

# EMonopoly

Mateusz Przybyła

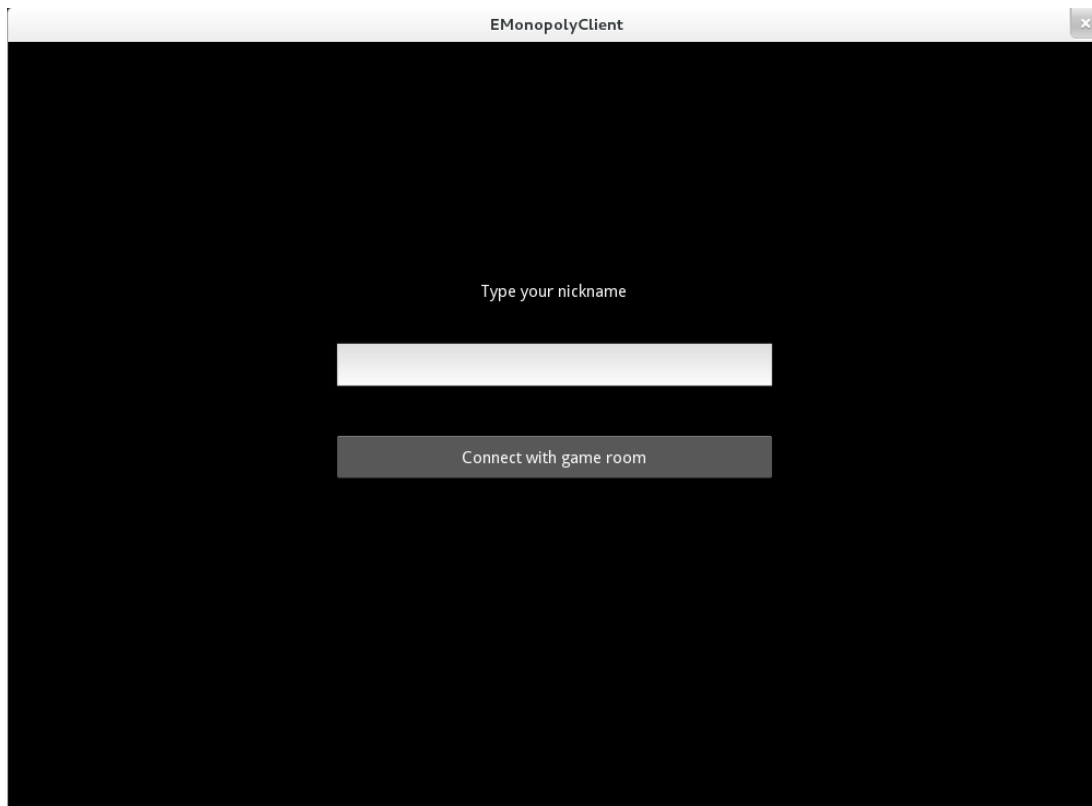
Przedmiotem projektu jest implementacja elektronicznej wersji znanej i popularnej gry planszowej Monopoly. EMonopoly będzie umożliwiać grę w Monopoly przez sieć z ustaloną liczbą graczy. Gracze mogą komunikować się ze sobą poprzez chat (dostępny w dwóch wersjach: pokojowej i serwerowej). Przed rozpoczęciem gry gracz musi połączyć się z serwerem nadzorującym grę i dołączyć do istniejącego pokoju, bądź założyć nowy.

Projekt składa się z dwóch części:

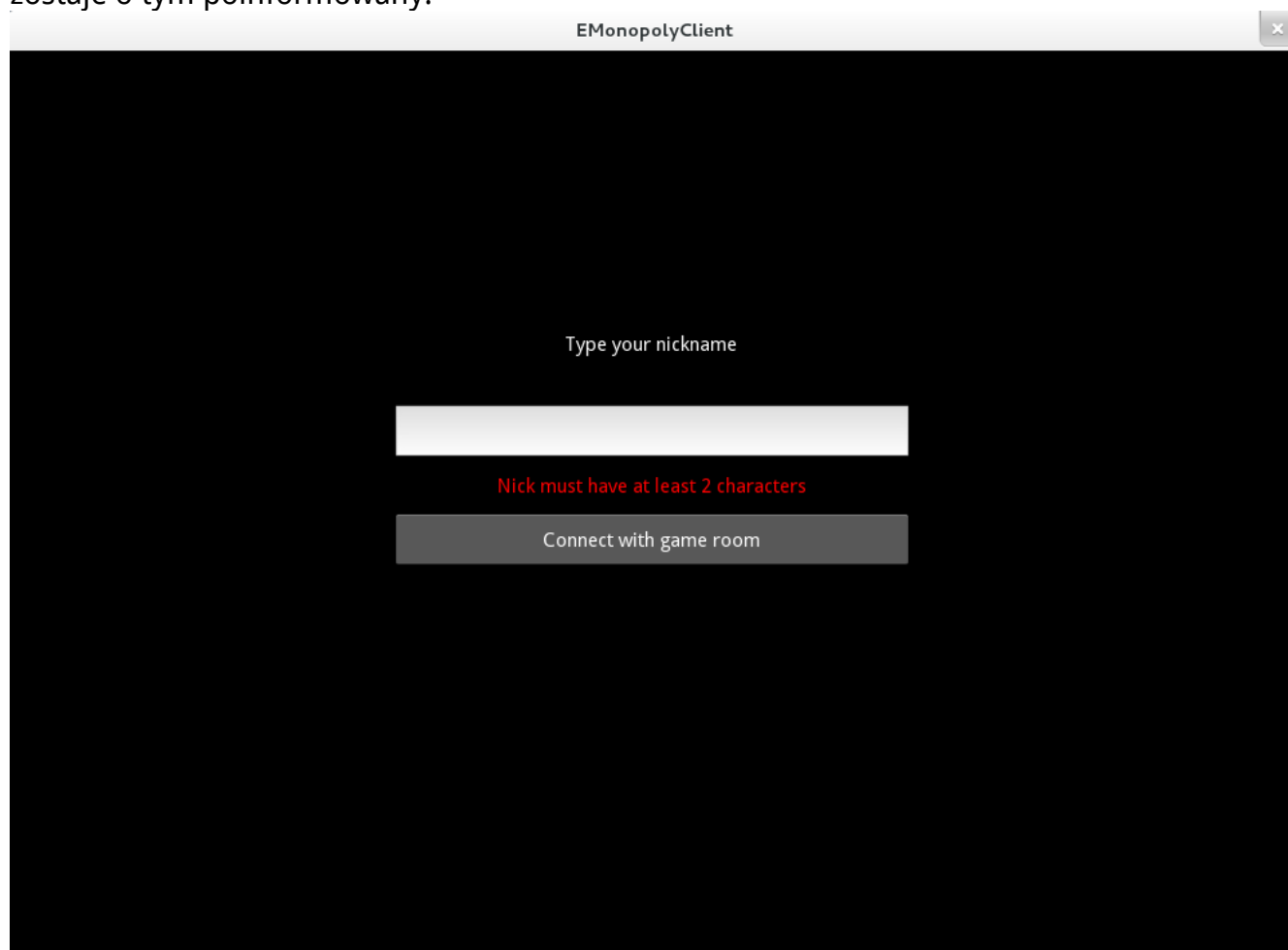
- serwera, do którego łączą się gracze i który zarządza wszystkimi grami, które rozgrywają podłączeni do niego gracze
- aplikacji klienckiej potrafiącej komunikować się z serwerem wg ustalonego protokołu i prezentującej przebieg gry w formie graficznej

## 1. Opis aplikacji klienckiej

Po uruchomieniu aplikacji klienckiej, następuje ustalenie połączenia z serwerem. Aby przejść dalej użytkownik musi wpisać nick, pod którym będzie identyfikowany na serwerze.



Nick musi być unikatowy w kontekście graczy przebywających obecnie na serwerze i zawierać co najmniej dwa znaki. Jeśli nie spełnia on powyższych warunków użytkownik zostaje o tym poinformowany:



The screenshot shows a window titled "EMonopolyClient" with a dark background. In the center, the text "Type your nickname" is displayed above a white rectangular input field. Below the input field, a red error message reads "Nick must have at least 2 characters". At the bottom of the central area, there is a grey button labeled "Connect with game room".

Po pomyślnym wprowadzeniu nicku, użytkownik zostaje przekierowany na główny pokój serwera, na którym może:

- komunikować się z innymi graczami poprzez chat
- przeglądać listę utworzonych pokoi
- utworzyć swój pokój ze swoją konfiguracją
- odświeżać listę pokoi
- opuścić pokój i ponownie wprowadzić nick



Ponieważ na początku na serwerze nie ma utworzonego żadnego pokoju założmy, że użytkownik tworzy swój pokój. Jego konfiguracja wymaga podania nazwy (unikatowej wśród obecnie utworzonych pokoi), wyboru liczby graczy (od 2 do 4) oraz ewentualnego oznaczenia pokoju jako prywatnego i podania hasła, które będzie wymagane przez graczy przy dołączeniu do pokoju.

The screenshot shows the EMonopolyClient application window. At the top, it says "Logged in as: Mateusz" and "Log out". Below this, there is a chat area with two messages: "Mateusz: witam wszystkich graczy!" and "Mateusz: hej hej hej". The main part of the window is a "Create new room" dialog box. It has four input fields: "Nazwa" (Name) with the text "Przykładowy Pokój", "Liczba graczy" (Number of players) with the value "2", "Gra prywatna" (Private game) with an unchecked checkbox, and "Hasło" (Password) with an empty text field. At the bottom of the dialog are "Submit" and "Cancel" buttons. Below the dialog, there are three buttons: "Create a room", "Refresh rooms", and "Send".

EMonopolyClient

Logged in as: Mateusz Log out

Mateusz: witam wszystkich graczy!  
Mateusz: hej hej hej

Create new room

Nazwa Liczba graczy

Przykładowy Pokój 2

Gra prywatna Hasło

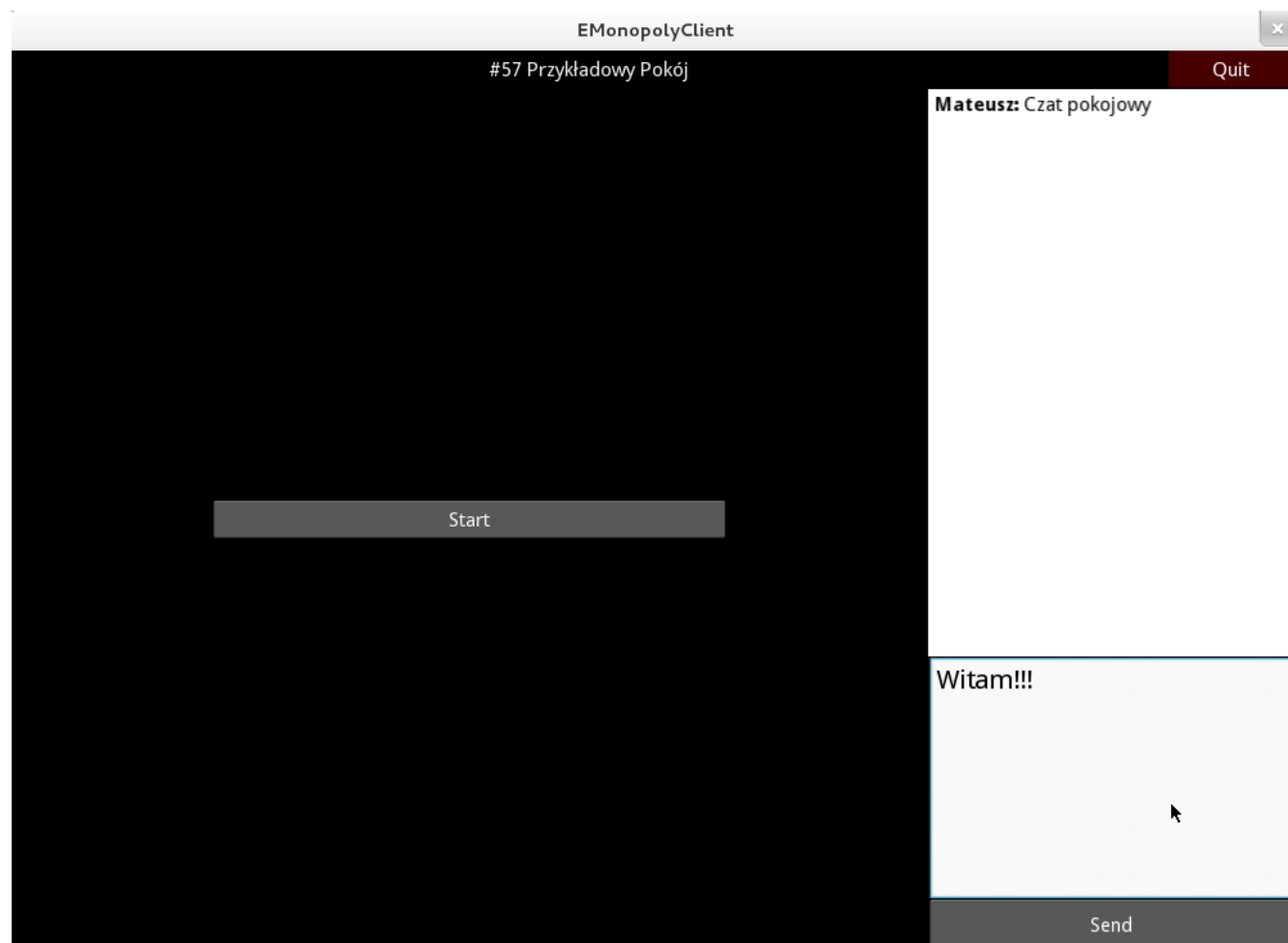
☐

Submit Cancel

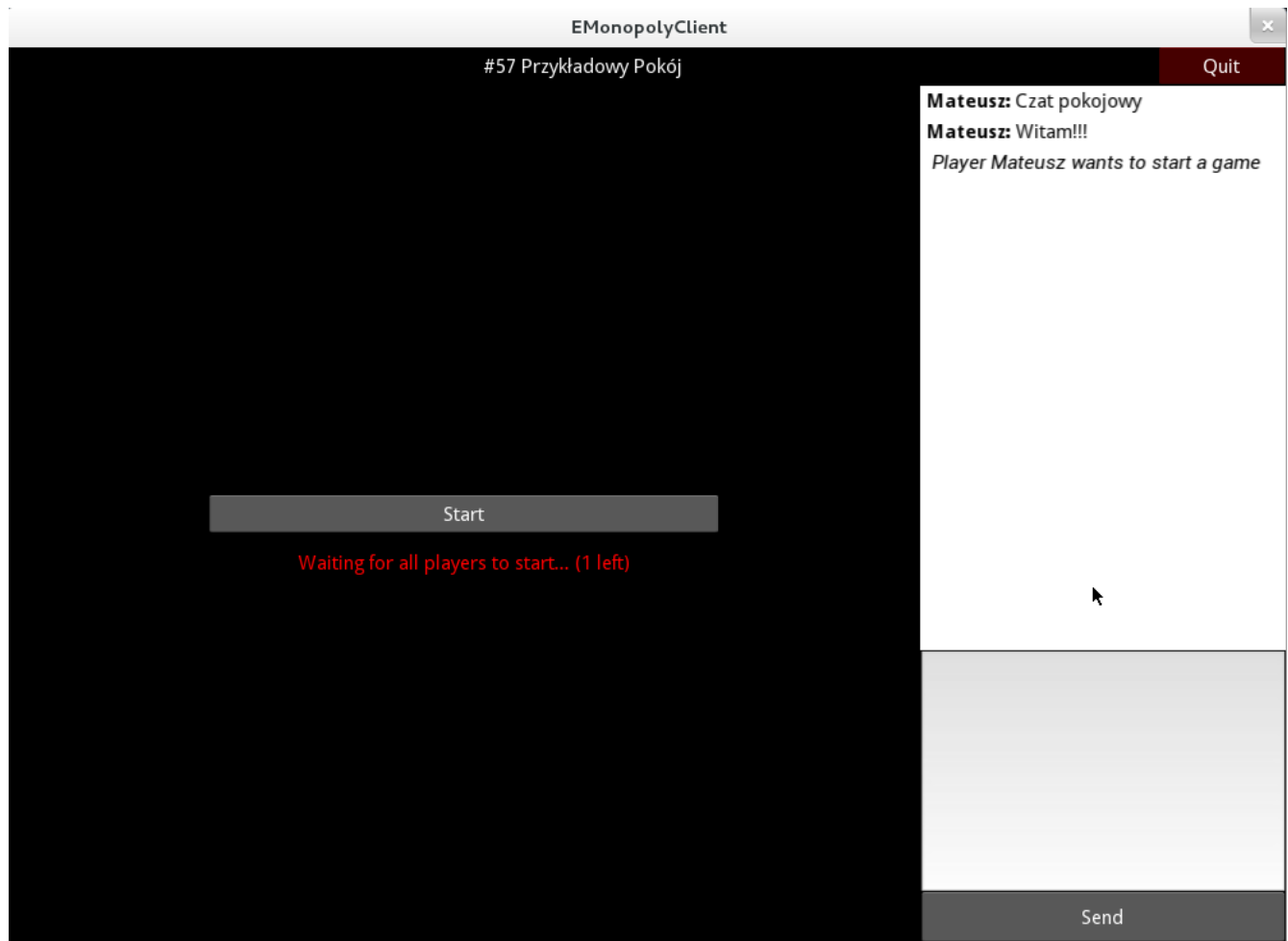
Create a room Refresh rooms Send

Po poprawnym utworzeniu pokoju, gracz (właściciel) zostaje do niego przeniesiony. W każdym pokoju znajduje się czat, który umożliwia komunikację wyłącznie z graczami wewnątrz pokoju.

Przed rozpoczęciem gry, od wszystkich graczy wymagane jest wciśnięcie przycisku start.



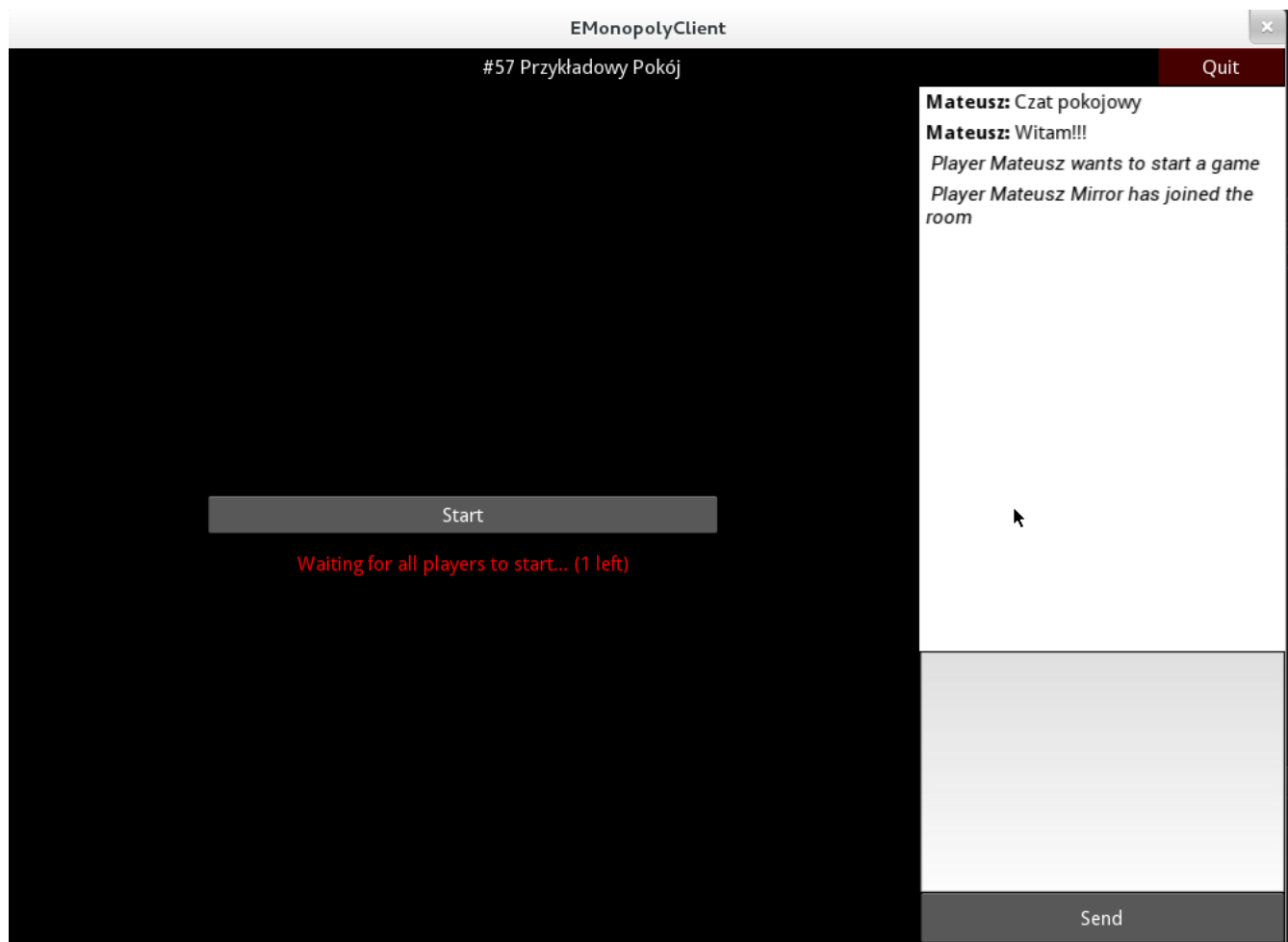
Ponieważ utworzony pokój jest 2 osobowy, poza właścicielem pokoju, ktoś jeszcze musi do niego dołączyć i wcisnąć przycisk start.



Przyjmijmy, że do serwera podłączył się drugi gracz. W głównym pokoju serwera po odświeżeniu listy pokoi zobaczy on pokój stworzony przez gracza pierwszego. Może on do niego dołączyć klikając przycisk **Join**.

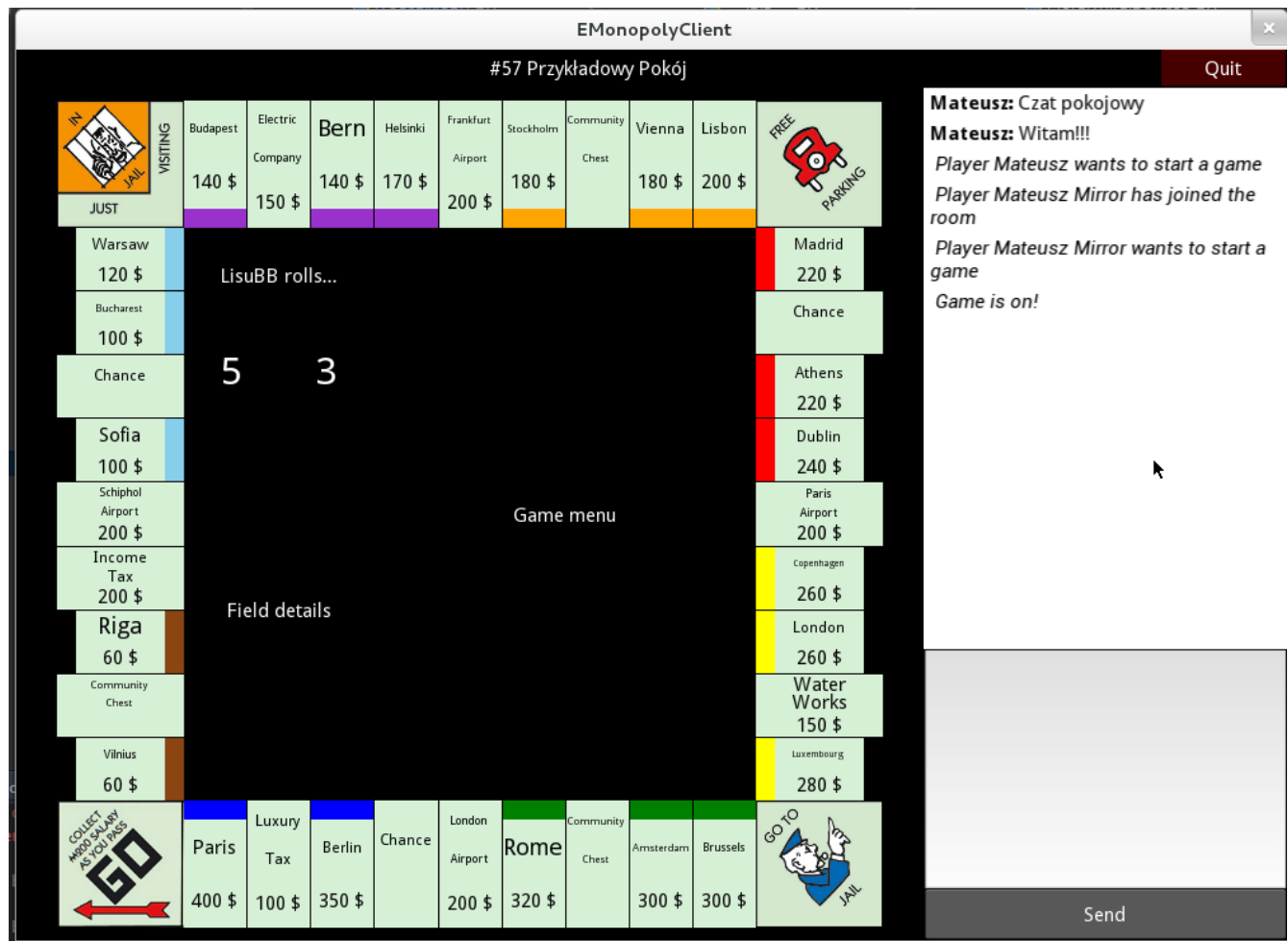


Po dołączeniu do pokoju właściciel widzi, że ktoś się przyłączył w oknie chatu.





Gdy drugi gracz wciśnie przycisk **Start** (wyrażając tym samym chęć rozpoczęcia gry), gracze mogą rozpocząć grę w Monopoly.



Obszar gry składa się z:

- planszy, która odzwierciedla wygląd gry planszowej, prezentuje ona status pól (do kogo należą, czy postawiono domki lub hotele) oraz pozycje graczy
- wyniku rzutu kostką przez graczy
- szczegóły dotyczące pola pojawiające się w sekcji **Field details** po wybraniu konkretnego pola, w tym polu można obejrzeć np. kto jest właścicielem pola, oraz stawki związane z danym polem, można tutaj też znaleźć dodatkowe informacje na temat pola
- menu gry, gdzie gracz podejmuje decyzje związane z grą, to tutaj decyduje się rzucić kostką, kupić pole, postawić dom/hotel, czy podejmować akcje związane z uniknięciem bankructwa
- dodatkowe pole, którego nie ma na obrazku to pole zawierające opis przebiegu gry, będą tutaj widoczne akcje wykonywane przez graczy oraz „przepływ gotówki”
- lista graczy będących w grze wraz z ich stanem konta i zaznaczeniem do kogo obecnie należy ruch (także niewidoczne na obrazku)

## 2. Serwer i architektura projektu

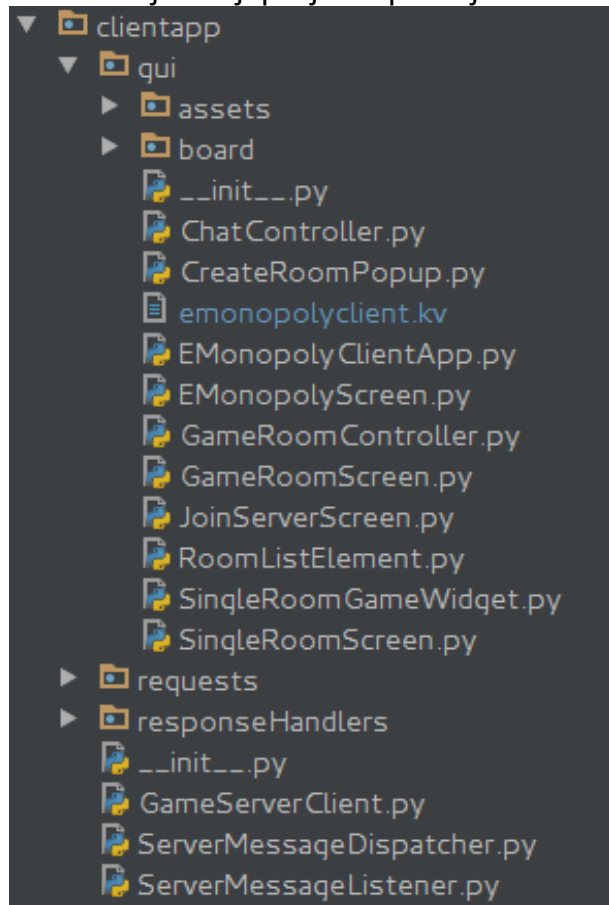
Sercem projektu jest aplikacja serwerowa. Zarządza ona połączeniami z klientami obsługując każdego niezależnie w innym wątku. Komunikacja z serwerem odbywa się wg ustalonego protokołu. Każda wiadomość składa się z dwóch części:

- pierwsze 4 bajty to liczba naturalna, określająca ile bajtów zawiera faktyczna wiadomość, która jest drugą częścią
- wiadomość w formacie JSON

To serwer przeprowadza walidację żądań od klienta oraz konkretnych pól. Kontroluje on też przebieg gry Monopoly w taki sposób, iż każda akcja klienta jest wpierw kierowana do serwera, który sprawdza czy jest ona poprawna (i dozwolona w danym momencie). Po poprawnej walidacji aktualizuje on stan gry oraz informuje o tym wszystkich graczy. W zależności od charakteru akcji gracze mogą zauważyć zmianę na planszy, lub w oknie statusu gry.

Interfejs graficzny aplikacji klienckiej został stworzony przy użyciu frameworku **Kivy**.

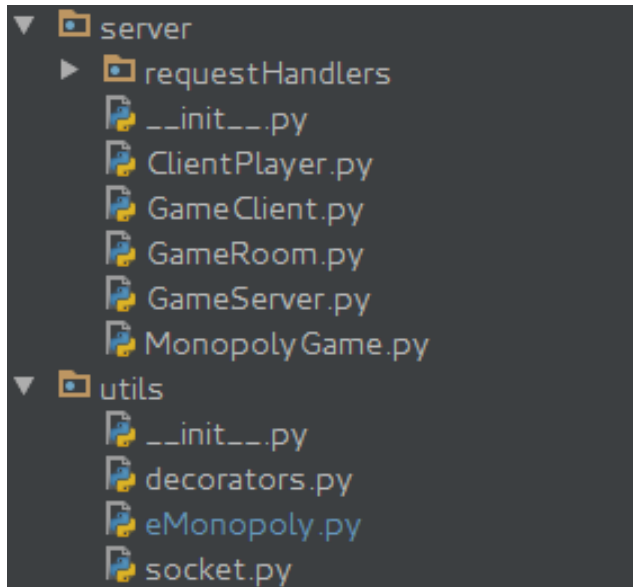
W obecnej wersji projekt aplikacji klienckiej zarysowuje się następująco:



- pakiet GUI zawiera klasy odpowiedzialne za zarządzanie interfejsem graficzny, ważnym plikiem jest *emonopolyclient.kv* zawierający hierarchiczną deklarację poszczególnych komponentów niemal wszystkich widgetów składających się na całą aplikację, pakiet assets zawiera obrazki, a pakiet board zawiera klasy odpowiedzialne za prezentowanie gry planszowej i jej kontroli w konkretnym pokoju
- pakiet requests zawiera obiektową reprezentację żądań jakie klient wysyła do serwera (przykładowe żądania to: ChatMsg, CreateRoomRequest, JoinServerRequest lub GetRoomsRequest), klasy te zawierają wszystkie potrzebne rzeczy do serializacji i przekształcenia ich do formatu JSON
- pakiet responseHandlers zawiera klasy odpowiedzialne za obsłużenie wiadomości od serwera, zarówno te w formie żądanie-

odpowiedź (jako wynik wysłanego żądania), jak i te w formie „push” (np. wiadomość z chatu lub ruch innego gracza podczas gry)

Serwer z kolei prezentuje się następująco:



- pakiet requestHandlers zawiera klasy odpowiedzialne za obsłużenie żądania otrzymanego przez klienta, wynik przetwarzania żądania może zostać odesłany do klienta, do wszystkich graczy pokoju, wszystkich graczy serwera lub może nie być wysłany do nikogo
  - klasa ClientPlayer zawiera informacje na temat podłączonego klienta, który ustalił już swoją tożsamość (nick)
- 
- klasa GameClient zajmuje się obsługą jednego użytkownika podłączonego do serwera
  - klasa GameRoom zawiera informacje na temat pokoju utworzonego przez gracza
  - klasa GameServer to główna klasa serwera, która zajmuje się ustaleniem połączenia z klientem i zdelegowania jego obsługi do GameClient, ma ona również informacje o stworzonych pokojach, podłączonych graczach i ich przypisaniu do poszczególnych pokoi, zajmuje się ona posprzątaniem zasobów związanych z graczem, gdy ten się rozłączy
  - klasa MonopolyGame zawiera model gry po stronie serwera i informacje o aktualnym stanie gry (stan konta, pozycje na planszy, do kogo należy ruch, hipoteki itp.)
  - moduły z pakietu utils zawierają użyte dekoratory, konfigurację rozgrywki (zbiór i rodzaj pól) oraz pomocnicze metody do komunikacji sieciowej wg ustalonego w projekcie protokołu (4 bajty rozmiaru + wiadomość)