UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA - INF
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
ALUNOS: ARTHUR SOUZA SALLY, CARLOS HENRIQUE RORATO, HELBERTH BORELLI

SISTEMAS DISTRIBUÍDOS
PROJETO DE APLICAÇÃO

DOCUMENTAÇÃO DO SISTEMA v. 0.2[1]

Link do Github: https://github.com/ArthurSouzaSally/ShoppingProject

1. DESCRIÇÃO GERAL DO SISTEMA

De maneira geral, o sistema apresenta uma solução para o controle de acesso de clientes e funcionários em um Shopping, em meio ao contexto de reabertura do comércio pós-pandemia da COVID-19. Neste cenário, os estabelecimentos comerciais precisam atender a uma série de restrições em seu funcionamento, evitando aglomerações e, por consequência, a propagação do vírus SARS-COV-2, causador da doença. As indicações da Organização Mundial da Saúde (OMS) trazem presente que a COVID-19 muito possivelmente se tornará endêmica nos próximos anos, o que faz com que soluções como esta sejam profundamente necessárias neste “novo normal”.

No caso específico dos Shoppings, é necessário pensar em controle de acesso, tanto do estabelecimento como um todo, como de cada andar e loja que o compõe. As restrições basicamente são colocadas nestes três níveis. Clientes e funcionários devem ter a temperatura aferida e os clientes só podem acessar as dependências caso não estejam com a quantidade máxima de consumidores.

Tendo em vista este contexto geral, a solução proposta apresenta um software que, utilizando técnicas e conceitos de sistemas distribuídos, auxilie no gerenciamento do controle de acesso, atendendo às restrições impostas pela situação atual e gerenciando todo o processo, desde a comunicação e coleta de dados dos sensores, até o gerenciamento de clientes em cada andar, contribuindo para o bem-estar do consumidor e buscando minimizar aglomerações.

2. REQUISITOS

Podemos elencar como requisitos funcionais do sistema:

Em relação ao controle de acesso de pessoas, podemos considerar: (1) funcionários do shopping, (2) funcionários das lojas, (3) prestadores de serviços, (4) clientes:

- **Trabalhadores (1, 2 e 3)**
  - possuem entrada separada

[1] Este documento está em desenvolvimento.

- exige identificação
- contagem de pessoas (contribuem para o total de pessoas no shopping)
- grupos 1 e 2 possuem turno de trabalho (deve barrar a entrada fora do turno)
- grupo 3 deve ser identificado e somente não entrarão caso o shopping já esteja lotado

- **Clientes**
- entradas
  - direto ao shopping
  - por lojas
  - contagem de pessoas
  - somente devem entrar caso shopping não esteja lotado

- **Shopping**
- multiplas entradas
  - dividido em pisos
  - entrada de funcionários
  - entrada de clientes
    - entradas para área comum ao shopping
    - entradas por lojas
  - áreas de transição (acessos)
    - shopping para lojas
    - lojas para shopping
    - pisos para pisos (escadas e elevadores)
- aglomerações (capacidades)
  - do shopping (soma de todos os ambientes)
  - dos pisos (compõe o quantitativo do shopping)
  - das lojas (compõe o quantitativo do shopping)
    - casos em que a loja oferece acesso ao shopping, devem respeitar o quantitativo máximo do shopping, mesmo que não estejam lotadas

- **Dispositivos de Acesso (IoT)**
- catracas (trabalhadores) ou câmeras (clientes)
- enviam ao setor de entrada mensagem de entrada (in) ou saída (out)
  - câmera (in- contagem e temperatura do cliente / out - contagem)
  - catraca (in - contagem, temperatura e checagem de turno / out - contagem)

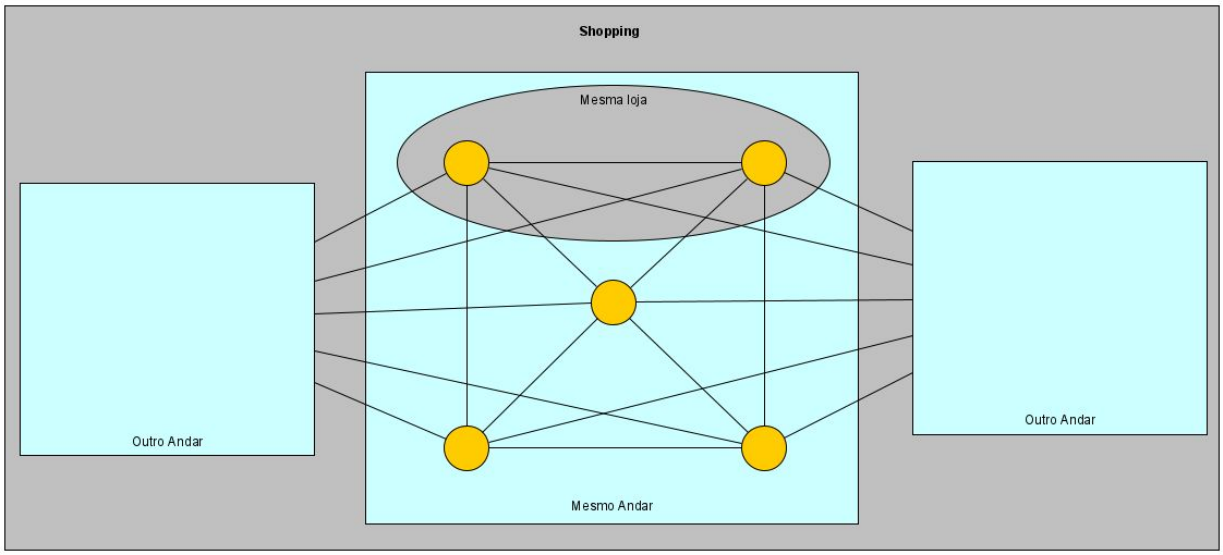
Em relação aos requisitos não-funcionais, podemos destacar:

- Construção de uma rede peer-to-peer, onde cada peer representa um ponto de controle de acesso.

- Utilização de recursos para coordenação, consenso e controle de concorrência: ordenação por timestamp e uso de bloqueio (lock) para garantir a consistência dos dados.

3. ARQUITETURA, DIAGRAMAS E ESQUEMAS

Em termos arquiteturais, o sistema corresponde ao modelo peer-to-peer (p2p), onde cada peer é um ponto de controle de acesso. O sistema é descentralizado, onde cada peer possui a informação necessária para seu funcionamento e, além disso, os peers compartilham as informações que são pertinentes. Todos os peers compartilham a quantidade de pessoas que está ocupando o shopping, para que se possa controlar a entrada. Peers do mesmo andar compartilham, além dessas, informações sobre a quantidade de pessoas neste andar específico, para também realizar o controle nas escadas rolantes. Por fim, peers na mesma loja compartilham a quantidade de pessoas dentro daquela loja, para limitar o acesso à entrada nessa. O diagrama abaixo ilustra o modelo arquitetural do sistema.



4. QUESTÕES RELATIVAS À IMPLEMENTAÇÃO

O software foi implementado em linguagem Python, por sua versatilidade e curva de aprendizado que permitiu que o processo de desenvolvimento ocorresse de maneira mais fluida. Por ser uma linguagem considerada de “alto nível”, propõe uma série de abstrações que evitam preocupações desnecessárias, dado o contexto do projeto e a natureza do problema a ser desenvolvido.

Em questões de implementação, optou-se por implementar o sistema, à priori, utilizando a tecnologia de sockets. Dadas as possibilidades e vantagens de se utilizar outros serviços (RabbitMq por exemplo), o software está em processo de migração e adaptação.

Ademais, optou-se por construir um Tracker, recurso que organiza as conexões entre os peers. Após a organização da comunicação entre os peers, este processo pode ser finalizado, dado que não é mais utilizado (os peers se comunicam entre si).

Para a implementação dos requisitos não-funcionais mencionados anteriormente, optou-se por utilizar o recurso de bloqueio (lock) para que a quantidade de pessoas em cada andar e no shopping fosse consistente em todos os peers. Somente um peer altera este valor por vez, e o repassa aos outros peers. A partir do momento em que eles são desbloqueados, mais um pode modificar o valor, bloqueando os outros peers. Acontece ainda a ordenação por timestamp e o uso de um recurso de “sorteio” para o controle de concorrência: dois peers pretendem realizar o bloqueio ao mesmo tempo, cada um sorteia um número inteiro e o peer com maior número ganha o direito de fazer por primeiro a modificação nos dados, bloqueando o acesso dos outros.

5. CENÁRIO PARA TESTES

Pensou-se, ainda, em um cenário para os futuros testes envolvendo um caso mais próximo da realidade. Para a realização desses testes, considerou-se o cenário do *Shopping INF*, que possui três andares, sendo um subterrâneo, que contém os seguintes elementos:

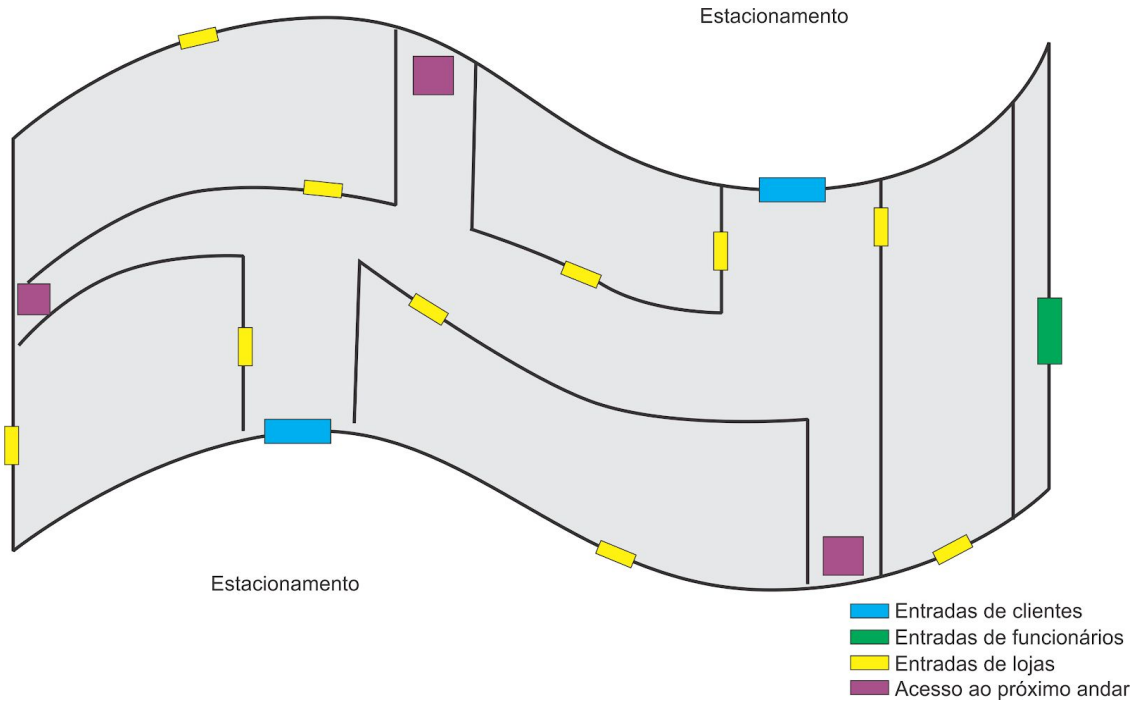
- **ANDAR SUBTERRÂNEO:** 2 entradas/saídas para clientes, 1 entrada para funcionários, 5 lojas de departamentos com duas entradas cada, e 3 acessos ao próximo andar.
- **ANDAR TÉRREO:** 3 entradas/saídas para clientes, 2 entrada para funcionários, 15 lojas com uma entrada apenas, 3 acessos ao subterrâneo e 3 acessos ao próximo andar.
- **SEGUNDO ANDAR:** Praça de alimentação, então não existe controle na porta. Cinema (1 controle de acesso) e parque de diversões (1 controle de acesso). 3 acessos ao andar térreo.

A capacidade máxima do shopping é de 1.500 clientes, sendo 500 clientes por andar. Nas lojas do andar subterrâneo, a capacidade máxima é de 50 clientes e no andar térreo, 15 clientes por loja. No segundo andar, é permitido que 100 pessoas estejam no Cinema e 80 crianças no parque de diversões.

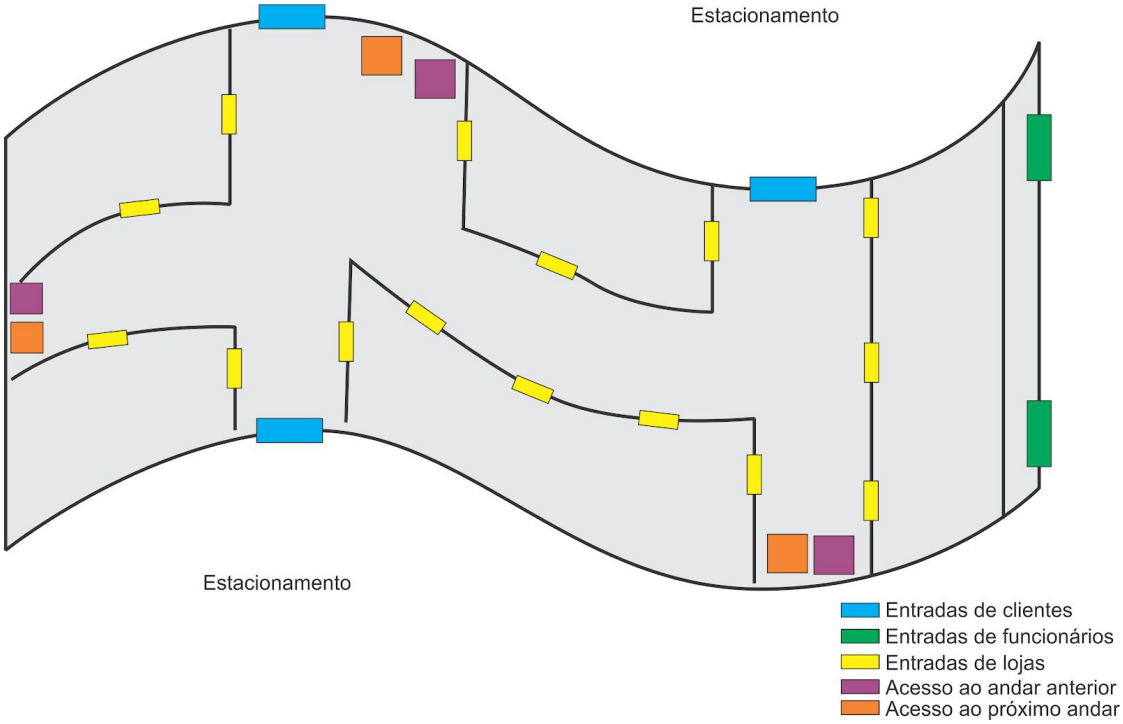
Podemos ilustrar o shopping com seus andres como é mostrado abaixo:

5.1. PLANTA - ANDAR SUBTERRÂNEO

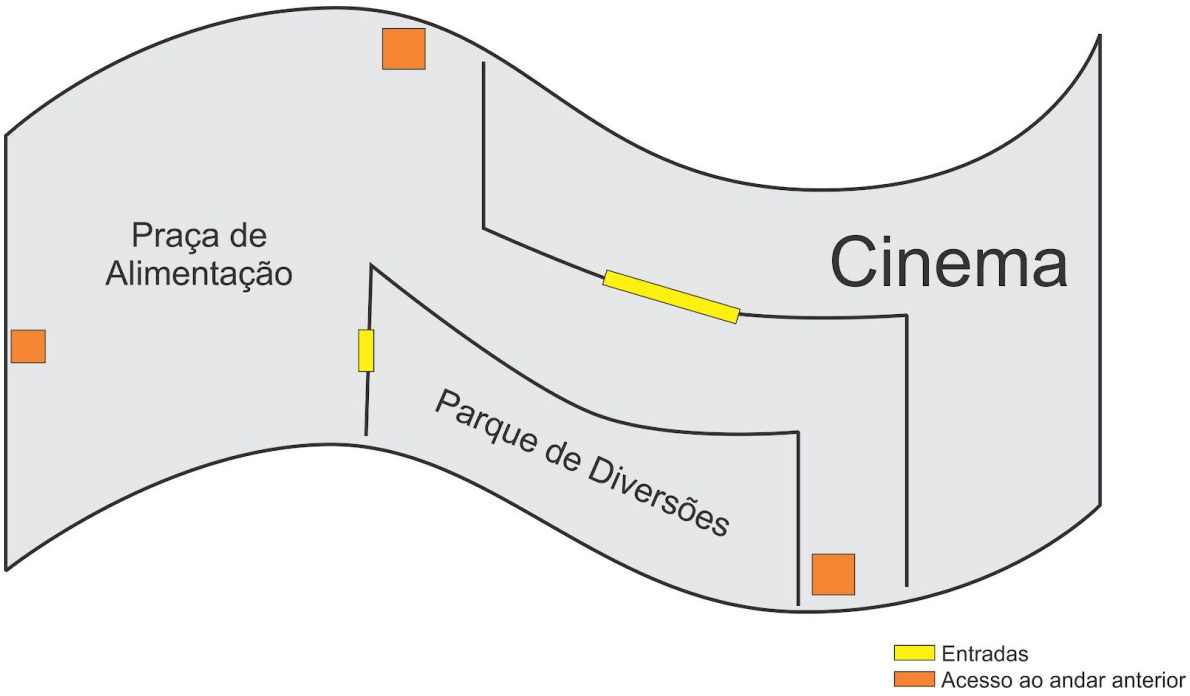
UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA - INF
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO
ALUNOS: ARTHUR SOUZA SALLY, CARLOS HENRIQUE RORATO, HELBERTH BORELLI



5.2. PLANTA - ANDAR TÉRREO



5.3. PLANTA - SEGUNDO ANDAR



6. INSTALAÇÃO E USO

Pré-requisitos: Python 3 instalado.

Existem dois componentes: `tracker.py` e `peer.py`. Proceder da seguinte forma:

- Executar primeiramente, via terminal, o código `tracker.py`;
- Informar ao *tracker* a quantidade máxima de pessoas no shopping. Pressionar “enter”;
- A partir da execução do *tracker*, o sistema está pronto para receber os *peers*.
- A medida que os *peers* vão entrando no sistema o *tracker* exibe as conexões (linha 5)
- Ao final da conexão dos *peers*, o *tracker* não é mais necessário e pode ser finalizado. Os *peers* podem comunicar-se e realizar as operações como é mostrado na tela.

```
1   $ python3 tracker.py
3   Iniciando Servidor...
3   Criando Tracker em: 192.168.75.138:15000
4   Limite de Pessoas no Shopping: 300
5   Novo Peer: 192.168.75.138:62834
6   Atualizando Peers
```

- Executar, de forma análoga, o arquivo `peer.py`, para criar um peer.
- O `peer.py` pode possuir várias instâncias de execução, visto que representa cada cenário de entrada e saída do shopping.
- Quando executado, o *peer* solicita o endereçamento do *tracker* (convencionou-se que a porta padrão do *tracker* é a 15000)
- Fazer isso com todos os *peers* necessários na rede;
- Os peers podem representar três cenários de entrada no shopping, sendo elas:
    - shopping - s
        - Funcionários
        - Clientes
    - Andar - a
    - Loja - l

```
1    $ python3 peer.py
2    IP do Tracker: 192.168.75.138
3    Atualizando informações...
4    Informe o Login: admin
5    Informe a Senha: admin
6    s - Shopping
7    a - andar
8    l - Loja
```

- Ao escolher o cenário de entrada loja, informar andar, limite e identificador da loja:
    - Neste cenário é definido o tipo de entrada/saída (funcionário (f) ou cliente (c)).
- O comando *help* pode ser executado em qualquer cenário e tem como objetivo exibir os comandos disponíveis no *peer*.
    - Os comandos entrou e saiu indicam o movimento das pessoas no cenário escolhido (shopping, andar e loja)

```
      -------- cenário de escolha s - loja --------
9     L
10    Qual andar eu estou?
11    1
12    Qual o limite de pessoas?
13    300
14    Digite o identificador da loja:
15    1
16    f - Funcionarios
17    c - Clientes
18    c
19    help
20    Lista de Comandos:
21    mPeer -> Ver o meu Peer
22    mPeers -> Ver lista de Peers
23    status -> Ver Estado do Shopping
24    saiu -> Informar saída de pessoas
25    entrou -> Informar entrada de pessoas
26    entrou
27    >> 10
28    Vindo de:
29    s - Entrando no Shopping
30    a - Vindo de outro Andar
31    l - Vindo de outra Loja
32    s
33    Status
34    Limite:      300
35    Atualmente:  10
36    Eu sou:      loja no 1º andar
37    Dados atualizados
```

- Cada comando de entrada e saída de pessoas vai atualizando o estado do shopping, que pode ser consultado por intermédio do comando *status*.

Esta representação de um cenário reduzido, com somente um peer, pode ser aplicada para cenários com vários peers executando os comandos descritos acima.