

Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie

Wydział Elektrotechniki, Automatyki, Informatyki
i Inżynierii Biomedycznej



Studio Projektowe 2

Symulacja kolejek przy użyciu systemu wieloagentowego

Autorzy:

Łukasz Bednarski

Patryk Papiór

Informatyka, IV rok,
Rok akademicki 2019/20

Prowadzący:

dr inż. Radosław Klimek

1. Wstęp

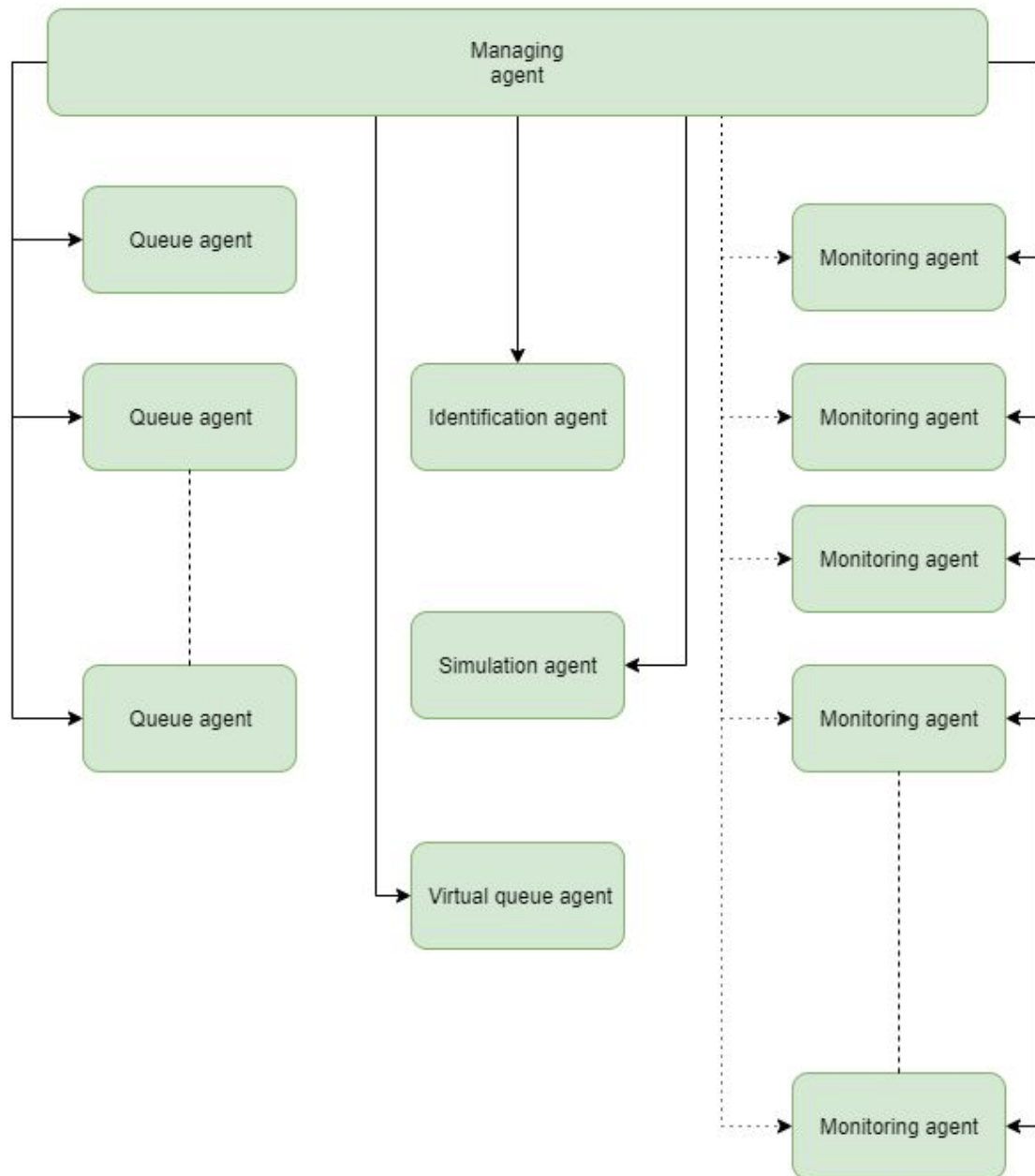
Celem niniejszego projektu było zaprojektowanie oraz implementacja symulacji kolejek występujących w różnych miejscach użytku publicznego (np. w supermarketach) oraz systemu wieloagentowego (z ang. multi-agent-system) złożonego z agentów (niżej szerzej opisanych) wykonujących powierzone im zadania.

Kolejka jest rozumiana przez pewien uporządkowany ciąg jednostek (klientów) oczekujących na obsługę.

2. System wieloagentowy (MAS)

Taksonomia agentów:

- **Agent dokonujący identyfikacji nowego klienta** (Identification agent - **ID**) - Agent odczytujący dane o nowym kliencie (przy pomocy współpracującym z nim sensorów odczytujących odcisk palca). Informuje cały system o pojawieniu się nowego klienta. Istnieje w systemie na stałe.
- **Agent monitorujący** zachowanie klienta i określający jego atrybuty (Monitoring agent - **MA**) - Dokonywana jest klasyfikacja klienta wchodzącego do sklepu. Określane są atrybuty takie jak: wiek, niepełnosprawność, płeć, temperatura ciała (może sugerować chorobę).. Agent jest tworzony dla każdego klienta.
- **Agent kolejki wirtualnej** (Virtual queue agent - **VQ**) - Monitoruje stan kolejki wirtualnej, podejmuje decyzję o przyporządkowaniu klienta do odpowiedniej kolejki rzeczywistej na podstawie rozpoznania wyżej wymienionych atrybutów. Istnieje w systemie na stałe.
- **Agent kolejki** (Queue agent - **QA**) - Zarządza stanem konkretnej, przypisanej do niego kolejki (która z kolei jest przypisana do konkretnej kasy). Dodaje/usuwa klientów. Posiada wiedzę o typie kolejki do której jest przypisany, ilość klientów znajdujących się w danej kolejce, średni czas oczekiwania na obsługę etc. Agent istnieje w systemie na stałe (Ilość agentów jest równa ilości kolejek oraz jest predefiniowana)
- **Agent zarządzający** (Managing agent - **MGA**) - Definiuje/Inicjalizuje szereg stałych wykorzystywanych w systemie (definiujących go), udostępnia dane innym agentom oraz nimi zarządza.



Rys. 1.. Powyższy diagram przedstawia podstawową architekturę omawianego systemu wieloagentowego. Strzałki nieprzerwane przedstawiają relację konstrukcji obiektu agenta(przez zarządcę) natomiast strzałki przerywane odnoszą się do niszczenia danego agenta.

3. Kolejki

Istnieją 4 rodzaje kolejek:

- **VIP** - dla klientów, którzy posiadają status vip'a
- **THERMAL** -dla klientów, którzy zdiagnozowani jako chorzy lub śpieszący się.
- **SPECIAL** - dla osób starszych(elderly), kobiet w ciąży(pregnant), osób z niepełnosprawnościami(handicapped)
- **NORMAL** - dla pozostałych osób(ordinary people)

Rodzaje kolejek są przechowywane w klasie **QueueTypes**, która jest gotowa na ewentualne rozszerzenia.

Konfiguracją kolejek zajmuje się Managing Agent. Oprogramowanie dostarcza możliwość określenia rodzajów kolejek występujących w systemie oraz ich ilość. Domyślnie w systemie są zainstalowane wszystkie rodzaje określonych wyżej kolejek, a ich ilość jest równa 1.

4. Urządzenia fizyczne

- Czytnik linii papilarnych - współpracuje z **IDA**, zczytuje dane biometryczne klienta.
- Kamera - współpracuje z **MA**, jej zadaniem jest wysłać dane do agenta. W systemie istnieje siatka kamer, tworzących graf oraz śledzących ruchy klientów.
- Kamera termiczna - współpracuje z **MA**, czyta dane termiczne (temperatura) poruszającej się jednostki

```
# queue_type: count, there must at least one queue for each type
queues_config = {
    QueueType.VIP: 1,
    QueueType.THERMAL: 1,
    QueueType.SPECIAL: 1,
    QueueType.NORMAL: 1
}
```

Rys.2. - Podstawowa konfiguracja kolejek w klasie Managing Agent

5. Symulacja - opis

a. Wejście

- Czas symulacji (wyrażony w sekundach [s])
- Tryb symulacji - związany bezpośrednio z częstotliwością pojawiania się nowych klientów w systemie [1].
- Określenie wielkości puli klientów do losowania
- Prędkość symulacji
- Rozmiar obszaru, po którym może poruszać się każdy klient. Ten obszar jest przedstawiony jako zbiór kamer monitorujących ruchy klientów
- Dane symulacji są ładowane z pliku **/in/config.json** znajdującego się w głównym katalogu projektu

b. Wyjście

- Plik z logami w formacie JSON. Dane w nim zawarte obejmują całą wiedzę zdobytą po przeprowadzonej symulacji np. dane o klientach, którzy uczestniczyli w systemie, ich ścieżki, czasy poszczególnych przejść, wiedzę o kolejkach, ich typach, przypisanych do nich klientów. Logi dostarczają również informacji o różnych statystykach np. średni czas oczekiwania w kolejce, ilość kobiet w ciąży etc.
- Wyświetlanie informacji w konsoli na temat stanu poszczególnych kolejek podczas trwania symulacji.
- Wizualizacja ścieżki losowo wybranego klienta podczas trwania symulacji
- Wizualizacja stanu kolejek przy pomocy histogramu

c. Tryby symulacji

- **VERY LOW** - Odstęp między pojawieniem się nowych klientów leży w przedziale [50, 800] [s]
- **LOW** - [30, 600] [s]
- **MEDIUM** - [30, 180] [s]
- **HIGH** - [10, 100] [s]
- **VERY HIGH** - [10, 60] [s]
- **ULTIMATE** - [1, 10] [s]

Przedział [a, b] oznacza, iż odstęp czasu t, po którym pojawi się nowy klient jest: $a < t < b$.

d. Przebieg symulacji

1. Na samym początku generowana jest pula klientów, z których część z nich będzie uczestniczyć w systemie (lub wszyscy - zależne od innych parametrów np. czas trwania symulacji).

2. Na potrzeby symulacji tworzymy historię systemu (zapisujemy niektórych klientów, którym jest przyznawany status VIP. Klienci, którzy znani są systemowi, zostają zapisani w bazie danych).

3. Tworzymy kolejki i przypisujemy im odpowiedni typ oraz wszystkich "stałych" agentów (oprócz **MGA** który już istnieje).

4. Po stworzeniu środowiska systemu zaczynamy właściwą część symulacji, każdy klient, który pojawia się w systemie jest identyfikowany przez **IDA**, który wykorzystuje do tego czytnik linii papilarnych. Porównywany jest z istniejącymi już klientami w bazie, nadawany jest mu odpowiedni status. Dane

biometryczne są reprezentowane w bazie klientów jako unikalny hash - generowany w trakcie "tworzenia" klienta.

Klient otrzymuje "własnego" unikalnego agenta monitorującego(**MA**), który współpracuje z szeregiem urządzeń(IoT) takimi jak: kamera, kamera termowizyjna etc.

Klient znajduje się także pod ciągłą obserwacją agenta kolejki wirtualnej(VQ), który podejmuje decyzje na podstawie zebranych danych o kliencie.

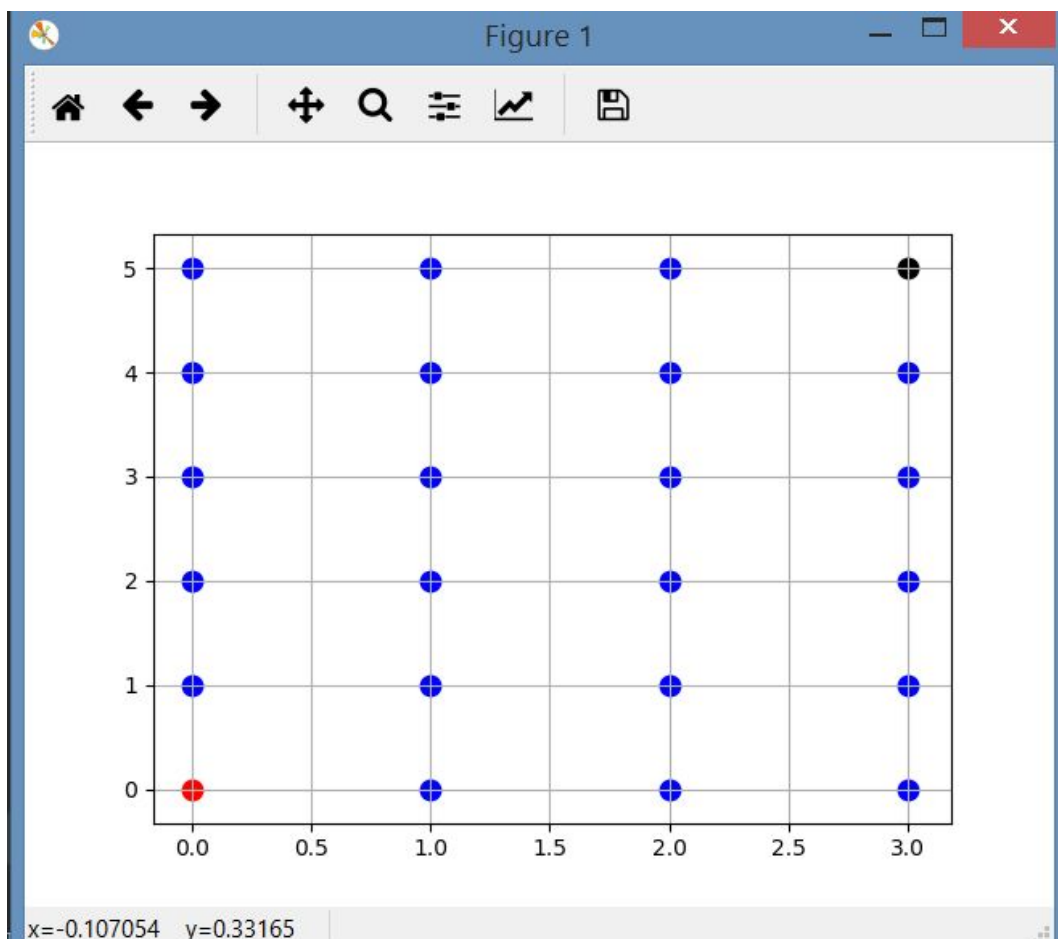
Agent kolejki fizycznej (**QA**) dodaje do/usuwa z kolejki klienta oraz zarządza jej stanem.

Pod sam koniec symulacji klient jest usuwany z systemu a jego agent jest dezaktywowany.

Na sam koniec symulacji następuje prezentacja danych w wyżej wymienionej formie, rozważana jest również możliwość logowania stanu symulacji podczas jej trwania.

e. Obszar monitorowany - scena

Każdy punkt, węzeł na Rys.3 odpowiada faktycznemu urządzeniu umiejscowionym na terenie, które miałoby być monitorowane(np. teren sklepu). Każda kamera monitoruje pewien wybrany przez zarządcę obszar, który przedstawiany w tym wypadku jako punkt.



Rys.3. Przedstawienie sceny
Przyjęta konwencja kolorystyczna na grafie:

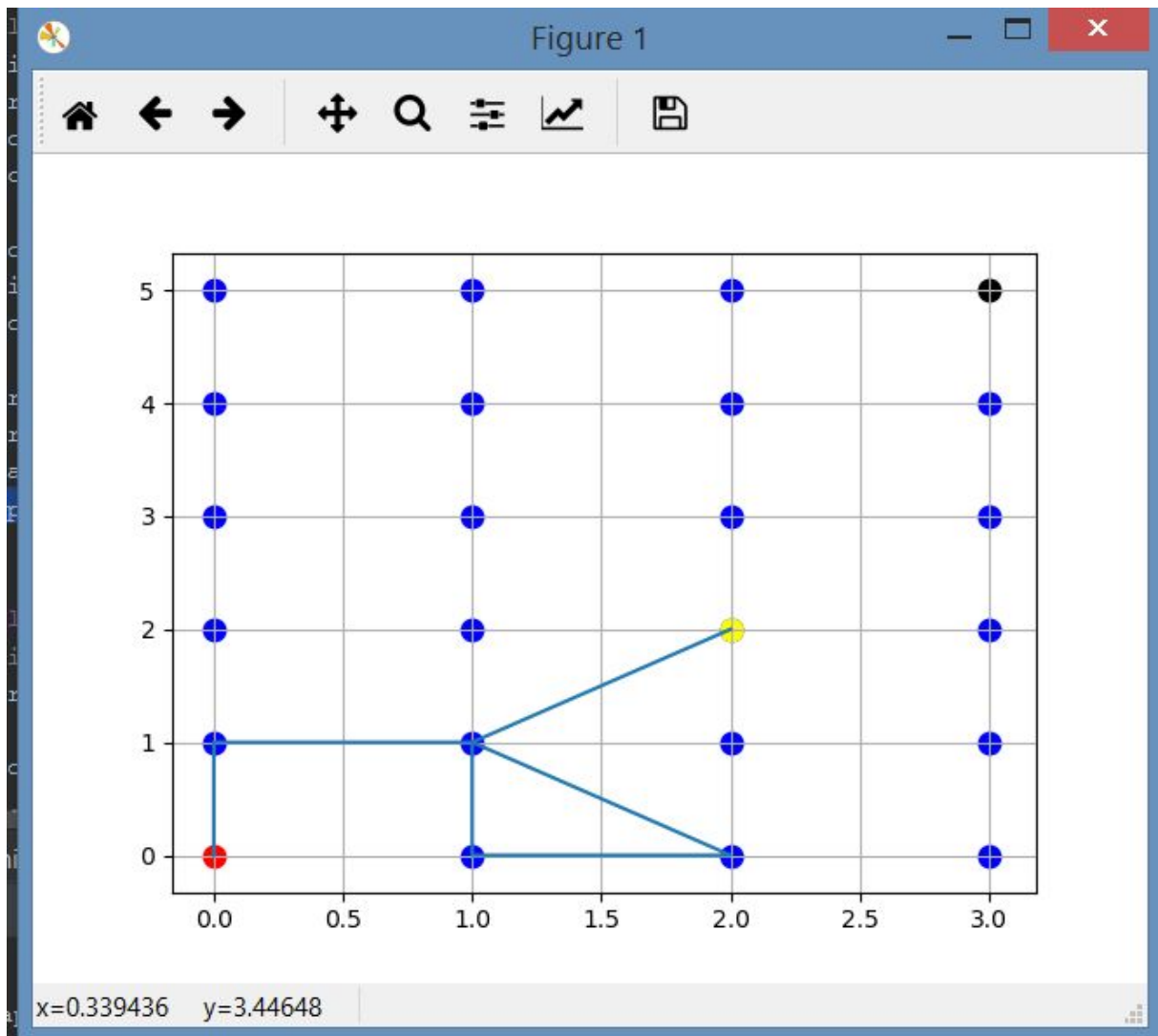
Czerwony - Punkt startowy, klient po wstępnej identyfikacji/rejestracji przy wejściu, otrzymuje swojego agenta monitorującego oraz zaczyna się poruszać po grafie

Czarny - Punkt końcowy, klient kończy swój pobyt na terenie monitorowanym. Ten węzeł odpowiada obszarowi wirtualnej kolejki, w której klient dostaje informację do której kolejki fizycznej należy się udać.

Żółty - Punkt, w którym obecnie znajduje się monitorowana jednostka. Gdy przejdzie ona całą ścieżkę, żółty punkt będzie pokrywał się z punktem końcowym.

f. **Śledzenie ruchów klienta w obszarze monitorowanym**

W poprzednim podpunkcie zaprezentowano model sceny, po której poruszać się mogą wszystkie monitorowane jednostki przebywające w systemie.



Rys.4. Przykładowa ścieżka(fragment), którą przebywa klient

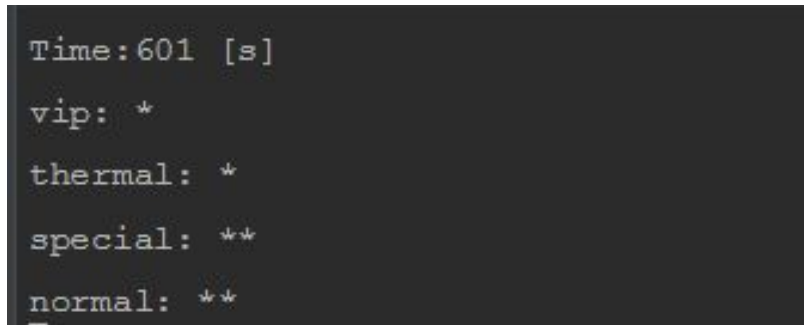
Logi powyższej ścieżki są następujące(dostępne w pliku

src/data/tracked_customer.log):

0,0 -> 0,1 -> 1,1 -> 1,0 -> 2,0 -> 1,1 -> 2,2

g. Logowanie stanu kolejek w konsoli

W trakcie działania programu, użytkownik ma możliwość do wglądu stanów poszczególnych kolejek “na żywo” poprzez analizę logów w konsoli.

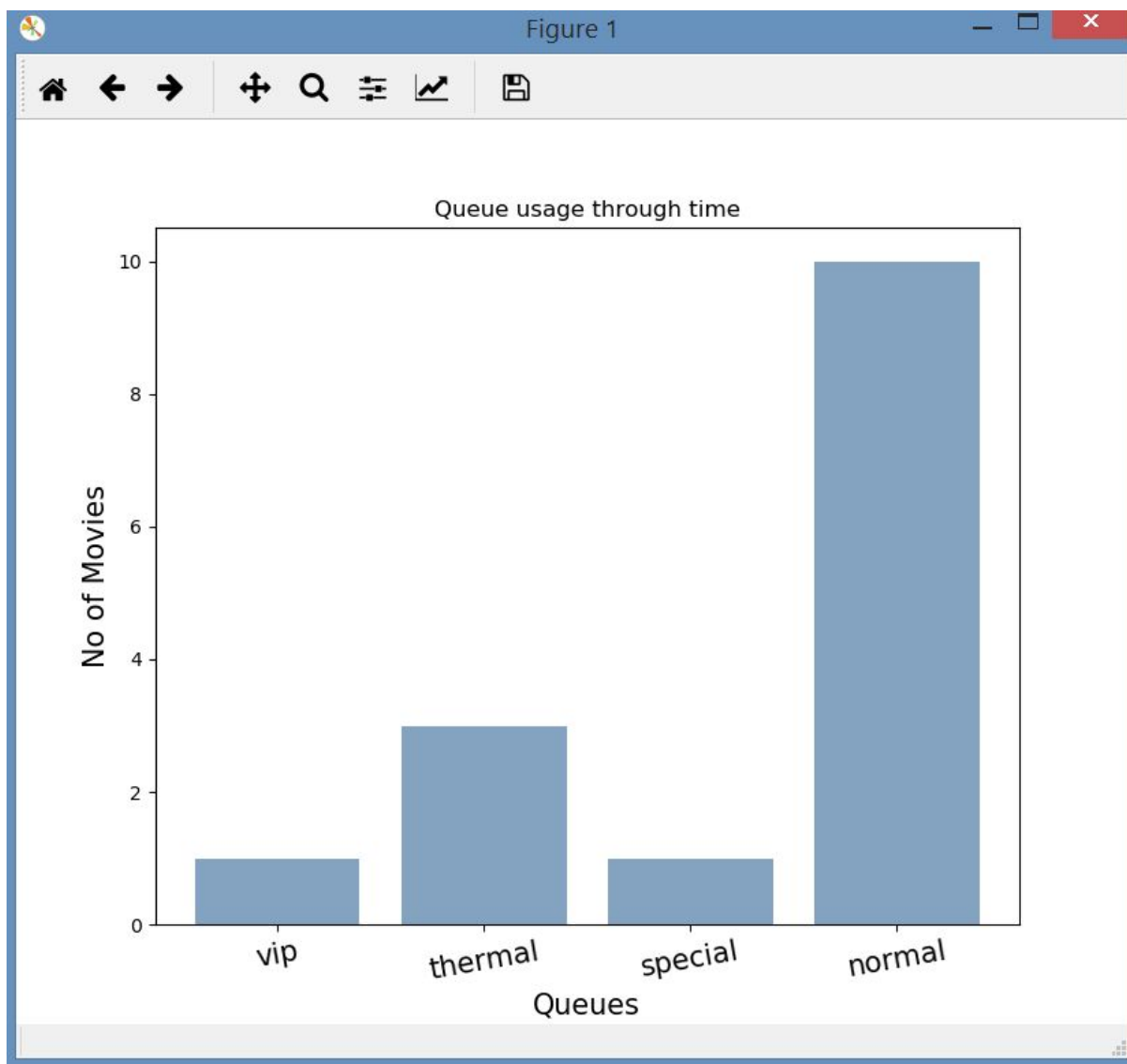
A screenshot of a terminal window with a dark background and light-colored text. The text shows the time as 601 seconds and the status of four queues: vip (1 person), thermal (1 person), special (2 people), and normal (2 people).

```
Time:601 [s]
vip: *
thermal: *
special: **
normal: **
```

Rys. 5. Przykładowy zapis stanu kolejki. Każda gwiazdka “*” reprezentuje jedną osobę zajmującą miejsce w kolejce.

h. Wizualizacja stanu kolejek

Stan kolejek można również sprawdzić przy pomocy histogramu



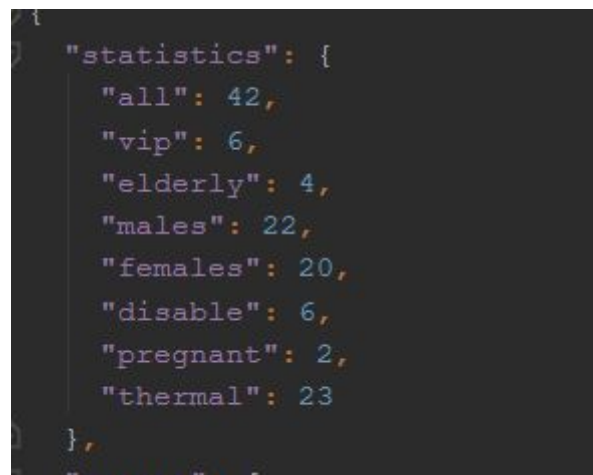
6. Opis pliku wyjściowego

We wcześniejszych rozdziałach można było zobaczyć różne sposoby odczytu stanu symulacji tj. logi w konsoli, histogramy kolejek. Symulacja na swoim wyjściu produkuje plik wyjściowy w formacie JSON znajdujący się w **/out/output.json** w głównym katalogu projektu.

Plik wyjściowy zawiera informację na temat przebiegu symulacji w dowolnym jej momencie.

Jest on podzielony na sekcje:

- statystyczna - zawiera informację na temat ilości klientów, atrybuty tychże klientów np. czy są vip'ami, płeć etc.



```
{
  "statistics": {
    "all": 42,
    "vip": 6,
    "elderly": 4,
    "males": 22,
    "females": 20,
    "disable": 6,
    "pregnant": 2,
    "thermal": 23
  },
  "queues": [

```

Rys. 7. Przykładowe dane z sekcji statystycznej

- Informację o kolejkach(queues) - składa się z:
 - ❖ id(index) kolejki - identyfikator
 - ❖ typ - VIP, THERMAL... etc
 - ❖ total - ilość osób, której przyporządkowano tą kolejkę w czasie całej symulacji
 - ❖ waiting_at_the_end - ilość osób stojących w kolejce w momencie zakończenia symulacji

- ❖ queue - lista identyfikatorów klientów, którzy stali w tej kolejce podczas całej symulacji
- ❖ mean_waiting_time - średni czas oczekiwania na obsługę
- ❖ active_customers_per_time - ilość klientów stojących w kolejce podczas każdego momentu(każdej sekundy) symulacji. Dla przykładu jeśli symulacja trwała godzinę, lista zawiera 3600 elementów.

```

"queues": [
  {
    "index": 0,
    "type": "VIP",
    "total": 4,
    "waiting_at_the_end": 0,
    "queue": [
      761,
      444,
      604,
      366
    ],
    "mean_waiting_time": 67.0,
    "active_customers_per_time": [
      0,
      0,
      0,
      0,
      0,
      0,
      0,
      0
    ]
  }
]

```

Rys.8. Przykładowe dane z sekcji kolejek(queues). W tym przypadku dane z kolejki typu "VIP"

- Informacje o klientach(customers)
 - ❖ index - identyfikator klienta w bazie danych(unikalny)
 - ❖ biometric - dane biometryczne pobrane z sensora(unikalne)
 - ❖ is_new - Czy klient już wcześniej był w bazie danych?
 - ❖ sex - płeć {"M", "F"}
 - ❖ disable - Niepełnosprawność
 - ❖ pregnant - dotyczy "F", informacja o ciąży
 - ❖ thermal - czy jest "termiczny"?
 - ❖ tracked_path - Ścieżka jaką klient przebył podczas symulacji
 - ❖ times - czasy przejść(Jeśli tracked_path ma długość n, to lista times ma długość n-1)

- ❖ `total_shopping_time` - całkowity czas spędzony podczas przechodzenia grafu, jest on równy sumie czasów z listy `times`
- ❖ `waiting_time` - czas oczekiwania w kolejce

7. Baza danych

Baza danych jest bardzo prosta - składa się z jednej tabeli o nazwie "customers". Przechowuje informację o "szczególnych klientach - VIPach", którzy są znani systemowi.

Fizyczny model bazy jest następujący:

```
CREATE DATABASE queue simulation;
CREATE TABLE `customers` (
  `id` INT(1) UNSIGNED ZEROFILL NOT NULL AUTO INCREMENT
  COMMENT 'Primary key',
  `biometric` VARCHAR(36) NULL DEFAULT NULL COMMENT
  'Contains biometric data',
  `customer status` ENUM('NORMAL','REGULAR','VIP') NULL
  DEFAULT NULL COMMENT 'Describes the customer judging the
  frequency',
  `is new` SMALLINT(1) NULL DEFAULT NULL COMMENT 'Informs
  if the customer was known before the simulation started
  running',
  PRIMARY KEY (`id`),
  UNIQUE INDEX `biometric` (`biometric`)
  COLLATE='utf8mb4 general ci'
  ENGINE=InnoDB
  AUTO INCREMENT=128;
```

8. Instrukcja obsługi

1. Instalacja języka programowania python

Pierwszym krokiem jest instalacja języka python na maszynie, na której będzie uruchamiany program. Oprogramowanie zostało napisane w python v3.6.5(istotne jest aby instalowany język był w wersji 3 lub wyższej).

2. Stworzenie oraz konfiguracja bazy danych

Przed próbą uruchomienia programu należy uruchomić usługę MySQL na serwerze(np. na lokalnym).

Uruchomić skrypt tworzący bazę znajduje się w następującym katalogu: `/scripts/db_setup.sql`. Należy wykonać kolejno znajdujące się w nim komendy.

Domyślnie ustawionym użytkownikiem jest **root**, a hasło **admin**.

Należy upewnić się, iż ten użytkownik ma prawo dostępu do stworzonej bazy danych.

3. Instalacja bibliotek/zależności

Przy zainstalowanym menedżerze pakietów PIP uruchamiamy następującą komendę będąc w katalogu głównym projektu:

```
pip install -r requirements.txt
```

4. Uruchomienie programu

Gdy wszystkie poprzednie warunki zostały spełnione, uruchamiamy następującą komendę w katalogu `/src`:

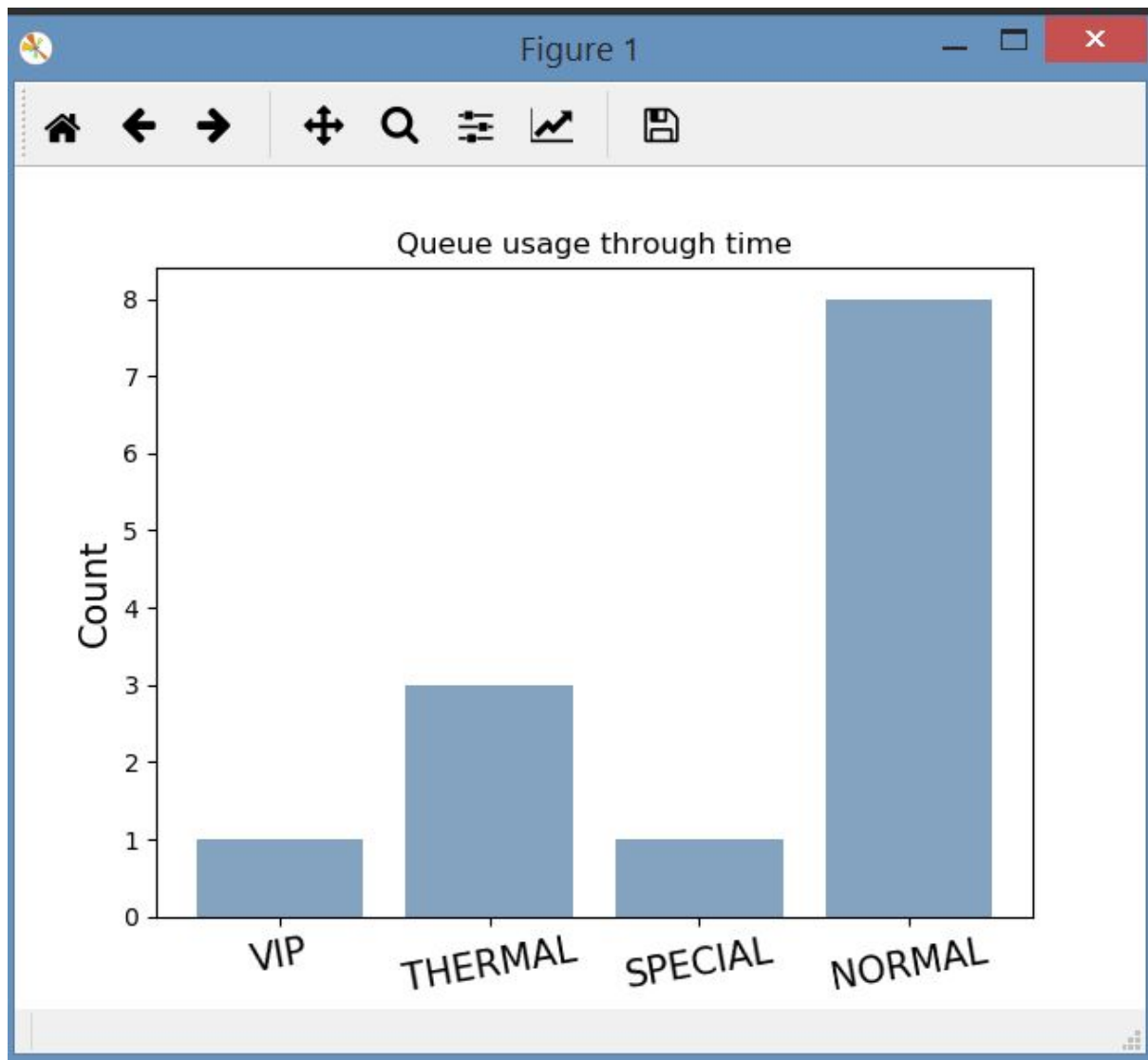
```
python app.py
```

5. Wizualizacja stanu kolejek

W katalogu **src/Visual/** uruchamiamy program komendą:

```
python queue_bar_plot.py
```

Program nasłuchuje zmiany w pliku **src/data/histogram.log** oraz nanosi zmiany na widoczny histogram.



Rys. 6. Przykładowy wykres stanu kolejki

6. Wizualizacja grafu

W katalogu **src/Visual/** uruchamiamy program komendą:

```
python graph.py
```

Program nasłuchuje zmiany w pliku **src/data/tracked_customer.log** oraz nanosi zmiany na przedstawionym grafie.

9. Opis technologii

Cała logika symulacji zaimplementowana została przy pomocy języka python wersji ≥ 3 .

Do wizualizacji ścieżki przejścia obiektów monitorowanych
zastosowano bibliotekę Matplotlib
System zarządzania bazą danych(RDBMS) - MySQL.

10. Literatura

- Artykuł "Context-aware and pro-active queue management systems in intelligent environments". Autorem tej pozycji jest dr inż Radosław Klimek.
- Artykuł "PDES-MAS: Distributed simulation of multi-agent systems" Vinoth Suryanarayanan , Georgios Theodoropoulosb,* , Michael Leesc