Raport z Projektu - System Obsługi Lotniska

Aleksander Dygoń 151856

Spis treści

1. Założenia projektowe

Projekt implementuje system symulujący działanie lotniska z następującymi głównymi komponentami:

- Generator pasażerów:
  Proces, który w nieskończonej pętli generuje nowych pasażerów, nie przekraczając ustawionego limitu procesów określonego przez użytkownika.
- Kontrola biletowo-bagażowa:
  Pobiera wygenerowanych pasażerów i sprawdza ich bilety oraz bagaż pod kątem zgodności z kryteriami wagowymi lotniska. Po pozytywnej weryfikacji przydziela pasażerów do odpowiednich stanowisk kontroli bezpieczeństwa, wykorzystując algorytm *Round Robin*.
- Kontrola bezpieczeństwa:
  Sprawdza pasażerów pod kątem posiadania niebezpiecznych przedmiotów. Po pozytywnej kontroli kieruje ich do hali odlotów. Kontrola odbywa się przy maksymalnie dwóch osobach na stanowisko, przy zachowaniu zasady tej samej płci. Pasażerowie mogą przepuścić maksymalnie trzy osoby w kolejce. System obsługuje także pasażerów VIP, którzy mają priorytet i mogą ominąć kolejkę.

- System zarządzania bramkami (gates):
    Po otrzymaniu sygnału o gotowości samolotu kieruje pasażerów do odpowiedniej bramki i organizuje boarding.
- Dyspozytor lotów:
    Komunikuje się z bramkami oraz samolotem. Po sprawdzeniu odpowiedniej liczby pasażerów wydaje zgodę na rozpoczęcie boardingu, a następnie na start samolotu. W przypadku nadmiernego zatłoczenia lotniska może natychmiast zatrzymać pracę lotniska.
- Proces obsługi samolotu:
    Po zakończeniu boardingu i otrzymaniu zgody na start rozpoczyna lot. Po określonym czasie wraca na lotnisko, kończąc swój cykl operacyjny.

1.1 Główne wymagania funkcjonalne:

- Odprawa biletowo-bagażowa z limitem wagowym
- 3 równoległe stanowiska kontroli bezpieczeństwa
- Maksymalnie 2 osoby tej samej płci na stanowisku
- Limit 3 przepuszczeń w kolejce dla każdego pasażera
- System obsługi VIP
- Zarządzanie schodami pasażerskimi o ograniczonej pojemności
- Koordynacja oraz zarządzanie flotą samolotów przez dyspozytora

2. Implementacja wymagań obowiązkowych

2.1 Dokumentacja przypadków użycia

Projekt zawiera szczegółową dokumentację w postaci komentarzy w kodzie oraz struktury modułowej reprezentującej poszczególne komponenty systemu. Główne przypadki użycia są zaimplementowane w oddzielnych modułach:

- ./src/generator.py - generowanie pasażerów
- ./src/luggageControl.py - kontrola bagażowa
- ./src/securityControl.py - kontrola bezpieczeństwa
- ./src/gate.py - obsługa bramek
- ./src/dispatcher.py - zarządzanie lotami
- ./src/airplane.py - obsługa samolotu
- ./src/consts.py - stałe oraz konfiguracja programów

2.2 Walidacja danych

Walidacja została zaimplementowana w module utils.py w funkcji validate_config()

2.3 Obsługa błędów

Obsługa błędów systemowych jest zaimplementowana w module utils.py w funkcji handle_system_error()

2.4 Minimalne prawa dostępu

Projekt implementuje minimalne prawa dostępu dla tworzonych struktur:

- Kolejek
- Plików

3. Realizacja wymagań

a. Tworzenie i obsługa plików

- Tworzenie pliku
- Zapis do pliku
- Odczyt z pliku
- Unlink

b. Tworzenie procesów

- Fork
- Exit
- Wait

c. Obsługa sygnałów

- Obsługa KeyboardInterrupt
- Kill

d. Synchronizacja procesów

- Synchronizacja przez pliki

e. Kolejki komunikatów

- Tworzenie kolejek
- Dodawanie do kolejki
- Pobieranie z kolejki

4. Problemy i rozwiązania

4.1 Synchronizacja dostępu do plików

Problem: Równoległy dostęp do plików przez różne procesy.

Rozwiązanie: Wykorzystanie mechanizmu fcntl do blokowania i odblokowywania plików

4.2 Koordynacja procesów

Problem: Koordynacja wielu procesów i przekazywanie sygnałów.
Rozwiązanie: Wykorzystanie kolejek komunikatów oraz komunikację przy pomocy plików:

- Tworzenie kolejek
- Dodawanie do kolejki
- Pobieranie z kolejki
- Synchronizacja przez pliki

5. Testy

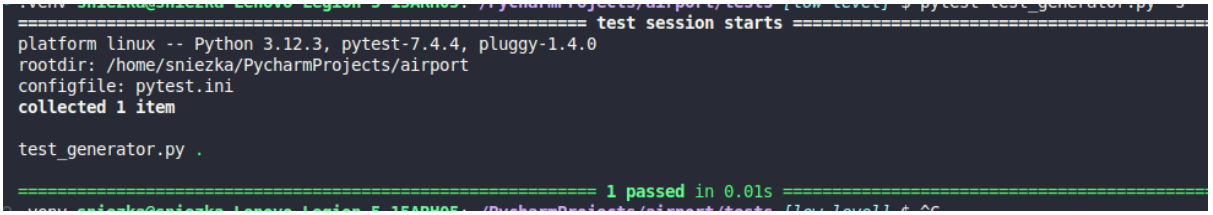
5.1 Struktura testów

Projekt zawiera kompleksowy zestaw testów:

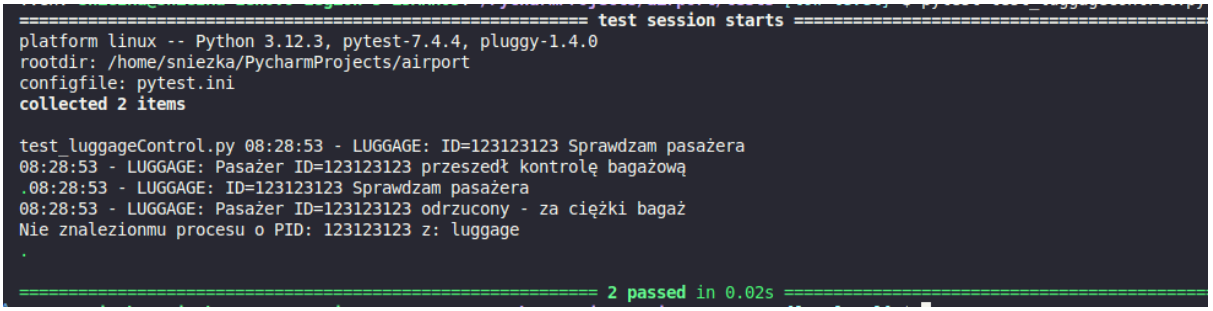
1. Testy jednostkowe poszczególnych komponentów:

Wszystkie testy zakładają konfigurację stałych parametrów zgodnie z branchem master.

- ./tests/test_generator.py - testy generatora pasażerów
  - TEST_1: Wygenerowanie oraz sprawdzanie poprawności generowanych danych zgodnie z konfiguracją parametrów w module consts



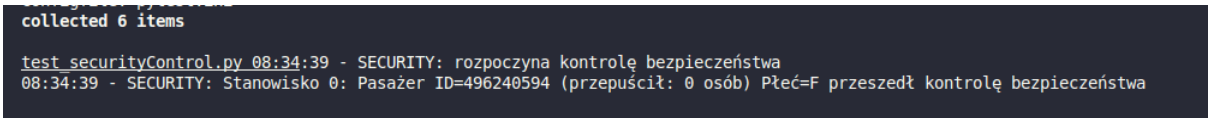
- ./tests/test_luggageControl.py - testy kontroli bagażowej
  - TEST_1: Kontrola pasażera z prawidłową wagą bagażu. Test zakłada że w kolejce do kontroli bagażowej stoi jeden pasażer z prawidłowa wagą bagażu. Kontrola przebiega pomyślnie.
  - TEST_2: Kontrola pasażera z nadwagą bagażu podręcznego. Test zakłada że w kolejce do kontroli bagażowej znajduję się jeden pasażer z nadwagą bagażową. Pasażer zostaje odrzucony.



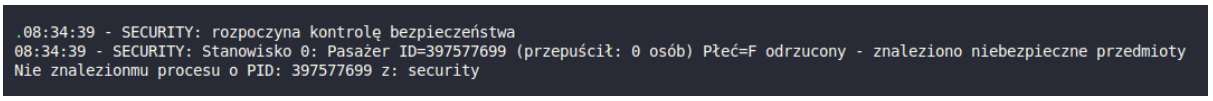
Na końcu widnieje informacje o braku procesu pasażera do usunięcia, wynika to ze środowiska, testy weryfikują poprawność algorytmiczną procesów bez faktycznego używania ich.

- ./tests/test_securityControl.py - testy kontroli bezpieczeństwa

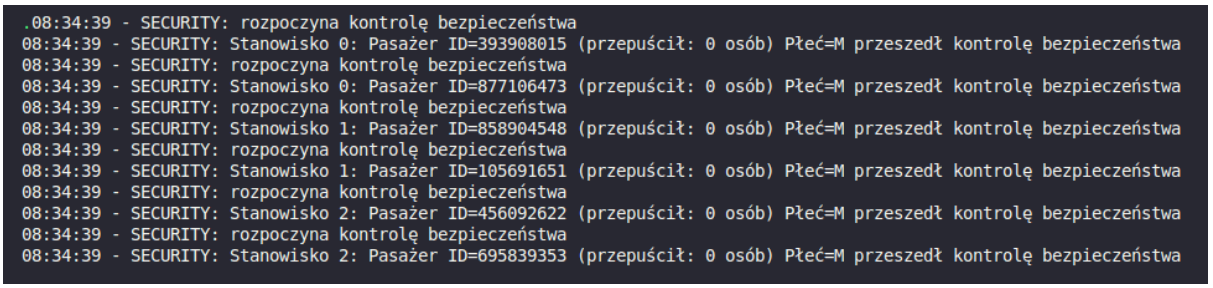
  - TEST_1: Kontrola pojedynczego pasażera bez niebezpiecznych przedmiotów. W kolejce do kontroli oczekuje jeden pasażer bez niebezpiecznych przedmiotów. Kontrola przebiega pomyślnie



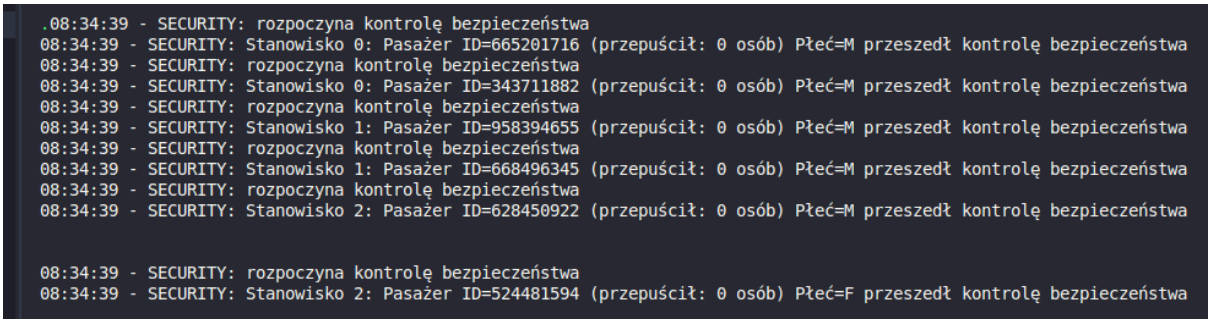
  - TEST_2: Kontrola pasażera z niebezpiecznym przedmiotem. W kolejce do kontroli oczekuje jeden pasażer z niebezpiecznymi przedmiotami. Zostaje odrzucony.



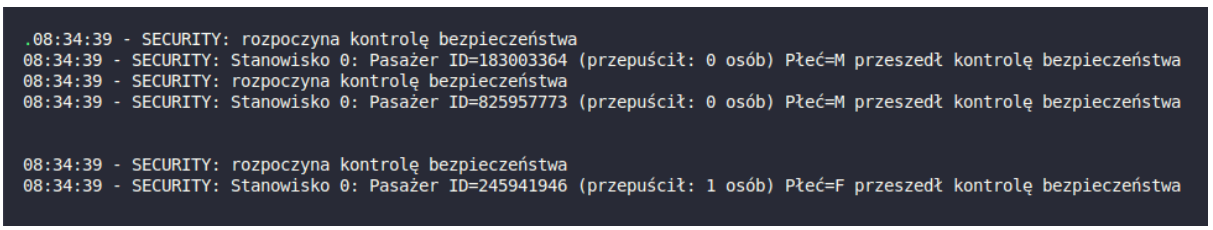
  - TEST_3: Kontrola wielu pasażerów. Do każdego punktu kontroli bezpieczeństwa oczekuję po 2 mężczyzn bez niebezpiecznych przedmiotów. Są oni równocześnie kontrolowani po 2 osoby na każdym stanowisku. Przechodzą kontrole pomyślnie



  - TEST_4: Weryfikacja zasady zachowania płci na stanowisku. Do punktu 1 i 2 w kolejce oczekuję 2 mężczyzn, do punktu 3 oczekuje 1 mężczyzna i 1 kobieta w podanej kolejności. Początkowo kontrolowana jest para mężczyzn na stanowisku 1 i 2, oraz jeden mężczyzna na stanowisku 3. Po zakończeniu kontroli, kontrolowana jest oczekująca kobieta.



  - TEST_5: Weryfikacja przypuszczeń pasażerów w kolejce do pojedynczego punktu kontroli bezpieczeństwa ustawieni są w kolejności: M,F,M. Do kontroli zostaje przepuszczony 3 z kolei mężczyzna. Kontrolowana są pary w kolejności M,M oraz później F.



- TEST_6: Weryfikacja obsługi VIP. Do pojedynczego punktu kontroli bezpieczeństwa ustawieni są M,M,F(pasażer VIP). Kontrola przebiega według schematu F (pasażer VIP pojedyńczo), i poźniej MM.

```
.08:45:22 - SECURITY: rozpoczyna kontrolę bezpieczeństwa
08:45:22 - SECURITY: Stanowisko 0: Pasażer ID=731992592 (przepuścił: 0 osób) Płeć=F przeszedł kontrolę bezpieczeństwa


08:45:22 - SECURITY: rozpoczyna kontrolę bezpieczeństwa
08:45:22 - SECURITY: Stanowisko 0: Pasażer ID=419252521 (przepuścił: 1 osób) Płeć=M przeszedł kontrolę bezpieczeństwa
08:45:22 - SECURITY: rozpoczyna kontrolę bezpieczeństwa
08:45:22 - SECURITY: Stanowisko 0: Pasażer ID=772106013 (przepuścił: 0 osób) Płeć=M przeszedł kontrolę bezpieczeństwa
```

- ./tests/test_gate.py - testy systemu bramek
  - TEST_1: Sprawdzenie odlotu samolotu przy idealnych warunkach. W hali odlotów oczekuje idealna liczba pasażerów do miejsc w samolocie. Pasażerowie nie przekraczając limitu pasażerów na schodach wsiadają do samolotu

```
test_gate.py 08:50:41 - GATE: Otrzymano sygnał o gotowości samolotu
08:50:41 - GATE: wchodzi 5 (Waga bagaży: 50.15kg) pasażerów na schody
08:50:43 - GATE: wchodzi 5 (Waga bagaży: 23.94kg) pasażerów na schody
```

  - TEST_2: Sprawdzenie odlotu samolotu przy braku pasażerów do zapełnienia samolotu. W hali odlotów oczekuje mniej pasażerów niż miejsc w samolocie. Pasażerowie wsiadają do samolotu przy wykorzystaniu schodów maksymalizując liczbę osób przy każdym użyciu schodów

```
.08:50:45 - GATE: Otrzymano sygnał o gotowości samolotu
08:50:45 - GATE: wchodzi 5 (Waga bagaży: 28.59kg) pasażerów na schody
08:50:47 - GATE: wchodzi 4 (Waga bagaży: 28.77kg) pasażerów na schody
```

  - TEST_3: Sprawdzenie odlotu samolotu przy nadmiarze pasażerów oczekujących na odlot. W hali odlotów oczekuje więcej pasażerów niż jest wolnych miejsc w samolocie Pasażerowie wsiadają do samolotu przy wykorzystaniu schodów nie przekraczając limitu miejsc w samolocie

```
.08:50:49 - GATE: Otrzymano sygnał o gotowości samolotu
08:50:49 - GATE: wchodzi 5 (Waga bagaży: 41.19kg) pasażerów na schody
08:50:51 - GATE: wchodzi 5 (Waga bagaży: 32.86kg) pasażerów na schody
.
```

5.2 Zakres testów

Testy jednostkowe

Każdy moduł ma dedykowany zestaw testów sprawdzających:

- Poprawność walidacji danych
- Obsługę skrajnych przypadków

6. Elementy wyróżniające projekt

- Zaawansowany system logowania i statystyk:
- Moduł stats.py zbierający kompleksowe statystyki
- System logowania z timestampami
- Wielopoziomowa walidacja.

7. Wnioski

Projekt spełnia wszystkie wymagane funkcjonalności i implementuje zaawansowane mechanizmy synchronizacji i komunikacji między procesami. Szczególnie warte uwagi są:

1. Kompleksowa obsługa błędów i walidacja danych
2. Efektywna synchronizacja procesów
3. Modułowa struktura kodu
4. Rozbudowany system logowania i statystyk

Podczas implementacji szczególnie wymagające okazały się:

- Koordynacja wielu procesów
- Zapewnienie spójności danych przy równoległym dostępie

8. Włączenie symulacji

8.1 Wymagania

- Python 3.x

8.2 Uruchomienie

```
git clone git@github.com:Sniezka1927/airport-threads.git
cd airport-threads/src && python3 main.py
```

8.3 Zebranie statystyk

Pliki ze statystykami zostaną zapisane w katalogu
`./stats/stats_{SIMULATE_START_TIMESTAMP}.json`

```
# Z katalogu src
python3 stats.py
```

8.4 Testowanie aplikacji

Przed każdym uruchomieniem każdego testu należy się upewnić, że wszystkie pliki w `./data/*.txt` oraz `./tests/tmp*` nie zawierają informacji o pasażerach.

```
# Z katalogu tests
pytest test_generator.py -s
pytest test_luggageControl.py -s
pytest test_securityControl.py -s
pytest test_gate.py -s
```