

Sistema de Controle de Acesso de Pessoas em um Shopping

Bruna Michelly Cordeiro e Thalia Santos de Santana

Resumo

Uma corrida pelo desenvolvimento de aplicações utilizando Tecnologias de Informação e Comunicação (TIC) tem sido presenciada dia a dia devido a pandemia causada pelo COVID-19. Visando esse cenário, o propósito deste trabalho é desenvolver um sistema que seja capaz de realizar o controle de acesso de pessoas em um Shopping Center. Esse sistema deverá fornecer o monitoramento de entrada e saída de indivíduos pelos acessos principais, além de verificar a quantidade de pessoas presentes em cada andar e em cada loja, com a finalidade de evitar aglomerações em pontos específicos do ambiente. O presente trabalho será dividido em quatro partes sendo: I. Núcleo funcional do sistema composto pelos componentes e suas interações; II. Coordenação, consenso e controle de concorrência; III. Tolerância a falhas e replicação; IV. Aspectos de segurança da interação entre os componentes. Como resultado, espera-se prover a criação de um sistema distribuído com vistas a prover um controle em tempo real do fluxo de pessoas em uma instalação de shopping, tendo como referência o Passeio das Águas Shopping, situado em Goiânia-GO.

Introdução

Em 31 de dezembro de 2019, a Organização Mundial da Saúde (OMS) foi alertada sobre vários casos de pneumonia na cidade de Wuhan, província de Hubei, na República Popular da China. Tratava-se de um novo tipo de coronavírus que não havia sido identificado antes em seres humanos. Os coronavírus são a segunda principal causa de resfriado comum e, até as últimas décadas, raramente causavam doenças mais graves do que o resfriado comum em seres humanos. Ao todo, sete coronavírus humanos (HCoVs) já foram identificados. O novo tipo que foi nomeado em 11 de fevereiro de 2020, recebeu o nome de SARS-CoV-2. Esse novo coronavírus é responsável por causar a doença COVID-19 [1].

Devido a pandemia é necessário que haja um controle de acesso de pessoas em locais de uso público e coletivo, a exemplo de Shoppings Centers, com a finalidade de evitar aglomerações. O sistema de controle de acesso apresentado neste trabalho será desenvolvido por meio de simulação dos dados, tanto a nível de cadastro dos colaboradores autorizados à entrada, como funcionários e prestadores de serviços, quanto a nível de visitantes no contexto de movimentação de entradas e saídas.

Além da introdução, o trabalho está organizado do seguinte modo: a próxima seção apresenta o cenário de referência; posteriormente, há a descrição dos dispositivos; descrição dos requisitos definidos para o sistema; etapas e aspectos descritivos; metodologia empregada, com informes do status atual de cada parte da implementação, seguido das referências bibliográficas.

Um shopping da cidade de Goiânia foi escolhido como modelo para o projeto. Inaugurado em 2013, o Passeio das Águas Shopping é o maior centro de compras e lazer do estado de Goiás [3]. A Figura 1 apresenta uma visão geral do shopping que é composto por mais de 78 mil metros quadrados (m²) de Área Bruta Tributável (ABI). O local é administrado pela Aliansce Sonae [4].



Em sua estrutura, o shopping contém dois pavimentos, cerca de 283 lojas, sendo 15 âncoras e 1 hipermercado, além de 7 salas de cinema [3]. A Figura 2 apresenta a planta baixa do shopping com indicações das lojas e possíveis entradas de clientes e funcionários.



Figura 2 - Indicações das partes do shopping. Fonte: Adaptada pelas autoras (2020).

Analisando a estrutura física do shopping, é possível definir o local de instalação e a quantidade de dispositivos que serão necessários para o monitoramento da quantidade de pessoas. Para a realização de um monitoramento com mais eficiência, três tipos de dispositivos serão necessários: i) Dispositivos instalados nas portas de acesso e escadas; ii) Dispositivos instalados nas portas das lojas e iii) Dispositivos instalados nos corredores, áreas comuns e áreas de alimentação.

Para fins de desenvolvimento de um protótipo inicial, será considerado a instalação de um único dispositivo na entrada de acesso ao interior da loja, independente de seu tamanho físico. Os dispositivos instalados nas portas das lojas serão configurados com informações referentes a dimensão em m², a quantidade prevista de funcionários por hora, e a capacidade máxima da loja.

O dispositivo instalado realizará a contabilização da entrada e saída de pessoas. Essa informação estará visualmente presente em um painel que será instalado na porta de entrada da loja, com o objetivo de informar aos clientes a quantidade de pessoas no local. Dessa forma, estes poderão tomar a decisão de entrar ou não na loja naquele momento.

Nos corredores e nos locais de acesso ao shopping como portas de entrada e escadas de acesso ao primeiro andar, painéis também serão instalados com a finalidade de informar aos clientes sobre a quantidade total de pessoas por pavimento, bem como o quantitativo total presente no shopping. Essas informações serão fornecidas pelos dispositivos individuais instalados em cada entrada de acesso, inclusive nas escadas e elevadores.

Em futuras implementações, o controle de funcionários também será realizado. Desta forma, tanto os que trabalham nas lojas quanto os prestadores de serviços, e.g, entregadores de mercadorias, deverão ingressar no shopping por meio de uma entrada específica conforme indicado na Figura 2. Na entrada dos funcionários, será necessário a instalação de um sistema de biometria que será responsável pela autorização do trabalhador, desde que este esteja no horário previamente cadastrado. A entrada do funcionário será contabilizada no sistema de contagem de pessoas. Caso o funcionário não esteja no seu horário de trabalho, então seu ingresso deverá ocorrer por meio da entrada de clientes. No interior de cada loja também deverá ser instalado o sistema de biometria para a identificação e contagem de funcionários.

Descrição do dispositivo

O dispositivo instalado no shopping apresentará uma arquitetura padrão. Assim, será utilizado o Raspberry Pi [5] que é um computador de baixo custo e tamanho reduzido. O Raspberry Pi atuará no processamento dos dados coletados e envio das informações para outros dispositivos. Para contagem das pessoas, uma câmera com ângulo de 60° será acoplada a placa. A Figura 3 apresenta os componentes descritos.

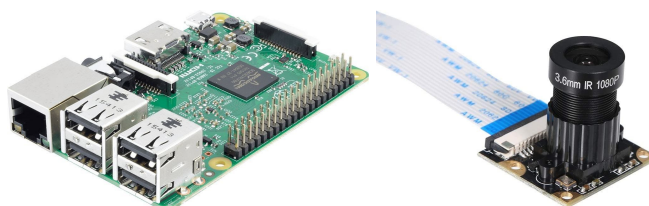


Figura 3 - Componentes utilizados na construção do dispositivo. Fonte: [5] [6].

Não está no escopo do trabalho a montagem e instalação do dispositivo físico, deste modo os componentes serão meramente ilustrativos. Os dados para processamento das informações estão sendo simulados e gerados aleatoriamente, em vista de uma compreensão factível do sistema.

Descrição dos Requisitos do Sistema

Locais públicos como shoppings centers apresentam vários acessos que permitem a entrada e saída de pessoas, e normalmente possuem andares compostos por estabelecimentos comerciais, áreas de alimentação e áreas comuns onde os presentes podem se reunir. Desta forma, o sistema de controle de acesso deverá realizar o monitoramento de diferentes pontos dentro de um shopping, possuindo um conjunto definido de regras para determinado ponto monitorado. Tais regras serão definidas no decorrer deste trabalho.

Normalmente as pessoas que frequentam um shopping se dividem em quatro categorias: *funcionários do shopping*, *funcionários das lojas*, *prestadores de serviço e clientes*. Os funcionários do shopping podem ser seguranças, administradores, profissionais da limpeza, etc. Os funcionários das lojas são as pessoas que atuam nos diferentes departamentos, em cada uma das lojas, lanchonetes, restaurantes, dentre outros. Já os prestadores de serviços são as pessoas que ocasionalmente entram no shopping, como entregadores e transportadores. E por fim, os clientes são todos os visitantes que vão ao shopping, a exemplo de atividades de lazer.

Os requisitos funcionais (RFs) referem-se a declarações de serviços que o sistema deve propiciar, com descrições de como este deve reagir a determinadas entradas bem como seu comportamento em situações específicas [2]. Portanto, os RFs atendidos pelo sistema são:

- RF01: O sistema deverá permitir ao administrador(a) do shopping cadastrar e alterar as capacidades máximas do ambiente como um todo, a taxa permitida de ocupação, além da quantidade de entradas disponíveis;
- RF02: O sistema deverá possibilitar ao gerente da loja cadastrar e alterar os funcionários autorizados, com suas respectivas categorias e horário de entrada/saída;
- RF03: O sistema deverá permitir, nos painéis de entrada das portas, escadas e elevadores, visualizar a quantidade de pessoas por andar e no shopping, em sua totalidade;
- RF04: O sistema deverá permitir, nos painéis de entrada das lojas e dos banheiros, visualizar a quantidade de pessoas presentes no respectivo ambiente;
- RF05: O sistema deverá exibir, nos painéis dos corredores do shopping, a quantidade total de pessoas presentes em cada andar;
- RF06: O sistema deverá realizar a autenticação de funcionários e prestadores de serviço por meio de biometria, somente por entrada específica e desde que estejam no horário de trabalho cadastrado
- RF07: O sistema deverá permitir a entrada de prestadores de serviço somente se o total de ocupantes desta categoria não tenha sido excedido;
- RF08: O sistema deverá realizar a contagem de funcionários e prestadores de serviço, visto que estes integram a contagem total de pessoas dentro do ambiente (combinada ao número de clientes);

- RF09: O sistema deverá armazenar em cada um dos dispositivos espalhados pelo shopping o status de entrada/saída e a data/hora da requisição;

Além das funcionalidades descritas anteriormente, o sistema deve prover a asseguaração dos seguintes requisitos não-funcionais (RNFs):

- RNF01: O sistema deverá permitir o acesso de forma segura à suas funcionalidades, realizando, para tal, a autenticação de usuários;
- RNF02: O sistema deverá possuir alta disponibilidade, de modo que os dados precisam ser mantidos de maneira distribuída, sem pontos únicos de falha;
- RNF03: O sistema deverá atender ao tempo de resposta das funções de acordo com limites estabelecidos previamente, de maneira independente à carga de uso;
- RNF04: O sistema não deverá demorar mais que 30 segundos para realizar a atualização dos painéis de fluxo de pessoas;
- RNF05: O sistema não deverá exibir quaisquer dados de cunho privativo, como dados sensíveis de funcionários, para usuários não autorizados, sendo a informação vedada somente ao administrador(a) do shopping.

Etapas e Descrição do Trabalho

O trabalho divide-se em quatro etapas que seguirão um cronograma específico de entrega conforme Tabela 1. Os arquivos estarão disponíveis no repositório do Github no seguinte endereço eletrônico: <<https://github.com/brunacordeiro/SCAPS>>.

Tabela 1: Cronograma das atividades do trabalho. Fonte: Própria (2020).

Etapa		Data Entrega (2020)
E1	Núcleo funcional do sistema composto pelos componentes e suas interações	16 de julho
E2	Coordenação, consenso e controle de concorrência	18 de agosto
E3	Tolerância a falhas e replicação (confiabilidade, disponibilidade, performance)	10 de setembro
E4	Aspectos de segurança da interação entre os componentes e apresentação final	01 de outubro

A primeira etapa do trabalho consiste no desenvolvimento do projeto e implementação das funcionalidades de um sistema distribuído que atenda as necessidades do cenário descrito. Assim sendo, uma arquitetura do sistema, incluindo os componentes e as interações entre eles será projetada, implementada e testada em um ambiente distribuído real, mas com simulação dos dispositivos de IoT e dos usuários envolvidos. Nesta primeira etapa, os requisitos não-funcionais apresentados anteriormente não serão considerados.

A segunda etapa consiste em introduzir mecanismos de coordenação, consenso e controle de concorrência, de modo a garantir que a consistência e coerência do estado de todos os componentes do sistema. Ou seja, elementos diferentes do sistema não poderão apresentar dados conflitantes. Nessa etapa serão descritos os mecanismos utilizados e os cenários para demonstração do estado consistente do sistema.

A terceira etapa será composta pela introdução dos mecanismos de tolerância a falhas e replicação, visando obter confiabilidade, disponibilidade, performance. Técnicas de balanceamento de carga poderão ser aplicadas para melhorar o desempenho em relação ao tempo de resposta e à vazão, bem como para aumentar a escalabilidade do sistema e a disponibilidade de seus componentes críticos. Além disso, o sistema deverá permanecer operando com 100% da funcionalidade na presença de falhas de alguns de seus componentes.

Testes de desempenho serão realizados por meio da simulação de uma grande quantidade de operações por um período de tempo representativo. Diferentes configurações do sistema em termos do número de réplicas dos componentes críticos serão consideradas para análise. Além de testes de desempenho, serão realizados testes da funcionalidade do sistema na presença de falhas de componentes.

Por fim, na quarta etapa do trabalho serão introduzidos mecanismos de autenticação e criptografia para atender ao requisito não-funcional de segurança. Usuários cadastrados deverão ser identificados de forma segura. A comunicação entre os componentes do sistema deverá ser autenticada e resistente a ataques de modificação de conteúdo. Os testes de desempenho realizados na E3 serão realizados novamente para demonstrar o impacto dos mecanismos de segurança no desempenho do sistema.

Metodologia da Primeira Etapa

A primeira etapa do projeto foi dividida em quatro partes que serão apresentadas nessa seção. Nesta primeira parte, os requisitos não-funcionais não foram considerados, porém o sistema será capaz de funcionar de modo a atender o cenário descrito.

1. Projeto e implementação das funcionalidades do sistema:

As funcionalidades do sistema são definidas conforme os requisitos que necessitam ser atendidos no momento. Para que os requisitos possam ser atendidos, definições na estrutura física do projeto deverão ser consideradas. Para a primeira entrega, de forma parcial, buscou-se atender ao RF03, levando em consideração a entrada de pessoas em uma das portas principais. O projeto considerou em primeiro momento a simulação com somente uma porta principal de entrada de clientes.

2. Definição da arquitetura do sistema, incluindo os componentes e suas interações:

A arquitetura do sistema proposto consistirá na estruturação dos componentes de software, das propriedades desses componentes e dos relacionamentos entre estes componentes. Dessa forma, dentro do estilo de arquitetura distribuída, foi utilizado o *middleware* RabbitMQ, que ao introduzir protocolos como o AMQP 0-9-1 e/ou AMQP 1.0, bem como demais protocolos de mensagens suportados, possuem em comum a característica de serem inerentemente distribuídos, haja vista que em grande parte conectam-se em *hosts* remotos [7].

Para ingresso no shopping, o sistema deverá verificar como dados de entrada as informações das câmeras (de modo simulado), que após processamento, proverão uma sequência de números binários representáveis como “0” para saída e “1” para entrada. Os números serão utilizados para contabilização da quantidade de pessoas dentro do recinto. Deste modo, espera-se ao final de todas as etapas que o programa deverá comportar-se da seguinte maneira (Figura 4):

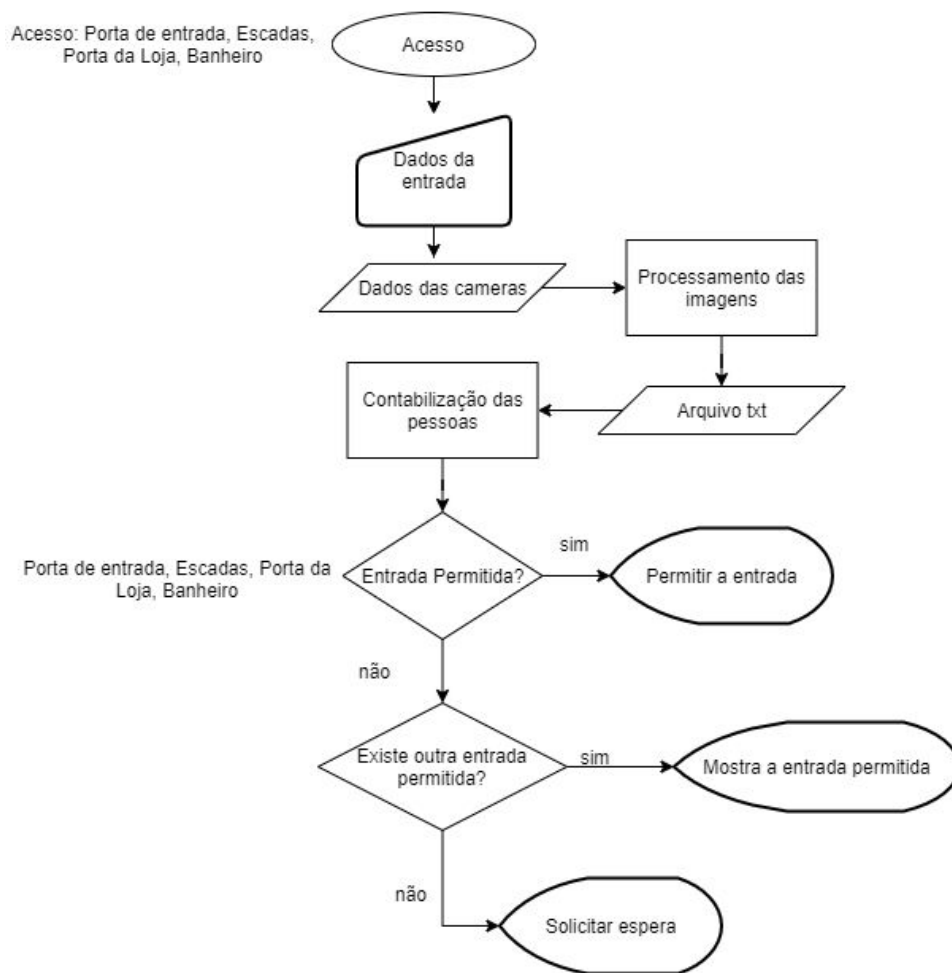


Figura 4 - Diagrama de fluxo da sequência após entrada do cliente dentro do shopping. Fonte: Própria (2020).

3. Utilização dos paradigmas de interação apropriados:

O RabbitMQ é o *middleware* responsável para que haja comunicação entre os serviços, considerado como um dos sistemas de mensagens de código aberto mais utilizados [8]. Desenvolvido em linguagem Erlang, implementa diversos protocolos, como o HTTP e AMQP, além de possuir grande desacoplamento entre serviços. Suas mensagens por padrão caem na memória da máquina, sendo citado como extremamente rápido e poderoso.

Em seu funcionamento tem como base para comunicação o TCP. Para este projeto, o RabbitMQ trata-se do meio de campo entre o padrão *publish/subscribe*. O *publisher* publica a mensagem para que for consumir e o *subscribe*, é o consumidor - capaz de ler a mensagem e utilizá-la como desejado. Entretanto, o publicador não envia sua mensagem diretamente para quem vai consumi-la, ela cai numa fila de mensagens, a qual é lida pelo consumidor.

Destaca-se que a mensagem enviada perpassa pelo *exchange* antes de chegar a fila - que pega a mensagem, processa e descobre para qual fila a mensagem vai ser enviada. Existem diversos tipos de *exchange*, como o *direct exchange* - que envia especificamente para uma determinada fila, ou mesmo por *fanout* (tipo usado por este projeto), que é uma *exchange* que vai mandar para todas as filas que estão relacionadas [9], sendo a forma desejada para o shopping, onde as informações de entrada devem comunicar-se entre si.

Por *fanout*, conforme apresentado na Figura 5, quando manda-se a mensagem, ela cai na *exchange*, sendo replicadas pelas filas, ou seja, envia-se uma única mensagem e todas as filas vão receber.

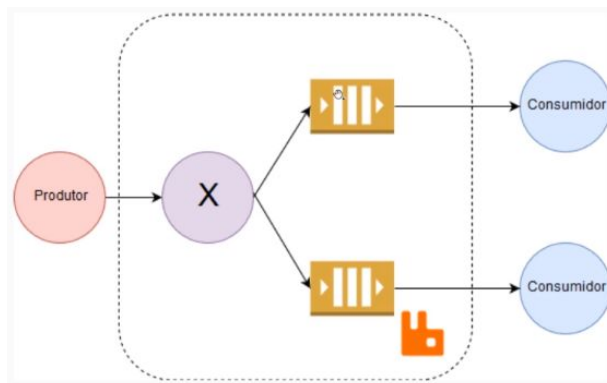


Figura 5 - Esquema representativo do envio de mensagens via *exchange* Fanout. Fonte: [8].

Desta maneira, para esta entrega, levando em conta a porta principal, a representação pelo RabbitMQ pode ser melhor vista na Figura 6. Há um *publisher* e um *consumer* (*subscriber*), que perpassa a *exchange* em uma fila de mensagens, em um único âmbito.

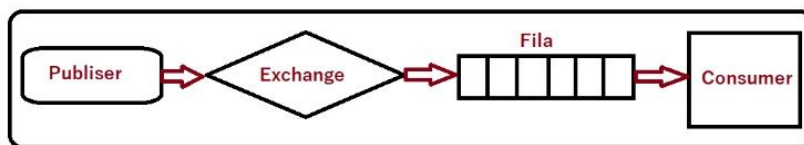


Figura 5 - Esquema publicador e consumidor para a fila da porta principal. Fonte: Própria (2020).

4. Implementação da arquitetura e teste:

Para trabalhar com o RabbitMQ, foi necessário seu download e instalação, conforme tutorial em <<https://www.rabbitmq.com/download.html>>. Foi utilizada a instalação para Windows, por meio do Chocolatey, um gerenciador de pacotes para este sistema operacional, dado que a primeira entrega foi realizada e simulada de forma totalmente local (sem aplicação no ambiente da nuvem Amazon Web Services - AWS).

O Chocolatey deve ser instalado previamente por meio do Power Shell do Windows, habilitando a Política de Execução do terminal para executar scripts e realizar o download de pacotes (conforme detalhado em <<https://hcode.com.br/blog/conhecendo-o-chocolatey-o-gerenciador-de-pacotes-do-windows>>). Após esta etapa, é necessário executar um comando específico de instalação (como descrito em <<https://chocolatey.org/install>>). Se a instalação ocorrer corretamente, ao digitar “choco -?”, será exibida a versão do choco.

Após a instalação, é necessário de fato efetuar a instalação do RabbitMQ por meio do comando “choco install rabbitmq” (como descrito em <<https://www.rabbitmq.com/install-windows.html#chocolatey>>). Posteriormente, para verificar o envio/recebimento das mensagens, é preciso inicializar a interface do RabbitMQ, com o comando “.\rabbitmq-plugins enable rabbitmq_management” (como descrito em <<https://documentacao.senior.com.br/seniorxplatform/instalacao/rabbitmq.htm>>), e foi possível acessar a URL do <<http://localhost:15672/>>, como visualizado na Figura 7.

Quanto ao código de envio/resposta das mensagens, foram criados três arquivos: *entradaPrincipal.py*, *receiver.py* e *receiver_log.py*. O primeiro refere-se a simulação da entrada principal do shopping, buscando em primeiro momento, funcionar em uma entrada do ambiente para posterior replicação. O *receiver* envia a mensagem da quantidade de pessoas entrando ou saindo, e o *receiver_log* recebe a mensagem, enquanto tudo isso é passível de monitoramento no RabbitMQ.

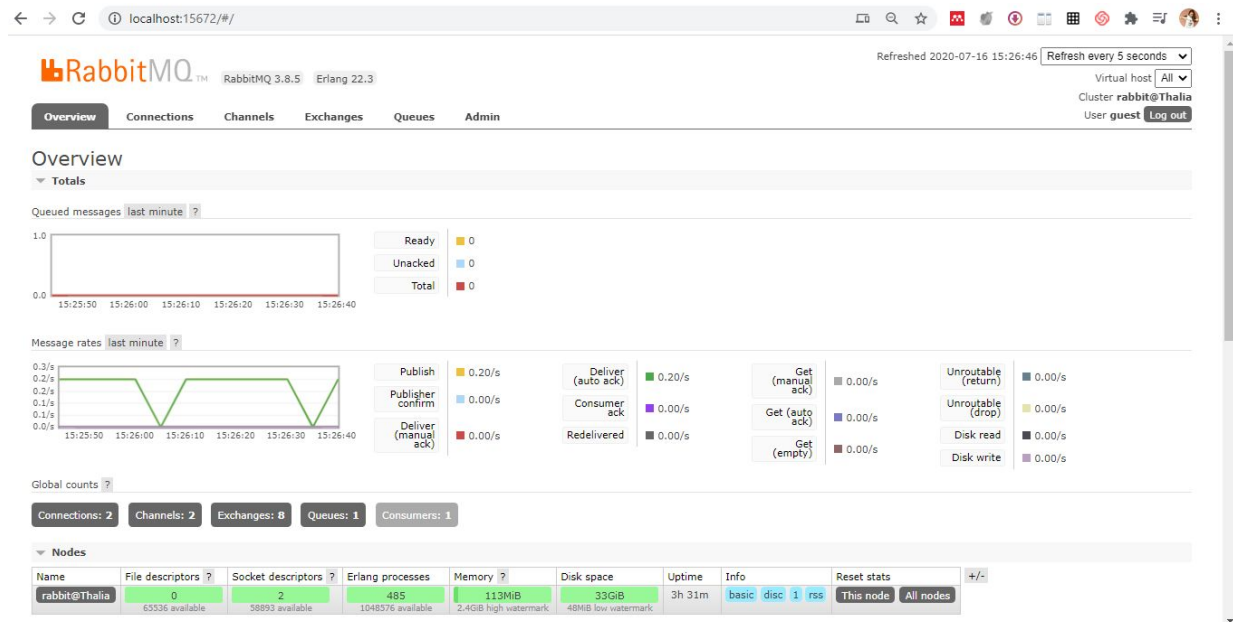


Figura 7 - Interface do RabbitMQ para visualização das mensagens. Fonte: Própria (2020).

Para melhor entendimento, segue capturas de tela (Figuras 8, 9, 10 e 11) do atual status de funcionamento do sistema:

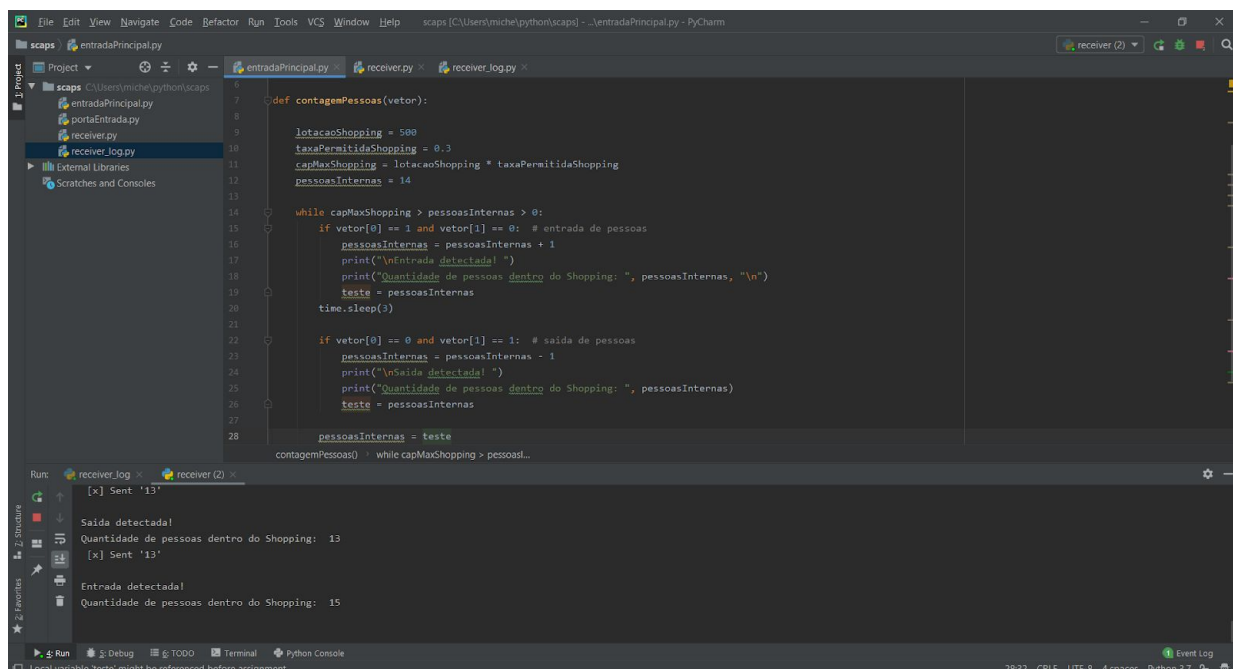


Figura 8 - Arquivo *entradaPrincipal.py*. Fonte: Própria (2020).

```

10 def connection():
11     connection = pika.BlockingConnection(
12         pika.ConnectionParameters(host='localhost')
13     )
14     channel = connection.channel()
15     entradaPrincipal.pessoasInternas = 14
16
17     channel.exchange_declare(exchange='entradaShopping', exchange_type='fanout')
18
19     # message = ' '.join(sys.argv[1:]) or "info: Hello World!"
20
21     while 1:
22         message = entradaPrincipal.contagemPessoas(vetor=random.sample(range(2), 2))
23         channel.basic_publish(exchange='entradaShopping', routing_key='', body=message)
24         print("[x] Sent %s" % message)
25         time.sleep(3)
26
27     connection.close()
28
29 if __name__ == "__main__":
30     connection()

```

Run: receiver_log x receiver (2)

```

[x] Sent '13'
Saída detectada!
Quantidade de pessoas dentro do Shopping: 13
[x] Sent '13'
Saída detectada!
Quantidade de pessoas dentro do Shopping: 13
[x] Sent '13'

```

Figura 9 - Arquivo receiver.py. Fonte: Própria (2020).

```

1 import pika
2
3 connection = pika.BlockingConnection(
4     pika.ConnectionParameters(host='localhost')
5 )
6 channel = connection.channel()
7
8 channel.exchange_declare(exchange='entradaShopping', exchange_type='fanout')
9
10 result = channel.queue_declare(queue='', exclusive=True)
11 queue_name = result.method.queue
12
13 channel.queue_bind(exchange='entradaShopping', queue=queue_name)
14
15 print("[*] Waiting for logs. To exit press CTRL+C")
16
17 def callback(ch, method, properties, body):
18     print("[x] %s" % body)
19
20 channel.basic_consume(
21     queue=queue_name, on_message_callback=callback, auto_ack=True)
22
23

```

Run: receiver_log x receiver (2)

```

[x] b'13'
[x] b'13'
[x] b'13'
[x] b'13'
[x] b'15'
[x] b'13'
[x] b'15'
[x] b'15'
[x] b'13'

```

Figura 10 - Arquivo receiver_log.py. Fonte: Própria (2020).

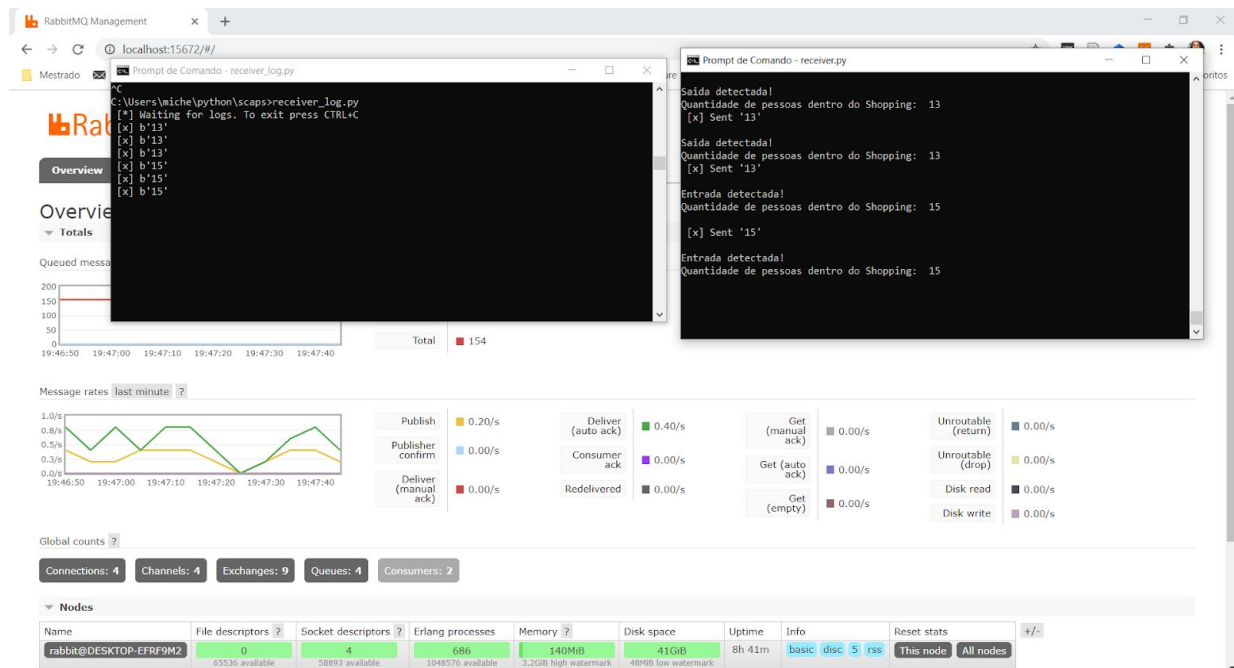


Figura 11 - Interface do Middleware em relação ao envio e resposta. Fonte: Própria (2020).

Como forma de atualização desta etapa, o código anterior foi atualizado e inserido no Github, já realizando a modificação da quantidade de pessoas como variável global entre as funções, bem como replicação em escadas e elevadores, além de demais portas.

Metodologia da Segunda Etapa

A segunda etapa do trabalho tem como objetivo introduzir mecanismos de coordenação, consenso e controle de concorrência, de modo a garantir a consistência e coerência do estado de todos os componentes do sistema. Assim, espera-se que elementos diferentes do sistema não tenham dados conflitantes e portanto, dados da quantidade de pessoas presentes no shopping possam ser idênticos para quaisquer dispositivos distribuídos.

Nessa seção, os mecanismos utilizados serão descritos, fazendo uso de cenários simulados para demonstrar a manutenção do estado consistente do sistema. O projeto realiza exemplificação com três (3) portas principais, duas (2) escadas e três (3) elevadores (considerando subida e descida). A ideia é o uso destas aplicações por intermédio de soluções em nuvem, como a AWS, considerada uma das plataformas com maior abrangência e uso mundialmente [10].

Portanto, utilizando-se a AWS por meio do serviço EC2, é mantida a necessidade de instalação do RabbitMQ, seguindo os passos descritos por Mutai [11] em sistema operacional Linux (Ubuntu), distribuição padrão selecionada para criação das máquinas virtuais (VMs) deste projeto, e foram escolhidas seguindo as seguintes características:

- Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-0bcc094591f354be2 (64-bit x86) / ami-0bc556e0c71e1b467 (64-bit Arm)
- Currently selected: t2.micro (Variable ECUs, 1 vCPUs, 2.5 GHz, Intel Xeon Family, 1 GiB memory, EBS only)

Ademais, também foi criado um grupo de segurança específico denominado *ScapsSecurityGroup*, com regras de entrada quanto à porta padrão de conexão com o RabbitMQ (5672).

O código da Parte I foi replicado em quatro máquinas distintas (*publisher* Porta1, *consumer* Porta1 e *publisher* Porta2, *consumer* Porta2), a fim de verificar o funcionamento e erros quanto à consistência das informações de indivíduos no shopping, essencial para esta etapa. Foi percebido então que, considerando duas portas principais, consumindo informações das filas em diferentes máquinas por meio da AWS, haveria a necessidade de inserção de mecanismos de comunicação entre as mesmas, trocando informações entre si para promoção de estados consistentes do número de presentes no recinto.

Logo, para a Parte 2 isso foi trabalhado e é possível verificar melhorias quanto à comunicação. A exemplo disso, para as portas, foram criadas três portas principais (três *publishers*) e há somente um *consumer* para as mesmas. Cada *publisher* refere-se a uma entrada. Assim, cada porta, elevador ou escada possui um *publisher* (referenciando-se aos dispositivos Raspberry dispostos na estrutura física do shopping).

Conforme a Figura 12, retornando para a ideia das portas de entrada, os três *publishers* (um de cada porta) passarão por uma *exchange*, que encaminha para fila e todos são consumidos por apenas um *consumer* (de todas as portas). Nas demais representações de escadas e elevadores, a mesma lógica é seguida.

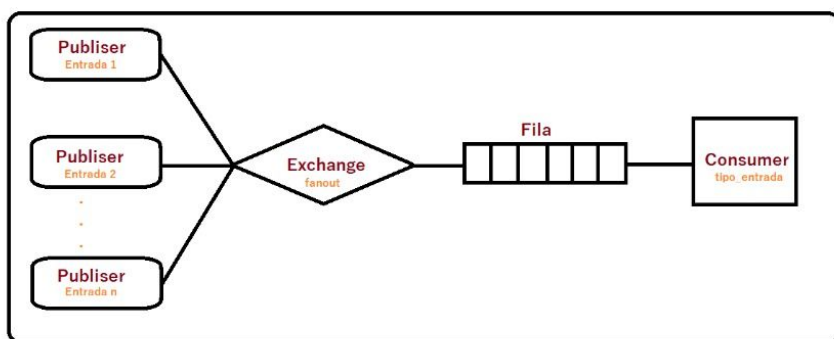


Figura 12 - Esquema publicador e consumidor para as portas principais de entrada. Fonte: Própria (2020).

Logo, haverá três grupos de *publishers* (para os três tipos de contabilização de pessoas) e assim, três *consumers* (derivados de cada tipo). Quanto ao consumidor, a Figura 13 exemplifica os *consumers* de portas, escadas e elevadores (que ao mesmo tempo acabam atuando tanto como *publisher* quanto *consumer*), e um *consumer* final, que deste, extrai-se dados da quantidade de pessoas dentro do shopping, bem como as informações necessárias para envio de dados visualizáveis em uma aplicação web.

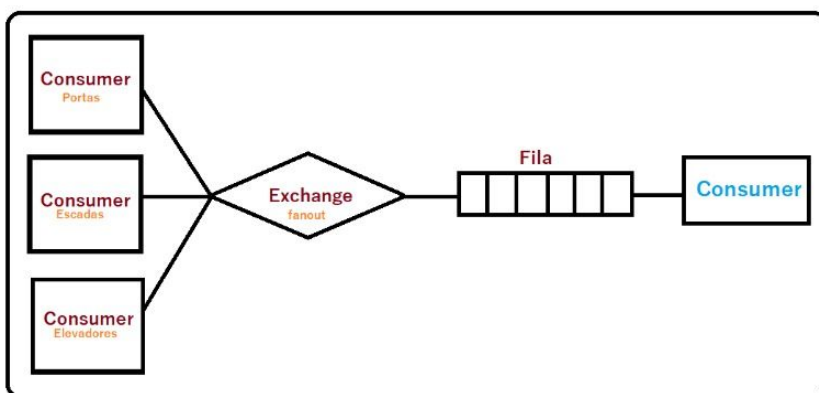


Figura 13 - Esquema entre tipos de *consumers* e consumidor final. Fonte: Própria (2020).

Portanto, para as portas (neste caso demonstrado com a porta 1), o código segue conforme a Figura 14, realizando o envio de dados como uma *string*.

The screenshot shows the PyCharm IDE with the file `senderDoor1.py` open. The code defines a function `consumer()` that interacts with a door. The Run console shows the following output:

```

Saída detectada!
Porta : '[(-1, 0, 1, 1)]'

Entrada detectada!
Porta : '[(0, 1, 1, 1)]'

Saída detectada!
Porta : '[(-1, 1, 2, 1)]'

```

Figura 14 - Arquivo `senderDoor1.py`. Fonte: Própria (2020).

Após o envio, o `consumerDoors.py` (Figura 15) recebe os dados inerentes à todas as portas em funcionamento e realiza o envio destes dados ao `consumer.py`, o qual realiza o papel de coordenador.

The screenshot shows the PyCharm IDE with the file `consumerDoors.py` open. The code defines a function `consumer()` that processes data from multiple doors. The Run console shows the following output:

```

Informações Porta: 1
Entradas: 11
Saída 6
Entradas no Shopping: 11
Saídas no Shopping: 6
Quantidade de Pessoas dentro do Shopping: 5

Informações Porta: 3
Entradas: 1
Saída 1
Entradas no Shopping: 12
Saídas no Shopping: 7
Quantidade de Pessoas dentro do Shopping: 5

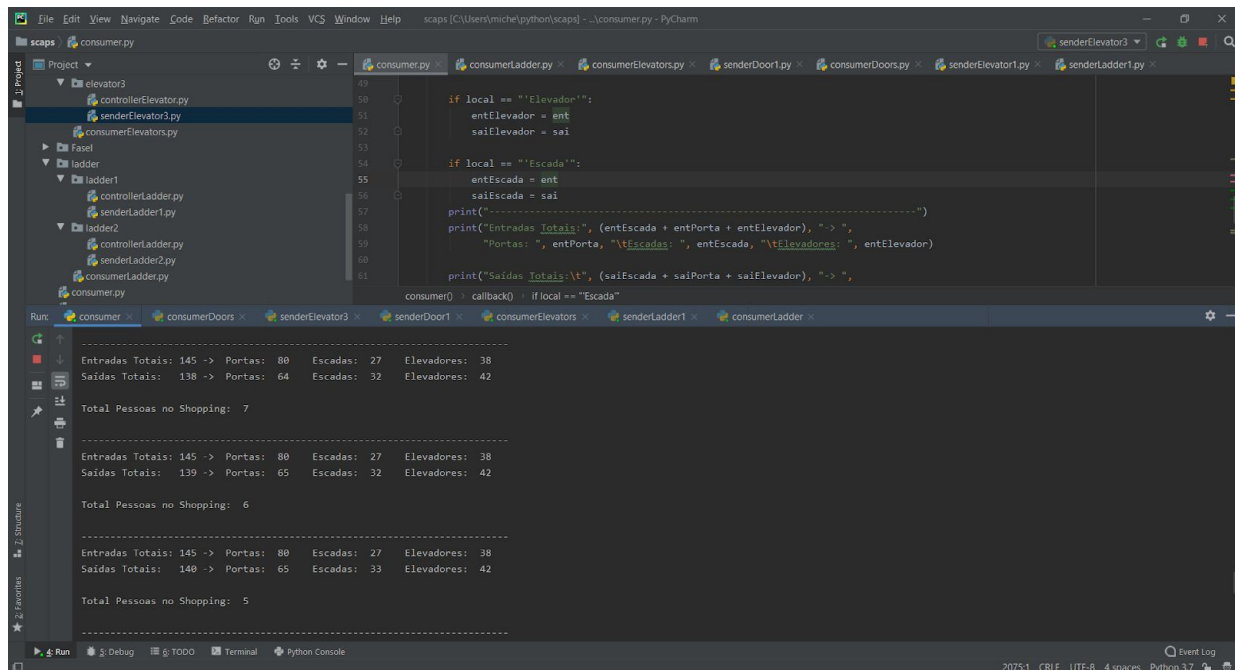
Informações Porta: 2
Entradas: 11
Saída 9
Entradas no Shopping: 23
Saídas no Shopping: 16
Quantidade de Pessoas dentro do Shopping: 7

```

Figura 15 - Arquivo `consumerDoors.py`. Fonte: Própria (2020).

Deste modo, o coordenador (Figura 16) é responsável por prover uma única informação consistente entre todos os dados de indivíduos presentes no shopping, haja vista realizar o cálculo de entrada e saída de pessoas, considerando os cenários de portas,

escadas e elevadores, por intermédio do `channel.basic_qos(prefetch_count=1)`, possibilita que o `consumer.py` receba e leia uma mensagem de cada vez.



The screenshot shows a PyCharm IDE with a project named 'scaps'. The file explorer on the left shows a directory structure with files like `controllerElevator.py`, `senderElevator3.py`, `consumerElevators.py`, `senderDoor1.py`, `consumerDoors.py`, `senderElevator1.py`, and `senderLadder1.py`. The main editor displays the `consumer.py` file, which contains a `consumer()` function. The function uses `channel.basic_qos(prefetch_count=1)` and processes messages for elevators and stairs. The console at the bottom shows the output of the program, displaying statistics for entrances, exits, doors, stairs, and elevators, as well as the number of people in the shopping area.

```
Entradas Totais: 145 -> Portas: 80 Escadas: 27 Elevadores: 38
Saídas Totais: 138 -> Portas: 64 Escadas: 32 Elevadores: 42
Total Pessoas no Shopping: 7

Entradas Totais: 145 -> Portas: 80 Escadas: 27 Elevadores: 38
Saídas Totais: 139 -> Portas: 65 Escadas: 32 Elevadores: 42
Total Pessoas no Shopping: 6

Entradas Totais: 145 -> Portas: 80 Escadas: 27 Elevadores: 38
Saídas Totais: 140 -> Portas: 65 Escadas: 33 Elevadores: 42
Total Pessoas no Shopping: 5
```

Figura 16 - Arquivo `consumer.py`. Fonte: Própria (2020).

Referências

- [1] OPAS/OMS Brasil. Folha informativa – COVID-19 (doença causada pelo novo coronavírus). 2020. Disponível em: <https://www.paho.org/bra/index.php?option=com_content&view=article&id=6101:covid19&Itemid=875>. Acesso em: 1 julho. 2020.
- [2] SOMMERVILLE, Ian. Engenharia de Software. 9. ed. São Paulo: Pearson. Prentice Hall, 2011.
- [3] Passeio das Águas Shopping. Sobre o Shopping. Disponível em: <<https://passeiodasaguasshopping.com.br/sobre-o-shopping/>>. Acesso em: 3 julho. 2020
- [4] Aliansce Sonae. Excelência e comprometimento. Disponível em: <<http://www.aliansce sonae.com.br/>>. Acesso em: 3 julho. 2020
- [5] Raspberry Pi. What is a Raspberry Pi?. Disponível em: <<https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/>>. Acesso em: 3 julho. 2020.
- [6] Amazon. Docooler Wide Angle Camera 5M Pixel Adjustable Angle Compatible for Raspberry Pi 3 Model B B+. Disponível em: <<https://www.amazon.com/Docooler-Camera-Adjustable-Compatible-Raspberry/dp/B07K56KFX5>>. Acesso em: 07 julho. 2020.
- [7] RabbitMQ. Distributed RabbitMQ. Disponível em: <<https://www.rabbitmq.com/distributed.html>>. Acesso em: 14 julho 2020.
- [8] RabbitMQ. Quorum queues. Disponível em: <<https://www.rabbitmq.com/quorum-queues.html>>. Acesso em: 14 julho 2020.

[9] Gist Github (Renato Costa). RabbitMQ com Python. Disponível em: <<https://gist.github.com/renatoapcosta/2a4b6c7a5933edf09e9226e11f1ca989>>. Acesso em: 16 julho 2020.

[10] Amazon Web Services. Computação em nuvem com a AWS. Disponível em: <https://aws.amazon.com/pt/what-is-aws/?nc1=f_cc>. Acesso em: 18 agost. 2020.

[11] Mutai, Josphat. Computer for Geeks. How To Install Latest RabbitMQ Server on Ubuntu 20.04 | 18.04 LTS. Disponível em: <<https://computingforgeeks.com/how-to-install-latest-rabbitmq-server-on-ubuntu-linux/>>. Acesso em: 06 agost. 2020.