Instituto de Computação UFF
Departamento de Ciência da Computação

PROJETO DOMINÓ MANIA

Grupo

Marcelo Vieitas da Fonseca Filho
213031127

Eduardo Zimelewicz
213031109

Universidade Federal Fluminense - UFF

Niterói – Rio de Janeiro
2017

Linguagem e recursos utilizados

Para o desenvolvimento do jogo foi utilizada a linguagem Python. Bibliotecas utilizadas foram nativas do Python, são elas, system, random e socket.

Regras de funcionamento dos protocolos

Com relação ao protocolo TCP implementado:

Na troca de mensagens, os buffers foram setados de forma a não receber mais do que a quantidade de bytes necessária, o que evita o vazamento de informação para futuras trocas de mensagem. Por exemplo, na linha de código 47 no arquivo domino.py (clientes[acks].recv(3)) o servidor espera receber de um dos clientes uma mensagem de tamanho igual a 3 bytes.

Portanto, se existe um .recv no lado servidor necessariamente deve haver um send do lado cliente para que a aplicação possa dar continuidade.

O protocolo utilizado funciona por informação contida na mensagem, por exemplo, se a mensagem recebida é ‘c’ então se envia uma informação x, caso a mensagem recebida seja ‘t’ se envia uma informação y.

Os clientes e servidores da aplicação funcionam da seguinte forma: cada cliente se conecta assim que for executado. O servidor só é conectado qual existem exatamente 4 clientes conectados, depois de 4 clientes conectados o servidor não aceita mais conexões.

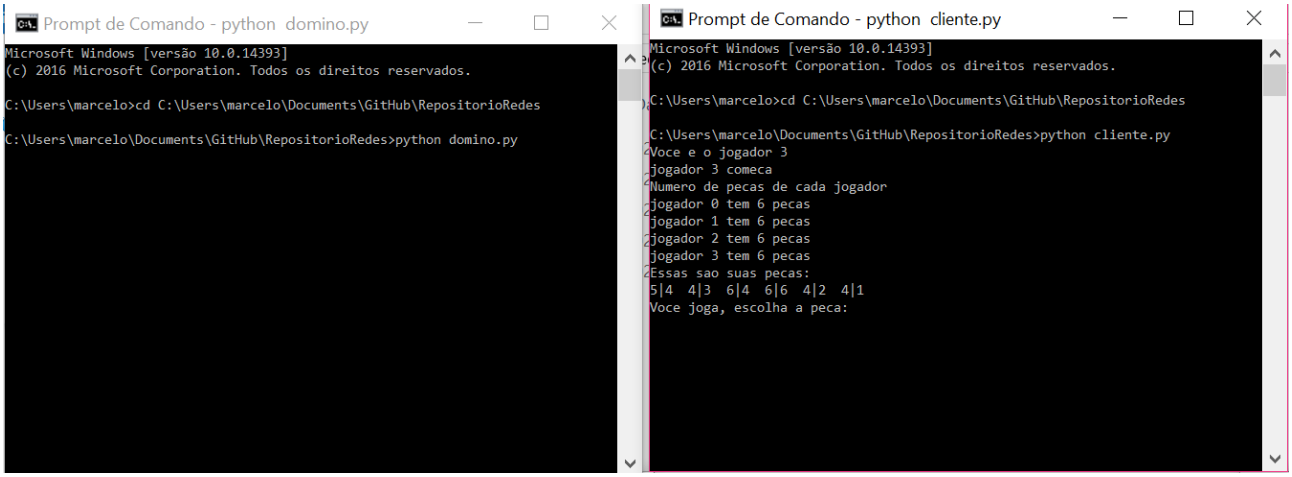
Tabela de mensagens:

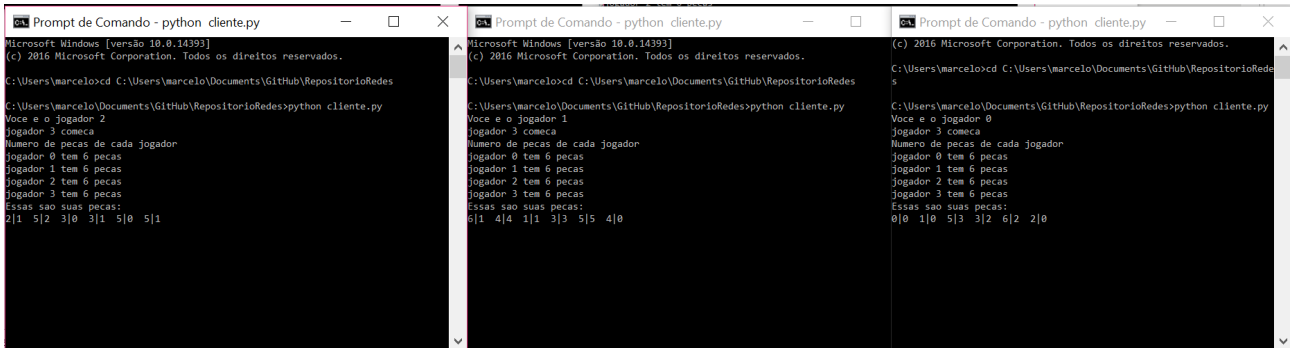
MENSAGEM	SIGNIFICADO	LADO
server_socket.send('p')	Envia requisição. Espera receber player que começa a partida. Ou Envia requisição. Espera receber mensagem “ponta”	Cliente
server_socket.send('ack')	Envia requisição. Espera receber ack de confirmação de servidor.	Cliente
server_socket.send('c')	Envia requisição. Espera receber confirmação que jogo começou.	Cliente
server_socket.send("pe")	Envia requisição. Espera receber número de peças de jogador	Cliente
server_socket.send('j')	Envia requisição. Espera receber 0 ou 1, onde 1 significa que é a vez do jogador.	Cliente
server_socket.send('e')	Envia requisição. Espera receber mensagem	Cliente

	“você joga, escolha sua peça”	
server_socket.send('m')	Envia requisição. Espera receber mensagem “modo da peça”	Cliente
server_socket.send('t')	Envia requisição. Espera receber número de peças na tabela.	Cliente
server_socket.send("tab")	Envia requisição. Espera receber tabela.	Cliente
server_socket.send('a')	Envia requisição. Espera receber pontuação de A.	Cliente
server_socket.send('b')	Envia requisição. Espera receber pontuação de B.	Cliente
clientes[x].send(MESSAGE)	Envia mensagem requisitada pelo cliente x	Servidor
clientes[x].recv(n)	Espera mensagem de tamanho n vinda de cliente x	Servidor

Dificuldades

Tivemos como dificuldades a implementação de uma interface mais aprimorada. Foi visto como uma prioridade a implementação de protocolo ao invés da interface, então foi preferido continuar com a interface apresentada no próprio cmd mas com uma funcionalidade boa do protocolo implementado.





Como interpretar a interface e responder de maneira correta as perguntas que darão continuidade ao jogo estão no arquivo Como jogar.txt presente no projeto. O placar do jogo aparece no domino.py executado.

Além da interface tivemos dificuldades em relação ao controle de mensagens, em qual tempo de execução cada mensagem deveria chegar para que a implementação fosse realizada de maneira correta.

Conclusão

Logo, obtivemos uma aplicação que atende a requisitos de uma aplicação padrão de cliente-servidor, onde somente o cliente envia requisições e o servidor as responde. Não foi realizada a implementação do UDP, devida a quantidade de tempo gasta na implementação do TCP e na tentativa de implementação de uma interface mais amigável.