diversas plataformas. Destaca-se pela simplicidade e redução no tamanho dos códigos criados, aproximando-se mais da linguagem natural e facilitando seu entendimento por parte do programador. Seu nome deriva-se da série de televisão britânica Monty Python's Flying Circus. O nome foi escolhido por seu criado na tentativa de fugir à nomeação de linguagens de programação com às tradicionais siglas ou letras, como B, C, ABC.

A linguagem possui um visual agradável, fácil de ler e bem organizada visualmente. Ao contrário de muitas linguagens, utiliza palavras em vez de pontuações, códigos e símbolos especiais para definir funções e a separação do código é feita

por meio de espaços em branco e indentação, contrapondo os delimitadores visuais, como chaves e palavras, por esse motivo, a indentação é obrigatória para o bom funcionamento do código. Atualmente possui muitas bibliotecas implementadas com as mais diversas funcionalidades, uma delas é a SimPy, que é uma das bases para este trabalho.

2.2 SimPy

O SimPy é um ambiente de simulação de eventos discretos baseado em processos, faz parte da linguagem de programação Python. Possui vários recursos para modelar pontos de congestionamento como contadores de verificação, servidores e túneis. Pode ser usado também na modelagem de componentes ativos como clientes, veículos e agentes. Simulações contínuas são teoricamente possíveis no SimPy, porém ele não em recursos que ajudem nesse tipo de processo. Em contrapartida, simulações com tempo fixo em que seus processos não interagem entre si ou com recursos compartilhados são bastante favorecidas e encontram muitas facilidades de implementação.

Baseado originalmente em Simscript e Simula, usa recurso de geradores do Python, implementando eficientemente co-rotinas. Combina dois pacotes anteriores, SiPy (Simula) e SimPy (Simscript). Na sua versão 3, o SimPy apresentou uma API completamente diferente e fácil de usar, além de tornar-se mais flexível e extensível, porém ainda dependendo dos geradores do Python, pois estes se mostraram bastante eficientes. Para essa nova versão, algumas das mudanças mais importantes foram: maior foco nos eventos, podendo combiná-los para criar eventos condicionais, o processo pode ser definido por qualquer função geradora, suíte de base de testes aprimorada e o ambiente teve sua documentação completamente revisada. Em suma, Esta será a ferramenta principal na geração dos modelos criados pelos usuários, pois suas funções são a base da implementação deste trabalho.

Podemos exemplificar o uso do SimPy através da modelagem do atendimento dos caixas de um supermercado. O cliente, ao escolher os produtos que irá comprar, dirige-se a um dos caixas, ao chegar ao caixa inicia-se o processo de atendimento, que leva um tempo determinado previamente ou uma variação de tempo, no nosso caso, iremos utilizar o tempo fixo, no final, o cliente dirige-se à saída do supermercado. O fluxo dessa modelagem é mostrado na Figura 3:

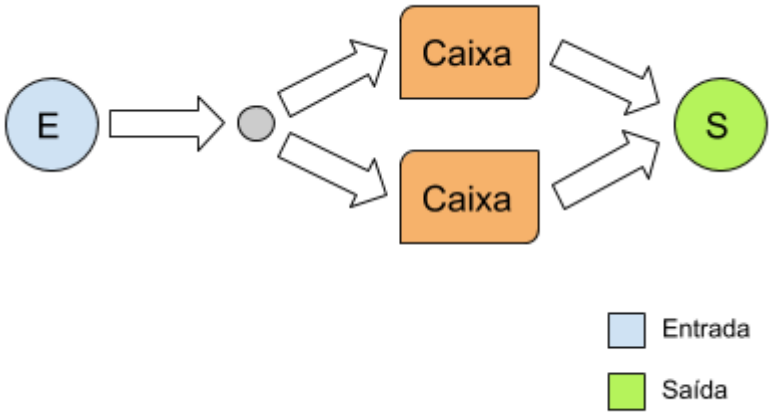

Figura 3 – Atendimento de caixas de Supermercado.

3 APLICAÇÕES WEB

Com o crescimento acelerado da velocidade das conexões de internet e o aumento contínuo do acesso das pessoas à ela, podemos dizer que vivemos em um mundo altamente conectado. De acordo com o estudo Network Skills in Latin América que aponta um crescimento exponencial na oferta de vagas para as áreas de TI, podemos observar que a busca por profissionais nessa área é cada vez maior, assim como há um crescimento na quantidade de profissionais formados a cada ano. Quanto mais disponibilidade de programadores, maior é o desenvolvimento da tecnologia, técnicas e ferramentas que são utilizadas nesse ramo. Isso proporcionou um salto no desenvolvimento de tecnologias voltadas a web, como versões mais modernas de HTML, ferramentas desenvolvidas para javascript e CSS. O aumento no número de usuários de dispositivos portáteis também é um dos principais responsáveis por isso. De acordo com um estudo do The Statistics Portal em 2014 a quantidade de usuários de smartphones no mundo era de 1,57 bilhões e estima-se que em 2020 a quantidade será de aproximadamente 2,87 bilhões, ou seja, quase o dobro de usuários em um curto período de tempo, como mostra a figura 4.

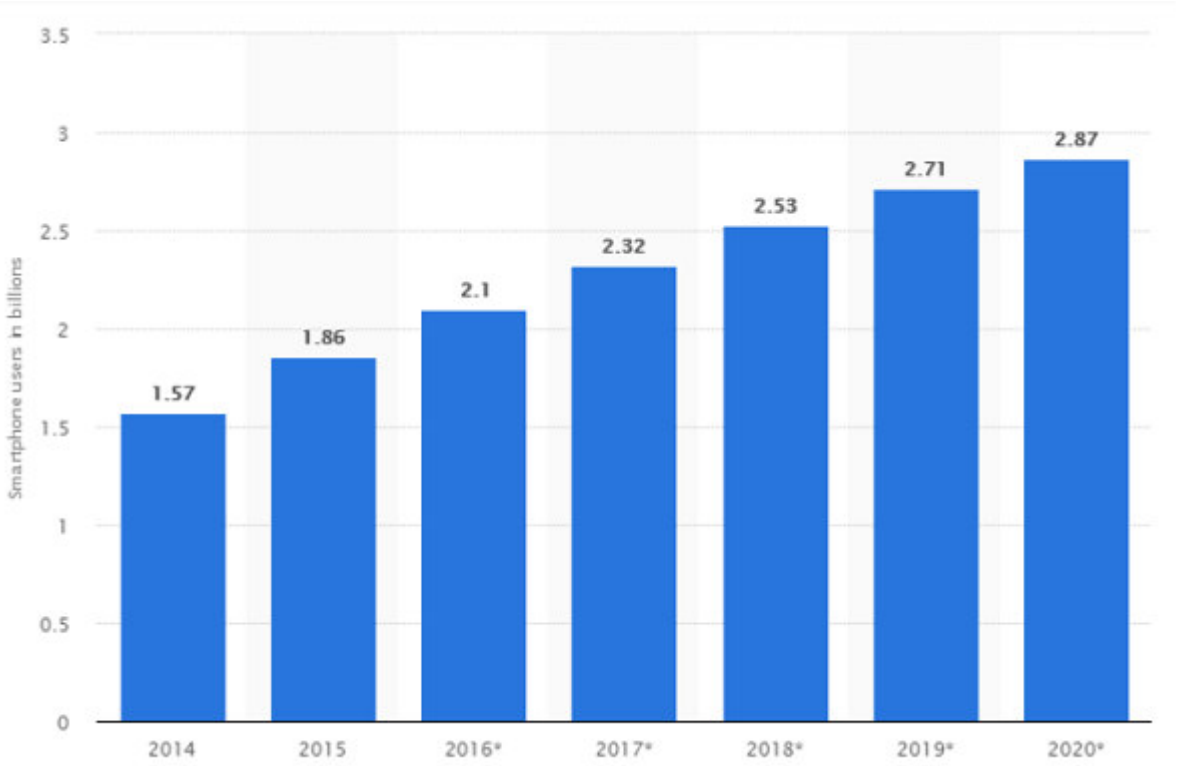

Figura 4 – Gráfico do usuários de smartphone pelo mundo (em bilhões).

Com isso, aplicações web se tornam cada vez mais comuns, seja pela facilidade de acesso ao conteúdo sem necessidade de instalação prévia ou pela mobilidade

e portabilidade inerentes a esse modelo de aplicação. Antigamente era mais complexo o desenvolvimento de sistemas pela necessidade da criação de uma camada de serviços que fizesse a integração com sistemas corporativos convencionais, pois cada empresa possuía protocolos e padrões próprios, não existindo um consenso. Nos últimos anos, houve uma mudança significativa deste cenário com a utilização de uma notação de objetos Javascript chamada JSON que integrou o tráfego de informações, com protocolos padronizados e meios de autenticação e autorização mais seguros, tornando-se confiável e aceito mundialmente.

É comum que alguns termos gerem dúvidas quando se trata de desenvolvimento Web mesmo entre as pessoas que convivem nesse ramo. Podemos destacar a dúvida entre website e aplicação Web, pois até programadores interpretam erroneamente tais conceitos. Website ou simplesmente site é o termo que designa uma página ou um conjunto de páginas da Web com os mais diversos tipos de informação e para diversos propósitos, como:

- Institucional: Sites feitos para empresas com intuito de exibir informação e gerar um meio de contato com seus clientes.
- Informações: Veículos de informação utilizam esse tipo de site para mostrar notícias ou dar algum tipo de informação como jornais, revistas e agências de notícias.
- Portais: São sites que possuem vários tipos de informação e conteúdo, geralmente criados por uma mesma empresa.
- Armazenagem de informação: São denominados sites de armazenamento os que funcionam como banco de dados, que catalogam registros de outros sites e possuem ferramentas de busca desse conteúdo.

Aplicação Web refere-se a um tipo de sistema que roda em um ambiente de internet. Nesse tipo de aplicações o usuário tem a possibilidade de realizar muitas tarefas que um simples site não permite. São capazes de gerar uma interação do usuário com o servidor, site ou mesmo outros usuários que frequentem o mesmo ambiente. O principal foco das aplicações Web é permitir um acesso rápido a ferramentas que resolvam vários tipos de problemas sem a necessidade de downloads e instalação de softwares, devido a facilidade de acesso às mesmas por estarem disponíveis 24 horas por dia na internet e poderem ser acessadas de qualquer lugar pelo usuário.

Podemos concluir que um site tem o objetivo principal de gerar informação e fornecer o contato de empresas e clientes de maneira rápida e eficiente. Já uma aplicação Web é uma ferramenta de automação de processos, tendo em vista que o usuário poderá acessá-la em qualquer lugar e a qualquer hora para resolver os problemas para os quais ela foi desenvolvida.

As aplicações Web possuem várias vantagens em relação à eficácia no processo de resolução de problemas, principalmente quando são utilizadas em ambientes corporativos, como no caso de uma empresa que possui diversos funcionários e precisa que eles utilizem um software para determinado trabalho. É muito mais efetivo implementar uma aplicação que funcione em um ambiente Web do que instalar o software em cada máquina, além da facilidade na alteração, comunicação e atualização dos serviços do mesmo. Também pode-se destacar a facilidade de utilizar essa mesma aplicação em outro lugar e a não necessidade de reinstalação da mesma em caso de substituição de máquinas na empresa.

Levando-se em consideração o aumento da acessibilidade a nível mundial e a evolução exponencial das tecnologias voltadas para o ambiente Web, além de observar todas essas facilidades, segurança e rapidez no que se refere a utilização de aplicações para esse ambiente, podemos concluir que é inegável a importância de se desenvolver cada vez mais ferramentas que sejam baseadas nesse formato para que possamos em um futuro próximo resolver problemas de diversos tipos com a maior facilidade e velocidade possível.

3.1 HTML

A linguagem de marcação de hipertexto ou HTML é considerada o principal idioma da World Wide Web e é derivada da junção entre os padrões HyTime e SGML que não são mais utilizados. Criada por Tim Berners Lee, a princípio foi projetada originalmente para descrever semanticamente documentos científicos e auxiliar a comunicação e disseminação de pesquisas entre ele e seus colegas. Porém seu design permitiu que fosse adaptada para descrever diversos outros tipos de documentos e aplicativos. Por esse motivo foi utilizada para formar a rede de comunicação de dados pública da época, o que veio a se tornar a internet que conhecemos. É uma linguagem baseada em marcações ou TAGs, há diversos tipos de TAGs que marcam diferentes elementos em um documento HTML, elas servem para separar e delimitar o espaço entre esses elementos. É uma linguagem bem flexível e de fácil aprendizado cujo objetivo é tornar a criação de páginas Web de maneira rápida e eficiente. Suas TAGs geram agrupamentos que organizam a estrutura do documento HTML, ao

utilizá-las, dizemos ao navegador o que significa cada informação que introduzimos, o que é um título, um formulário, um parágrafo, um botão, entre outros. Elas dizem também o que significa cada elemento para os mecanismos de busca. Para exibir o resultado de uma determinada busca, o mecanismo tem que saber diferenciar um título de um parágrafo, por exemplo e isso é feito através das TAGs. As principais TAGs estruturais são mostrados a seguir:

Tags estruturais:

<!– –>

Cria um comentário.

<html> </html>

Envolve todo um documento html.

<head> </head>

Envolve o cabeçalho de um documento html.

<meta>

Fornece informações gerais sobre o documento.

<style> </style>

Informações de estilo.

<script> </script>

Linguagem script.

<noscript> </noscript>

Conteúdo alternativo para quando a linguagem script não for suportada.

<title> </title>

O título do documento.

<body> </body>

Envolve o corpo (texto e tags) do documento html.

** **

Divide o conteúdo em seções.

3.1.1 HTML 5

Atualmente em sua quinta versão, é aceita por todos os navegadores da web e é utilizada no processamento, renderização e apresentação de conteúdo dispo-

nível online. Utilizada como base principal para o desenvolvimento da interface, possibilitando a apresentação da ferramenta ao usuário. Esta versão traz a evolução no padrão de definição da linguagem HTML. Além de ser a mais nova versão, com novos atributos, elementos e comportamentos, possui um conjunto maior de tecnologias, permitindo um avanço na criação e poder de processamento de aplicações e websites. Este conjunto recebeu o nome de HTML 5 & friends, abreviado e mais comumente conhecido como HTML 5.

A estrutura básica de um documento HTML é mostrada na figura 5:

```
<!DOCTYPE html>

<html lang="pt-br">
<head>
    <meta charset="utf-8">
    <title>Título da página</title>
</head>
<body>
    ... código HTML que compõe o site ...
</body>
</html>
```

Figura 5 – Código HTML

Entre as principais novidades dessa versão, podemos citar as novas estruturas e elementos de semântica, elementos de áudio e vídeo que incorporam e permitem a manipulação de conteúdos multimídia mais recentes e melhorias no formulários Web. Houveram também melhorias na conectividade através da implementação de Web Sockets, Eventos do servidor, que são eventos enviados pelo servidor para o cliente sem a necessidade do segundo enviar requisições, comunicação em tempo real através de WebRTC, eliminando a necessidade de plugins ou aplicações externas. Podemos destacar também uma melhoria no que diz respeito aos recursos de armazenamento e recursos offline, o que permite que aplicações e extensões percebam oscilações na conexão.

Houve melhorias significativas na performance e integração com linguagens de script, como o Javascript através de Web Workers que permitem a utilização do javascript como threads em segundo plano, o que permite que essas aplicações continuem rodando e não prejudiquem o funcionamento ou causem lentidão nas interações dos eventos. Motor JIT-compiling para JavaScript que faz parte da nova geração de motores JavaScript muito mais poderosos levando a uma melhoria na performance das aplicações. o History API permite a manipulação do histórico salvo no navegador, o que é bastante útil para sites que carregam informações interativas. A padronização do atributo *contentEditable* que permite que qualquer página possa se tornar uma página Wiki, ou seja, páginas que podem ser editadas coletivamente,

TAG	DESCRIÇÃO
<canvas>	Usado para desenhar gráficos, em tempo real, através de scripting (geralmente JavaScript).
<audio>	Áudio no site.
<video>	Vídeo no site.
<source>	Define vários recursos de mídia para <video>e <audio>.
<embed>	Define um container para uma aplicação externa de conteúdo interativo (um plug-in).
<track>	Define faixas de texto para <video>e <audio>.
<datalist>	Especifica uma lista de opções pré-definidas para controles de entrada.
<keygen>	Define um campo gerador de par de chaves (para formulários).
<output>	Define o resultado de um cálculo.
<article>	Define um artigo (notícia, post etc.).
<aside>	Define o conteúdo, além do conteúdo da página.
<bdi>	Isola uma parte do texto que pode ser formatado em uma direção diferente de outro texto fora dela.
<command>	Define um botão de comando que um usuário pode invocar.
<details>	Define detalhes adicionais que o usuário pode exibir ou ocultar.
<dialog>	Define uma caixa de diálogo ou janela.
<summary>	Define um título visível de um elemento <details >.

além da inclusão da TAG canvas que permite a criação de desenhos e gráficos na página.

Algumas das principais TAGS implementadas no HTML 5 são:

Na parte de acesso aos dispositivos, houveram diversas melhorias, como a manipulação das câmeras dos computadores e dispositivos móveis. Possibilidade de eventos touchscreen, utilização da geolocalização, detecção na orientação do dispositivo, o que permite que o navegador detecte a mudança na orientação do dispositivo e se adeque a isso, uma melhoria muito bem vinda por causa do aumento significativo do número de usuários em dispositivos móveis. Pointer Lock API permite que o cursor seja limitado pelas medidas da aplicação que estiver sendo utilizada, muito útil em jogos para que o cursor não saia da área desejada e não gere uma experiência ruim ao usuário.

Para aplicações Web, os principais recursos dessa nova versão são os Recursos e eventos offline, armazenamento persistente de dados das aplicações no lado do cliente, o atributo contentEditable e a utilização de arquivos de aplicações web, o que inclui o suporte para selecionar vários arquivos usando um elemento chamado Input.

Com a chegada dessa nova versão podemos perceber que temos muito a

ganhar, pois podemos contar com diversos novos recursos que trazem novas funcionalidades, uma maior estruturação no conteúdo, suporte nativo a conteúdos multimídia e facilidade na implementação de aplicações, além de reduzir a necessidade de utilizar bibliotecas e plugins externos que geram muitos problemas ou serem acessados em diferentes navegadores.

3.2 JavaScript

Atualmente Javascript é a principal ferramenta utilizada no desenvolvimento dos mais diversos tipos de páginas web, sejam aplicações ou simples páginas informativas. Em conjunto com o HTML e o CSS, formam os pilares da Web. Praticamente toda página que você acessar terá algum tipo de conteúdo JavaScript, ou seja, ele está presente no nosso cotidiano, mesmo que não notemos isso, seja em uma notícia de algum portal ou em um post em uma rede social.

O JavaScript foi desenvolvido por Brendan Eich, que nasceu em 1961 nos Estados Unidos. Originou-se a partir do projeto da linguagem ScriptEase, quando o mesmo foi adquirido pela Netscape, batizando-a com o nome de Mocha. A linguagem mudaria mais uma vez de nome, no ano de 1995, antes de seu lançamento, foi rebatizada para LiveScript, que começou a fazer parte do Netscape a partir da versão 2.0, lançada ainda em 1995. Ainda em 1995, é lançada a linguagem Java* pela Sun Microsystems, parceira da Netscape. Devido ao sucesso do Java e por meio da parceria entre as duas empresas, ao entrar para a família Netscape, a linguagem recebeu o nome de Javascript.

Brendan Eich iniciou sua carreira na Silicon Graphics, onde trabalhou por 7 anos em sistemas operacionais e sistemas de rede. Após isso, trabalhou 3 anos na MicroUnity Systems Engineering, porém Eich ficou realmente conhecido por seu trabalho na Netscape e na Mozilla. Na primeira começou a trabalhar em 1995, justamente no JavaScript. Depois foi um dos fundadores da Mozilla.org em 1998, atuando como arquiteto chefe. Quando a empresa AOL finalizou o suporte para o navegador Netscape, ele ajudou a alavancar a Mozilla Foundation.

A linguagem tem um papel de fundamental importância para o processo de evolução da Web. Sendo uma linguagem aberta, de fácil aprendizado e livre do controle e manipulação de organizações comerciais, o JavaScript possui as características essenciais que vão de acordo com a ideia central da Internet que é ser aberta e livre para todos. Sua evolução foi fundamental para a eliminação de plugins e ferramentas proprietárias que atrapalhavam o desenvolvimento de muitas

aplicações e que possuíam performance ruim. Com a padronização do uso dessa linguagem, houve uma corrida pela utilização mais eficiente da mesma, fazendo com que os programadores buscassem maneiras de melhorar e adequar seus trabalhos em volta do JavaScript, trazendo benefícios tanto para desenvolvedores que tinham mais ferramentas baseadas na linguagem para trabalhar quanto para usuários que recebiam aplicações mais poderosas e com performance superior. As etapas da nomeação da linguagem ocorreram como mostra a figura 6.

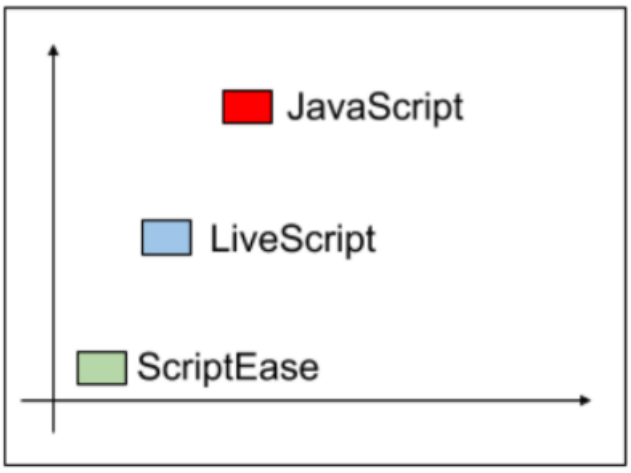

Figura 6 – Etapas da nomeação do JavaScript.

3.2.1 Características

É inegável que o Javascript é uma das linguagens mais fáceis de aprender e rápidas de implementar, apresenta uma performance excelente nos navegadores atuais e é a principal linguagem de script da Web. A facilidade de testar a linguagem de forma rápida contribui para que o desenvolvedor visualize o resultado de seu código à medida que vai implementando. Entre as principais características da linguagem, podemos destacar: Código fonte incluído no HTML

- Linguagem Interpretada
- Baseada em Objeto
- Sintaxe parecida com C,C++ e Java
- Programação dirigida por eventos
- Independente de plataforma

3.3 JSON

JSON (JavaScript Object Notation - Notação de Objetos JavaScript) é definida como um meio de formatação de troca de informações entre linguagens de programação. Esta notação é baseada na linguagem Javascript, mais precisamente no subconjunto Standard ECMA-262, que é bastante usada em diversas implementações de tecnologias da internet. Um dos principais atrativos do JSON para ser usado na troca de informações é o fato de ser escrito em formato de texto e de não depender de nenhuma linguagem específica, pois utiliza convenções que são reconhecidas por diversas linguagens baseadas em C e familiares, como C++, C, Java, JavaScript, Perl, Python e diversas outras. Esta linguagem é apresentada baseando-se em duas estruturas, object e array. A primeira é um conjunto desordenado de pares nome/valor. Sua estrutura se dá na forma ilustrada na Figura 7.

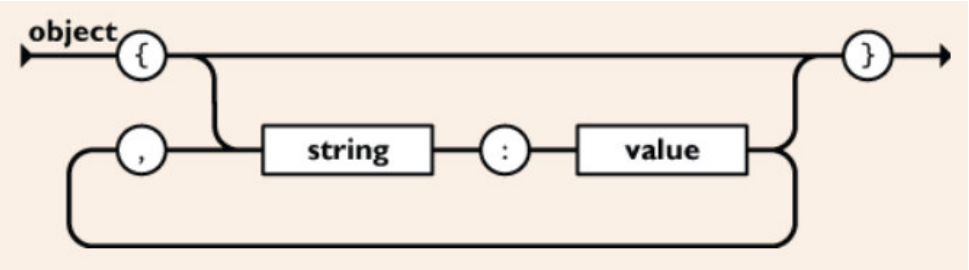

Figura 7 – Object

Já a segunda, o array, é definido como um conjunto ordenado de valores separados por vírgula, como mostra a figura 8.

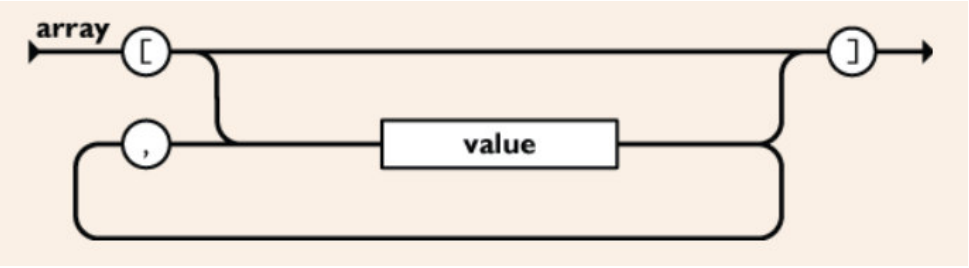

Figura 8 – Array

4 AMBIENTE VISUAL DE SIMULAÇÃO

Composta por quatro tecnologias que trabalham em conjunto, o ambiente tem como principal linguagem de programação o Javascript, que compõe a maior parte das funções fundamentais e controla o fluxo das informações que são enviadas à linguagem intermediária, o JSON. Na linguagem Javascript, foi utilizada uma biblioteca chamada GO.js, um ambiente criado para o desenvolvimento de aplicações que necessitem de diagramas e gráficos de fluxo de dados, como exemplificado na figura 9.

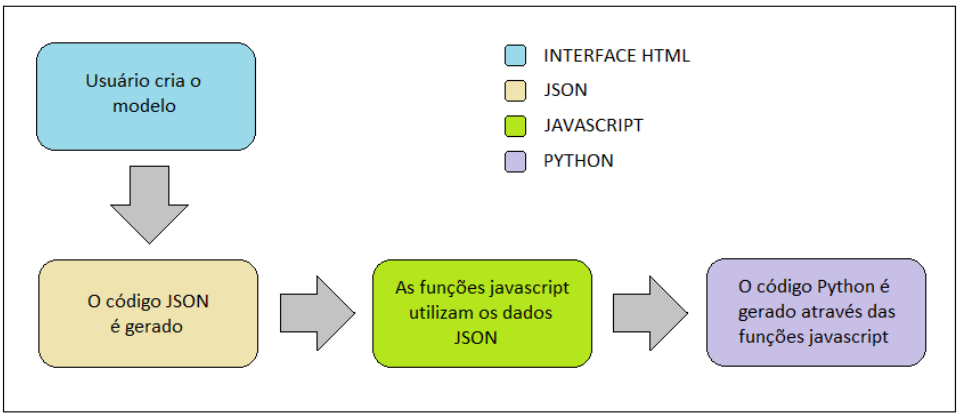

Figura 9 – Modelo da implementação do ambiente.

A interface é simples e os ícones representam as funções necessárias para criar os principais modelos, como mostra a figura 10.

Figura 10 – Ícones

Tomando como base um de seus exemplos, chamado Logic Circuit (Circuito Lógico) que demonstra uma aplicação do fluxo de um circuito e de todas as portas lógicas correspondentes. Este exemplo foi escolhido por ter similaridades no que diz respeito ao fluxo dos dados e de algumas de suas portas lógicas, porém, diversas alterações foram feitas para que correspondesse aos requisitos apresentados. O código-fonte é apresentado no arquivo js/scripts.js. As principais funções serão apresentadas a seguir:

A primeira delas é a função “pythonCreator()” que trata da alteração do documento python à partir dos dados recebidos do documento JSON. Ao ser iniciada, uma variável recebe os dados da área de texto “mySavedModel” presente no documento HTML, logo após, são obtidos e alterados os dados referentes ao tempo de simulação, que por definição são 10 (dez) unidades de tempo. Em seguida, os recursos necessários para a criação das filas são inseridos no arquivo python. Através do primeiro laço (linha 270) são criados os recursos necessários para o funcionamento dos processos, além da chamada dos processos de entrada (Input) presentes no modelo criado pelo usuário, como mostra a listagem 1.

```
268
269         /* Creating resources */
270         for (var i = 0; i < jsonList.nodeDataArray.length; i++ ){
271
272           var nodeKey = jsonList.nodeDataArray[i].key;
273           nodeKey = nodeKey-2*nodeKey;
274
275           if (jsonList.nodeDataArray[i].category == "Queue"){
276
277             edited.splice(reqLine,0,`req${nodeKey} = simpy.Resource(env, capacity = 1)`);
278             reqLine++;
279
280           }
281
282           if (jsonList.nodeDataArray[i].category == "Input"){
283
284             edited.splice(inputLine+reqLine-22,0,`env.process(arrivals_${nodeKey}(env))`);
285             inputLine++;
286
287           }
288         }
289
```

Listagem 1 - Criando Recursos

Logo após são inseridos no arquivo os nós criados ou alterados pelo usuário, bem como os valores presentes nos mesmos através dos laços presentes entre as linhas 292 e 398, como se pode notar na listagem 2.

```
/* Creating processes */
 // Input
for (var i = 0; i < jsonList.nodeDataArray.length; i++ ){

  var nodeVal1 = jsonList.nodeDataArray[i].value1;
  var nodeVal2 = 100-nodeVal1;
  nodeKey = jsonList.nodeDataArray[i].key;
  nodeName = nodeKey-2*nodeKey;


  if (jsonList.nodeDataArray[i].category == "Input"){

    edited.splice(line,0,`def arrivals_${nodeName}(env):`);
    edited.splice(line+1,0,"");
    edited.splice(line+2,0,"  global clients");
    edited.splice(line+3,0,"");
    edited.splice(line+4,0,"  while True:");
    edited.splice(line+5,0,"     # Arrivals");
    edited.splice(line+6,0,`    yield env.timeout(${nodeVal1}) # Time between arrivals`);
    edited.splice(line+7,0,"    clients += 1");
    edited.splice(line+8,0,`    print('client', clients, 'arrives in Input (${nodeName}) in', env.now, '!')`);
    edited.splice(line+9,0,"     # Next node");
    console.log("input line1: ", line);
    line = findNextLeftNode(jsonList, edited, nodeKey, line+10);
    console.log("input line2: ", line);
  }
}

line+=2;
```

[...]

```
//Output
for (var i = 0; i < jsonList.nodeDataArray.length; i++ ){

  var nodeVal1 = jsonList.nodeDataArray[i].value1;
  var nodeVal2 = 100-nodeVal1;
  nodeKey = jsonList.nodeDataArray[i].key;
  nodeName = nodeKey-2*nodeKey;

  if (jsonList.nodeDataArray[i].category == "Output"){
    console.log("Output line: ", line);
    edited.splice(line,0,`def output_${nodeName}(clients):`);
    edited.splice(line+1,0,"");
    edited.splice(line+2,0,"  global exits");
    edited.splice(line+3,0,"");
    edited.splice(line+4,0,"  print('client', clients, 'leaves system in', env.now, '!')");
    edited.splice(line+5,0,"  exits +=1");
    edited.splice(line+6,0,"");

  }

}
```

Listagem 2 - Criando Processos

Por fim, as alterações feitas são inseridas na área de texto pythonarchive que apresenta o código-fonte Python, o objetivo principal do software. A segunda e a terceira funções são praticamente idênticas e servem para realizar a busca dos nós a serem inseridos na saída de cada nó do tipo Fork, pois este apresenta duas saídas distintas, diferentemente dos outros que apresentam apenas uma ou nenhuma saída. Elas são denominadas findNextLeftNode() e findNextRightNode(), que realizam a busca dos nós referentes às saídas à esquerda e à direita do Fork, respectivamente. Esse formato de código permite a fácil alteração para a mudança nas funções do python para atender a quaisquer demandas de filas, seja com a implementação de novas funções de chegada, filas com capacidade de atendimento maior que um cliente por vez, até a mudança nas métricas utilizadas, pois cada sessão está em uma área específica e que não precisa de alterações em outras para realizar suas mudanças.

5 CASOS DE TESTE

Neste capítulo veremos alguns casos de teste que demonstram o funcionamento do software, foram implementados exemplos diversos, desde filas simples, a filas com feedback que é o processo no qual o elemento que sai de um nó entra em outro pelo qual ele já passou, gerando assim um laço no qual ele poderá ficar infinitamente ou sair, dependendo do modo como foi implementada a sequência de nós.

5.1 Caso 1 - Fila simples

Este exemplo demonstra um sistema formado por uma fila simples, com apenas uma fonte geradora de elementos que passarão pelo sistema e uma saída, como podemos ver na figura 11.

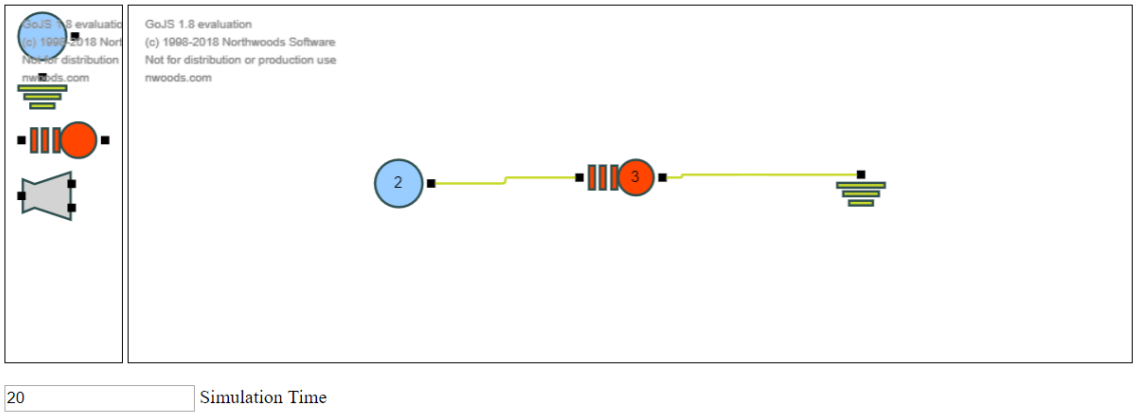

Figura 11 – Fila simples (Interface).

A figura 12, mostra o log de eventos resultante da execução do código python gerado. A versão do Python utilizada no exemplo é a 3.6.4 e parte do log foi suprimido intencionalmente para facilitar a visualização.

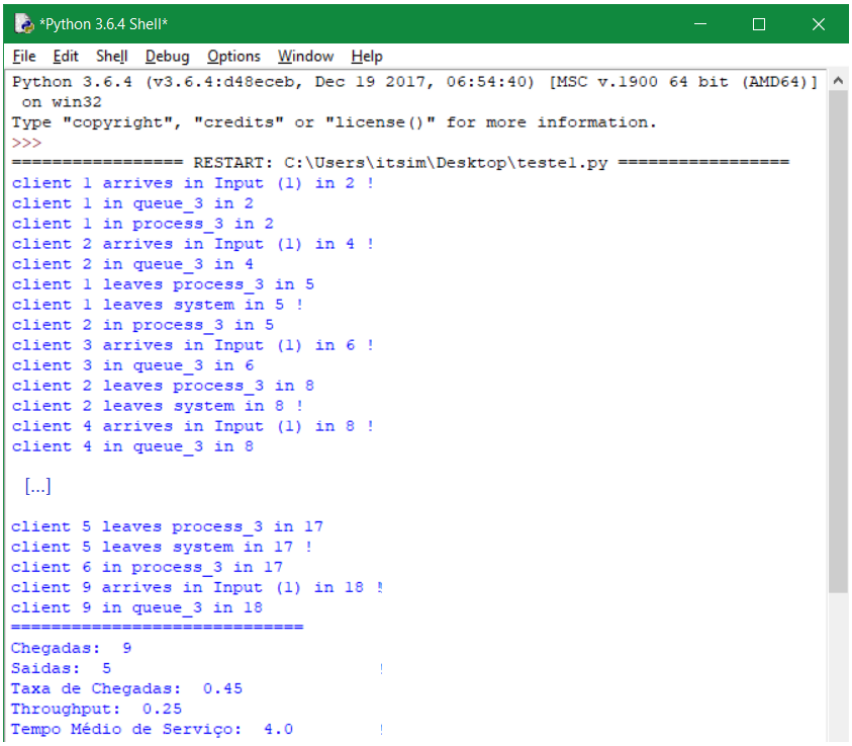

Figura 12 – Fila simples (Log do arquivo python).

A seguir, podemos observar através dos gráficos que o sistema não se encontra em equilíbrio, pois a taxa de chegadas é maior que a taxa de saída, isso ocorre porque o tempo entre as chegadas é menor que o tempo de execução do processo na fila, como podemos observar na Figura 14.

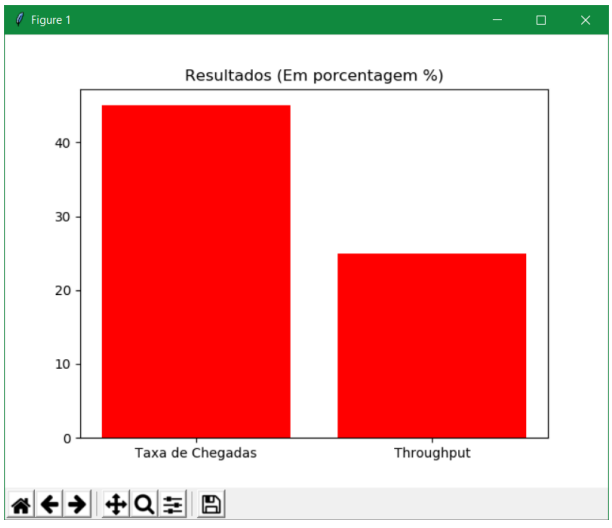

Figura 13 – Fila simples (Gráfico de Taxa de Chegadas e Throughput).

O tempo médio de serviço é obtido ao dividirmos o tempo total de simulação pelo pela quantidade de clientes atendidos, neste caso, 4 unidades de tempo. Na Listagem 3, a seguir, um trecho do código python gerado pelo ambiente de simulação, demonstrando a fila e a saída:

```
# Queue List
def queue_3(env, clients):

    print('client', clients, 'in queue_3 in', env.now)
    req = req3.request()
    yield req
    print('client', clients, 'in process_3 in', env.now)
    yield env.timeout(3) # Process time
    req3.release(req)
    print('client', clients, 'leaves process_3 in', env.now)
     # Next node
    output_2(clients)

# Fork List

# Output List
def output_2(clients):

  global exits

  print('client', clients, 'leaves system in', env.now, '!')
  exits +=1


# ------------------------------------------------ #

# Create environment and start processes
env = simpy.Environment()

# Resources
req3 = simpy.Resource(env, capacity = 1)
```

Listagem 3 - Fila simples, código Python.

5.2 Caso 2 - Fila dupla

Neste caso, veremos a demonstração de um sistema composto por duas filas em sequência, com entrada e saída únicas. A interface é mostrada na figura 14.

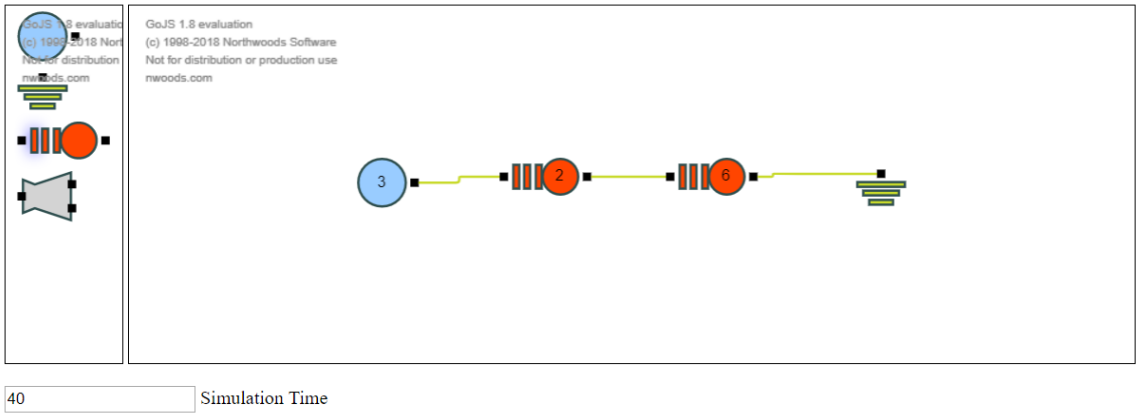

Figura 14 – Fila dupla (Interface).

Na figura 15, podemos observar o log da execução do código Python.

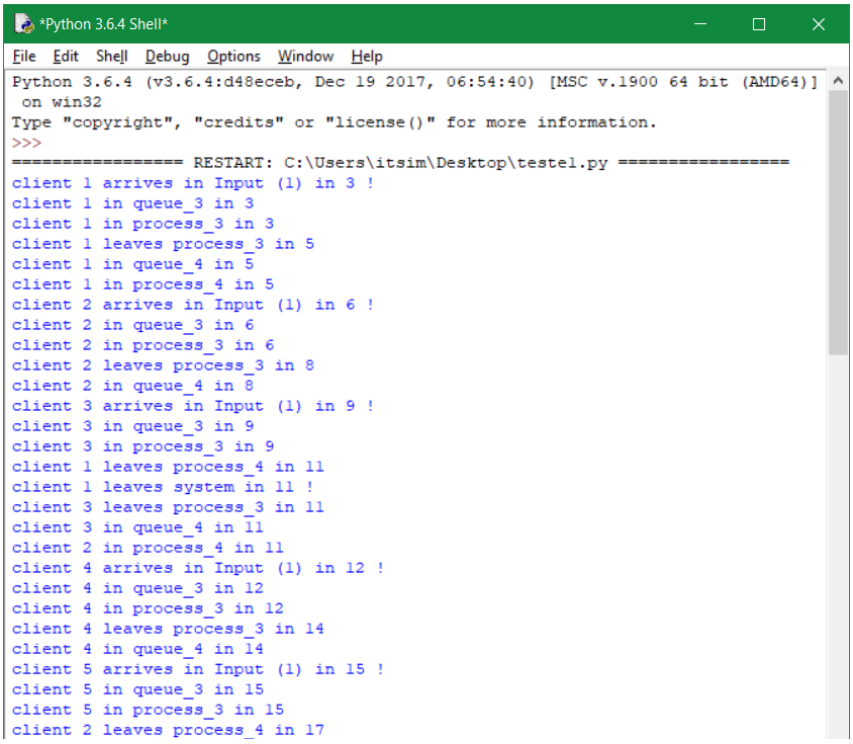

Figura 15 – Fila dupla (Log do arquivo python).

Podemos observar no gráfico da figura 16 que para este exemplo o sistema também não está em equilíbrio, pois as filas continuam com um tempo maior que a chegada dos clientes.

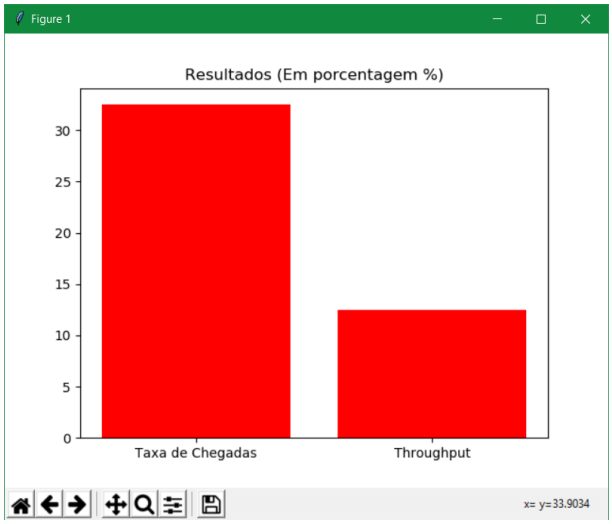

Figura 16 – Fila dupla (Gráfico de Taxa de Chegadas e Throughput).

A seguir, na listagem 4, um trecho do código gerado em Python, mostrando algumas métricas e saídas dos dados.

```
# ------------------------------------------------- #

 # Create environment and start processes
env = simpy.Environment()

 # Resources
req3 = simpy.Resource(env, capacity = 1)
req4 = simpy.Resource(env, capacity = 1)


 # Input
env.process(arrivals_1(env))


 # Running simulation
env.run(until = sim_time)

print("============================")
print("Chegadas: ", clients)
print("Saidas: ", exits)
print("Taxa de Chegadas: ", clients/sim_time)
print("Throughput: ", exits/sim_time)
print("Tempo Médio de Serviço: ", sim_time/exits)

plt.title("Resultados (Em porcentagem %)")
tags = ['Taxa de Chegadas', 'Throughput']
porcentagens = [clients/sim_time*100, exits/sim_time*100]
plt.bar(tags, porcentagens, color='red')
plt.show()
```

Listagem 4 - Fila dupla, código Python.

5.3 Caso 3 - Fila dupla com desvio (Fork)

Este exemplo da figura 17 mostra como o sistema se comporta ao adicionar-se um desvio em uma sequência de filas, além de utilizar duas saídas para os elementos que passam pelo mesmo.

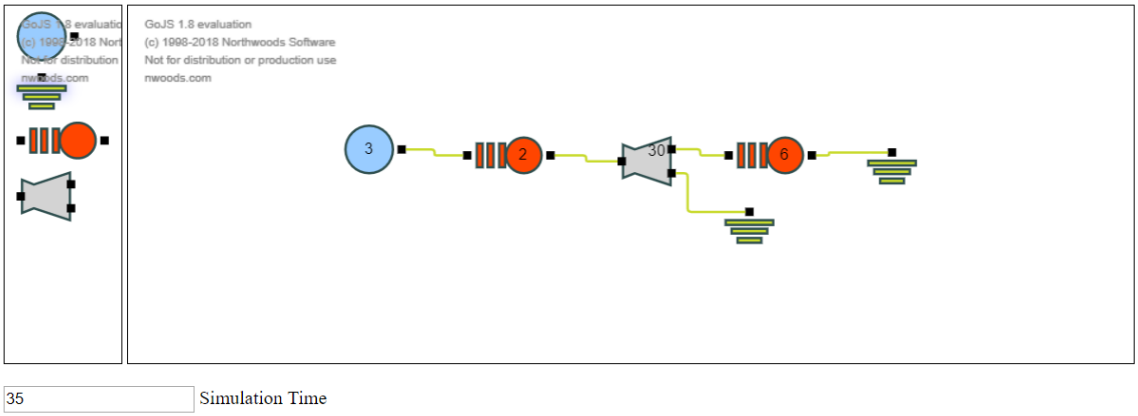

Figura 17 – Fila dupla com desvio (Interface).

No log da figura 18, podemos observar um dos maiores tempos médios de serviço observados nos exemplos deste trabalho.

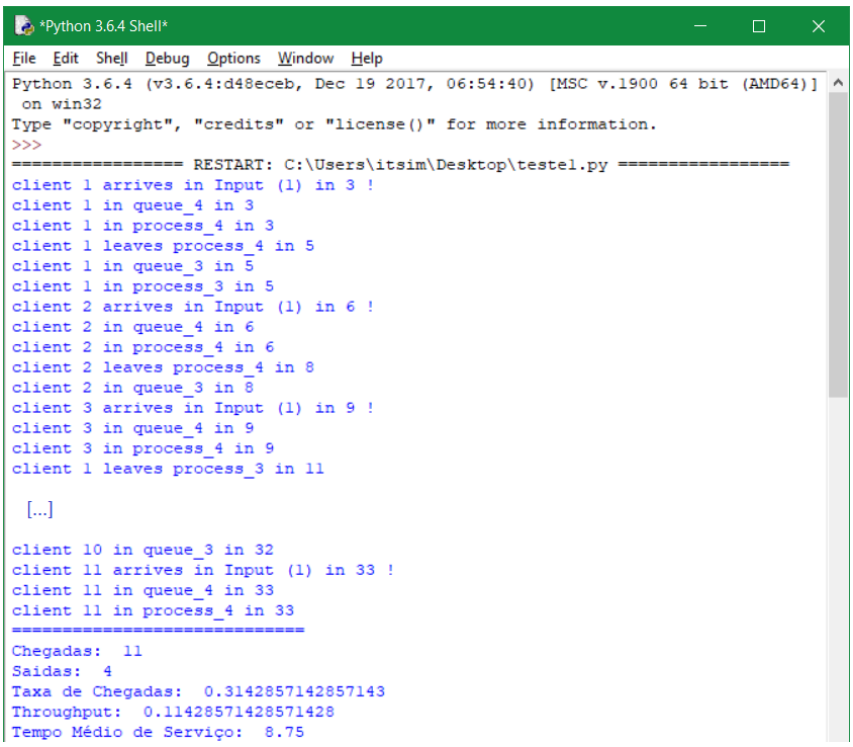

Figura 18 – Fila dupla com desvio (Log do arquivo python).

O gráfico da figura 19, demonstra os resultados obtidos nesse teste.

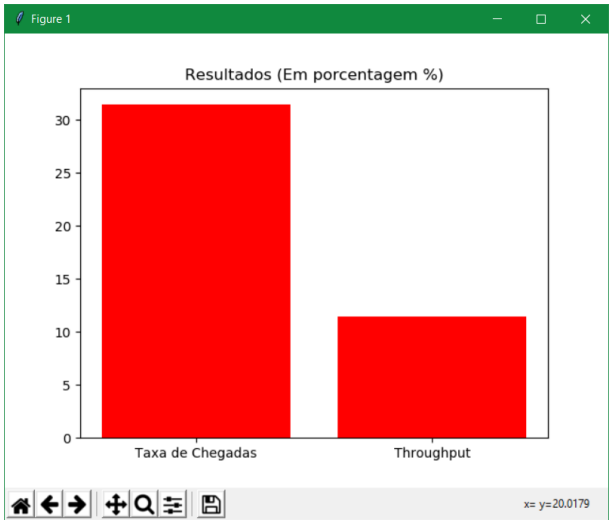

Figura 19 – Fila dupla com desvio (Gráfico de Taxa de Chegadas e Throughput).

5.4 Caso 4 - Feedback

Este caso demonstra um exemplo de feedback, onde um elemento que sai da fila pode retornar à ela mesma ou ir para um novo nó do sistema, como mostra a interface na figura 20.

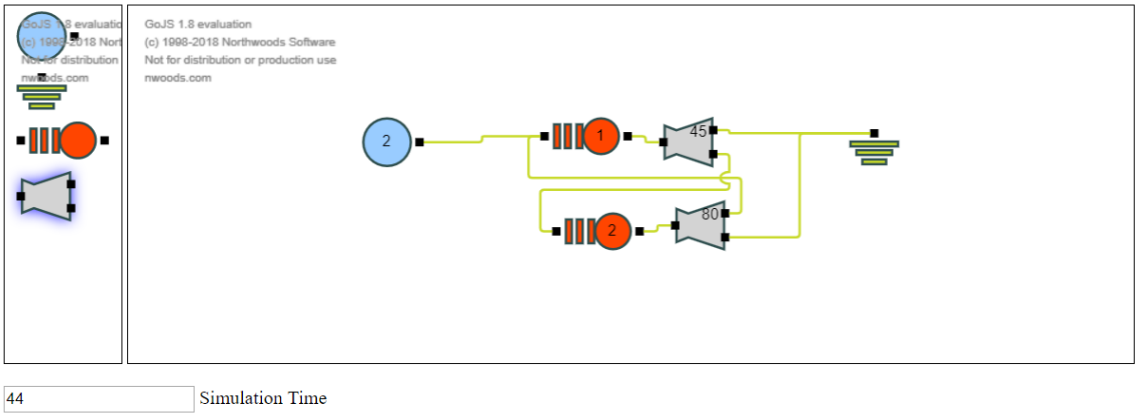

Figura 20 – Feedback (Interface).

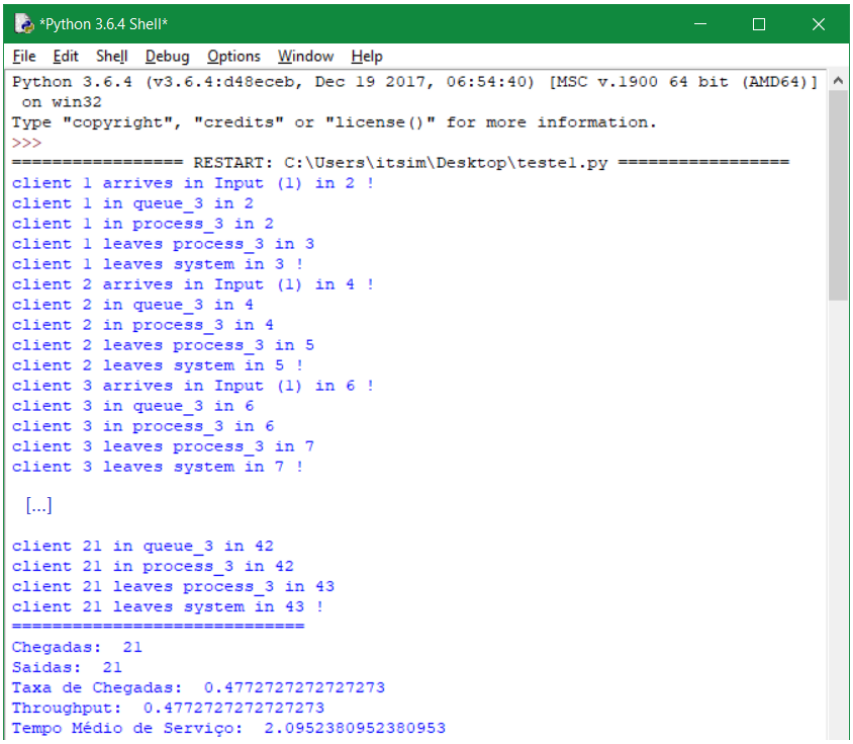

Figura 21 – Feedback (Log do arquivo python).

No log de eventos da Figura 21 e no gráfico da Figura 22, podemos observar que o sistema está em equilíbrio, pois a taxa de chegadas é igual à taxa de saídas, isto ocorreu porque os clientes não tiveram que ficar tempo na fila maior que a o tempo médio de chegadas de novos clientes no sistema.

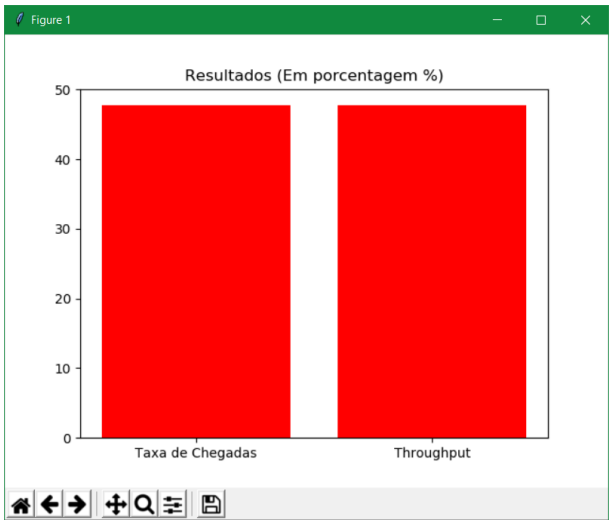

Figura 22 – Feedback (Gráfico de Taxa de Chegadas e Throughput).

A seguir, na listagem 5, um trecho do código Python gerado, mostrando os devios (Forks) e suas respectivas saídas.

```
# Fork List
def fork_5(clients):

  rand = random.uniform(1, 100)

  if (rand >= 1 or rand <= 45):
   # Next left node
    output_2(clients)
  else:
   # Next right node
    env.process(queue_4(env, clients))
def fork_6(clients):

  rand = random.uniform(1, 100)

  if (rand >= 1 or rand <= 80):
   # Next left node
    env.process(queue_3(env, clients))
  else:
   # Next right node
    output_2(clients)

# Output List
def output_2(clients):

  global exits

  print('client', clients, 'leaves system in', env.now, '!')
  exits +=1
```

Listagem 5 - Feedback, código Python.

5.5 Caso 5 - Múltiplas filas

Neste exemplo, demonstramos uma soma de vários formatos anteriormente mostrados. A interface pode ser observada na figura 23.

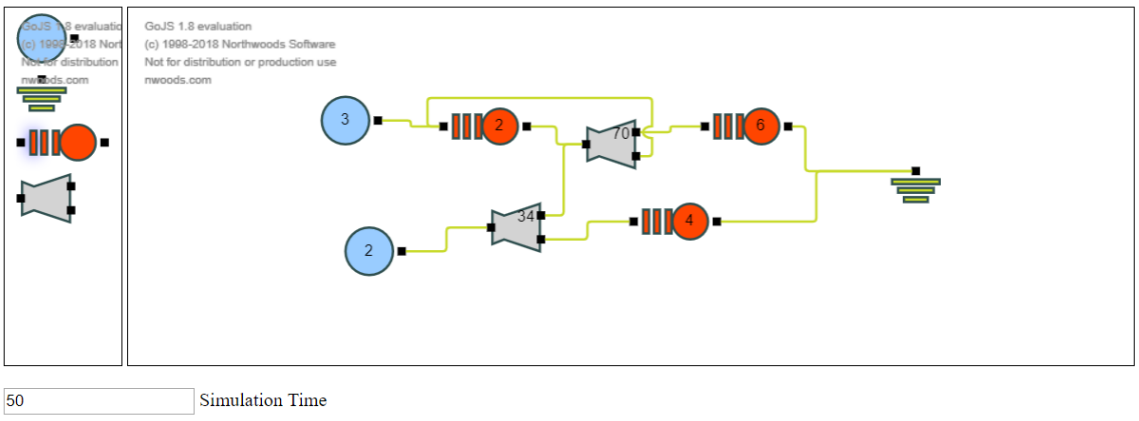

Figura 23 – Múltiplas filas (Interface).

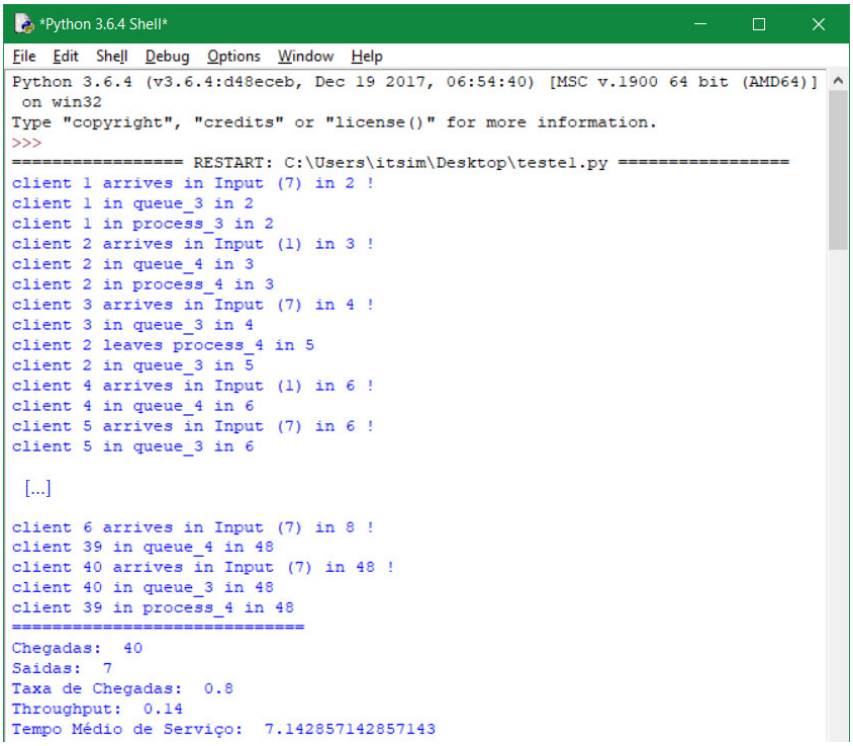

Figura 24 – Múltiplas filas (Log do arquivo python).

Acima, na figura 24 vimos um trecho do log da execução do exemplo de feedback.

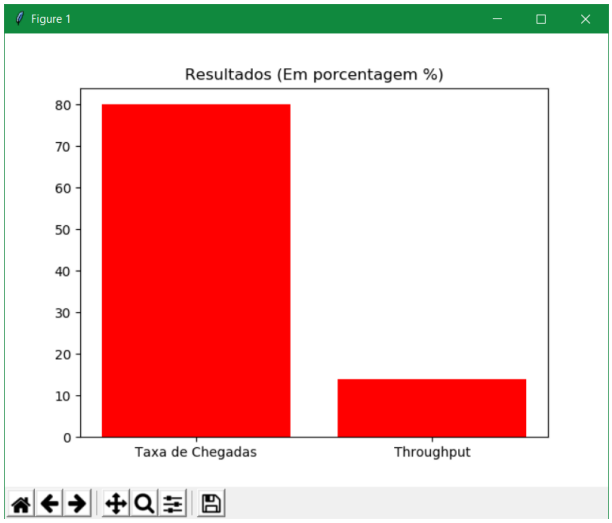

Figura 25 – Múltiplas filas (Gráfico de Taxa de Chegadas e Throughput).

Na figura 25 podemos observar a maior diferença entre as taxas de chegada e throughput, pois os vários caminhos percorridos pelos elementos dificultaram que eles fossem atendidos no tempo de execução que era baixo.

A seguir, na listagem 6, um trecho do código de múltiplas filas, mostrando algumas das funções referentes às filas geradas.

```
# Queue List
def queue_3(env, clients):

    print('client', clients, 'in queue_3 in', env.now)
    req = req3.request()
    yield req
    print('client', clients, 'in process_3 in', env.now)
    yield env.timeout(2) # Process time
    req3.release(req)
    print('client', clients, 'leaves process_3 in', env.now)
     # Next node
    fork_5(clients)
def queue_4(env, clients):

    print('client', clients, 'in queue_4 in', env.now)
    req = req4.request()
    yield req
    print('client', clients, 'in process_4 in', env.now)
    yield env.timeout(6) # Process time
    req4.release(req)
    print('client', clients, 'leaves process_4 in', env.now)
     # Next node
    output_2(clients)
def queue_8(env, clients):

    print('client', clients, 'in queue_8 in', env.now)
    req = req8.request()
    yield req
    print('client', clients, 'in process_8 in', env.now)
    yield env.timeout(4) # Process time
    req8.release(req)
    print('client', clients, 'leaves process_8 in', env.now)
     # Next node
    output 2(clients)
```

Listagem 6 - Múltiplas filas, código Python.

6 CONCLUSÃO

Ao final deste trabalho, podemos observar que um ambiente como este pode se tornar base para a utilização em diversas áreas, desde utilização em sala de aula para facilitar o aprendizado da linguagem e da simulação de eventos discretos até realizar cálculos de simulação de eventos para obras reais.

Simulação de eventos está presente nas mais diferentes áreas do conhecimento e um ambiente desse tipo pode facilitar bastante o crescimento de projetos que a utilizem, além de ajudar os próprios desenvolvedores a aprender sobre o tema e sobre as linguagens envolvidas no processo.

O intuito deste trabalho foi de criar um ambiente base e com facilidade de alteração e melhorias por parte dos desenvolvedores, além de ser uma ferramenta utilizável em sala de aula por professor e alunos. Também foi de mostrar a integração entre diferentes linguagens de programação e como elas podem ser utilizadas para resolver questões inerentes à diversas áreas do conhecimento, além de mostrar a capacidade da biblioteca SymPy para a simulação de eventos discretos.

Com as especificações cumpridas, podemos observar novas fronteiras a serem ultrapassadas no processo de desenvolvimento de ambientes que utilizam essa biblioteca, bem como o melhoramento deste, que através da integração de desenvolvedores, poderá torná-lo um amplo método de realizar cálculos e tarefas referentes ao uso de simulação.

Foi muito proveitoso o desafio de criar um ambiente que integre tantas tecnologias distintas, as dificuldades encontradas, no que tange o aprendizado de novas linguagens de programação e a compreensão do tema eventos discretos puderam ser superadas e o trabalho pôde ser concluído, o aprendizado de novas linguagens e o aprofundamento nas já conhecidas foi parte fundamental no crescimento intelectual e ajudou bastante a melhorar a ideia do próprio ambiente enquanto o projeto seguia, tornando-a algo melhor do que a ideia inicial e inserindo elementos e características que facilitem o processo de atualização e incremento novos elementos que o tornem uma ferramenta mais poderosa e que ela faça cada vez mais parte do cotidiano de quem utiliza simulação de eventos discretos.

Como sugestão para futuros trabalhos, pode-se citar a implementação de exemplos predefinidos no ambiente para a visualização dos usuários e a expansão do mesmo, adicionando novos tipos de métricas ou novos tipos de nós.

REFERÊNCIAS

BORGES, Luiz Eduardo. Python para desenvolvedores: aborda Python 3.3. Novatec Editora, 2014.

BRAY, Tim. The javascript object notation (json) data interchange format. 2017.

FLANAGAN, David. JavaScript: O guia definitivo. Bookman Editora, 2004.

FREEMAN, Elisabeth. Use a cabeça!: HTML com CSS e HTML. Alta books, 2008.

GO.js. Circuito lógico utilizado como base para a interface da ferramenta, disponível em <https://gojs.net/latest/samples/logicCircuit.html>

GUHA, Arjun; SAFTOIU, Claudiu; KRISHNAMURTHI, Shriram. The essence of JavaScript. In: European conference on Object-oriented programming. Springer, Berlin, Heidelberg, 2010. p. 126-150.

MCKINNEY, Wes. Python for data analysis: Data wrangling with Pandas, NumPy, and IPython. "O'Reilly Media, Inc.", 2012.

PRESCOTT, P.; TORRES, P. A. F. M. HTML 5. Babelcube Incorporated. Disponível em, 2015.

THE STATISTICS PORTAL. Gráfico de usuários de smartphone pelo mundo, disponível em <https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/>