Spis treści

[Memorsy! 1](#_Toc11977352)

[Główne założenia protokołu 1](#_Toc11977353)

[Połącznie, rozłączenie 1](#_Toc11977354)

[Sposób obsługi klientów przez serwer 2](#_Toc11977355)

[Przepływ danych w aplikacji 2](#_Toc11977356)

[Rodzaje pakietów 2](#_Toc11977357)

[Generowanie mapy 2](#_Toc11977358)

[Przykładowa rozgrywka 2](#_Toc11977359)

[Logowanie 2](#_Toc11977360)

# Memorsy!

Program jest realizacją gry w „memories”. Gra polega na znalezieniu na planszy dwóch pasujących do siebie kart.

Aplikacja umożliwia grę na dwóch poziomach (6 ,12 kart w talii).

# Główne założenia protokołu

Protokół realizowany jest z użyciem szyfrowanego połączenia.

Wszystkie komunikaty są kończone znakiem końca linii – '\r\n'.

Tablice są formatowane za pomocą biblioteki pickle[[1]](#footnote-1).

Strony komunikacji porozumiewają się za pomocą kodów : mapowane z pliku 'codes.py',

# poza wyjątkiem gdy klient chce opuścić rozgrywkę, nie wysyła kodu dla "QuitGame" tylko komunikat tekstowy

# Takie rozwiązanie eliminuje konflikt z zapytaniem o treść karty, która przysłana jest poprzez zapytanie zależne od pozycji karty.

# Połącznie, rozłączenie

Komunikacja jest realizowana z użyciem szyfrowanego połączenia. Po połączeniu, klient pobiera i sprawdza ważność certyfikatu serwera. Jeśli jest ważny, nawiązywane jest połączenie szyfrowane, realizowane przy użyciu modułu ssl.[[2]](#footnote-2)

Jeśli klient chce rozłączyć się z serwerem, wysyłamy kod komunikatu ‘QuitGame’. (Poza wyjątkiem opisanym w założeniach).

Po zakończeniu, zerwaniu połączenia z serwerem klient jest usuwany z listy.

W przypadku błędu serwera (zerwania połączenia), aplikacja klienta zamyka się.

# Sposób obsługi klientów przez serwer

* Każdy klient jest obsługiwany w niezależnym wątku.
* Dla każdego klienta generowana jest mapa zależna od poziomu, klient generuje pustą mapę.
* Mapa przechowywana jest tylko po stronie serwera, klient wysyła zapytania o zawartość kart.
* Klient wysyła tablice z pozycjami kart, z zapytaniem czy do siebie pasują.
* Gdy skończą się karty tj. klient odgadnie wszystkie możliwe pary, serwer wysyła komunikat informujący o wygranej.
* Po skończeniu gry, klient ma możliwość zaczęcia kolejnej rozgrywki.

# Przepływ danych w aplikacji

## Rodzaje pakietów

Protokół wykorzystuje trzy różne sposoby przekazywania informacji.

* Informacja o wybranym poziomie rozgrywki jest przekazywana jako wartość typu integer.
* Zapytanie o kartę, należy przekazać string zawierający pozycję karty.
* Sprawdzenie dwóch kart – tablica id: [id, id], przekształcona za pomocą modułu pickle[[3]](#footnote-3).
* Wszystkie komunikaty przekazywane są za pomocą kodów stanu (plik ‘codes.py’).

## Generowanie mapy

* Mapa jest generowana z losowo wybranych kart z pliku ‘/store/cards.txt’.
* Karty losowane są przy pomocy modułu random[[4]](#footnote-4).

## Przykładowa rozgrywka

Klient – C, Serwer – S.

* + K wysyła wybrany poziom
  + S odpowiada czy przygotował grę, jeśli tak obie strony generują mapy
  + K wysyła zapytania o zawartość kart
  + S odpowiada tekstem.
  + K wysyła parę kart [id, id]
  + S odpowiada czy karty do siebie pasują.
  + Gdy wszystkie karty zostały odkryte – S wysyła komunikat „GameOver”
  + Klient kończy rozgrywkę, lub rozpoczyna nową ( wysyła - „QuitGame” /„NewGame”).
  + NewGame -> pkt 1 / QuitGame -> zamknięcie połączenia z C

# Logowanie

Wszystkie czynności podejmowane w aplikacji, błędy, ostrzeżenia logowane są do plików.

* server.log - serwer
* client.log - klient

1. https://docs.python.org/2/library/pickle.html [↑](#footnote-ref-1)
2. https://docs.python.org/2/library/ssl.html [↑](#footnote-ref-2)
3. https://docs.pyton.org/2/library/pickle.html [↑](#footnote-ref-3)
4. https://docs.python.org/2/library/random.html [↑](#footnote-ref-4)