Laboratorio No.8

- Andrea Lam 20
- Paola de León 20361
- Gabriela Contreras 20213


```
import numpy as np
import simpy


#Genarado con ayuda de Chatgpt
class Servidor:
    def __init__(self, env, capacidad,i):
        self.env = env
        self.id = i + 1
        self.capacidad = capacidad
        self.solicitudes_atendidas = 0
        self.tiempo_ocupado = 0
        self.tiempo_desocupado = 0
        self.tiempo_servicio_total = 0
        self.tiempo_ultimo_fin_servicio = 0

    def atender_solicitud(self, solicitud):
        if self.solicitudes_atendidas < self.capacidad:
            tiempo_servicio = np.random.exponential(0.5)  # Tiempo de servicio exponencial
            yield self.env.timeout(tiempo_servicio)
            self.solicitudes_atendidas += 1
            self.tiempo_ocupado += tiempo_servicio
            self.tiempo_servicio_total += tiempo_servicio
            self.tiempo_ultimo_fin_servicio = self.env.now  # Actualiza el tiempo del último fin de servicio
            print(f"Servidor {self.id} atendió Solicitud {solicitud['id']} en {round(self.env.now, 4)} segundos")

def llegada_solicitudes(env, servidores, cola, tasa_llegada):
    solicitud_id = 0
    while True:
        yield env.timeout(np.random.exponential(1 / tasa_llegada))  # Llegada de solicitudes con proceso de Poisson
        solicitud_id += 1
        solicitud = {'id': solicitud_id, 'tiempo_llegada': env.now, 'tiempo_en_cola': 0}
        cola.append(solicitud)
        env.process(servicio_solicitud(env, servidores, cola, solicitud))

def servicio_solicitud(env, servidores, cola, solicitud):
    servidor_disponible = next((s for s in servidores if s.solicitudes_atendidas < s.capacidad), None)

    if servidor_disponible is not None:
        yield env.process(servidor_disponible.atender_solicitud(solicitud))
        tiempo_llegada = solicitud['tiempo_llegada']  # Guarda el tiempo de llegada
        tiempo_en_cola = env.now - tiempo_llegada  # Calcula el tiempo en cola
        solicitud['tiempo_en_cola'] = tiempo_en_cola
    else:
        print(f"Solicitud {solicitud['id']} en espera")
        yield env.timeout(1)  # Espera si no hay servidores disponibles



env = simpy.Environment()
tasa_llegada = 2400 / 60
capacidad_servidor = 100
numero_servidores = 1

servidores = [Servidor(env, capacidad_servidor,i) for i in range(numero_servidores)]
cola = []

env.process(llegada_solicitudes(env, servidores, cola, tasa_llegada))
env.run(until=3600)  # 1H en seg

# Calcular y mostrar las métricas
total_solicitudes_atendidas = sum(servidor.solicitudes_atendidas for servidor in servidores)
tiempo_total_solicitudes_en_cola = sum(solicitud['tiempo_en_cola'] for solicitud in cola)
promedio_tiempo_en_cola = tiempo_total_solicitudes_en_cola / len(cola)
```

```
Servidor 1 atendió Solicitud 44 en 1.0069 segundos
Servidor 1 atendió Solicitud 39 en 1.0571 segundos
Servidor 1 atendió Solicitud 38 en 1.0673 segundos
Servidor 1 atendió Solicitud 37 en 1.0678 segundos
Servidor 1 atendió Solicitud 30 en 1.1244 segundos
Servidor 1 atendió Solicitud 25 en 1.1556 segundos
Servidor 1 atendió Solicitud 46 en 1.1777 segundos
Servidor 1 atendió Solicitud 23 en 1.1978 segundos
Servidor 1 atendió Solicitud 53 en 1.2377 segundos
Servidor 1 atendió Solicitud 48 en 1.2404 segundos
Servidor 1 atendió Solicitud 43 en 1.2459 segundos
Servidor 1 atendió Solicitud 18 en 1.2858 segundos
Servidor 1 atendió Solicitud 58 en 1.2999 segundos
Servidor 1 atendió Solicitud 45 en 1.3252 segundos
Servidor 1 atendió Solicitud 54 en 1.3711 segundos
Servidor 1 atendió Solicitud 49 en 1.3998 segundos
Servidor 1 atendió Solicitud 47 en 1.4134 segundos
Servidor 1 atendió Solicitud 57 en 1.4449 segundos
Servidor 1 atendió Solicitud 61 en 1.4605 segundos
Servidor 1 atendió Solicitud 32 en 1.5667 segundos
Servidor 1 atendió Solicitud 59 en 1.6191 segundos
Servidor 1 atendió Solicitud 50 en 1.6549 segundos
Servidor 1 atendió Solicitud 41 en 1.7016 segundos
Servidor 1 atendió Solicitud 42 en 1.7081 segundos
Servidor 1 atendió Solicitud 67 en 1.7773 segundos
Servidor 1 atendió Solicitud 71 en 1.8133 segundos
Servidor 1 atendió Solicitud 70 en 1.815 segundos
Servidor 1 atendió Solicitud 62 en 1.8276 segundos
Servidor 1 atendió Solicitud 65 en 1.8637 segundos
Servidor 1 atendió Solicitud 75 en 1.8665 segundos
Servidor 1 atendió Solicitud 51 en 1.8927 segundos
Servidor 1 atendió Solicitud 68 en 1.9045 segundos
Servidor 1 atendió Solicitud 84 en 1.9436 segundos
Servidor 1 atendió Solicitud 82 en 1.9507 segundos
Servidor 1 atendió Solicitud 83 en 1.9764 segundos
Servidor 1 atendió Solicitud 22 en 1.9846 segundos
Servidor 1 atendió Solicitud 79 en 2.0036 segundos
Servidor 1 atendió Solicitud 86 en 2.0051 segundos
Servidor 1 atendió Solicitud 77 en 2.0092 segundos
Servidor 1 atendió Solicitud 69 en 2.0416 segundos
Servidor 1 atendió Solicitud 78 en 2.0594 segundos
Servidor 1 atendió Solicitud 63 en 2.0896 segundos
Servidor 1 atendió Solicitud 89 en 2.1019 segundos
Servidor 1 atendió Solicitud 88 en 2.1019 segundos
Servidor 1 atendió Solicitud 93 en 2.153 segundos
Servidor 1 atendió Solicitud 97 en 2.253 segundos
Servidor 1 atendió Solicitud 81 en 2.2686 segundos
Servidor 1 atendió Solicitud 72 en 2.2703 segundos
Servidor 1 atendió Solicitud 99 en 2.2993 segundos
Servidor 1 atendió Solicitud 95 en 2.3789 segundos
Servidor 1 atendió Solicitud 66 en 2.3934 segundos
Servidor 1 atendió Solicitud 91 en 2.4126 segundos
Servidor 1 atendió Solicitud 73 en 2.4235 segundos
Servidor 1 atendió Solicitud 105 en 2.4374 segundos
Servidor 1 atendió Solicitud 74 en 2.5 segundos
Servidor 1 atendió Solicitud 107 en 2.5448 segundos
```

```
# a. Solicitudes que atendió cada servidor
print(f"Solicitudes atendidas por cada servidor: {[servidor.solicitudes_atendidas for servidor in servidores]}\n")

# b. y c. Tiempo ocupado y desocupado de cada servidor
for servidor in servidores:
    tiempo_total_simulacion = env.now
    tiempo_desocupado = tiempo_total_simulacion - servidor.tiempo_servicio_total
    print(f"Servidor {servidor.id} - Tiempo ocupado: {round(servidor.tiempo_servicio_total, 4)} segundos")
    print(f"Servidor {servidor.id} - Tiempo desocupado: {round(tiempo_desocupado, 4)} segundos\n")

# d. Tiempo en total estuvieron las solicitudes en cola
print(f"Tiempo total en cola de las solicitudes: {round(tiempo_total_solicitudes_en_cola, 4)} segundos\n")

# e. promedio tiempo estuvo cada solicitud en cola
print(f"Promedio de tiempo en cola: {round(promedio_tiempo_en_cola, 4)} segundos\n")

# f. Promedio de solicitudes en cola cada segundo
solicitudes_en_cola = [0] * 3600  # Inicializa una lista para cada segundo
for solicitud in cola:
    segundo = int(solicitud['tiempo_llegada'])
    solicitudes_en_cola[segundo] += 1
promedio_solicitudes_en_cola = sum(solicitudes_en_cola) / len(solicitudes_en_cola)
print(f"Promedio de solicitudes en cola cada segundo: {round(promedio_solicitudes_en_cola, 4)}\n")

# g. Momento de salida de la última solicitud
```

```
salida_ultima_solicitud = max(solicitud['tiempo_llegada'] + solicitud['tiempo_en_cola'] for solicitud in cola)
print(f"Momento de salida de la última solicitud: {round(salida_ultima_solicitud, 4)} segundos")
```

```
Solicitudes atendidas por cada servidor: [121]

Servidor 1 - Tiempo ocupado: 55.3243 segundos
Servidor 1 - Tiempo desocupado: 3544.6757 segundos

Tiempo total en cola de las solicitudes: 55.3243 segundos

Promedio de tiempo en cola: 0.0004 segundos

Promedio de solicitudes en cola cada segundo: 40.025

Momento de salida de la última solicitud: 3599.9774 segundos
```

```
def simulacion(tasa_llegada, max_servidores=10):
    for numero_servidores in range(1, max_servidores + 1):
        env = simpy.Environment()  # Crea un nuevo entorno de simulación
        servidores = [Servidor(env, 100) for _ in range(numero_servidores)]
        cola = []
        env.process(llegada_solicitudes(env, servidores, cola, tasa_llegada))
        env.run(until=3600)  # Simula 1 hora

        if not cola:  # Si no hay solicitudes en cola, se cumple la condición
            return numero_servidores

    return max_servidores  # Si se llega al límite máximo, regresa el máximo

# Tasa de llegada inicial
tasa_llegada_inicial = 2400 / 60
num_servidores_necesarios = simulacion(tasa_llegada_inicial)

print(f"Número de servidores necesarios para {tasa_llegada_inicial} solicitudes por minuto: {num_servidores_necesarios} servidores")
```

```
Número de servidores necesarios para 40.0 solicitudes por minuto: 10 servidores
```

```
env = simpy.Environment()
tasa_llegada = 6000 / 60  # Max de llamadas
capacidad_servidor = 100
numero_servidores = 1
servidores = [Servidor(env, capacidad_servidor, i) for i in range(numero_servidores)]
cola = []

env.process(llegada_solicitudes(env, servidores, cola, tasa_llegada))
env.run(until=3600)  # Simulación de 60 segundos para cumplir un minuto

# Calcular y mostrar las métricas
total_solicitudes_atendidas = sum(servidor.solicitudes_atendidas for servidor in servidores)
tiempo_total_solicitudes_en_cola = sum(solicitud['tiempo_en_cola'] for solicitud in cola)
promedio_tiempo_en_cola = tiempo_total_solicitudes_en_cola / len(cola)
```

```
Servidor 1 atendió Solicitud 102 en 1.2924 segundos
Servidor 1 atendió Solicitud 73 en 1.293 segundos
Servidor 1 atendió Solicitud 119 en 1.3067 segundos
Servidor 1 atendió Solicitud 98 en 1.3303 segundos
Servidor 1 atendió Solicitud 96 en 1.3322 segundos
Servidor 1 atendió Solicitud 107 en 1.3483 segundos
Servidor 1 atendió Solicitud 71 en 1.351 segundos
Servidor 1 atendió Solicitud 113 en 1.3714 segundos
Servidor 1 atendió Solicitud 111 en 1.3747 segundos
Servidor 1 atendió Solicitud 116 en 1.3813 segundos
Servidor 1 atendió Solicitud 122 en 1.3872 segundos
Servidor 1 atendió Solicitud 79 en 1.4005 segundos
Servidor 1 atendió Solicitud 128 en 1.4074 segundos
Servidor 1 atendió Solicitud 104 en 1.4176 segundos
Servidor 1 atendió Solicitud 74 en 1.4211 segundos
Servidor 1 atendió Solicitud 125 en 1.4345 segundos
Solicitud 134 en espera
Solicitud 135 en espera
Servidor 1 atendió Solicitud 93 en 1.4935 segundos
Servidor 1 atendió Solicitud 92 en 1.4966 segundos
Servidor 1 atendió Solicitud 129 en 1.5023 segundos
Servidor 1 atendió Solicitud 101 en 1.5113 segundos
Solicitud 136 en espera
Solicitud 137 en espera
Solicitud 138 en espera
Solicitud 139 en espera
Servidor 1 atendió Solicitud 124 en 1.5235 segundos
Solicitud 140 en espera
Solicitud 141 en espera
Solicitud 142 en espera
Solicitud 143 en espera
Solicitud 144 en espera
```

```
# a. Solicitudes que atendió cada servidor
print(f"Solicitudes atendidas por cada servidor: {[servidor.solicitudes_atendidas for servidor in servidores]}\n")

# b. y c. Tiempo ocupado y desocupado de cada servidor
for servidor in servidores:
    tiempo_total_simulacion = env.now
    tiempo_desocupado = tiempo_total_simulacion - servidor.tiempo_servicio_total
    print(f"Servidor {servidor.id} - Tiempo ocupado: {round(servidor.tiempo_servicio_total, 4)} segundos")
    print(f"Servidor {servidor.id} - Tiempo desocupado: {round(tiempo_desocupado, 4)} segundos\n")

# d. Tiempo en total estuvieron las solicitudes en cola
print(f"Tiempo total en cola de las solicitudes: {round(tiempo_total_solicitudes_en_cola, 4)} segundos\n")

# e. promedio tiempo estuvo cada solicitud en cola
print(f"Promedio de tiempo en cola: {round(promedio_tiempo_en_cola, 4)} segundos\n")

# f. Promedio de solicitudes en cola cada segundo
solicitudes_en_cola = [0] * 3600  # Inicializa una lista para cada segundo
for solicitud in cola:
    segundo = int(solicitud['tiempo_llegada'])
    solicitudes_en_cola[segundo] += 1
promedio_solicitudes_en_cola = sum(solicitudes_en_cola) / len(solicitudes_en_cola)
print(f"Promedio de solicitudes en cola cada segundo: {round(promedio_solicitudes_en_cola, 4)}\n")

# g. Momento de salida de la última solicitud
salida_ultima_solicitud = max(solicitud['tiempo_llegada'] + solicitud['tiempo_en_cola'] for solicitud in cola)
print(f"Momento de salida de la última solicitud: {round(salida_ultima_solicitud, 4)} segundos")
```

```
Solicitudes atendidas por cada servidor: [133]

Servidor 1 - Tiempo ocupado: 57.8603 segundos
Servidor 1 - Tiempo desocupado: 3542.1397 segundos

Tiempo total en cola de las solicitudes: 57.8603 segundos

Promedio de tiempo en cola: 0.0002 segundos

Promedio de solicitudes en cola cada segundo: 99.9858

Momento de salida de la última solicitud: 3599.9994 segundos
```

```
def simulacion(tasa_llegada, max_servidores=10):
    for numero_servidores in range(1, max_servidores + 1):
        env = simpy.Environment()  # Crea un nuevo entorno de simulación
        servidores = [Servidor(env, 100) for _ in range(numero_servidores)]
        cola = []
        env.process(llegada_solicitudes(env, servidores, cola, tasa_llegada))
```

```
            env.run(until=3600)  # Simula 1 hora

            if not cola:  # Si no hay solicitudes en cola, se cumple la condición
                return numero_servidores

        return max_servidores  # Si se llega al límite máximo, regresa el máximo

    # Tasa de llegada inicial
    tasa_llegada_inicial = 6000 / 60
    num_servidores_necesarios = simulacion(tasa_llegada_inicial)

    print(f"Número de servidores necesarios para {tasa_llegada_inicial} solicitudes por minuto: {num_servidores_necesarios} servidores")
```

```
Número de servidores necesarios para 100.0 solicitudes por minuto: 25 servidores
```