Documentazione delle classi

Classi di logica

XeviousVS

Classe applicazione. Istanzia l'interfaccia, carica le impostazioni e istanzia di conseguenza gli oggetti di modello che costituiscono l'applicazione oltre ad inizializzare le componenti per il log e le interazioni con il database.

Gestisce gli eventi quali la pressione dei pulsanti, l'utilizzo del campo username, la pressione dei tasti di gioco e la modifica dell'interfaccia per l'utente, in collaborazione con le altre classi.

ServerLogXml

Classe main del server di log. Riceve dai client di gioco gli eventi per il log in formato xml, li valida e li aggiunge al file di log cumulativo a comune per tutti i client.

RicercatoreAvversario

Classe dedicata ad implementare i protocolli di ricerca di un avversario per iniziare una nuova partita o recuperarne una interrotta. Genera la connessione con il client dell'avversario e gli oggetti TrasmettitoreComandi e RicevitoreComandi per la comunicazione.

Espone un metodo per chiudere la connessione, da utilizzare in caso di terminazione o interruzione della partita. In caso di terminazione, introduce un ritardo per assicurare la sincronizzazione degli eventi fra i due client.

Utilizza OperazioniDatabase per accedere al database, e RicevitoreConnessioneAvversario per porsi in attesa della connessione dell'altro utente.

Essendo utilizzata in concorrenza dal thread di interfaccia e dal thread di ascolto per la connessione, i suoi metodi sono eseguiti in mutua esclusione.

RicevitoreConnessioneAvversario

Thread dedicato all'ascolto sulla porta indicata nelle impostazioni, per attendere che l'avversario stabilisca la connessione.

Una volta stabilita la connessione comunica all'oggetto RicercatoreAvversario associato il Socket da utilizzare e termina.

L'attesa può essere interrotta con il metodo interrupt(), tuttavia a causa della concorrenza con il client remoto l'interruzione non ha efficacia garantita: si dà precedenza alla richiesta del client remoto.

Il metodo ottientiStato() dell'oggetto indica, alla terminazione del thread, se è stata stabilita la connessione o meno.

TrasmettititoreComandi

Classe designata all'inoltro via socket dei comandi all'utente remoto.

Viene creato da XeviousVS e da questi riceve i comandi da inoltrare.

Il Socket non viene creato né distrutto da questa classe, deve invece essere passato già connesso e il metodo inoltraComando() non va utilizzato dopo che il Socket è stato chiuso.

RicevitoreComandi

Thread designato alla ricezione via socket dei comandi dall'utente remoto.

Viene avviato e terminato da XeviousVS, mentre i comandi ricevuti vengono inoltrati a ModelloGioco tramite il metodo ModelloGioco.eseguiComandoRemoto().

Il Socket non viene creato né distrutto da questa classe, deve invece essere passato come argomento già connesso.

Quando la connessione viene chiusa, controlla che la partita sia terminata correttamente ed in caso contrario avvia le procedure di recupero partita. Dopodiché, in ogni caso, il thread termina.

ModelloGioco

Classe che costituisce il modello del gioco, riceve i comandi da eseguire da XeviousVS (del giocatore locali) e RicevitoreComandi (dall'avversario) e li esegue. Mantiene lo stato della partita in termini di vite e stato di attività, mentre gli elementi grafici di gioco vengono acceduti tramite VistaGioco.

VistaGioco

Classe che gestisce le componenti visuali del gioco. Istanzia e posizione gli elementi grafici del gioco, fornisce metodi per accedervi o indicarne posizioni specifiche (per il recupero di una partita interrotta).

Fornisce metodi per avviare e sospendere le animazioni, e quindi controllare il proseguire del gioco.

PannelloViteUtente

Classe che gestisce gli elementi grafici che costituiscono il pannello delle vite di un utente.

Istanzia e posiziona gli elementi grafici che costituiscono il pannello, ed espone un metodo impostaViteUtente per modificare il numero di vite mostrate.

Navicella

Elemento grafico che rappresenta una navicella guidata da un utente.

Espone metodi per eseguire i comandi elaborati da ModelloGioco, e per sospendere/avviare le animazioni su richiesta di VistaGioco.

Il metodo sparaProiettile() controlla che prontaASparare sia vero, e dopo aver sparato l'imposta a falso. Utilizza poi le classi Timer e TimerTask per schedulare la reimpostazione a vero di prontaASparare, implementando così il periodo di attesa tra un proiettile e l'altro. I metodi per far fuoco e impostare prontaASparare richiedono la mutua esclusione.

Proiettile

Elemento grafico che rappresenta un proiettile sparato dalla navicella guidata da un utente.

Viene creato dall'esecuzione del comando Fuoco da parte di una Navicella.

Contiene la logica di rilevazione ed esecuzione degli impatti, provvedendo a rimuovere i proiettili coinvolti e modificare i conteggi delle vite in caso sia coinvolta una navicella.

RilevatoreSpostamento

Classe che realizza ChangeListener, è utilizzata all'interno della classe proiettile per rilevarne lo spostamento e quindi verificare la presenza di impatti.

Classi statiche

OperazioniCacheLocale

Classe statica che contiene metodi per salvare e recuperare la cache locale.

Accede alle classi applicative per salvare e ripristinare lo stato della partita.

OperazioniDatabase

Classe statica contenente i metodi per accedere al database, in particolare per le operazioni di lettura e aggiornamento delle statistiche e registrare, cancellare e ottenere le informazioni di contatto per creare o recuperare le partite.

LoggerEventoXML

Classe statica che contiene i metodi per creare un oggetto LogEventoXML a partire dalla descrizione dell'evento, serializzarlo e inviarlo via socket al server di log.

Valida il messaggio XML tramite XSD prima dell'invio, inviando solo i messaggi che superano la validazione.

LettoreImpostazioniXml

Classe statica che contiene i metodi per leggere il file di impostazione xml, validarne il contenuto e ottenere l'oggetto ImpostazioniXml da utilizzare nell'applicazione. Nel caso la validazione del contenuto fallisce, vengono restituite impostazioni di default mentre il file xml viene sovrascritto con queste.

Strutture dati

StatisticheUtente

Struttura dati dedicata a mantenere le statistiche dell'utente.

Richiede vittorie e sconfitte come argomento, mentre la percentuale delle vittorie è calcolata nel costruttore.

RegistrazioneUtenteInAscolto

Struttura dati dedicata a contenere username, indirizzo e porta di un utente in ascolto.

Viene utilizzata sia per la ricerca di una nuova partita che per il recupero di una partita interrotta.

LogEventoXML

Classe che contiene le informazioni di log da inviare via socket al server di log.

Richiede come parametro la descrizione dell'evento, mentre ricava le altre informazioni (indirizzo IP e timestamp) alla creazione.

Viene creato, serializzato e inviato via socket dalla classe LoggerEventoXML.

DatiCache, DatiCachePartita, DatiCacheElementoGioco

Strutture dati serializzabili utilizzate da OperazioniCacheLocale per salvare e caricare lo stato dell'applicazione via file binario.

ImpostazioniXML, AssociazioniTasti, IndirizzoServer

Strutture dati utilizzate per la deserializzazione delle impostazioni xml, e per serializzazione in caso di scrittura delle impostazioni default.

All’avvio XeviousVS ottiene l’oggetto ImpostazioniXML da LettoreImpostazioni, e con questo oggetto vengono configurati i vari componenti dell’applicazione.

Enumerazioni

Comando

Enumerazione che indica il comando di gioco inviato da un utente.

Viene creato da XeviousVS quando un tasto associato ad un comando viene premuto, trasmesso da TrasmettitorreComandi e ricevuto da RicevitoreComandi, interpretato ed eseguito da ModelloGioco.

Fazione

Enumerazione che si applica ad un elemento di gioco (Navicella, Proiettile, PannelloViteUtente) per indicare il giocatore a cui appartiene. Viene utilizzato anche nei metodi di accesso ai campi di ModelloGioco e VistaGioco.

I possibili valori sono Giocatore e Avversario, dove il Giocatore è l’utente del client e l’Avversario è l’utente connesso attraverso la rete.

StatoPartita

Enumerazione utilizzata da ModelloGioco per indicare lo stato della partita, in particolare riguardo l’attività e le condizioni di pausa/attesa.

Viene utilizzato per l’interpretazione dei comandi e la verifica dello stato della partita da parte di altre classi.

StatoRicevitoreConnessione

Enumerazione utilizzata per indicare lo stato del thread RicevitoreConnessioneAvversario, e in particolare, dopo la terminazione, le condizioni per cui è terminato.

È utilizzato per affrontare consistentemente i casi di corsa critica tra la richiesta di connessione dal client remoto e la richiesta di annullamento della ricerca dal client locale.

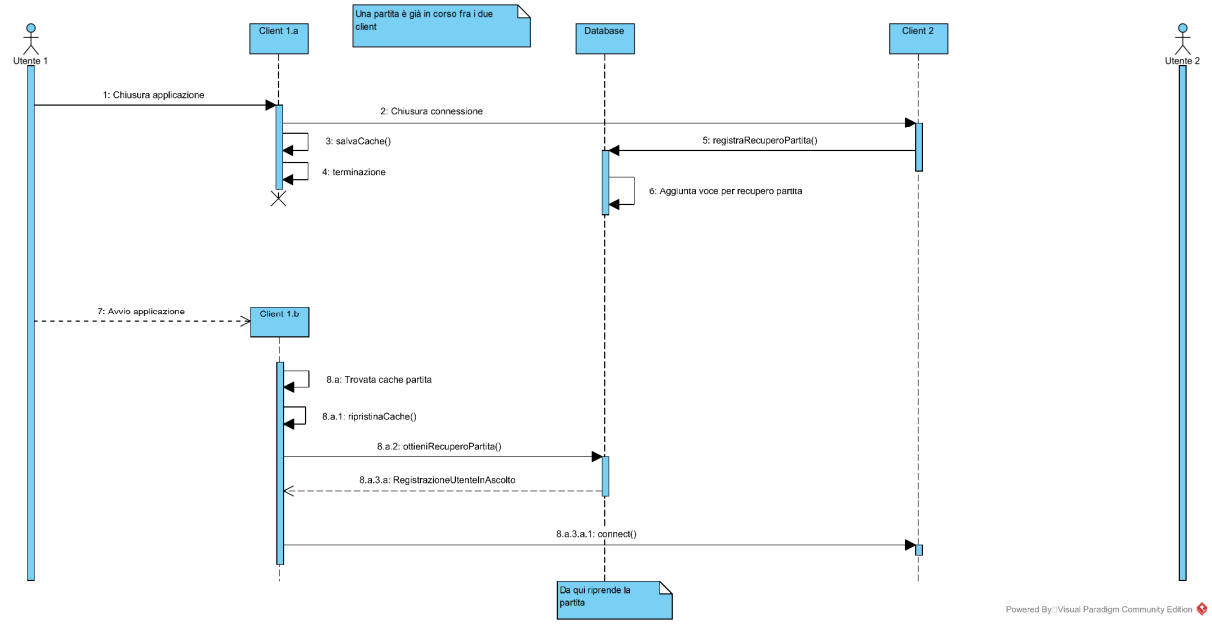
TipoRicerca

Enumerazione utilizzata per indicare il tipo di ricerca che è in corso da parte di RicercatoreAvversario.

È utilizzata dalla classe nei casi in cui si utilizza RicevitoreConnessione, per distinguere le operazioni da compiere una volta stabilita la connessione o per interrompere la ricerca in corso.

Diagrammi

Diagrammi di sequenza per il protocollo di recupero di una partita interrotta



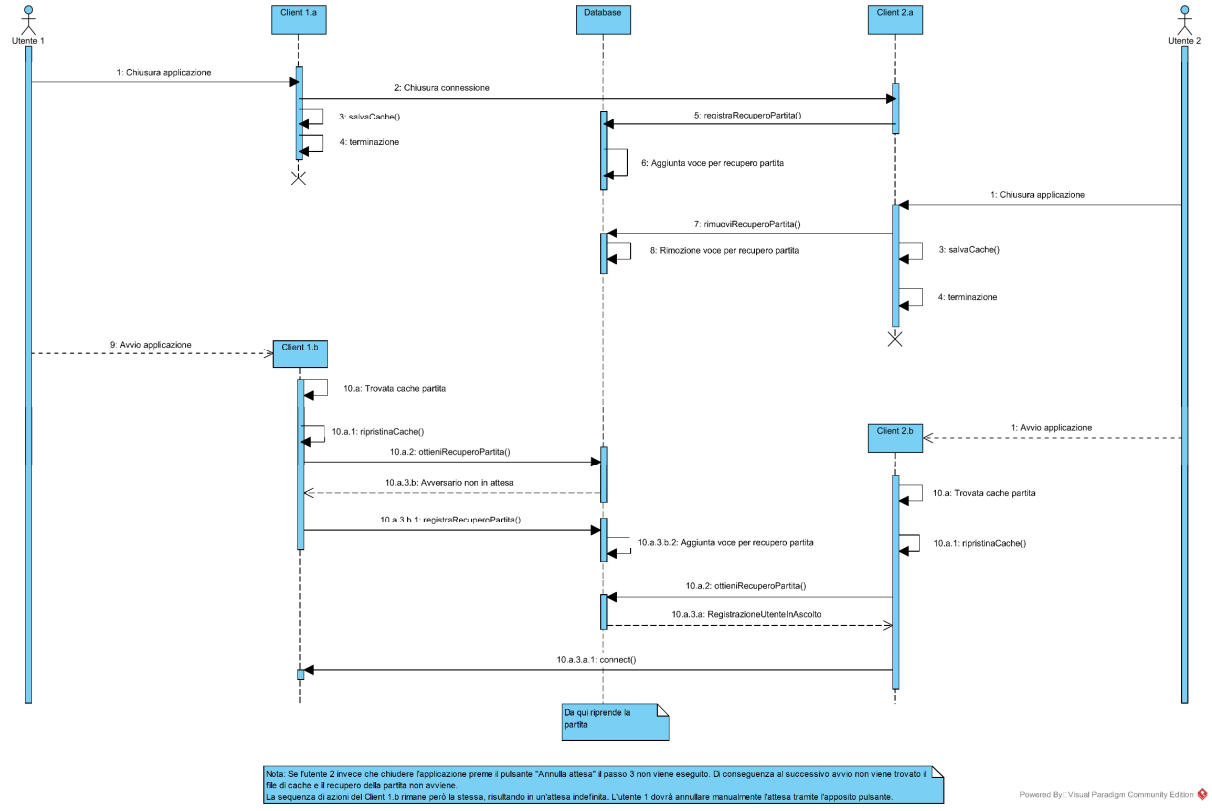


Diagramma delle classi per il file di impostazione xml

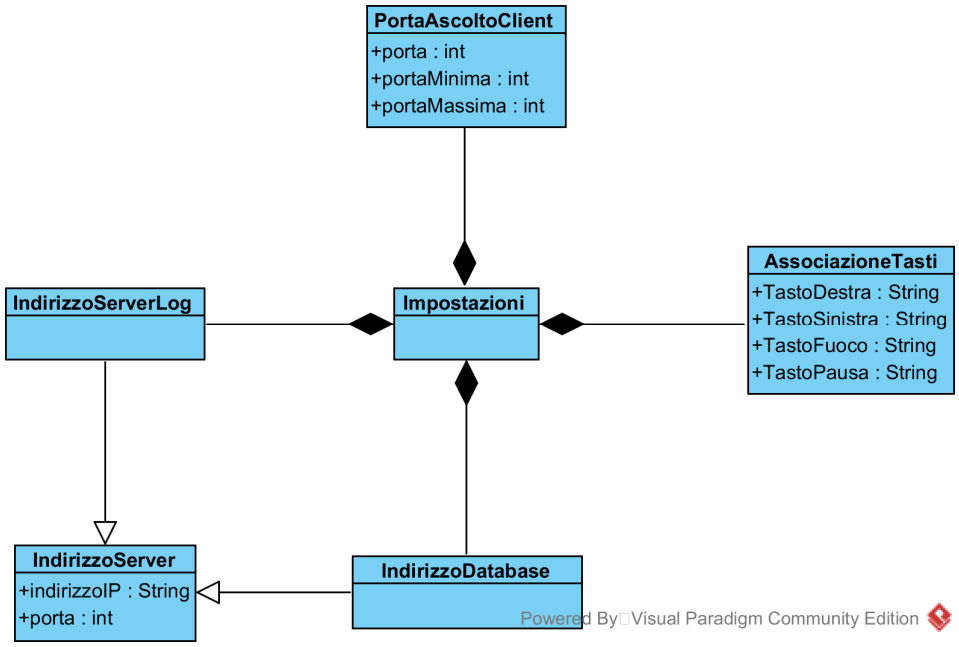


Diagramma delle classi per il file di impostazione xml

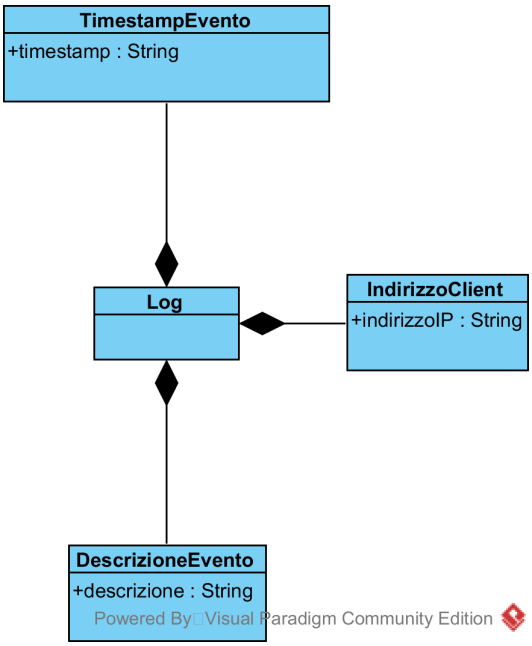


Diagramma Entità-Relazione per il database MySQL

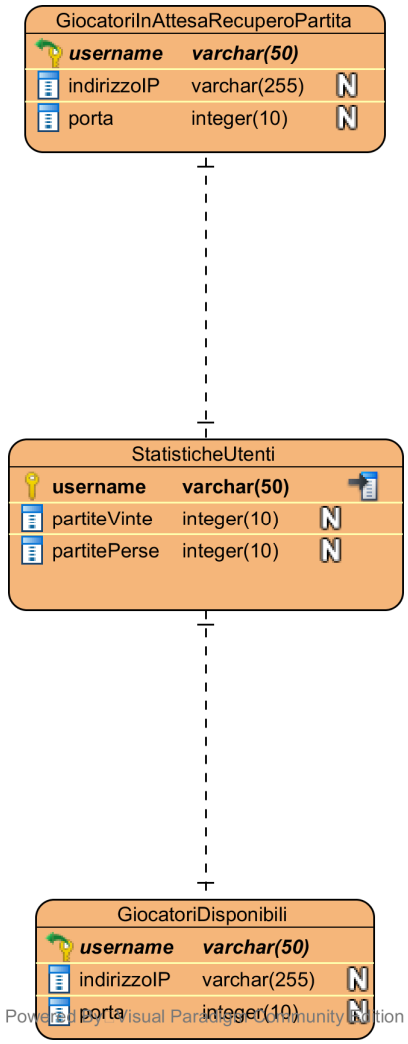


Diagramma delle classi dell'applicazione