UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA - CCET
COORDENADORIA DO CURSO DE CIÊNCIA DA COMPUTAÇÃO – COCOM

EDUARDO JOSÉ TORRES ROCHA

# **UM AMBIENTE PARA VISUALIZAÇÃO DE SIMULAÇÃO ORIENTADA A EVENTOS:** estudo de caso com a biblioteca simpy

São Luís- MA
2023

EDUARDO JOSÉ TORRES ROCHA

# UM AMBIENTE PARA VISUALIZAÇÃO DE SIMULAÇÃO ORIENTADA A EVENTOS: estudo de caso com a biblioteca simpy

Projeto de pesquisa apresentado ao Curso de Ciência da Computação, da Universidade Federal do Maranhão, a ser utilizado como diretrizes para manufatura do Trabalho de Conclusão de Curso.

Orientador: Prof. Dr. Mário Antônio Meireles Teixeira.

São Luís- MA
2023

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Rocha, Eduardo José Torres.
UM AMBIENTE PARA VISUALIZAÇÃO DE SIMULAÇÃO ORIENTADA A EVENTOS: : estudo de caso com a biblioteca simpy / Eduardo José Torres Rocha. - 2023.
56 p.

Orientador(a): Mário Antonio Meireles Teixeira.
Curso de Ciência da Computação, Universidade Federal do Maranhão, São Luís, 2023.

1. Ensino de computação. 2. Redes de filas. 3. Simpy. 4. Simulação. 5. Visualização. I. Teixeira, Mário Antonio Meireles. II. Título.

EDUARDO JOSÉ TORRES ROCHA

**UM AMBIENTE PARA VISUALIZAÇÃO DE SIMULAÇÃO ORIENTADA A EVENTOS:** estudo de caso com a biblioteca simpy

Projeto de pesquisa apresentado ao Curso de Ciência da Computação, da Universidade Federal do Maranhão, a ser utilizado como diretrizes para manufatura do Trabalho de Conclusão de Curso.

BANCA AVALIATIVA

_____

Mário Antônio Meireles Teixeira (UFMA)
**Orientador**

_____

Simara Vieira da Rocha (UFMA)
**1º Avaliadora**

_____

Anselmo Cardoso de Paiva (UFMA)
**2º Avaliador**

Dedico este trabalho ao amor da minha vida, Iandara Porto, que me ajudou a aguentar a pressão e a corrida para o desenvolvimento desse projeto, que não foi simples e a força e incentivo que ela me forneceu com muito amor, foram fundamentais para que isso se tornasse real.

## AGRADECIMENTOS

A construção desse trabalho levou como contribuição a ajuda e força de várias pessoas, estas as quais eu agradeço:

Ao professor Mário Meireles, por ter sido mais que o meu orientador, mas o meu guia em todo o desenvolvimento do projeto, esse que foi bem intenso e cercado de desafios.

Aos professores do curso de Ciência da Computação, que me forneceram a base acadêmica necessária para proporcionar uma boa criação desse projeto e por construírem o que eu tenho como conhecimento que usarei para além da minha vida acadêmica.

A todas as pessoas que auxiliaram e participaram das pesquisas, pela colaboração no momento de busca de dados para o projeto.

Aos meus pais, que me deram o apoio e suporte necessários para que esse sonho se realizasse.

Ao meu amor, Iandara, por ter me apoiado e me incentivado a cada dia, a cada momento, que foi fundamental para eu continuar construindo esse projeto, me ajudando a acreditar que isso era capaz de se realizar.

Aos meus amigos, que sempre estiveram próximos nos momentos mais conturbados.

“Educação não transforma o mundo. Educação muda as pessoas. Pessoas mudam o mundo” (Paulo Freire)

## RESUMO

O presente trabalho tem como propósito, elaborar um ambiente de visualização de simulações. Isso se deve ao dilema levantado, a dificuldade de elaborar, metrificar e interpretar os dados de simulações construídas com programas de computadores, que utilizam linguagem de programação para a implementação. Assim, busca-se mitigar essa complexidade provinda da realidade computacional para o consumo na realidade humana, aproximando as pessoas das simulações. Para isso, apresenta-se a base teórica para levantar as soluções idealizadas. Após isso, por meio da codificação, constrói-se a aplicação que representa o ambiente visualizador propriamente dito, aplicando todas as soluções encontradas. Dessa forma, realizou-se os testes do programa desenvolvido, buscando o cumprimento de solucionador da problemática, suprindo o papel de ambiente de visualização. Por fim, interpreta-se a viabilidade do estudo, através da codificação e a experimentação do programa resultante, concluindo-se que o ambiente elaborado é viável para o que se propõe.



**Palavras-chave:** Simulação. Redes de Filas. Visualização. Simpy. Ensino de Computação.

## ABSTRACT

The purpose of this work is to elaborate a simulation visualization environment. This is due to the dilemma raised, the difficulty of elaborating, measuring and interpreting data from simulations built with computer programs, which use programming language for implementation. Thus, we seek to mitigate this complexity arising from computational reality for consumption in human reality, bringing people closer to simulations. For this, the theoretical basis is presented to raise the idealized solutions. After that, through coding, the application that represents the viewer environment itself is built, applying all the solutions found. In this way, the tests of the developed program were carried out, seeking to solve the problem, fulfilling the role of visualization environment. Finally, the viability of the study is interpreted, through coding and experimentation of the resulting program, concluding that the developed environment is viable for what is proposed.



**Keywords:** Simulation. Queueing Networks. Visualization. Simpy. Computer teaching.

## LISTA DE FIGURAS

## LISTA DE TABELA

## SUMÁRIO

# 1 INTRODUÇÃO

Ao realizar quaisquer processos, normalmente é necessário elaborar um estudo, para que tudo ocorra de uma maneira satisfatória. Para isso, são criadas simulações para prever como estas ações serão executadas e de que forma irão desempenhar. Entretanto, essa não é uma tarefa trivial, já que existem diversas maneiras de simular. No presente estudo, usou-se a simulação desenvolvida por meio de programas de computadores. Essas ferramentas necessitam de um conhecimento especializado, pois possui uma saída de dados de difícil interpretação.

Nesse sentido, surge a problemática: como aproximar as pessoas com pouco conhecimento especializado a simulações feitas em programas de computadores? A fim de responder essa questão, apresenta-se a idealização do ambiente para visualização de simulações, que busca solucionar esse cenário dilemático. Ademais, essa ferramenta também possui o potencial de ser utilizada para o ensino de simulação no âmbito de redes de filas que contribui com o cenário acadêmico. A partir dessa inquietação, objetivou-se elaborar o presente estudo de caráter bibliográficos e de práxis para compreender as necessidades dos usuários no/do ambiente.

Seguindo esse cenário, pode-se relacionar alguns objetivos na elaboração do estudo. Inicialmente, é possível citar a criação de uma interface simples e de fácil entendimento por pessoas com diferentes níveis de conhecimento e que abranja todos os requisitos de simulações. Além disso, também é necessário objetivar uma aplicação desacoplada de quaisquer bibliotecas de simulações e compatível com as principais plataformas, sendo elas: MacOS, Windows e Linux. Por fim, é crucial buscar a liberdade de reprodução das simulações, aumentando a precisão das análises delas.

Além disso, após o desenvolvimento da base teórica desta pesquisa, apresentou-se a idealização das soluções consequentes dos objetivos levantados. A codificação é realizada para trazer todo o planejamento para o mundo real, para colocar todo o estudo em prática. Fazendo a escolha das tecnologias e padrões de desenvolvimento mais adequados para a realidade do projeto, essa ação torna-se possível. Dessa forma, o estudo avança para a posterior experimentação do que foi construído.

Desse modo, ao término do projeto levantado, há a conclusão dos resultados obtidos desde a prospecção do tema à implementação dele. Ao decorrer do estudo,

pode-se observar de forma detalhada, como a aplicação proposta foi construída, assim como a sua utilização foi feita. Ao final, tem-se o veredito da viabilidade do projeto do estudo, relacionando à problemática apresentada com as soluções executadas, assim como as possíveis melhorias.

# 2 DESENVOLVIMENTO

## 2.1 TEORIA DAS FILAS

Agner Krarup Erlang publicou primeira vez em 1909, a Teoria das Filas. Segundo as pesquisas do engenheiro dinamarquês, a Teoria das Filas foi um estudo em que considera a formal de filas que faz o uso de modelos matemáticos e métricas das propriedades compostas pelos clientes.

### 2.1.2 Sistema de filas

É um sistema caracterizado por um banco de canais de atendimento paralelos com um fluxo de clientes que entram em horário distintos e são atendidos, possivelmente aguardando em uma fila se todos os servidores estiverem ocupados.

Ele é composto pelos seguintes pontos e suas respectivas variáveis:

Tabela 1- Variáveis de fila

| | |
|---|---|
| A | O tipo de distribuição de probabilidade utilizada para o processo de chegada, com um ou mais parâmetros. |
| S | O tipo de distribuição de probabilidade utilizada para o processo de serviço, com um ou mais parâmetros. |
| s | O número de servidores. |
| K | O número máximo de clientes que podem estar no sistema, se limitado. |
| N | O tamanho da população de clientes potenciais, se limitado. |
| Q | Disciplina de atendimento |

Fonte: elaborado pelo autor.

Dessa forma, pode-se delimitar estes pontos, utilizando as variáveis apresentadas, com a seguinte forma:

A / S / s / K / N / Q

Além disso, tem-se distribuições mais comuns de probabilidade, sendo elas:

Tabela 2- Principais distribuições de probabilidade

| | |
|---|---|
| M | Distribuição exponencial (Markovian) |
| Ek | Distribuição Erlang |
| D | Distribuição determinística |
| G | Distribuição geral |

Fonte:

Assim, temos como exemplo, M/M/5, que consiste em:

Tabela 3- Notação do sistema M/M/5

| | |
|---|---|
| M | Processo de chegada exponencial |
| M | Processo de serviço exponencial |
| 5 | 5 servidores |

Fonte: elaborado pelo autor.

### 2.1.2 Processo de chegada

Em um sistema de filas, um cliente pode chegar a qualquer momento, assim como o próximo cliente, dessa forma, o intervalo entre eles variável. Por conta disso, podemos pensar no processo de chegada como um processo estocástico. Com isso, a partir do agregado, pode-se prever um valor médio para a taxa de chegada. E para que isso seja possível, deve-se saber a distribuição de probabilidade que descreve o processo, podendo ser alguma das que foram citadas na tabela 2.

### 2.1.3 Distribuição Do Tempo De Serviço

Seguindo a mesma ideia do processo de chegada, aqui há uma distribuição de probabilidade vinculada com o tempo de serviço, conforme as exemplificadas na tabela 2.

### 2.1.4 Limites Do Sistema

A capacidade está vinculada com a quantidade máxima de clientes, podendo ser finita ou infinita, assim como os clientes potenciais, como mostrado na tabela 1.

### 2.1.5 Disciplina De Atendimento

Esta propriedade dita de que forma os clientes são atendidos na fila, sendo as disciplinas:

Tabela 4- Disciplinas de filas

| | |
|---|---|
| First In, First Out (FIFO) | Primeiro a chegar, primeiro a ser atendido |
| Last In, First Out (LIFO) | Último a chegar, primeiro a ser atendido |
| Fila com prioridade | Clientes possuem uma prioridade atribuída. Pode ser preemptivo: onde o atendimento de um cliente com menor prioridade é interrompido para atender um de maior prioridade; ou não-preemptivo: onde o de maior prioridade apenas é colocado no início da fila, sem interromper o atendimento do cliente atual. |
| Serve In Random Order (SIRO) | Os clientes são atendidos aleatoriamente. |

Fonte: elaborado pelo autor.

### 2.1.6 Quantidades Operacionais

Dentro do sistema de filas, há quantidades operacionais relacionadas, que através delas, pode-se realizar cálculos de métricas, que serão apresentados mais adiante. São elas:

$\lambda$ - taxa de chegada
$\mu$ - taxa de atendimento
$Ta$ - tempo de atendimento
$c$ - número de servidores em paralelo
$W$ - tempo de espera no sistema
$W_f$ - tempo de espera na fila

### 2.1.7 Lei Da Utilização

Diz a porcentagem do tempo em que o servidor está ocupado.

$$\rho = \frac{\lambda}{c\mu} \text{ ou } \rho = \lambda Ta$$

### 2.1.8 Lei de Little

Número de clientes na fila:

$$N_f = \lambda W_f$$

Número de clientes no sistema:

$$N = \lambda W$$

## 2.2 O QUE É SIMULAÇÃO?

De uma ótica objetiva, menciona-se a definição conceitual do objeto, simulação, que de acordo com o dicionário Cambridge consiste em: “um modelo de um conjunto de problemas ou eventos que podem ser usados para ensinar alguém sobre como fazer alguma coisa, ou o processo de fazer tal modelo”. Portanto, é possível notar que de uma maneira geral, simulações, se limitam a modelos de algo, algum processo, com a tentativa de se aproximar da realidade, permitindo assim uma fácil visualização do objeto em estudo, sem a necessidade de implementação no mundo real.

## 2.3 POR QUE SIMULAR?

A simulação é uma das ferramentas que permite a diversos profissionais, como cientistas da computação, engenheiros, físicos, realizar as atividades a que se propõem. Através dela, eles podem adquirir capacidade de identificar, formular e solucionar problemas ligados às atividades de projeto, operação e gerenciamento do trabalho e de sistemas de produção de bens e/ou serviços.

Para mais, as simulações possuem a finalidade de prever o comportamento futuro dos sistemas usando modelos, isto é, antecipar os efeitos produzidos por alterações ou pelo emprego de outros métodos em suas operações. Com isso, é possível construir teorias e hipóteses considerando observações efetuadas através dos instrumentos de análise anteriormente citados, ou seja, os modelos (TEIXEIRA, 2018).

Por fim, nos problemas de gerência, no qual o “tempo-dinheiro” assume proporções “alarmantes”, pois as decisões de negócios reclamam um teste prévio em simuladores os mais avançados possíveis, para que se possa avaliar as suas consequências dentro de um tempo e confiabilidade admissíveis.

## 2.4 MODELOS DE SIMULAÇÕES

Um modelo, comparado ao sistema real que ele representa, pode lidar com informação a um baixo custo, além do conhecimento ser obtido mais rapidamente e em condições não observáveis na vida real. O modelo basicamente busca resumir o funcionamento do sistema em um pequeno número de variáveis que permita sua apreensão pelo intelecto humano (GAVIRA, 2003).

Dentro da definição de modelos, temos algumas classificações, sendo elas: linear ou não-linear, estático ou dinâmico, estável ou instável, discreto ou contínuo e determinístico ou estocástico.

### 2.4.1 Linear ou Não-Linear

Os modelos são lineares quando o sistema que representam segue uma lei linear, ou seja, independente do nível em que esteja sempre haverá a mesma resposta da variável endógena a uma variável independente ou exógena. Modelos não-lineares são aqueles em que o relacionamento das variáveis endógenas e exógenas não se ocorre por meio de relações lineares.

### 2.4.2 Estático ou Dinâmico

Modelos estáticos são usados para representar sistemas em que o tempo não desempenha nenhum papel. Por outro lado, modelos dinâmicos representam sistemas que evoluíram no tempo, como um sistema de uma fábrica.

### 2.4.3 Estável ou Instável

Modelos dinâmicos podem ser estáveis ou instáveis. Nos estáveis, sempre se volta à condição inicial após algum distúrbio. Já os modelos instáveis representam sistemas que depois de ativados tendem a desenvolver a sua amplitude de movimentação e, consequentemente, a se afastar sobremaneira de sua condição inicial.

### 2.4.4 Discretos ou Contínuos

Em modelos contínuos, os valores das variáveis se alteram de forma gradativa no tempo e são geralmente representados por equações diferenciais, como por exemplo, no crescimento de uma planta, no enchimento de um pneu de carro ou na variação do nível de um tanque de combustível. Eventos discretos, por outro lado, evoluem à medida que os estados do sistema são alterados e facilmente identificados, como uma parada de trens em estações ou a montagem da base de uma cadeira ou o escalonamento de um processo na CPU.

### 2.4.5 Determinísticos ou Estocásticos

Modelos determinísticos não possuem componentes aleatórios. Exemplos de modelos determinísticos incluem um sistema de equações diferenciais que descrevem uma reação química, ou modelos de programação linear inteira-mista. O resultado destes modelos também não possui componentes aleatórios. A presença de elementos aleatórios gera a necessidade de elaborar modelos estocásticos. A maioria dos sistemas de filas são modelados estocasticamente.

## 2.5 SIMULAÇÃO ORIENTADA A EVENTOS

Em um sistema de eventos discretos, um ou mais fenômenos de interesse mudam seu valor, ou estado, em pontos discretos (ao invés de continuamente) no tempo. A ocorrência destes eventos muda o estado do sistema em cada momento. Dessa forma, assumimos que não há mudanças no sistema entre um evento e outro. Mesmo em caso de haver incrementos fixos de avanço no tempo, o que não é muito comum, a evolução do sistema não ocorre de forma contínua no tempo. Neste caso, supõe-se que o estado do sistema não se altera ao longo do intervalo compreendido entre dois eventos consecutivos (ALMEIDA, JOÃO, 2016, p. 3).

### 2.5.1 Vantagens

Como mostrado por Muriel Gavira (2003), no contexto das simulações orientada a eventos, têm-se diversas vantagens, como exemplo: modelos mais realistas, já que o modelo mais apropriado para o problema será utilizado, evitando a

ocorrência obrigatória do inverso; o modelo evolui de acordo com a complexidade do problema, de forma gradual, analisando todos os detalhes do problema a ser tratado; altamente adaptável aos diversos tipos de problemas.

### 2.5.2 Desvantagens

Em contrapartida, apesar das vantagens, há alguns pontos negativos a serem considerados por Muriel Gavira (2003), sendo eles:

- ❖ Pelas saídas serem comumente aleatórias, os resultados podem ser complexos de serem analisados e interpretados;
- ❖ Dificuldade de modelagem;
- ❖ Especialização é necessária para implementar os modelos de acordo com o problema proposto, o que exige treino e experimentação, resultando em um gasto necessário de tempo.

## 2.6 SIMULAÇÕES COM A BIBLIOTECA SIMPY

Seguindo o objeto de estudo tratado nessa discussão, as simulações orientadas a eventos, menciona-se uma das bibliotecas mais populares que fazem o uso delas, a biblioteca SimPy[1]. De acordo com a documentação desta, pode-se conceituá-la como uma estrutura de simulações orientadas a eventos baseadas em processo, escrita na linguagem python.

Além disso, ela possui como proposta de abranger um amplo espectro de simulações e para que isso seja possível ela possui abstrações de recursos com filas, ademais de poder funcionar com tempo real, avanço de eventos ou até mesmo o menor tempo possível. Ainda mais, trazendo à tona a teoria das filas, há ainda a abstração acerca das filas com prioridade, sendo preemptivas ou não.

Dessa forma, a biblioteca SimPy oferece um ótimo cenário resolutivo na hora de elaborar simulações, por ser de amplo espectro, relativamente fácil, leve, com uma documentação completa e bem performática. Por conta dessas características e sua popularidade entre os desenvolvedores, ela foi a escolhida para receber a implementação referente a aplicação resultante dessa discussão, apesar de haver outras, como exemplo, a Java Simulation Library.

[1] "SimPy | Read the Docs." https://readthedocs.org/projects/simpy/. Acessado em 5 out. 2022.

## 2.7 MÉTRICAS E SEUS DESAFIOS

Levando em conta todo o fundamento teórico apresentado até o momento, nota-se os diferentes tipos de sistemas e os modelos que envolvem as simulações, além do conceito e a motivação dessas. Entretanto, como já dito anteriormente, é necessário um treinamento especial para a elaboração e leitura das simulações. Ainda mais, quando se envolve codificação, pois a dificuldade é acentuada.

Nesse sentido, evidencia-se um dos maiores desafios, o cálculo e interpretação das métricas que possuem uma complexidade intrínseca, tendo em vista que pode haver variáveis aleatórias. Se isso não bastasse, como já dito, simular por meio de programas de computadores, como a biblioteca SimPy, mostra-se ainda mais complexa, devido à especialização em código necessária. Somando-se a isso, apresentar as saídas, métricas, de uma simulação desenvolvida em código, é quase inviável para pessoas sem especialização.

Dessa forma, nota-se um dilema específico muito claro, fazer o uso e apresentação das métricas provenientes das simulações, dando enfoque para o âmbito computacional. Logo, fica claro qual seria a proposta solucionadora do problema, sair de um nível baixo, onde apenas se enxerga código e saídas de terminal, para um cenário de alto nível, com valores e representações gráficas. Dessa forma, o desafio das métricas seria amenizado, facilitando o acesso às simulações por pessoas de diferentes níveis de conhecimento, auxiliando inclusive o ensino-aprendizagem desse assunto em disciplinas de graduação em Computação, Engenharias e áreas afins.

# 3 UM AMBIENTE PARA VISUALIZAÇÃO DE SIMULAÇÕES

Finalmente, dentro desta seção, inicia-se a elaboração da temática do estudo trabalhado, no qual busca-se projetar um ambiente para visualizar simulações. Para isso, devem ser seguidas algumas etapas, relacionando-se objetivos, soluções e a construção do programa de computador, no qual, há descrito cada detalhe do seu desenvolvimento, como tecnologias utilizadas.

## 3.1 OBJETIVOS

Nesse sentido, acompanhando-se o que foi apresentado na construção teórica do projeto, as simulações cumprem um importante papel na resolução de problemas, dos mais diversos tipos. Entretanto, os resultados das simulações exigem um certo cuidado, pois, para possuir valor, eles devem ser alcançados por todas ou a maioria das pessoas envolvidas no dilema tratado.

Por conta disso, deve haver uma filtragem e organização dos resultados obtidos, para que sejam consumidos tanto por pessoas com muito quanto pouco conhecimento especializado. E esse trabalho de filtragem torna-se ainda mais difícil quando se envolve simulações por meio de programas de nível mais baixo, como os que utilizam a biblioteca SimPy, ou outras similares, devido ao frequente uso de saída de terminal, que apresenta uma linguagem verbal e visual não muito amigável.

Dessa forma, com essa problemática apresentada, o estudo corrente traz propostas para minimizá-la, citando-se a partir disso, os seguintes objetivos:

- Fornecer uma interface gráfica, simples e de fácil entendimento por pessoas com diferentes níveis de conhecimento, agregando a maior acessibilidade possível;
- Ser desacoplada de bibliotecas de simulações, fornecendo uma linguagem de comunicação universal;
- Atender todos os requisitos de simulações, abraçando uma porção majoritária de conceitos relacionados à avaliação de desempenho;
- Ser altamente disponível, permitindo o seu uso nas principais plataformas, sendo elas: Windows, Linux e MacOS;

- Possuir um alto nível de configuração de reprodução das simulações, permitindo acelerar, diminuir, pausar e reiniciar, permitindo assim um acompanhamento aguçado dos eventos que ocorrem nas mesmas.

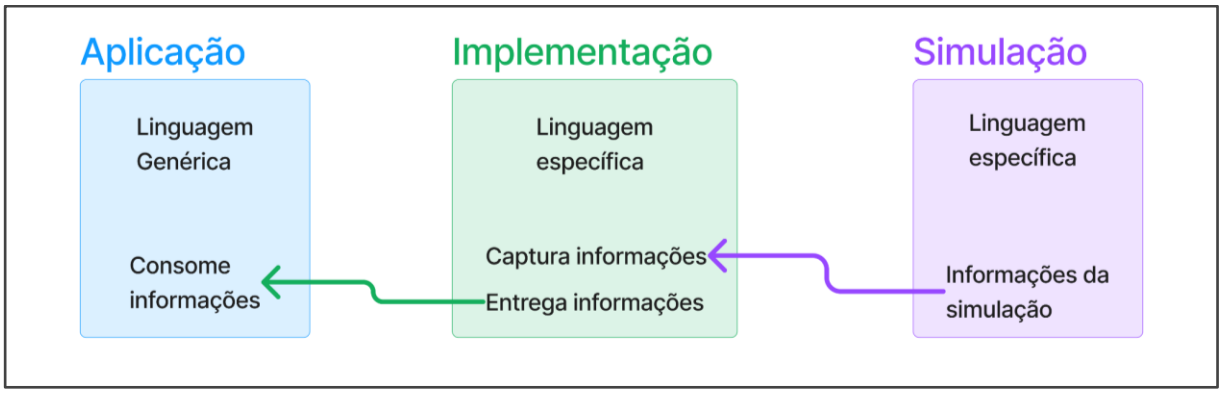
## 3.2 SOLUÇÕES

Com os objetivos acima citados, relacionando-os diretamente com a problemática apresentada, obtêm-se soluções importantes para o desenvolvimento da aplicação.

### 3.2.1 Esquema aplicação-implementação

Buscando atender o requisito de desenvolver um programa que seja desacoplado de quaisquer linguagem e suas bibliotecas de simulação, foi idealizada esta solução que se mostra como o núcleo de todo o desenvolvimento e funcionamento da aplicação resultante. Conceituando-a, essa solução propõe um esquema de relacionamento entre aplicação (o programa resultante) e a implementação, sendo a primeira sem detalhes de implementação e livre de qualquer acoplamento; e a segunda, fortemente acoplada a uma linguagem ou uma biblioteca. Nesse cenário, tem-se os papéis de leitor e tradutor, os quais são desempenhados, respectivamente, pela aplicação e implementação.

Figura 1-Esquema de aplicação-implementação

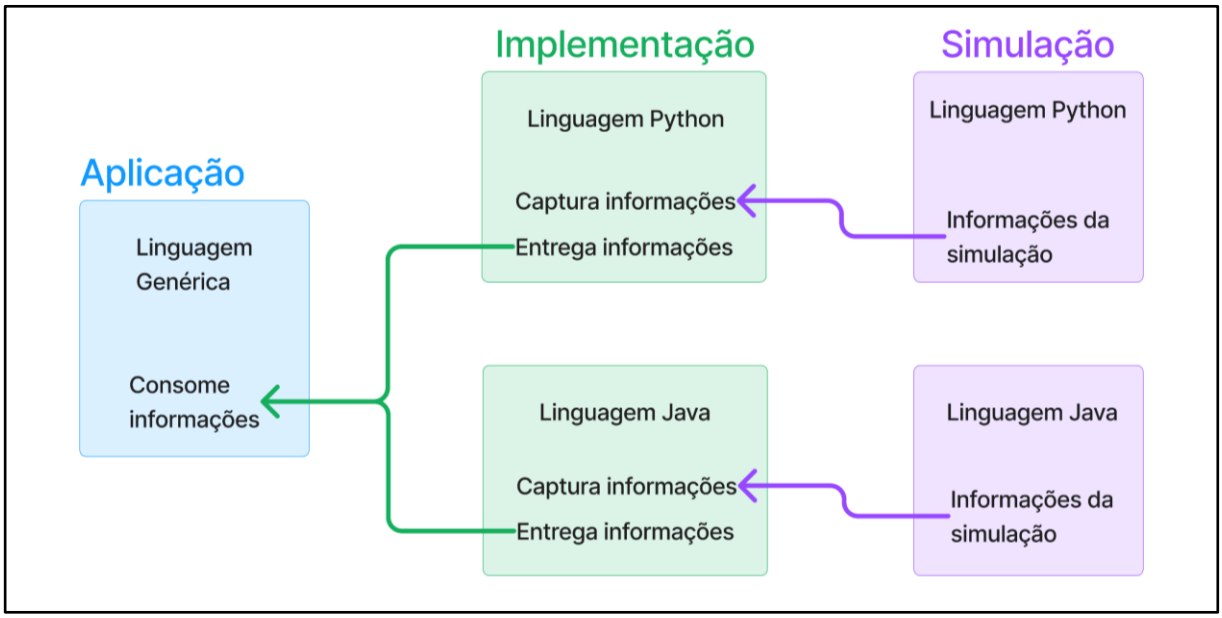



Fonte: Compilação do autor.

O funcionamento do esquema proposto pode ser mais bem analisado com a figura 1. Nela é possível observar os agentes aplicação e implementação relacionando-se diretamente, juntamente com a simulação. Dessa forma, pode ser

compreendido que as informações contidas na simulação, sendo esta contida em uma linguagem específica qualquer, estão sendo coletadas pela implementação, tendo a mesma linguagem que a simulação, que posteriormente são processadas e entregues para a aplicação, na qual as consome, tendo esta uma linguagem genérica. Com isso, é fácil de notar o trabalho de intermediador da implementação, que coleta os dados do objeto trabalhado, nesse caso, a simulação, já que ele consegue "compreender" aqueles dados, por possuir a mesma linguagem, os processa e os converte para uma linguagem genérica, entregando-os para a aplicação, que compreende esta linguagem. Por conta disso, é possível criar um único programa de visualização de simulações que atende as mais diversas simulações e as linguagens em que foram escritas, bastando desenvolver a implementação necessária para a necessidade criada.

Figura 2- Uma aplicação com duas implementações




Fonte: Compilação do autor.

Por fim, pode-se observar essa possibilidade de atender diversas simulações e suas linguagens com a figura 2. No esquema apresentado, tem-se uma aplicação, que consome de duas implementações, tendo elas, as linguagens Java e Python, capturando simulações com suas respectivas linguagens. Dessa forma, o esquema de aplicação-implementação mostra-se uma ótima solução para o desenvolvimento de um ambiente de visualização de simulação desacoplado de qualquer linguagem específica, atendendo com o requisito proposto anteriormente.

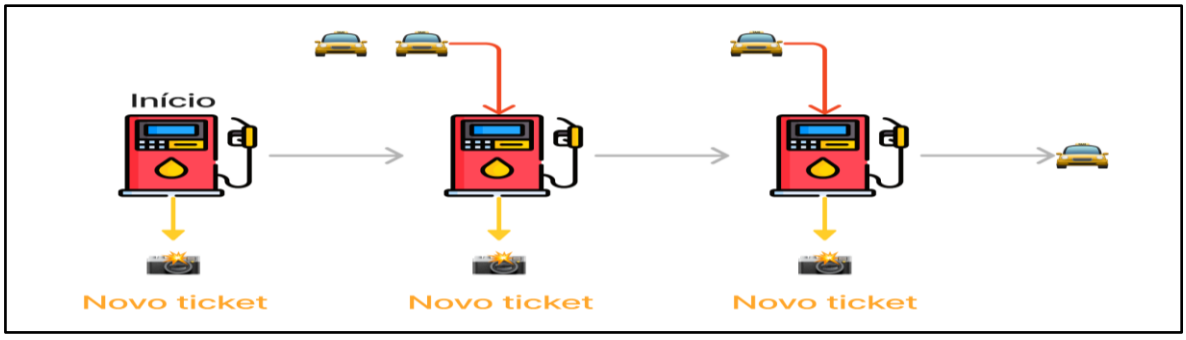
### 3.2.2 Relatório de Simulação

Com a solução do esquema de aplicação-implementação proposta é necessário elaborar uma estrutura para armazenar os dados das simulações que serão transportados para a aplicação com uma linguagem genérica. Para isso, desenvolveu-se o sistema de relatório de simulação, o qual possui como finalidade capturar simulações executadas em sua biblioteca e linguagem próprias, organizando os seus dados e colocando-os em uma linguagem universal para serem consumidos pela aplicação de visualização de simulações. Como pode ser visto, o sistema de relatório fará parte da implementação e da aplicação, tendo em vista que ele fará o papel de analisar, coletar e traduzir o conteúdo das simulações, no âmbito da implementação, e entregar um relatório, em uma linguagem universal das informações coletadas, já no âmbito da aplicação.

Nesse sentido, menciona-se o funcionamento básico deste relatório, que consiste em um esquema de registro de alterações em uma simulação executada. O que isso significa? A cada novo estado da simulação é realizado um registro, como uma fotografia, e foi escolhido chamar cada registro desse de ticket, como um ticket de um evento. Dessa maneira, obtém-se um relatório da simulação que foi executada de maneira leve, sem perder quaisquer eventos que podem ocorrer, permitindo assim, visualizar cada estado, cada ticket.

Dentro desse contexto, pode-se exemplificar o funcionamento do relatório de simulação através de uma situação de posto de combustível. Para isso, considera-se uma bomba de combustível com a capacidade hipotética de atender apenas um veículo por vez. Visualizando isso, tem-se a figura abaixo:

Figura 3- Simulação de bomba de combustível




Fonte: Compilação do autor.

Analisando a imagem, pode-se ver que há 3 estados diferentes apresentados na simulação, sendo eles, em ordem numérica:

1. O estado inicial, em que a bomba se encontra sem uso e pronta para atender;
2. O estado em que chegam dois veículos simultaneamente para serem atendidos, porém como a bomba só atende um por vez, cria-se uma fila.
3. O estado em que o veículo que chegou primeiro sai, após ser atendido, e o que chegou por último apresenta-se para ser atendido pela bomba de combustível.

Colocando dessa maneira, é facilmente perceptível o funcionamento do relatório. Cada estado que foi apresentado é "fotografado", ou seja, registra-se um ticket contendo as informações da simulação naquele estado. Exemplificando em forma de números, utilizando a fila da bomba de combustível como valor, teríamos os tickets dessa forma:

- Ticket 1: Fila vazia
- Ticket 2: Carro 1, Carro 2
- Ticket 3: Carro 2

Esse é um exemplo mínimo, contabilizando apenas a fila do recurso trabalhado, neste caso, a bomba de combustível. Entretanto, o relatório de simulação possui um poder impressionante, podendo armazenar em cada ticket desde filas de recursos até as métricas calculadas.

Por fim, por se apresentar como uma interface de persistência genérica, aceitando diferentes estruturas de dados, é de forma intrínseca, desacoplada de qualquer biblioteca, podendo ser construída e transportada em qualquer linguagem, detalhes estes que serão tratados no decorrer da aplicação resultante do estudo, mostrando-se assim, uma ótima solução.

### 3.2.3 Métricas Semiautomatizadas

Como dito anteriormente, o cálculo das métricas geralmente apresenta-se como um desafio no processo de simulação. Além de mostrarem-se difíceis de serem calculadas, elas são de suma importância para a construção dos resultados do

experimento, tornando-se assim, um trabalho obrigatório e de complexidade elevada. Para isso, pensou-se em um sistema semiautomatizado para o cálculo das métricas, que propõe uma redução dessa dificuldade supracitada.

Assim sendo, elaborou-se a estrutura desse sistema, que foi nomeado como calculadora de métricas. A ideia nuclear desse componente consiste em fornecer uma interface genérica para efetuar os cálculos das métricas, entregando operações já prontas, como a lei de utilização, abrindo possibilidade para estender a calculadora, para métricas específicas. É importante ressaltar que esse sistema atua, no esquema aplicação-implementação, no âmbito da implementação, tendo em vista que é nesse aspecto que é possível ter acesso aos dados necessários para poder realizar os cálculos. Com isso, após obter os resultados buscados, armazena-os no relatório de simulação, proposto logo acima, sendo assim, consumidos pela aplicação, sendo mostrados através da interface gráfica.

Dessa forma, é possível reduzir de forma acentuada a dificuldade de realizar os cálculos das métricas, de forma semiautomatizada, através de operações já prontas para serem utilizadas, sem a necessidade de saber detalhes da programação. Além disso, é possível elaborar cálculos específicos que ainda não tenham na implementação, estendendo a interface da calculadora, dando assim, um poderio excelente de cálculo. E com o armazenamento no relatório de simulação, tem-se um ótimo sistema de cálculo de métricas, complementando o conjunto de soluções até então apresentados.

### 3.2.4 Interface Genérica

Com a ideia principal da temática do estudo corrente, é de fundamental importância a elaboração de uma interface genérica, que atenda a todos os tipos de simulações. Para isso, foram abstraídos alguns elementos gerais para a aplicação, tendo diferentes características técnicas e visuais. Dessa forma, é possível cumprir com o objetivo de abraçar a porção majoritária das estruturas da avaliação de desempenho de forma visual.

Nesse sentido, cita-se o primeiro deles, que foi o elemento recurso, que representa qualquer recurso simulado, podendo ser um caixa de atendimento, um posto de combustível, um servidor, sendo assim, bem genérico. Ele possui uma capacidade de atendimento, sendo assim, detentor das métricas de desempenho,

como taxa de atendimento. Ainda mais, há a fila, que será atendida pelo recurso trabalhado. Vale mencionar que essa fila pode ser sem prioridade, com prioridade não-premptivo e preemptivo. Por fim, tem-se o contêiner, que corresponde há uma abstração de armazenamento de determinado conteúdo, podendo ser uma analogia de um tanque de água, gás, ou seja, que pode ser abastecido e consumido.

Desse modo, elabora-se uma interface genérica para atender uma grande amplitude de simulações diferentes, isso por meio dos elementos citados, recurso, fila e contêiner. Através deles, têm-se as métricas necessárias para elaboração dos resultados da simulação. Com isso, cumpre-se com o objetivo supracitado, obtendo uma solução adequada para o desenvolvimento do programa.

### 3.2.5 Controles de reprodução

Com a elaboração de simulações tanto no âmbito de implementação quanto aplicação, acrescenta-se os controles de reprodução, na aplicação. Ele é responsável pelo controle de execução do relatório de simulação, possuindo controles de velocidade e reprodução. Tem como finalidade uma melhor observação do relatório resultante, podendo de forma lenta, analisar as variações métricas, assim como visualizar as atividades dos recursos de forma mais acelerada, funcionando como um reprodutor de vídeos, porém nesse caso, reprodutor de tickets, provindos do relatório de simulação. Dessa forma, atende-se o objetivo de alto nível de configuração de reprodução das simulações.

## 3.3 A APLICAÇÃO – LEMONLAB

Como fruto do estudo corrente, foi desenvolvida uma aplicação, que cumpre com os objetivos propostos, adotando também as soluções elaboradas. Esta foi batizada de, lemonlab, que representa a junção de palavras lemon (limão, em português) e lab (laboratório, em português). Isso porque, o programa representa um laboratório de simulações, explicando o uso do lab. Já a utilização do lemon, é explicada pelo limão ser uma fruta versátil, com uma cor amigável, mostrando mais características do programa. Entretanto, antes de sua codificação propriamente dita, foi elaborado um estudo sobre que tecnologias iam fazer parte da sua construção. Além disso, também foi desenhada a interface gráfica que iria compor a aplicação,

tendo grande importância também, já que a proposta base do projeto é exatamente um ambiente visual para as simulações.

### 3.3.1 Tecnologias

Nesta etapa, uma pesquisa foi feita para escolher as tecnologias mais adequadas para o desenvolvimento do projeto. Em primeiro lugar, para atender com o objetivo de ser multiplataforma, foi escolhido o framework electronjs. Esse conjunto tem como proposta, fornecer a possibilidade de criar apenas uma aplicação que execute nos principais sistemas operacionais, Windows, Linux e MacOS, não necessitando conhecimento profundo dos elementos nativos desses três elementos, cumprindo assim com o objetivo.

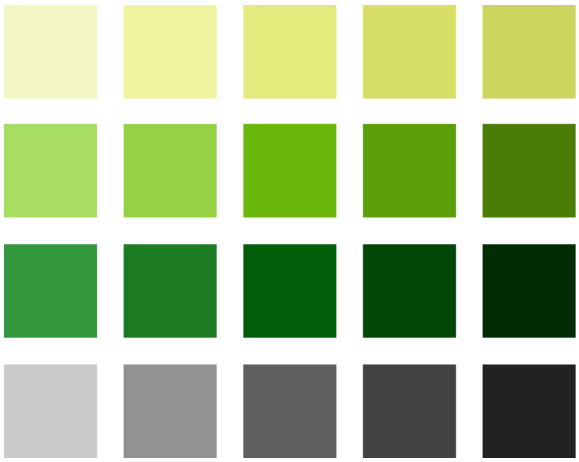
Além disso, como o electronjs funciona com tecnologia web, foi quase natural a escolha do React como biblioteca para construção da interface. Apesar dos concorrentes, como Vue e Angular, o React possui um ótimo suporte, uma comunidade enorme, com conteúdo diverso e atende todos os recursos web necessários para a construção da aplicação, sendo assim, uma escolha racional.

Acompanhando a temática web, utilizou-se também o pré-processador sass, que facilita a escrita das folhas de estilo. Por fim, foi realizado o uso do empacotador webpack, juntamente com babel e typescript, que juntos, fizeram o papel de checagem de tipo, transpilação e empacotamento do programa, finalizando assim o ciclo de desenvolvimento. Dando continuidade na descrição da temática, é importante ressaltar a linguagem universal trabalhada. Por ser um ambiente web, é natural a utilização do formato JSON (JavaScript Object Notation). Esse é um formato bem consolidado no meio da computação, e ele deriva da forma em que a linguagem JavaScript (também chamada formalmente de ECMAScript) trata os objetos e estruturas de dados. E como o ambiente web possui essa linguagem como cidadã de primeira classe, justifica a opção pelo JSON, além do fato dele ser aceito por diversos ambientes de execução e linguagens, sendo facilmente integrado em uma local que não seja ainda, sendo a linguagem universal mais adequada. Dessa forma, foram escolhidas as tecnologias adequadas e necessárias para a criação da aplicação resultante do estudo corrente.

### 3.3.2 Interface

Como resultado deste estudo, tem-se um ambiente para visualização de simulação, o que resulta em um cuidado especial com a questão da interface do programa. Para isso, foi utilizado o aplicativo figma para o esboço inicial do painel visual do ambiente. Com isso, pensou-se nas cores principais, que respeitando o nome do mesmo, foi escolhida uma paleta com tons de verde, com um tema escuro, exigindo assim diferentes tons de cinza, com o uso discreto da cor preta.

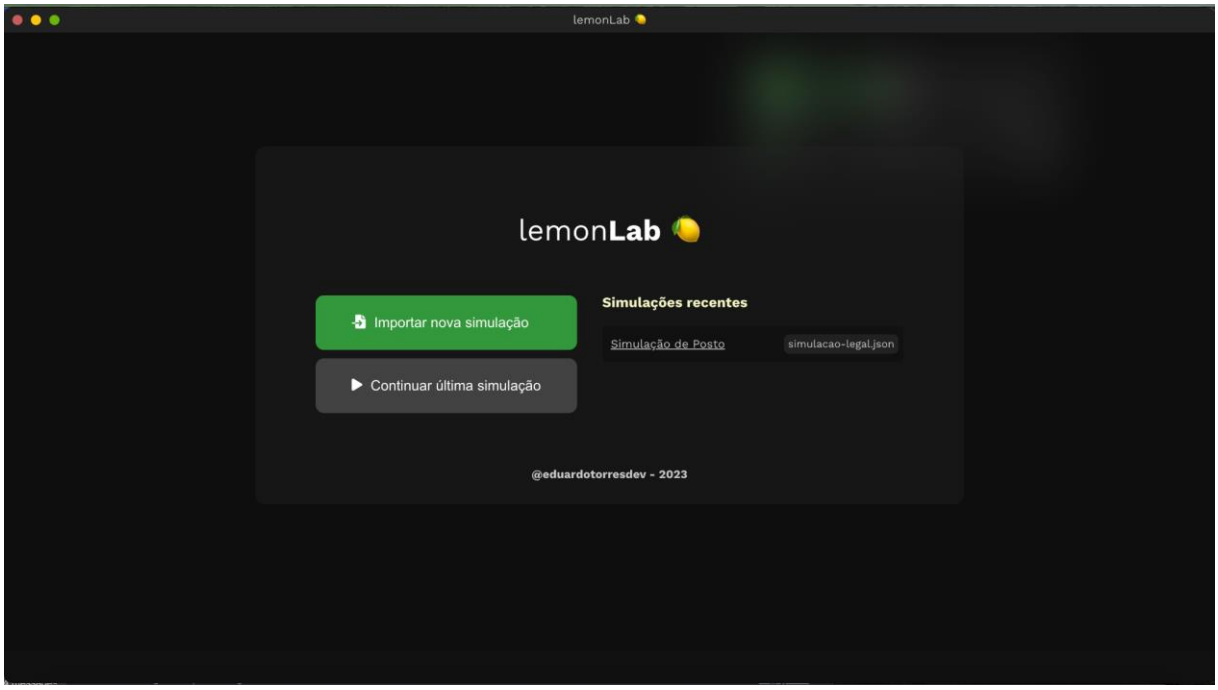
Figura 4- Paleta de cores do aplicativo



Fonte: Compilação do autor.

Com as cores escolhidas, começou-se a elaborar as telas necessárias para o funcionamento do programa. Para isso, estudou-se o fluxo do carregamento de uma simulação até a sua execução na aplicação, incluídos os passos após término de todo o fluxo. Como consequência, foram também desenhados os componentes de interface necessários para a interação do usuário com o sistema. Dessa forma, diversos elementos foram desenvolvidos, seguindo o padrão de cores juntamente com uma característica de flat design, que propicia um desenho mais sóbrio e minimalista de interface de usuário.

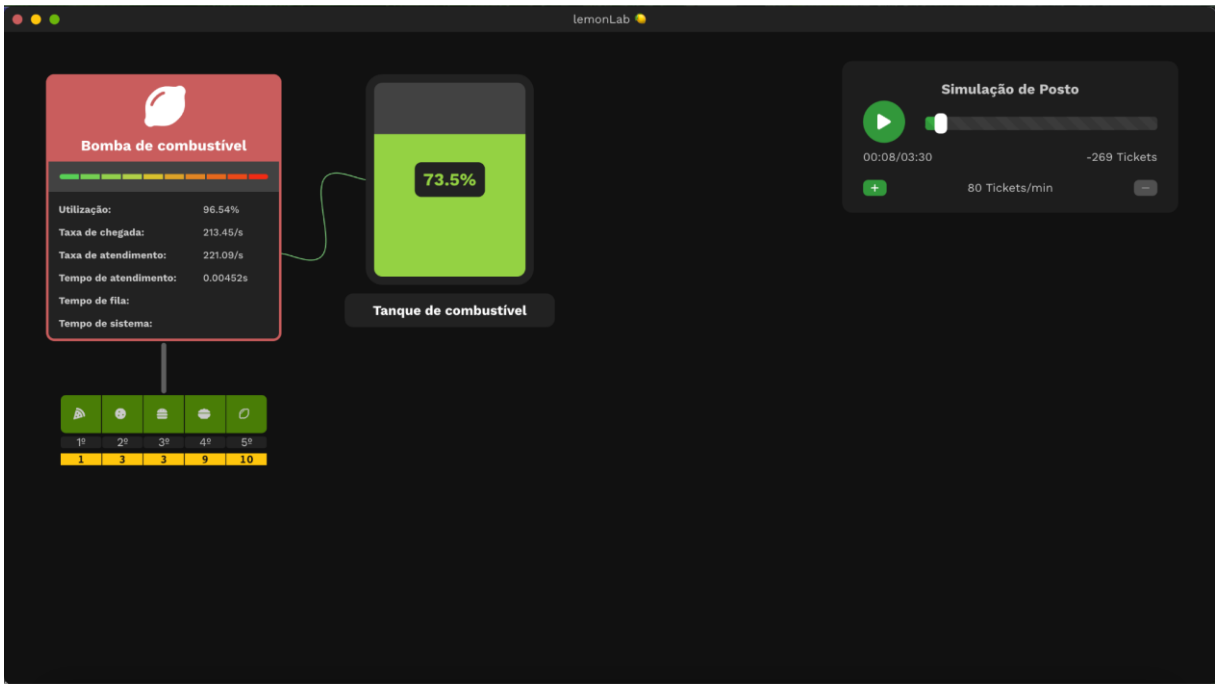
Figura 5- Tela inicial do programa rodando




Fonte: Compilação do autor

Através da figura 5, pode-se finalmente visualizar a interface resultante do programa. Trata-se da tela inicial, onde encontra-se as últimas simulações executadas, mostrando o seu título e o nome do arquivo carregado. Além disso, é possível notar dois botões em destaque, que possibilitam importar uma nova simulação ou reabrir a última simulação executada. Utilizando da ótica apenas visual, é possível notar um design simples, com conteúdo bastante destacado e organizado, corroborando com o objetivo de fornecer uma interface de fácil uso pelo usuário, fazendo também o consumo da paleta de cores proposta, tendo como ênfase, o logo da aplicação, o limão.

Figura 6- Aplicação com uma simulação em execução

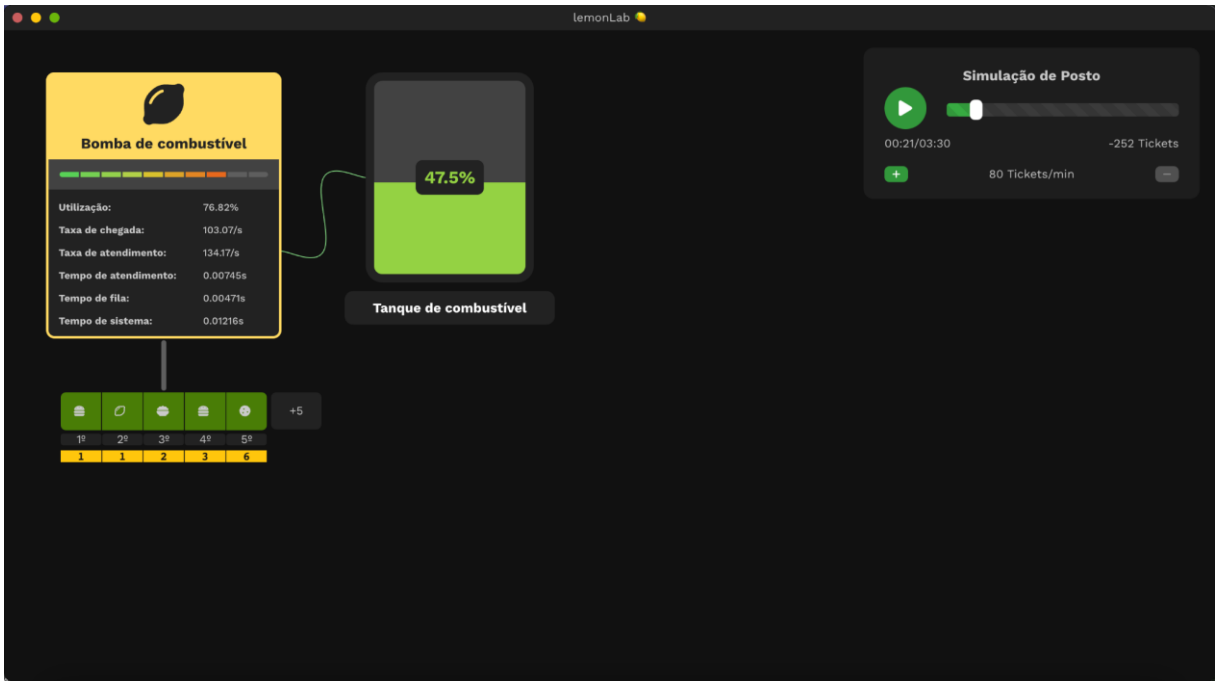



Fonte: Compilação do autor.

Dando continuidade a apresentação da interface, tem-se a figura 6, na qual mostra a tela do programa ao executar uma simulação. Nela, há uma gama superior de componentes especiais. Inicialmente, destaca-se o elemento recurso, que é representado por uma caixa com um limão de ícone, juntamente com o nome do recurso, nesse caso, bomba de combustível. Além desses detalhes, é possível observar as métricas em lista, sendo elas: utilização, taxa de chegada, taxa e tempo de atendimento, tempo de fila e tempo de sistema. E por fim, no aspecto do recurso, tem-se a fila de usuários que aguardam serem atendidos, juntamente com a sua posição e a sua prioridade (Este é um exemplo de fila com prioridade não-preemptiva).

Ademais, na figura 6, existe também a presença de um contêiner, que representa o tanque de combustível de um posto, com a sua porcentagem de preenchimento em destaque. Somando-se a isso, é possível observar no canto superior direito, a existência dos controles de reprodução, juntamente com o nome da simulação. É importante relevar, o cumprimento de algumas soluções propostas, sendo elas, os controles, a interface generalizada, tendo em vista que é possível com esses mesmos elementos visuais, representar inúmeras simulações e o funcionamento por tickets, através do relatório de simulação, que pode ser visto na reprodução, onde está escrito "80 tickets/min".

Figura 7-Variação da utilização




Fonte: Compilação do autor.

Por fim, para uma interface tornar-se mais amigável e orgânica, no sentido de ser natural a sua utilização, houve a proposta de trazer dinamismo para ela. E isso é efetuado através de mudanças de interface de forma gradual e suave. Através da figura 7, é possível notar a mudança de cor do recurso, que ao atingir uma utilização de 80%, tornou-se amarela, já que antes estava vermelha, servindo assim de alerta de utilização exagerada antes, e utilização moderada depois. Seguindo a ideia de utilização, há ainda um componente auxiliar, a barra de carga, logo acima das métricas, que quanto mais cheia, maior a utilização daquele recurso. Ainda mais, é possível notar a variação do nível do tanque de combustível, ou contêiner, representa assim, a quantidade de conteúdo que preenche esse espaço, tornando mais fiel a realidade.

Consolidando todo esse cenário visual apresentado, é possível notar que se concluiu o objetivo de desenvolver uma interface amigável, orgânica, simples e bem legível para o usuário utilizar. Após o desenho de toda a interface, que inclui os seus componentes e páginas, chegou o momento de começar a codificar as estruturas abstraídas nas soluções e nos objetivos, sendo a parte com maior complexidade e tempo de produção.

### 3.3.3 Desenvolvimento

Após a construção da interface de usuário referente ao programa, chegou o momento da codificação. Nesta etapa, junta-se os objetivos, as soluções, a interface e as tecnologias escolhidas para iniciar o processo de desenvolvimento. É importante ressaltar que, devido ao esquema aplicação-implementação utilizado, foi necessário codificar ambos. Como temática, tem-se o estudo de caso com a biblioteca Simpy, como mencionada anteriormente. Por conta disso, e pela forma como o esquema proposto funciona, a implementação foi realizada na linguagem Python, já que a biblioteca é nesta linguagem. Com esse momento apresentado, volta-se o foco para a aplicação.

Em primeiro lugar, foi necessário fazer a construção dos diretórios do projeto, sendo um desktop, referente ao programa propriamente dito, e implementações, referente a todas as implementações existentes, no momento, apenas em Python. Com isso, pode-se instalar as dependências necessárias para o desenvolvimento pleno e a sua configuração. Após esses processos serem concluídos, foi possível iniciar a codificação do programa.

Nesse contexto, é mencionada a relação da aplicação, implementação e simulação, onde a primeira consome os dados da segunda, sendo que esta captura os mesmos da simulação, convertendo-os na linguagem universal escolhida, neste caso, o JSON. Por conta desse fluxo, foi inicialmente codificado a implementação em python, que é responsável por coletar os dados de uma simulação Simpy, armazenando-os no relatório de simulação, composto por tickets, cada ticket contendo o estado da simulação no momento que foi registrado. Após essa coleta, e persistência em memória, a implementação fornece um método de salvamento, que gera o relatório no formato JSON.

Dando continuidade, com o aspecto referente à implantação pronta, segue-se o caminho para a aplicação. Nesse sentido, codificou-se como prioritário a tela de inicial, onde é possível importar uma nova simulação, ou seja, o relatório já no formato de linguagem universal, aquele entregue pela implementação. Após esse passo, já com a simulação podendo ser lida, foi iniciada a construção da tela de execução da simulação.

Já dentro dessa etapa, foram feitas micro iterações, de forma incremental, de toda a estrutura da aplicação. Isso resulta na construção dos componentes relevantes

para o funcionamento do aplicativo em si, como navegação, histórico de simulações de que já foram abertas, interações de arrastar os elementos na interface. Assim, como foi feito na implementação, que precisou adicionar os conteúdos relevantes a serem mostrados na interface, como por exemplo, a captura dos dados da fila.

Adentrando na parte mais avançada do desenvolvimento, foi realizada a construção da calculadora de métricas, no âmbito da implementação, assim como a captura das filas com prioridades, sendo preemptivas ou não. Desviando o foco para a aplicação, a atenção foi dada na codificação dos controles de reprodução de simulação e a apresentação das métricas resgatadas. Além disso, as animações dos elementos constituintes da interface generalizada, sendo eles, contêiner, fila e recurso, também foram programadas.

É importante ressaltar, que a fase mediana até o final do desenvolvimento, mostrou-se um grande desafio. Isso se deve ao fato da captura de dados não ser algo trivial. A biblioteca Simpy possui seus atributos públicos e protegidos, e fazer o uso desses primeiros, para montar o relatório de simulação, é bastante limitado. Por conta disso, um estudo a mais foi necessário para poder capturar os dados de forma confiável, e que mantivesse a integridade da simulação.

Além disso, houve uma complexidade considerável no lado da aplicação, dando ênfase em estruturas e interações específicas, sendo elas, o conector dos recursos e contêineres, e a interação arrastar os elementos. A primeira é devido ao desenho do conector necessitar um certo processamento gráfico, que não poderia ser abusado para que não houvesse perda de desempenho do programa, ou seja, que fosse otimizado. Já considerando a segunda, entra a questão da interação entre os componentes no momento da interação, além da preocupação do desempenho, focando novamente na otimização. Dessa forma, por conta desses tópicos apresentados, houveram grandes desafios no desenvolvimento desse projeto, que tornou-se viável após a transposição deles.

Por fim, o desenvolvimento da aplicação e da implementação foi realizado com sucesso, agregando aquelas soluções propostas, atendendo os objetivos. Ainda mais, o desenho da interface foi codificado cumprindo com todas as restrições propostas, juntamente com os requisitos. Com visto também, não foi um processo de desenvolvimento de software simples e curto, foi longo e complexo, como consequência da busca em atender todas as necessidades criadas, ou seja, obtendo a maior qualidade de programa possível neste aspecto.

### 3.3.4 Extensibilidade como pilar

No ponto de vista de atingir o maior número de usuários possíveis com o lemonlab, ele foi pensado tendo como base fundamental, a extensibilidade, com esta se diferenciando na visão da aplicação e implementação. Dando destaque para esta primeira, esse fundamento acabou sendo elaborado de forma natural, devido ao fato do aplicativo consumir um arquivo com uma linguagem universal, abrindo espaço para o acoplamento à qualquer simulação. Em contrapartida, no escopo da implementação, por ela ser a tradutora do código nativo para o código universal, são necessárias estratégias para possibilitar essa função, além de ser preciso deixar espaço para extensão, dando um código aberto a acoplamentos.

No contexto da implementação, vale ressaltar primeiramente a maneira em que as métricas principais foram calculadas, deixando ao mesmo tempo, espaço para calcular métricas que não foram cobertas inicialmente. Para isso, se implementou uma calculadora, seguindo o padrão de desenvolvimento orientado a objetos, acatando assim o cumprimento da funcionalidade, além de permitir a extensibilidade. Para melhor visualização, desenvolveu-se as seguintes classes para realizar os cálculos, como mostrado na tabela 5.

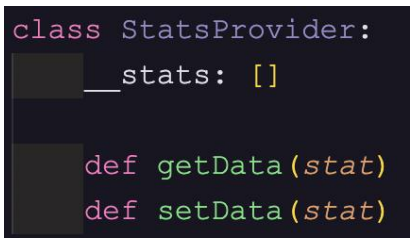
Tabela 5- Classes de cálculo de métricas

| Nome da classe | Função |
|---|---|
| BaseCalculator | A classe base, que fornece os métodos necessários para realizar os cálculos, além de configurar os valores internos, como o tempo em que o cálculo foi iniciado, e as estatísticas armazenadas no recurso. |
| Arrival Rate Calculator | Calcula a taxa de chegada |
| Await Queue Calculator | Calcula o tempo de espera na fila |
| Await System Calculator | Calcula o tempo de espera no sistema |
| Service Rate Calculator | Calcula a taxa de atendimento |
| Service Time Calculator | Calcula o tempo de atendimento |
| Usage Calculator | Calcula a utilização do recurso |

Fonte: elaborado pelo autor.

Dessa forma, as calculadoras de métricas foram elaboradas, além de abrir espaço para estender a classe base para implementar cálculos ainda não desenvolvidos. Ademais, com os valores obtidos, foi elaborado um sistema para armazenar os dados de cada recurso, registrando-os dentro dos tickets do sistema de relatório, transportando dessa forma, as métricas para a aplicação que irá visualizar a simulação. A imagem abaixo, mostra um pseudocódigo representando esse sistema citado, com a classe recebendo o nome de StatsProvider, onde mantém os dados e fornece métodos para manipular e retornar eles.

Figura 8- Classe de armazenamento das métricas




Fonte: Compilação do autor.

Por fim, elaborou-se soluções para a implementação que suprem as necessidades centradas no pilar de extensibilidade. Vale ressaltar que essas ideias não são regras, ou seja, outras implementações que forem criadas não precisam necessariamente segui-las, podendo até encontrar formas mais elegantes ou mesmo, mais eficientes. Assim, conclui-se o trabalho em cima do pilar supracitado, favorecendo o uso, além de também atrair futuras contribuições no projeto.

### 3.3.5 Código-Fonte

Ao fim do desenvolvimento, além da aplicação e implementação prontas, obteve-se o código-fonte resultante, que detém tanto os códigos base, quanto os códigos de configuração de ambiente de desenvolvimento. Toda essa codificação criada, é pública, aberta a contribuições, livre e sem quaisquer fins lucrativos, podendo ser acessada no repositório2 de códigos na plataforma github. Além disso, as

[2] "eduardotorresdev/lemon-lab" - https://github.com/eduardotorresdev/lemon-lab - Acessado em 9 de dez. de 2022

instruções de construção e publicação do programa também estão disponíveis neste mesmo local.

# 4 EXPERIMENTO: APURAÇÃO ELEIÇÕES 2022

Com a aplicação e a implementação construídas, chega-se o momento de realizar a experimentação. Para isso, foi escolhido o contexto das eleições de 2022, o principal assunto tratado no momento da escrita desse estudo, especificando-se na apuração das eleições, ou seja, o sistema em que entrega os resultados das eleições. Isso porque, ele traz vários aspectos onde a avaliação de desempenho atua, pois a otimização é uma questão crucial na hora desta entrega.

## 4.1 APRESENTAÇÃO

Colocando o contexto apresentado sobre a ótica de servidor http (Hypertext Transfer Protocol), onde o usuário realiza uma requisição e o servidor responde com o conteúdo da página, neste caso, a apuração das eleições, tem-se o cenário a ser testado. Isso se deve a quantidade de usuários acessando esse conteúdo, que ao realizar esse consumo, faz com que o servidor fique ocupado, processando o pedido, até o momento da entrega, ficando disponível novamente. Supondo que um servidor atenda um usuário por vez, e cada usuário consome o tempo aproximado de 200 milissegundos do servidor, para que seja atendido, temos uma situação em que a avaliação de desempenho entra, ou seja, que uma simulação se apresenta. Nesse sentido, caso 1000 usuários requisitem o acesso, tem-se uma demanda maior do que o que servidor atende, criando-se uma fila, entrando dessa forma, a teoria das filas, para maior otimização possível no atendimento desses usuários. Dessa forma, tem-se a apresentação do experimento que vai ser implementado adiante, que consiste em servidores http, que retorna o conteúdo da apuração das eleições de 2022, que atendem os usuários que necessitarem dessa informação.

## 4.2 IMPLEMENTAÇÃO

Nesta etapa, é necessário inicialmente elaborar a simulação no estudo de caso deste trabalho científico, na biblioteca SimPY. Para obter parâmetros de comparação, elabora-se três simulações: com um servidor, dois servidores e três servidores, atendendo entre 15 a 20 usuários por segundo, com um tempo de atendimento de 500 milissegundos por parte do servidor, com tempo de simulação de 1 hora e meia. Dessa

forma, obteve-se uma configuração de simulação, já com os recursos do lemonlab, como pode ser visto na figura 9, além dos parâmetros na figura 10.

Figura 9- Configuração inicial da simulação

```
# Setup and start the simulation
print('Web Servers')
random.seed(RANDOM_SEED)

env = simpy.Environment()
server1 = simpy.PriorityResource(env, 1)
# server2 = simpy.PriorityResource(env, 1)
# server3 = simpy.PriorityResource(env, 1)

lemon = lemonlab.Start("Apuração eleições 2022")
servidor1 = lemon.createResource(lemonlab.ResourcePriority("Servidor 1"))
# servidor2 = lemon.createResource(lemonlab.ResourcePriority("Servidor 2"))
# servidor3 = lemon.createResource(lemonlab.ResourcePriority("Servidor 3"))

env.process(user_generator(env, [server1], [servidor1]))

# Execute!
env.run(until=SIM_TIME)

lemon.save(
    'eleicoes-2022',
    '/Users/eduardo.torres/Desktop/LemonLabProject/implementations/python'
)
```

Fonte: Compilação do autor.

Figura 10- Parâmetros da simulação

```
TIME_TO_USER_PAGE = 0.5    # Tempo para o servidor atender em segundos
T_INTER = [1, 5]           # Cria uma usuário a cada [min, max] décimos de segundos
SIM_TIME = 60              # Tempo da simulação em segundos
```

Fonte: Compilação do autor.

Após todas essas etapas realizadas, roda-se o código em python, para executar a simulação e gerar o relatório da mesma. Após isso, tem-se a saída de terminal como mostrado na figura 11.

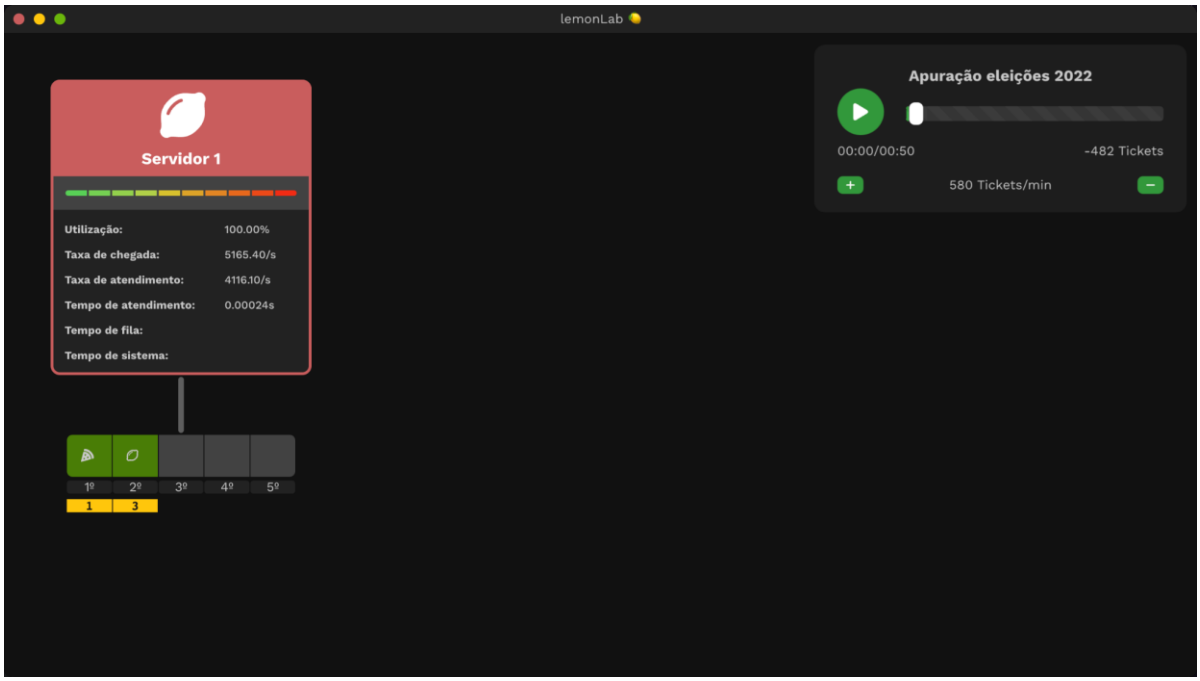
Figura 11- Saída de terminal após executar a simulação com o relatório

```
eduardo.torres@WP090191 python % python example.py
Web Servers
Simulation file created with success ✨
```

Fonte: Compilação do autor.

Dessa forma, já é possível visualizar a simulação na aplicação desse estudo, bastando abrir o arquivo resultante. Para isso, basta ir no programa, na tela inicial, clicar em "importar uma nova simulação" e escolher o arquivo. Após abrir o relatório, pode-se ver o servidor, o nome da simulação e as métricas, como mostrado na figura 12.

Figura 12- Simulação aberta no lemonlab




Fonte: Compilação do autor.

Com isso, foi possível criar uma simulação na sua linguagem específica, nesse caso, python, sem que haja qualquer apoio visual e verbal de alto nível. Desse modo, acoplou-se a implementação do lemonlab, para poder gerar o relatório de simulação. Por fim, carrega-se o relatório na aplicação do estudo, para poder visualizar com nível alto de abstração, obtendo o resultado da imagem acima, podendo usar os controles de reprodução, assim como analisar a fila e as métricas obtidas. Isso tudo foi apresentado com o exemplo inicial, com apenas um servidor, mas todo o processo executado é repetido com os outros cenários, com dois e três servidores, para motivos de comparação.

## 4.3 AMOSTRAGEM DE CASOS

Com a implementação realizada, podendo assim, gerar o relatório que será carregado pela aplicação, montou-se três casos diferentes para os mesmos parâmetros de consumo dos servidores da apuração das eleições de 2022. Dessa forma, obtêm-se três amostragens de dados para testagem da aplicação, possibilitando assim, a análise da viabilidade de todo o estudo realizado.

Com isso, primeiramente, apresenta-se os parâmetros de consumo fixos para os testes dos casos, como é mostrado na tabela 5.

Tabela 6- Parâmetros de consumo

| | |
|---|---|
| Tempo para o servidor atender a requisição (em milissegundos) | 500 ms |
| Intervalo de chegada de cada usuário | 1 a 5 segundos |

Fonte: elaborado pelo autor.

Além disso, é necessário também demonstrar os três casos a serem testados, como pode ser visto na tabela 6, valendo lembrar que em todos os casos, há o uso de fila com prioridade não-preemptiva.

Tabela 7- Casos de testagem

| | |
|---|---|
| Caso 1 | um servidor |
| Caso 2 | dois servidores |
| Caso 3 | três servidores |

Fonte: elaborado pelo autor.

Por fim, com os três casos montados, utilizando os mesmos parâmetros de consumo, executa-se os testes propriamente ditos, gerando os respectivos relatórios para cada montagem, analisando tanto a saída da implementação, quanto a saída da aplicação, coletando assim os resultados, que são apresentando logo adiante.

# 5 RESULTADOS OBTIDOS

## 5.1 CASO 1 - UM SERVIDOR

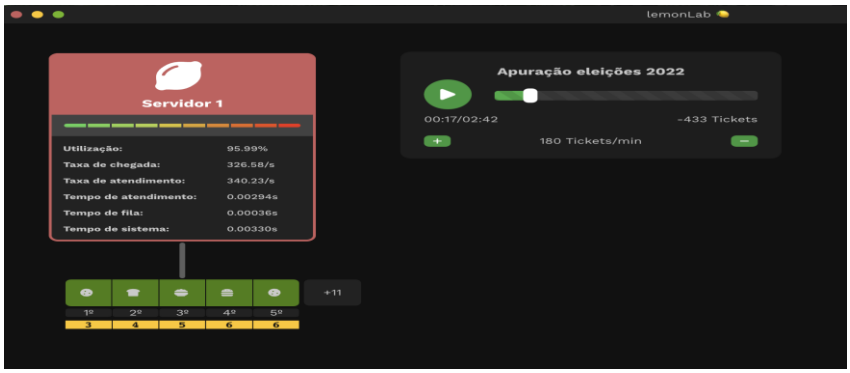
Em primeiro lugar, é realizada a testagem do caso 1, onde, como foi visto, é utilizado apenas um servidor para atender todas as requisições dos usuários que desejam consultar a apuração das eleições de 2022. Executando a implementação com essa configuração, obtém-se o relatório a ser carregado na aplicação, tendo a a estrutura de recursos vista na figura 13. Assim, corrobora-se com a proposta do caso do experimento, permitindo assim, prosseguir com o teste.

Figura 13- Recursos do relatório do caso 1


Fonte: Compilação do autor.

Dando continuidade, chega-se o momento de visualizar a simulação gerada no lemonlab. Assim, de início, analisa-se, através da figura 14, os primeiros 10% de execução, tendo algumas informações relevantes para a discussão.

Figura 14- 10% da simulação do caso 1

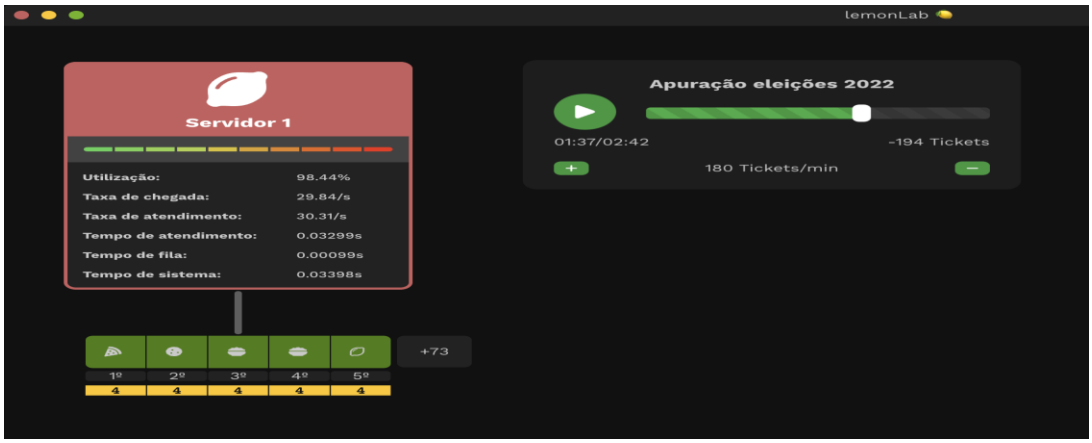



Fonte: Compilação do autor.

Nela, é possível notar uma utilização alta do servidor, que se refere a quanto tempo do tempo o recurso está sendo consumido, neste caso, 95.99%. Além disso, é facilmente notável a fila de usuários que aguardam para serem atendidos pelo servidor, que somando todos, contabiliza-se 16 pessoas aguardando. Colocando essas duas informações apresentadas sob o mesmo plano, conclui-se a existência de uma certa sobrecarga do servidor, tendo em vista que mesmo sendo consumido quase 100% do tempo proposto, ainda assim, uma fila considerável foi criada.

Em contraste com a ótica macroscópica supracitada, existe também a ótica microscópica do teste, onde analisa-se por exemplo, a métrica de taxa de atendimento, que resulta em 340,23 atendimentos por segundo, o tempo de fila, com o valor de 0.00036 segundos, além de outras métricas que podem ser visualizadas. Além disso, esses valores serão posteriormente comparados ao final da execução dos outros casos propostos, sendo relacionados ao final do estudo.

Figura 15- 60% da simulação do caso 1



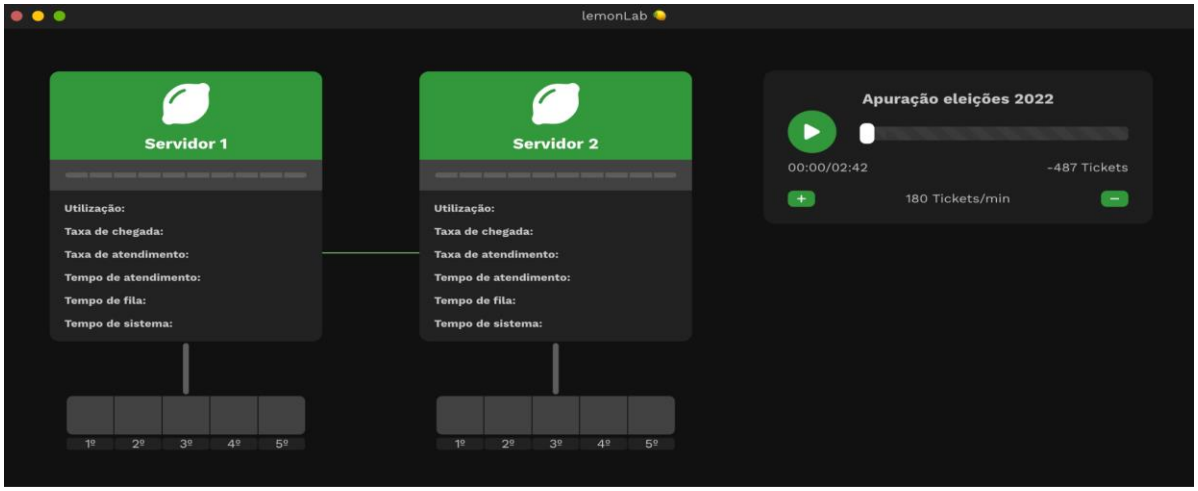

Fonte: Compilação do autor.

Ademais, vale mencionar um percurso mais adiante da visualização da simulação, atingindo agora cerca de 60% da reprodução total. Na visão macroscópica, é possível corroborar a afirmativa anterior de que o servidor se encontra sobrecarregado, com utilização em 98.44% e 78 usuários aguardando na fila, agravando o cenário apresentado nos 10%. Além disso, é interessante afirmar que na análise microscópica também confirma o agravamento, diminuindo a taxa de atendimento para 30,31 atendimentos por segundo e tempo de fila de 0,00099 segundos. Dessa forma, conclui-se a testagem do caso 1, que possui apenas um servidor para atender todas as requisições.

## 5.2 CASO 2 - DOIS SERVIDORES

Com a testagem do caso 1 finalizada, há o prosseguimento para o caso 2, em que são utilizados dois servidores para atender às requisições dos usuários, que em teoria, deve aliviar a carga atribuída ao serviço de apuração das eleições, tendo em vista que será os acessos serão balanceados entre ambos. Com isso, gera-se o relatório da simulação com o novo caso, sendo então, carregado dentro da aplicação, como é possível observar na figura 16.

Figura 16- Caso 2 carregado na aplicação

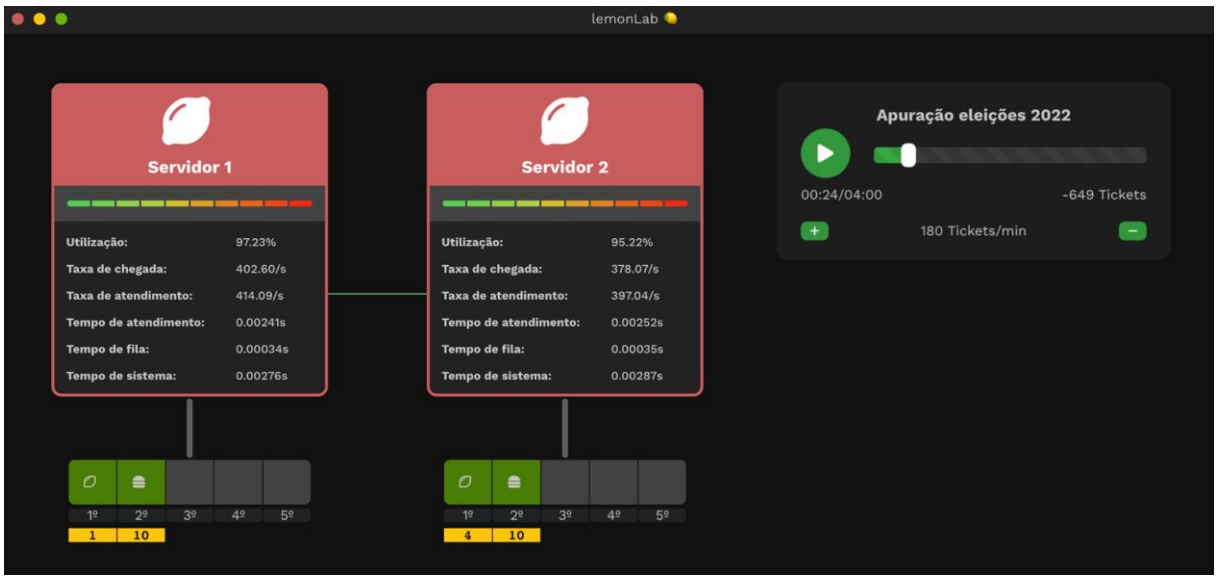



Fonte: Compilação do autor.

Como contato inicial, pode-se observar a existência dos dois servidores, assim como todas as suas métricas e respectivas filas, tendo assim, recursos separados,

prontos para atenderem as demandas necessárias. E para seguir o padrão de teste, estabelece-se a reprodução da simulação em aproximadamente 10% do tempo total, acompanhando o que foi realizado no primeiro caso.

Figura 17- 10% de reprodução do caso 2




Fonte: Compilação do autor.

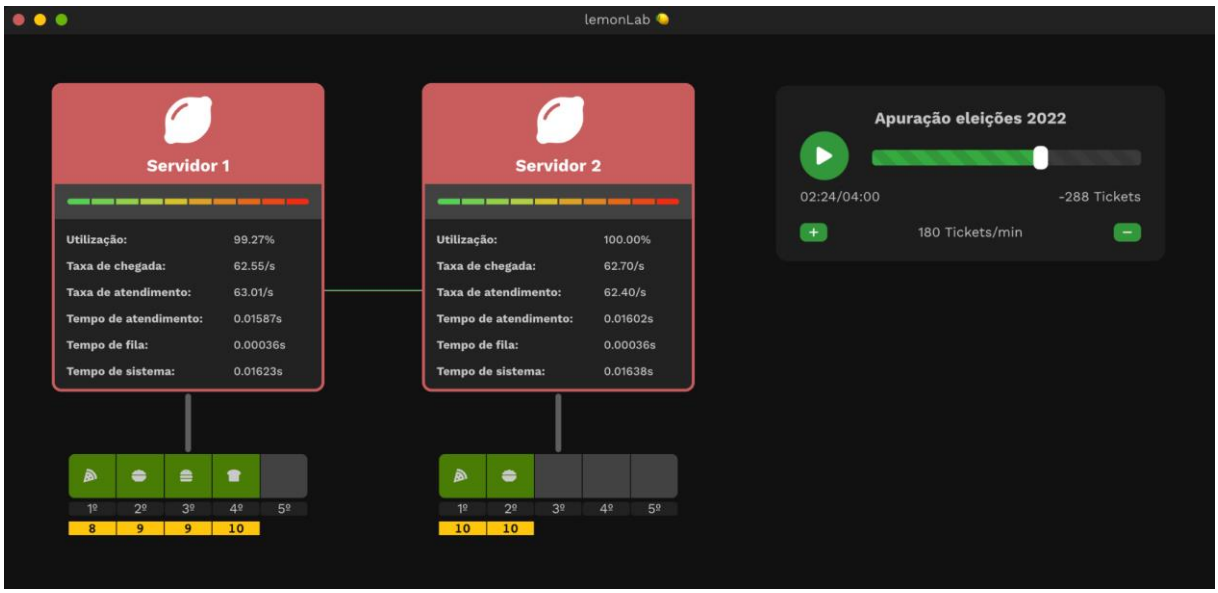
Através da figura 17, é possível analisar diferenças em relação ao primeiro caso, sendo elas não tão notáveis em uma visão superficial. Ao visualizar, percebe-se que a utilização do sistema continua alta, tendo ambos os servidores um valor acima de 95% de utilização, que poderia ser interpretada como uma sobrecarga. Entretanto, outros fatores devem ser levados em conta ao efetuar uma avaliação de desempenho.

Nesse sentido, revela-se dois dados relevantes para esta análise, taxa de atendimento e o tamanho da fila dos servidores. Assim, trazendo os valores do primeiro caso para realizar a comparação, nota-se um aumento na taxa de atendimento, antes eram aproximadamente 340 atendimentos por segundo, aumentando para cerca de 400 atendimentos por segundo, uma diferença considerável. Além disso, o tamanho da fila é observado, resultando em apenas quatro clientes aguardando para serem atendidos, um valor menor que os dezesseis contabilizados no caso 1, considerando a mesma porcentagem da simulação.

Dessa forma, torna-se palpável a melhoria ao adicionar um servidor no sistema de apuração das eleições, corroborando com a afirmativa teórica em que, ao realizar

essa adição, a carga deferida no sistema seria balanceada aos recursos disponíveis. Além disso, vale ressaltar a análise realizada de forma simples e direta, através da interface fornecida pela aplicação lemonlab, ressaltando novamente o cumprimento da proposta apresentada pelo projeto de estudo. Por fim, continua-se a execução do teste, avançando nesse momento para cerca de 60% da reprodução da simulação, seguindo o primeiro caso.

Figura 18- 60% de reprodução do caso 2




Fonte: Compilação do autor.

Nesse contexto, mostra-se a figura 18, que contém a reprodução em cerca de 60% do tempo total da simulação. Assim, pode-se observar, novamente, uma elevada utilização do sistema, acompanhada com métricas que apresentam melhorias de desempenho. Seguindo a análise inicial, a taxa de atendimento permanece superior àquela mostrada no caso 1, com apenas um servidor, tendo um aumento de cerca de 206%, corroborando o incremento de eficiência esperada de um balanceamento de carga com dois servidores no sistema. Além disso, é discrepante a diferença de fila, onde, enquanto no primeiro teste haviam 78 usuários aguardando para serem atendidos, neste caso, há apenas 6 usuários aguardando o atendimento em fila, resultando assim, em uma melhoria expressiva ao se considerar o desempenho do sistema.

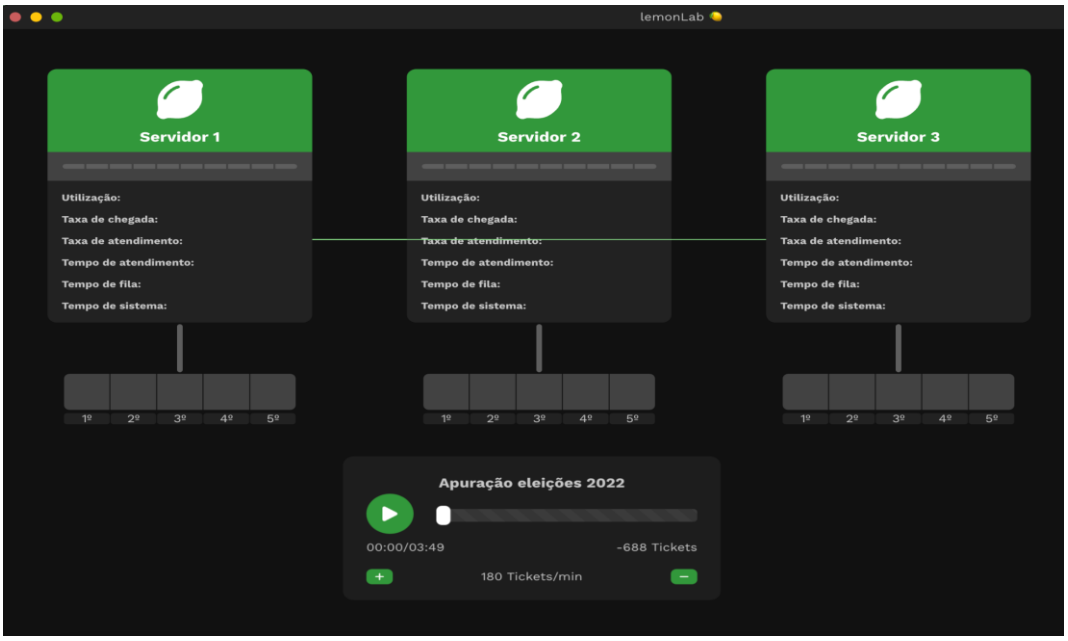
Dessa forma, finaliza-se a testagem do caso 2, que possui dois servidores para dividir a carga do sistema de apuração das eleições, corroborando com a premissa de

uma eficiência superior em relação ao primeiro caso, tendo em vista, os melhores números apresentados neste teste. Além disso, fica destacado novamente, o cumprimento dos objetivos desse estudo, que visa facilitar a interpretação e análise dos dados de uma simulação de forma visual, afirmativa que foi confirmada na execução dos experimentos até o momento, dando continuidade para o último caso colocado.

## 5.3 CASO 3 - TRÊS SERVIDORES

Por fim, chega-se ao teste final, onde há como proposta, o melhor desempenho em relação às testagens anteriores, devido ao balanceamento de carga entre três servidores para o sistema de apuração da eleição. Com isso, seguindo a metodologia levantada na experimentação do projeto de estudo, adiciona-se mais um servidor na simulação, para assim gerar o relatório. Após esta etapa, inicia-se a o lemonlab para realizar o carregamento da simulação. Desse modo, tem-se o cenário como mostrado na figura abaixo.

Figura 19- Simulação com três servidores

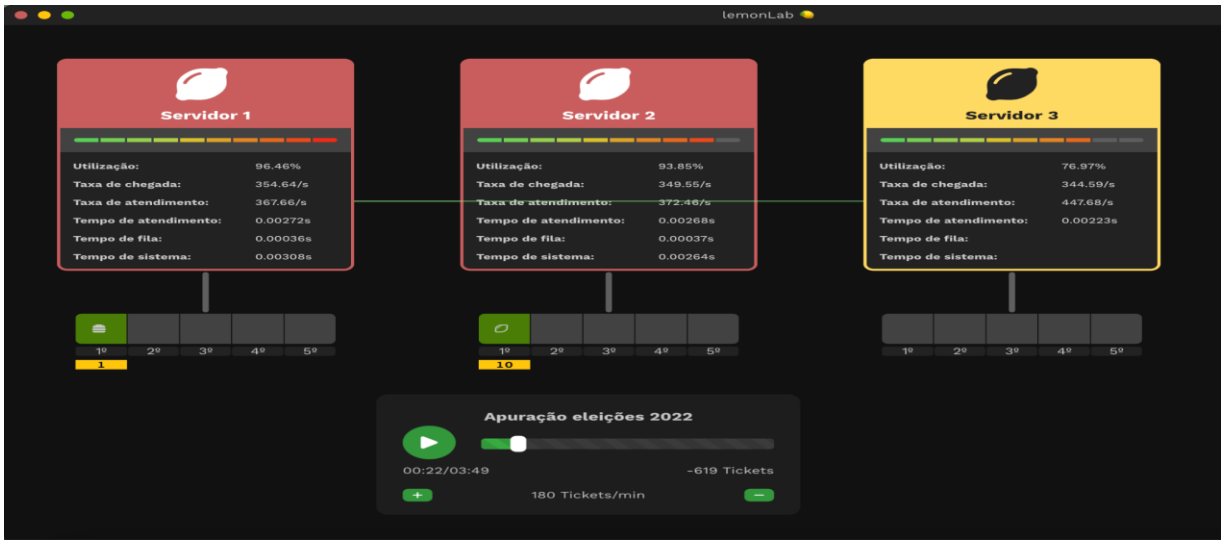



Fonte: Compilação do autor.

Com a simulação configurada e carregada, é possível realizar a análise dos valores da simulação. Seguindo a padronização, coloca-se a reprodução em 10% do

tempo total, para então visualizar os dados fornecidos na interface. Nesse sentido, apresenta-se a figura 20.

Figura 20- 10% de reprodução da simulação do caso 3




Fonte: Compilação do autor.

Em primeiro lugar, vale ressaltar a diferença nítida em relação a todos os outros testes, não necessitando uma visualização aprofundada dos dados para concluir um desempenho superior. Isso se deve aos recursos visuais apresentados, como exemplo, a utilização dos recursos na cor amarela, apresentando uma diferença não numérica para o usuário. Entretanto, para poder realizar a comparação com os resultados obtidos, analisa-se de forma mais detalhada os valores vistos na simulação.

Nesse sentido, cabe destacar as métricas encontradas nesse teste, começando pela utilização. Comparando com os casos anteriores, tem-se uma utilização destoante, tendo dois recursos com cerca de 96% e um com aproximadamente 77%, contrastando com a carga de utilização superior a 95% em todos os testes precedentes. Vale ressaltar que essa informação não é suficiente para afirmar se tal serviço está realmente sendo sobrecarregado ou não, apenas direciona se o recurso está com uma baixa carga ou uma alta carga de atendimento, podendo por exemplo, estar sendo muito requisitado, mas sem criar uma fila de espera, obtendo assim, um caráter de não sobrecarga.

Dando continuidade à análise, parte-se para a observação das métricas de dimensão microscópica, realçando a taxa de atendimento e o tamanho da fila. Neste

momento é possível notar uma melhoria considerável ao comparar com as outras experimentações. Dessa forma, utilizando em primeiro lugar a taxa de atendimento, observa-se um crescimento de forma indireta. Isso porque a taxa numérica em cada recurso possui um valor menor do que no caso 2. Entretanto, isso não significa que perdeu desempenho e sim que possui um alto índice de atendimento e, pelos usuários estarem mais divididos, acaba atendendo um número inferior de requisições.

Por fim, passa-se o foco para a próxima métrica, a fila de espera. É notável o tamanho inferior ao caso anterior, apesar de ser uma diferença de uma pequena proporção. Com isso, somando-se os dados de utilização, taxa de atendimento e a fila do serviço de apuração das eleições, é possível concluir que o mesmo não está sobrecarregado, possui uma melhoria pequena em relação ao caso 2, uma considerável em relação ao primeiro caso, e que há recurso disponível para caso haja um pico de acesso maior do que foi parametrizado, tendo assim, uma tolerância à uma carga superior de requisições.

Figura 21- 60% de reprodução da simulação do caso 3


Fonte: Compilação do autor.

Em conclusão, analisa-se a reprodução em 60% do tempo total. Nela, é possível observar uma utilização acima de 99%, que não representa novamente, uma

sobrecarga, e sim que os recursos estão sendo plenamente acessados no sistema de apuração. Como na análise anterior, há um valor menor na taxa de atendimento, devido a um maior balanceamento das requisições entre os servidores. Por fim, nota-se uma fila com apenas um usuário aguardando para ser atendido, enquanto no caso anterior haviam 6 usuários. Desse modo, corrobora-se a afirmação feita no início da observação deste caso, um ótimo desempenho do sistema com uma tolerância para cargas superiores ao planejado, tendo assim, uma reserva de recursos.

## 5.4 TABULAÇÃO DOS CASOS E SEUS RESULTADOS

Com a conclusão dos testes propostos ao projeto de estudo, divididos em três casos, juntamente com as comparações entre os respectivos desempenhos, construiu-se a tabela 8, levando em conta a reprodução avançada, ou seja, em 60%. Além disso, vale ressaltar que as métricas capturadas nos casos com múltiplos servidores, são as médias calculadas.

Tabela 8- Comparação dos casos e seus resultados

| Métrica | Um servidor | Dois servidores | Três servidores |
|---|---|---|---|
| Utilização em % | 98,44 | 99,63 | 99,81 |
| Taxa de chegada por segundo | 29,84 | 62,62 | 60,23 |
| Taxa de atendimento por segundo | 30,31 | 62,70 | 59,59 |
| Tempo de atendimento em segundos | 0,03299 | 0,01595 | 0,01678 |
| Tempo de fila em segundos | 0,00099 | 0,00036 | 0,00035 |
| Tempo de sistema em segundos | 0,33398 | 0,01631 | 0,01713 |
| Quantidade total de usuários na fila | 78 | 6 | 1 |

Fonte: elaborado pelo autor.

Dessa forma, analisando a tabela construída a partir dos dados coletados, é possível notar a evolução no desempenho de forma bem nítida, justificando as mudanças de estrutura no sistema de apuração das eleições, em relação a adição de servidores.

# 6 CONCLUSÃO

Após a realização dos experimentos propostos no trabalho apresentado, chega-se o momento de observar todo o conteúdo que foi coletado, selecionado, testado e analisado até o momento. Com isso, recapitula-se a partir da prospecção do projeto, dando ênfase ao caminho inicial que foi seguido para toda a construção elaborada. Dessa maneira, inicialmente destaca-se a problemática que possibilitou todo este desenvolvimento.

Assim, como problema encontrado no estudo corrente, houve o alto custo de realizar simulações, especificamente, aquelas construídas através de linguagens de programação, como Python, escolhida no projeto trabalhado. Além disso, a grande dificuldade de transmitir os resultados das simulações para pessoas com nível de conhecimento especializado baixo, mostrava-se também como um dilema. Dessa forma, o grande desafio era simular e transportar as informações coletadas de uma codificação de mais baixo nível e de alto aprofundamento, para um cenário de baixo aprofundamento, buscando a maior distância do código de computador possível, tornando assim, mais próximo da visão humana.

Nesse sentido, uma das principais soluções encontradas foi a elaboração de uma aplicação composta de uma interface gráfica, ou seja, uma linguagem não-verbal de ampla abrangência no entendimento das pessoas envolvidas nas simulações. Ademais, além de atender os visualizadores das simulações, também era necessário colocar o desenvolvedor como objetivo principal na elaboração do projeto. Com isso, codificou-se o padrão aplicação-implementação, para permitir de maneira fácil e simplificada a integração das simulações ao lemonlab, a aplicação desenvolvida no estudo. Dessa forma, em teoria, os dois lados, fornecedor e consumidor, seriam atendidos de maneira satisfatória, cumprindo com os objetivos do projeto.

Em contrapartida, a visão conceitual não é suficiente para afirmar se as soluções apresentadas são viáveis no mundo real, havendo assim um contraste com o mundo das ideias. Dessa forma, foi desenvolvida uma aplicação, o lemonlab, que faria o papel da visualização gráfica, fornecendo uma linguagem universal para permitir a integração de diferentes simulações, podendo ser originadas de distintas linguagens de programação, tendo assim, a maior compatibilidade possível. Com a ideia do programa codificada no mundo real, foram realizados os experimentos para analisar se as soluções cumpriam com os objetivos elaborados.

Como consequência, houve a interpretação dos resultados obtidos nos testes, sendo experimentado um sistema hipotético de apuração das eleições de 2022. E através disso, pôde-se analisar as melhores estruturas para atender os usuários que requisitavam aos servidores da apuração, podendo classificar o menos ao mais eficiente, inclusive sintetizando a informação de qual teria uma margem de aceitação de uma carga de usuários acima do planejado, obtendo assim, uma tolerância de acesso. Com isso, avaliou-se o desempenho de um sistema utilizando o lemonlab e todas as ferramentas que ele oferece.

Desse modo, é possível realizar a seguinte afirmação: as soluções foram colocadas em prática e conseguiram cumprir com as necessidades a que foram apresentadas. Isso se deve, a análise fácil e simples observada nas experimentações descritas no estudo corrente por meio da aplicação criada, que caso fosse realizada por meio da linguagem original, utilizando os métodos antigos para visualizar as métricas, teria uma dificuldade bastante acentuada, gastando um maior tempo, resultando em um maior custo. Assim, o lemonlab supre as necessidades levantadas no projeto, economizando tempo, diminuindo o custo.

Entretanto, há espaço para melhorias na aplicação, complementando o seu funcionamento. Em primeiro lugar, permitir exportar os dados da simulação em um determinado tempo de execução em um formato tabular, facilitando a transposição dos dados para outros programas, como os de planilhas, para então analisar, processar, elaborar gráficos, entre outras ações. Além disso, como estratégia de otimização do relatório de simulação, poderia ser feito um mecanismo de taxa de captura de tickets, limitando, por exemplo, a captura máxima de 30 tickets por segundo, permitindo diminuir o tamanho final do relatório, além de permitir regular a precisão do mesmo, ou seja, quanto maior taxa, maior precisão.

Desta maneira, finaliza-se o estudo, com a apresentação do dilema, as possíveis soluções, a execução das mesmas, a elaboração de testes e interpretação dos resultados. Com isso, percebeu-se a coesão entre os dados apresentados, mostrando assim, a integridade do estudo, juntamente com a viabilidade do programa desenvolvido. Por fim, houve a abertura para possíveis melhorias da aplicação, aumentando assim, o horizonte de casos de uso do lemonlab, aumentando a força do projeto desenvolvido e apresentado.

## REFERÊNCIAS

DEFENSE, **Department of. Modeling and Simulation Body of Knowledge**. Disponível em: https://www.acqnotes.com/Attachments/DoD%20M&S%20Book%20of%20Knowledge.pdf. Acesso em: 20 de abril de 2023.

LIMA, Thiago F. Melo de et al. **Modelagem de sistemas baseada em agentes:** alguns conceitos e ferramentas, In: Simpósio Brasileiro de Sensoriamento Remoto, nº 14, 2009, Natal. Anais. Florianópolis: INPE, 2009, p. 5279-5286.

GUTERRES, Marcelo Xavier. **SIMULAÇÃO - Módulo 01. UNIPAMPA.** Disponível em: https://cursos.unipampa.edu.br/cursos/engenhariadeproducao/files/2016/08/modulo-01-sim-a1-notas-de-aula-guterres-2017.pdf. Acesso em: 20 fev. 2022.

BRESSAN, Graça. **Modelagem e Simulação de Sistemas Computacionais.** Disponível em: https://edisciplinas.usp.br/pluginfile.php/2446242/mod_resource/content/1/modsim03.pdf. Acesso em: 10 abr. 2022.

ALMEIDA, João Flávio de Freitas. **Simulação por eventos discretos:** Teoria & prática. IFMG. Disponível em: https://cursos.unipampa.edu.br/cursos/engenhariadeproducao/files/2016/08/apostila-sim-simulacao-por-eventos-discretos.pdf. Acesso em: 31 de jan. 2022.

GAVIRA, Muriel de Oliveira. **Simulação computacional como umaferramenta de aquisição de conhecimento**. Dissertação (Mestre em Engenharia de Produção) – Universidade de São Paulo, São Carlos, 2005.