Dokument projektowy

## **Spis treści**

[1. Ogólny opis projektu 2](#_Toc74653384)

[2. Spis członków 2](#_Toc74653385)

[3. Instrukcja obsługi 2](#_Toc74653386)

[4. Przebieg gry 2](#_Toc74653387)

[5. Struktura sieci 3](#_Toc74653388)

[6. Protokół – wiadomości 3](#_Toc74653389)

[7. Protokół – kody wiadomości zwrotnych 4](#_Toc74653390)

[8. Komunikacja serwer - klient (diagram) 5](#_Toc74653391)

[9. Komunikacja host - klient (diagram) 5](#_Toc74653392)

[10. Schemat blokowy aplikacji 6](#_Toc74653393)

[11. Pliki aplikacji 7](#_Toc74653394)

# Ogólny opis projektu

Projekt ma na celu utworzenie aplikacji sieciowej, która pozwoli użytkownikom na wspólną grę w Państwa-Miasta. Aplikacja jest tworzona w języku Python, a do obsługi komunikacji sieciowej zostanie użyty moduł - *socket*.

# Spis członków

* Przemysław Sałek

[https://github.com/PituchaAleksander/country-city\_game/commits?author=PrzemyslawSalek](https://github.com/PituchaAleksander/country-city_game/commits?author=PituchaAleksander)

* Aleksander Pitucha

<https://github.com/PituchaAleksander/country-city_game/commits?author=PituchaAleksander>

* Szymon Sala

[https://github.com/PituchaAleksander/country-city\_game/commits?author=szymix1999](https://github.com/PituchaAleksander/country-city_game/commits?author=PituchaAleksander)

# Instrukcja obsługi

1. Uruchomienie serwera

python server.py <server\_host> <server\_port>

1. Uruchomienie serwera

python app.py <server\_host> <server\_port>

1. Następnie należy postępować zgodnie z poleceniami wyświetlanymi w konsoli i interfejsie graficznym klienta.

# Przebieg gry

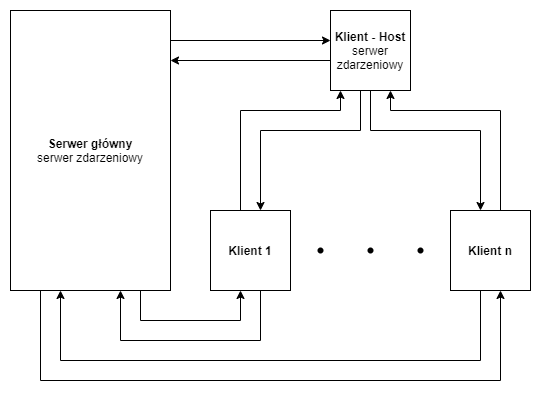
1. Wylosowanie litery przez system gry.
2. Od momentu wylosowania litery wszyscy gracze zaczynają wpisywać słowa zaczynające się na wylosowaną literę – po jednym słowie pasującym do danej kategorii. Wpisywane wyrazy nie powinny być wyrażeniami zbyt ogólnymi. Między innymi błędem jest użycie słowa dinozaur w kategorii zwierzęta. Wpisywanie słów kończy się, gdy skończy się czas przeznaczony na rundę.
3. Zliczanie punktów – wszyscy gracze otrzymują po jednym punkcie za każde poprawne słowo z danej kategorii.
4. Gra toczy się tak długo, jak tylko gracze mają na to ochotę.

# Struktura sieci

Serwer główny – jest to serwer zdarzeniowy, który przechowuje informacje o utworzonych pokojach. Klient-host wysyła mu informację o utworzeniu pokoju. Zwykły klient po wysłaniu poprawnego hasła pokoju dostaje dane hosta, dzięki którym może się z nim połączyć.

Klient-host – jest to zwykły użytkownik, który utworzył u siebie pokój gry. Jest zarządcą gry, czyli m.in. decyduje kiedy zaczyna się runda. Inni gracze mogą dołączyć do utworzonego przez niego pokoju i grać razem z nim.

Klient – jest to zwykły użytkownik, który dołączyć do pokoju innego gracza.



# Protokół – akcje (wiadomości)

Serwer do klienta

|  |  |
| --- | --- |
| CREATED password | Informacja od serwera o utworzeniu pokoju wraz z hasłem. |
| OK | Potwierdzenie przez serwer rozpoczęcia gry. |
| ERROR | Błąd. Nie można rozpocząć gry. |
| EXISTS host\_address | Informacja o istnieniu pokoju wraz z adresem hosta. |
| NOT\_EXISTS | Błąd. Taki pokój nie istnieje. |

Klient do serwera

|  |  |
| --- | --- |
| CREATE\_ROOM | Żądanie utworzenia pokoju. |
| GAME\_START password | Informacja o rozpoczęciu gry w danym pokoju. |
| JOIN password | Żądanie uzyskania informacji o hoście pokoju. |

Host do klienta

|  |  |
| --- | --- |
| OK nick | Informacja zwrotna na dołączenie gracza do pokoju wraz z nazwą hosta. |
| NEW\_PLAYER nick | Informacja o dołączeniu nowego gracza. |
| ROUND\_START letter time | Informacja o rozpoczęciu rundy wraz z literą i czasem o której ma się skończyć. |
| END\_ROUND | Informacja o końcu rundy. |
| RESULTS results | Wiadomość z wynikami wszystkich graczy w pokoju. |
| END\_GAME | Informacja o końcu gry. Zamknięto pokój. |

Klient do hosta

|  |  |
| --- | --- |
| CONNECT nick | Żądanie dołączenia do pokoju wraz ze swoją nazwą. |
| ANSWERS answers | Wiadomość z własnymi odpowiedziami. |

# Protokół – kody wiadomości zwrotnych

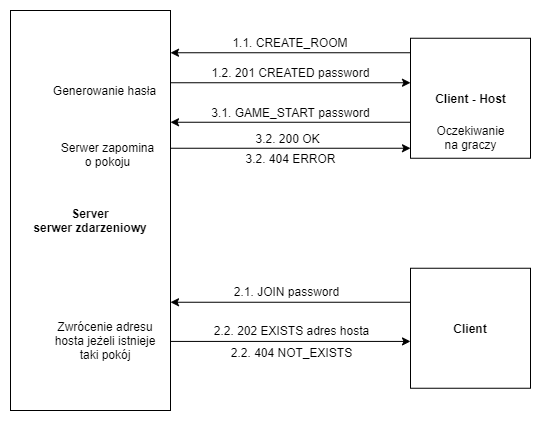
Akceptacja

|  |  |
| --- | --- |
| 200 | OK |
| 201 | CREATED |
| 202 | EXISTS |

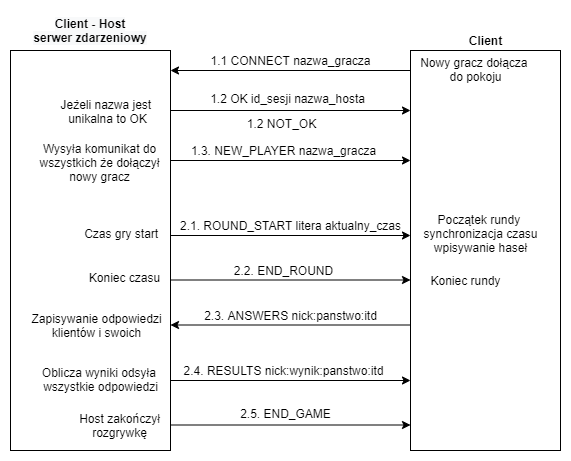
Błędy

|  |  |
| --- | --- |
| 404 | NOT\_EXISTS |

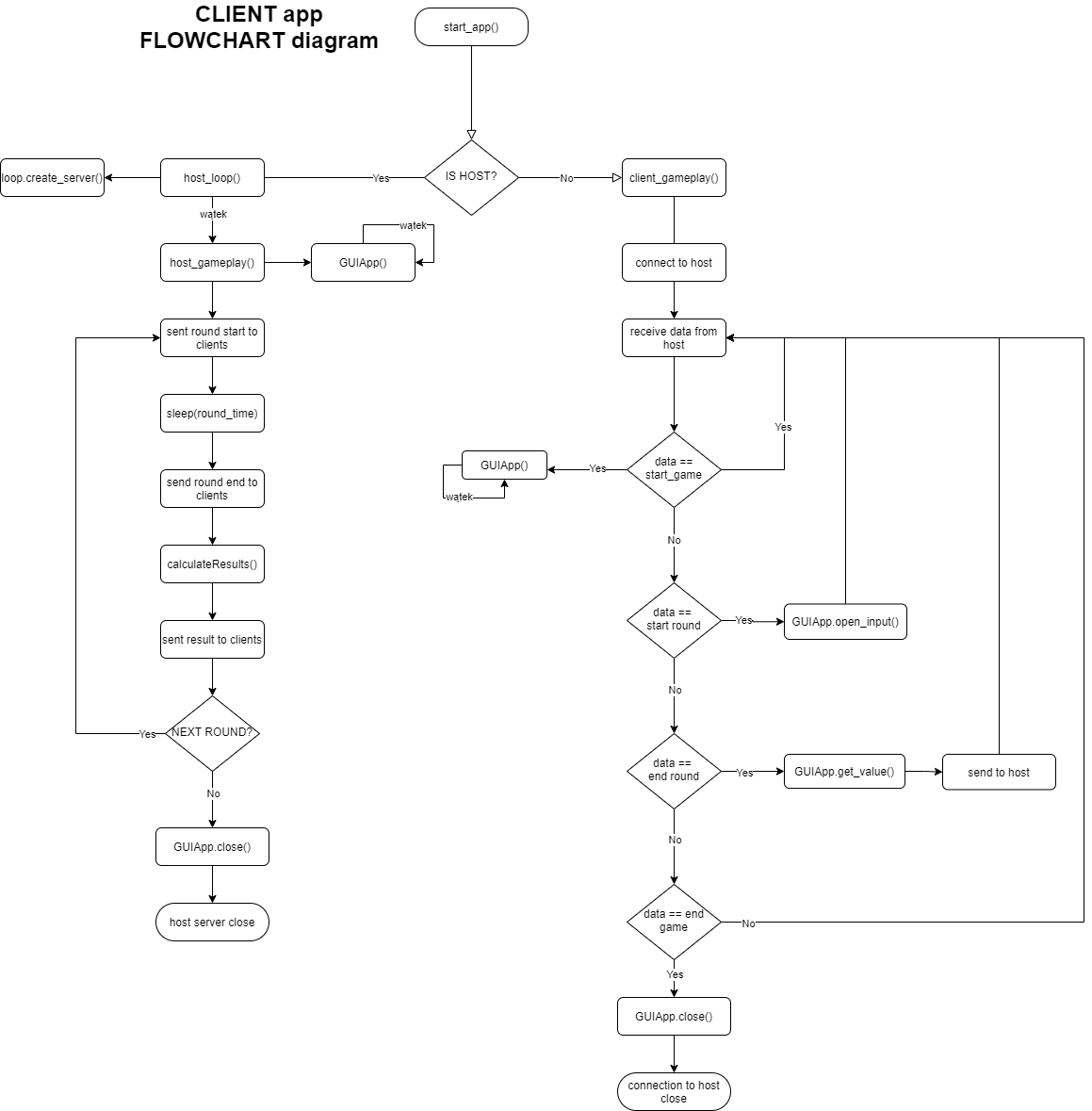
# Komunikacja serwer - klient (diagram)



# Komunikacja host - klient (diagram)



# Schemat blokowy aplikacji



Link do lepszej jakości: <https://github.com/PituchaAleksander/country-city_game/blob/main/docs/Schemat%20blokowy%20aplikacji.png>

# Pliki aplikacji

Source - <https://github.com/PituchaAleksander/country-city_game/tree/main/source>

server.py – zawiera klasę *CountryCityServerProtocol* oraz funkcje i metody odpowiedzialne za działanie serwera zdarzeniowego aplikacji, który przechowuje pokoje i zarządza informacjami o nich.

app.py – plik startowy który posiada funkcje odpowiadające za start programu klienta – gracza.

client.py – plik z funkcjami odpowiedzialnymi za zarządzanie połączeniem klienta z hostem oraz prowadzeniem gry po stronie zwykłego gracza.

host.py – plik w którym mamy serwer zdarzeniowy hosta oraz funkcję odpowiedzialną za prowadzenie gry po stronie hosta i komunikację ze wszystkimi klientami w pokoju.

GUI.py – plik zawierający klasę interfejsu graficznego działającego na własnym wątku zbudowanego z pomocą tkinter. Zawiera wszystkie metody budujące interfejs i zarządzające nim.

gameData.py – plik z klasą *GameData* odpowiedzialną za logikę gry.

categories.py – plik z klasą *Categories* przechowującą informacje o kategoriach i obliczającej wyniki.

Logs - <https://github.com/PituchaAleksander/country-city_game/tree/main/logs>

server\_logs – plik przechowujący logi z lokalnego serwera. Zostaje utworzony wraz z pierwszym uruchomieniem serwera.