INSTALACJA:  
  
Aplikacja została napisana dla Pythona w wersji 3.6 lub wyższej. Aby uruchomić serwer napisany w języku Python należy doinstalować (za pomocą polecenia pip install <nazwa\_modułu>) następujące moduły:

- Flask

- pygame

OPIS KOMUNIKACJI MIĘDZY FRONTENDEM A BACKENDEM:

Przebieg gry:  
1. Gracz odpala przeglądarkę, wyświetla mu się okno z prośbą o podanie nazwy użytkownika. Po wpisaniu i kliknięciu „Start” przeglądarka wysyła na serwer zapytanie **setConnection(username)**. W odpowiedzi dostaje potwierdzenie poprawnego połączenia (prawdopodobnie może to być pusta wiadomość lub treść błędu w przypadku errora).

2. Przeglądarka wyświetla graczowi okno oczekiwania na przeciwnika i wysyła na serwer zapytanie **getOponent()**. Informacja zwrotna powinna zawierać nazwę użytkownika przeciwnika.

3. Gra rozpoczyna się. Wysyłama zapytanie **playerReady()**, żeby zsynchronizować klienta obu graczy. Kiedy dostanę zwrotkę ‘ok’ startuje timer (np. 10 sekund na ruch) i oczekujemy na wybranie figury przez użytkownika. W momencie wybrania figury przeglądarka wysyła na serwer dane funkcją **chooseFigure(figura)**, gdzie figura to string o jednej z wymienionych wartości: ‘paper’, ‘rock’, ‘scissors’, ‘spock’, ‘lizard’. Funkcja ta oczekuje w odpowiedzi takiej samej informacji dotyczącej przeciwnika. Jeżeli przeciwnik nie zgłosi do serwera odpowiedzi w ciągu 10 sekund to serwer wysyła pusty string. Dodatkowo funkcja powinna zwracać drugi parametr „won” – 1 jeżeli gracz wygrał, 0 jeżeli przegrał (lub true/false). Na podstawie tej informacji zmienia wartość wyniku. Możemy też przesyłać z serwera cały wynik, czyli cała funkcja zwróci odpowiedź (‘figuraPrzeciwnika’, ’twójWynik’, ’wynikPrzeciwnika’).

4. Po otrzymaniu odpowiedzi przeglądarka wyświetla wyniki przez 3 sekundy i wysyła do serwera funkcję **playerReady()**. Jeżeli dostanie pustą wiadomość zwrotną to oznacza, że drugi gracz też jest gotowy na kolejną rundę (zapobiegnie to różnicom w czasie odliczania kiedy jeden z graczy ma laga) lub wiadomość ‘fail’ (czy cokolwiek podobnego) informującą, że przeciwnik się rozłączył. Po otrzymaniu takiej wiadomości przeglądarka wyświetla okno informujące o rozłączeniu przeciwnika przez 3 sekundy, a następnie oczekuje na przeciwnika (czyli wysyła **getOponent()**) i cały cykl się powtarza.

Kiedy gracze się rozłączają to zerujemy wyniki i „usuwamy” grę. Nowa rozpoczyna się od 0 i serwer musi wysłać graczom nazwy użytkownika przeciwnika.

Wszystko też zależy w jaki sposób będziemy się komunikować. Nie wiem jeszcze na co pozwala xml-rtc, ale gdyby udało nam się skomunikować poprzez zapytania XHR metodą POST to działałoby to tak jak opisałem (klient po wysłaniu wiadomości oczekiwałby na zwrotkę z serwera i dopiero po jej otrzymaniu podejmował kroki).

Funkcje przesyłane pomiędzy frontendem a backendem:

|  |
| --- |
| Schemat:  funkcja (parametry wysyłane na backend) {  zwrotka na frontend } |

setConnection(myUsername) {

status: ‘ok’ lub ‘fail’  
}

getOponent() {

status: ‘ok’ lub ‘fail’,

username: ‘nick przeciwnika’  
}

playerReady() {

status: ‘ok’ jeśli przeciwnik jest gotowy lub ‘fail’, gdy błąd

}

chooseFigure(figure) {

status: ‘ok’ lub ‘fail’,

myScore: liczba(int),

oponentScore: liczba(int),

oponentFigure: ‘paper’ lub ‘rock’ lub ‘scissors’ lub ‘spock’ lub ‘lizard’  
}