UNOxUNO: gioco di carte distribuito
Progetto di Sistemi Distribuiti

Luca Rinaldi, Octavian Bujor

Università di Bologna

Sommario Questa relazione descrive la progettazione e l'implementazione di un videogioco distribuito basato sul famoso gioco di carte UNO. Lo sviluppo è stato orientato alla tolleranza ai guasti di tipo crash. Il progetto è stato realizzato in JAVA con l'ausilio di RMI e Slick2D.

1 Introduzione

1.1 Obiettivo

L'obiettivo di questo progetto è stato quello di realizzare un gioco distribuito a più giocatori in cui siano tollerati i guasti di tipo crash, senza l'ausilio di un server centralizzato, neanche per la creazione della partita. Il progetto scelto si basa sul gioco di carte UNO e sfrutta le potenzialità di Java e RMI per la comunicazione peer-to-peer tra i diversi giocatori.

1.2 Regole del gioco

UNO (Bib. [1]) è un gioco di carte statunitense che può essere giocato da 2 a 10 persone. Il mazzo di gioco è composto da 108 carte così distribuite:

- 19 carte di colore Rosso che vanno dall'1 al 9 (2 serie) più uno 0
- 19 carte di colore Blu che vanno dall'1 al 9 (2 serie) più uno 0
- 19 carte di colore Giallo che vanno dall'1 al 9 (2 serie) più uno 0
- 19 carte di colore Verde che vanno dall'1 al 9 (2 serie) più uno 0
- 8 carte Pesca Due (2 per colore)
- 8 carte Cambio giro (2 per colore)
- 8 carte Salta turno (2 per colore)
- 4 carte Jolly Cambio colore
- 4 carte Jolly Pesca Quattro

Il mazziere viene scelto casualmente tra i giocatori e distribuisce 7 carte a testa. La porzione di mazzo rimanente viene detta "mazzo pesca" e la prima carta viene girata, andando così a formare il cosiddetto "mazzo scarti".

Durante il suo turno, un giocatore deve scartare una carta dalle proprie a disposizione, compatibile con quella in cima al mazzo scarti per colore o per numero. Ciò significa che la carta deve avere colore o numero uguale alla prima carta sul banco: ad esempio, se la carta sul banco è un 7 rosso, il giocatore può

scartare una carta numerata di colore rosso, o volutamente un 7 di qualsiasi colore; da allora il colore può cambiare e così via. Alternativamente, il giocatore può usare una carta Azione[1].Non si può scartare più di una carta per turno.

Se il giocatore non ha a disposizione carte da scartare, ha l'obbligo di pescarne una dal mazzo pesca. La carta pescata può essere giocata nello stesso turno.

Quando un giocatore scarta una delle sue due carte rimaste così rimanendo con una sola carta in mano, deve pronunciare "UNO!". La pronuncia della parola deve avvenire prima che la penultima carta che questi gioca raggiunga il mazzo scarti. Se non si esclama "UNO!" scartando la penultima carta, si devono pescare 2 carte dal mazzo pesca. Vince il gioco chi rimane per primo senza carte in mano.

Le funzioni delle carte "Azione" sono le seguenti:

Cambio giro Inverte il senso di gioco da orario ad antiorario o viceversa.
Salta turno Fa saltare un turno al giocatore successivo nel senso del gioco.
Pesca due carte Il giocatore successivo nel senso del gioco pesca due carte e salta il turno.
Cambio colore Permette al giocatore di scegliere uno tra i quattro colori disponibili. Può essere giocata in qualsiasi momento e su qualsiasi carta.
Pesca quattro carte Il giocatore successivo nel senso del gioco pesca quattro carte e salta il turno. La carta in questione consente, inoltre, a chi l'ha giocata di cambiare il colore del gioco.

2 Aspetti progettuali

2.1 Creazione della stanza

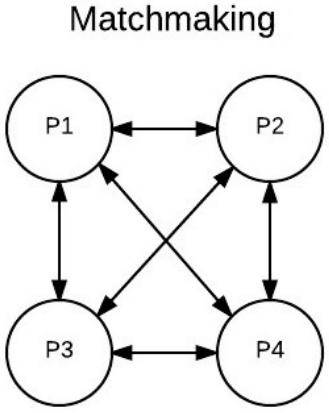


Figura 1. Struttura della rete nella fase di matchmaking

[1] Le carte azione colorate devono essere compatibili con la carta del mazzo scarti

Prima di iniziare una partita è necessario creare la stanza di gioco in cui si attende la conferma di tutti i giocatori che vogliono giocare tra di loro.

Per fare questo un giocatore crea la stanza e automaticamente diventa il primo partecipante in attesa degli altri giocatori. Ogni giocatore successivo deve inserire l'indirizzo e la porta di un giocatore già presente all'interno della stanza (non necessariamente l'indirizzo di colui che l'ha creata). In questo modo non è necessario utilizzare un server centrale per la creazione della partita ma ogni giocatore entra nella stanza connettendosi agli altri per formare la rete completamente connessa in figura 1.

Quando un giocatore entra in una stanza, chi riceve la richiesta si occupa di aggiungerlo alla lista dei giocatori e di mandare un messaggio di aggiornamento di stato a tutti i restanti componenti della rete.

La partita inizia quando tutti i giocatori nella stanza hanno premuto il tasto "Start", ovvero quando tutti sono pronti.

2.2 Durante la partita

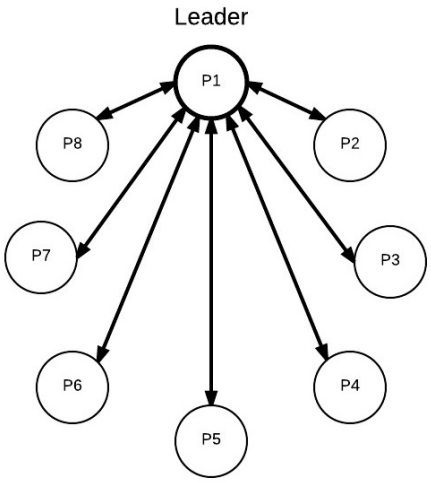


Figura 2. Struttura della rete distribuita

Una volta iniziata la partita, le connessioni sulla rete distribuita diventano come quelle mostrate in figura 2, dove il ruolo del Leader viene assegnato temporaneamente a chi ha attualmente il turno nel gioco. Il passaggio del turno, a differenza di un tipico gioco da carte, può essere effettuato in entrambi i sensi di gioco, a seconda del numero di carte "Cambio giro" giocate.

Il Leader è colui che si occupa dell'aggiornamento di stato di tutti i giocatori presenti nella partita. Lo stato del gioco include:

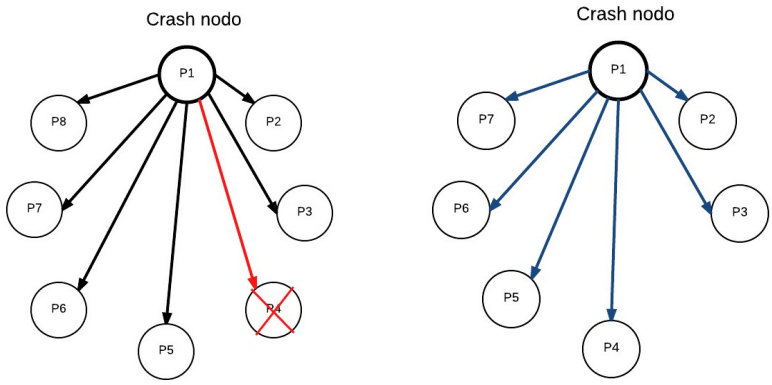
– Orologio logico (numero di modifiche);
– Il numero, la posizione, l'indirizzo e il nome dei giocatori;
– Il nome del Leader attuale, ovvero di colui che può giocare;
– Le carte presenti nel mazzo pesca e nel mazzo scarti;
– Le carte in mano a ciascun giocatore;
– Il senso di gioco;
– Le penalità per l'attuale giocatore.

Lo stato del gioco può essere cambiato solo dal Leader ed esso si occupa di mandare un messaggio con i cambiamenti apportati (pesca di una carta, scarto di una carta, passaggio di turno, etc.) a tutti i nodi della rete.

Un giocatore, durante il suo turno, può decidere se giocare una carta compatibile con l'ultima scartata oppure pescarne una dal mazzo pesca (e giocarla). Se non ha carte da scartare, può passare il turno premendo l'apposito pulsante.

Se il giocatore ha due carte in mano e sta per scartarne una, prima di giocare la carta deve premere il pulsante "UNO!" per non ricevere la penalità prevista dalle regole del gioco (2 carte). Quando un giocatore scarta una carta Jolly ("Cambio colore" o "Pesca quattro"), esso deve anche scegliere il colore desiderato. Un giocatore vince la partita quando scarta l'unica carta che ha in mano, notificando tutti gli altri nodi della rete della sua vittoria.

2.3 Rilevazione e gestione di crash



.a Il leader rileva il crash .b Sincronizzazione dello stato

Figura 3. Gestione del crash di un processo

Il crash di un nodo viene rilevato e gestito in modo diverso in base al suo ruolo attuale. Se il giocatore che fa crash non è il Leader, ai fini del gioco non è importante notificare immediatamente il suo crash agli altri giocatori, perché

esso non può apportare cambiamenti dello stato del gioco. Per questo motivo il Leader controlla la presenza degli altri utenti solo una volta ogni cambio turno.

In figura 3 viene mostrato un esempio di rilevazione e gestione del crash di un nodo secondario:

1. Il leader, al cambio del turno, manda a tutti i nodi un ping per accertarsi della loro presenza;
2. Il ping a P4 fallisce (figura 3.a) e il Leader modifica il proprio stato rimuovendolo dalla lista dei giocatori e le carte di P4 vengono rimesse nel mazzo pesca;
3. Il leader manda lo stato aggiornato ai giocatori restanti (figura 3.b), in modo che ogni giocatore venga a conoscenza del crash di P4.

Se il Leader fa crash, il gioco non può proseguire fino all'elezione di un nuovo Leader. A causa di questo il crash di un Leader deve essere rilevato il prima possibile e dunque mentre il Leader completa il suo turno, gli altri giocatori controllano che esso sia ancora presente mandandogli periodicamente dei ping.

In figura 4 viene mostrato un esempio di rilevazione e gestione di un crash del Leader di gioco:

1. I giocatori inviano ping al Leader ad intervalli regolari;
2. I ping al Leader falliscono (figura 4.a) e si avvia la fase di Elezione. I nodi mandano i ping al giocatore successivo a quello crashato:
   - Se il giocatore successivo non risponde ai ping (crash), si ripete la procedura fino a quando non si trova un nodo disponibile;
   - Se il giocatore successivo è colui che sta mandando il ping, esso si autoelegge Leader;
   - Altrimenti si manda una richiesta di elezione al nodo (figura 4.b).
3. Il giocatore che ha ricevuto le richieste di elezione richiede lo stato a tutti i giocatori e prende solo quello con orologio logico più alto, modificandolo e mettendo se stesso come Leader;
4. Il nuovo Leader invia lo stato aggiornato a tutti i giocatori (figura 4.c).

3 Aspetti implementativi

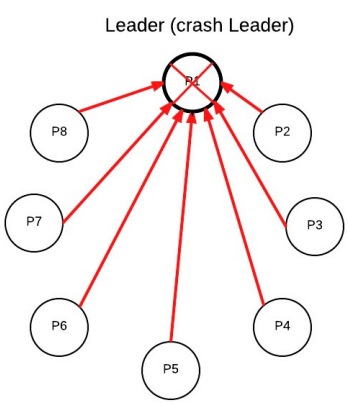
3.1 Strumenti utilizzati

Per realizzare il gioco sono stati utilizzati i seguenti strumenti:

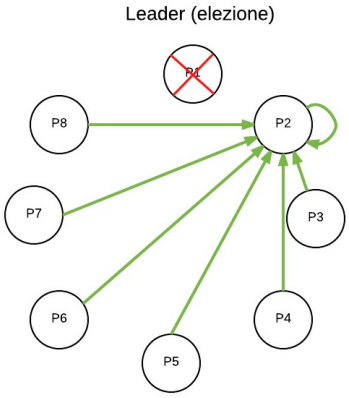
Java Linguaggio multipiattaforma orientato agli oggetti, progettato per essere il più possibile indipendente dalla piattaforma di esecuzione.

RMI (Remote Method Invocation) Una tecnologia che permette ai processi Java di comunicare attraverso una rete distribuita.

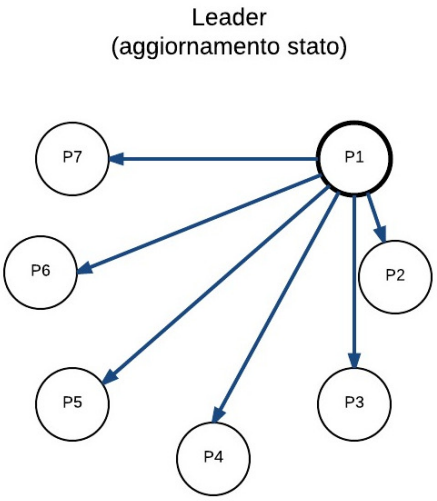
Slick2D Una libreria di Java che include un insieme di strumenti e utilities costruite intorno a LWJGL (OpenGL), in modo da rendere più semplice la realizzazione di giochi in 2D (rif. [2]).



.a I processi rilevano il crash



.b Elezione di un nuovo leader



.c Sincronizzazione dello stato

Figura 4. Gestione del crash del leader

Ant Strumento per l'automazione di un processo di build simile a Make ma scritto in Java.

Git Sistema software distribuito di controllo di versione del codice di un progetto.

Eclipse e NetBeans IDE per lo sviluppo ottimizzati per il codice Java, con funzioni avanzante quali l'autocompletamento, il refactoring e il debugging.

Gimp Strumento open-source per l'elaborazione di immagini raster.

Xubuntu, Manjaro e Debian Distribuzioni Linux utilizzate per lo sviluppo e il testing del software.

3.2 Interfaccia grafica

Figura 5. Stanza di gioco

L'interfaccia grafica è composta da una sequenza di schermate, così suddivise:

Menù principale In questo menù è possibile avviare il gioco, entrare nelle impostazioni o uscire dall'applicazione.

Impostazioni In questa schermata si può modificare la dimensione della finestra per metterla a schermo intero.

Play In questo menù si può scegliere se creare una stanza o entrare in una stanza già creata da un altro utente.

Join In questa schermata si possono inserire il nickname pubblico, i dati dell'utente a cui connettersi (indirizzo IP, porta) per entrare nella relativa stanza.

Crea stanza In questa schermata si possono inserire il nickname pubblico e la porta a cui gli altri utenti si devono connettere per entrare nella nuova stanza.

Stanza di matchmaking In questa schermata (figura 5) viene visualizzata la lista di giocatori connessi tra di loro, in attesa di giocare. Ogni giocatore può premere sul tasto "Start" per dire di essere pronto (in questo caso compare un tick di fianco al nome del relativo giocatore). Quando tutti i giocatori presenti nella stanza (minimo 3) hanno premuto "Start", il gioco comincia.

Gioco Questa schermata (figura 6) contiene tutte le informazioni utili ai fini del gioco vero e proprio. Ogni giocatore può vedere le sue carte, l'ultima carta giocata, il senso di gioco e il numero di carte in mano agli altri giocatori. Durante il suo turno, un giocatore può giocare una carta premendo su una delle carte in basso. Le carte uguali sono raggruppate in un'unica carta con il numero di carte indicato in alto. Chi ha il turno, inoltre, vede tre pulsanti:

Pesca Consente di pescare una carta. Questa operazione può essere fatta solo una volta per turno.

Passa Consente di passare il turno senza giocare una carta. Questa operazione può essere fatta solo dopo aver premuto il tasto Pesca.

UNO! Consente di giocare una delle due carte rimanenti in mano senza incorrere nella penalità prevista dalle regole del gioco.

Ogni schermata di gioco viene disegnata diverse volte al secondo, eseguendo una funzione di rendering dentro la quale viene rilevato un eventuale cambiamento di stato e in base ad esso vengono aggiornati i componenti (carte nella mano, numero di carte in mano agli altri giocatori, etc.) all'interno dell'interfaccia.

Figura 6. Schermata di gioco

3.3 Struttura del codice

In figura 8 sono illustrati due UML relativi alla struttura del codice del progetto. Il codice è strutturato in tre diversi pacchetti JAVA che comunicano tra di loro, ognuno con la sua relativa funzione:

Communication In questo pacchetto (figura 8.a) sono contenute tutte le classi per la gestione della logica del gioco e delle comunicazioni tra i processi. Questo pacchetto include la classe relativa allo stato del gioco, che per essere inviato sulla rete implementa Serializable.

Game Questo pacchetto (figura 8.b) contiene tutte le classi relative alla gestione dell'interfaccia di gioco (GUI) e degli eventi ad essa collegati.

Utilities Questo pacchetto contiene i valori di alcuni elementi degli altri pacchetti, come stringhe e numeri.

3.4 Gestione delle comunicazioni

In figura 7 viene illustrato un esempio di comunicazione tra processi e di gestione di crash di un nodo diverso dal leader.

All'inizio del suo turno, il leader manda un ping di richiesta a tutti gli altri giocatori per verificare la loro presenza ("controllo utente") e aspetta la loro risposta. Se una o più di queste richieste dovessero fallire con una eccezione di tipo RemoteException, si avvia sul leader la procedura di gestione dei crash dei relativi utente. Questa procedura consiste nella rimozione dei giocatori[2] dallo stato del gioco e l'invio dello stato aggiornato a tutti gli altri giocatori ancora presenti in partita. L'aggiornamento dello stato viene fatto su un thread a parte e questo richiede dunque un lock su ogni processo per gestire le sezioni critiche derivate dalla lettura e scrittura contemporanea sugli stessi insiemi di dati.

I nodi che aspettano il loro turno inviano un ping di richiesta al leader ogni 3 secondi utilizzando un timer su un thread separato. Quando uno di questi ping fallisce e restituisce una eccezione di tipo RemoteException, il processo rileva il crash del leader ed inizia la fase di elezione, cercando di eleggere il nodo immediatamente successivo (in base al senso di gioco) a quello crashato. Quando un nodo riceve una o più richieste di elezione accetta solo la prima e si auto-elegge leader. Il nuovo leader invia a tutti gli altri nodi una richiesta di stato per prendere quello con orologio logico maggiore. Se questa richiesta dovesse fallire con una RemoteException, il nuovo leader si occupa di rimuovere il relativo utente dallo stato appena aggiornato. Successivamente il leader modifica lo stato inserendosi come leader, incrementa il clock e invia lo stato aggiornato a tutti gli altri giocatori. In caso di crash di tutti i giocatori tranne uno, quest'ultimo si auto-elegge leader e si proclama vincitore.

Oltre che in questi casi particolari, il leader invia una richiesta di aggiornamento stato (incrementando sempre il valore del clock) ogni volta che:

- Pesca una nuova carta;

[2] Le carte in mano ai giocatori rimossi dal gioco vengono messe in fondo al mazzo scarti.

– Gioca una nuova carta (passando il turno);
– Salta il turno;
– Applica una penalità ricevuta dal giocatore precedente;
– Vince la partita.

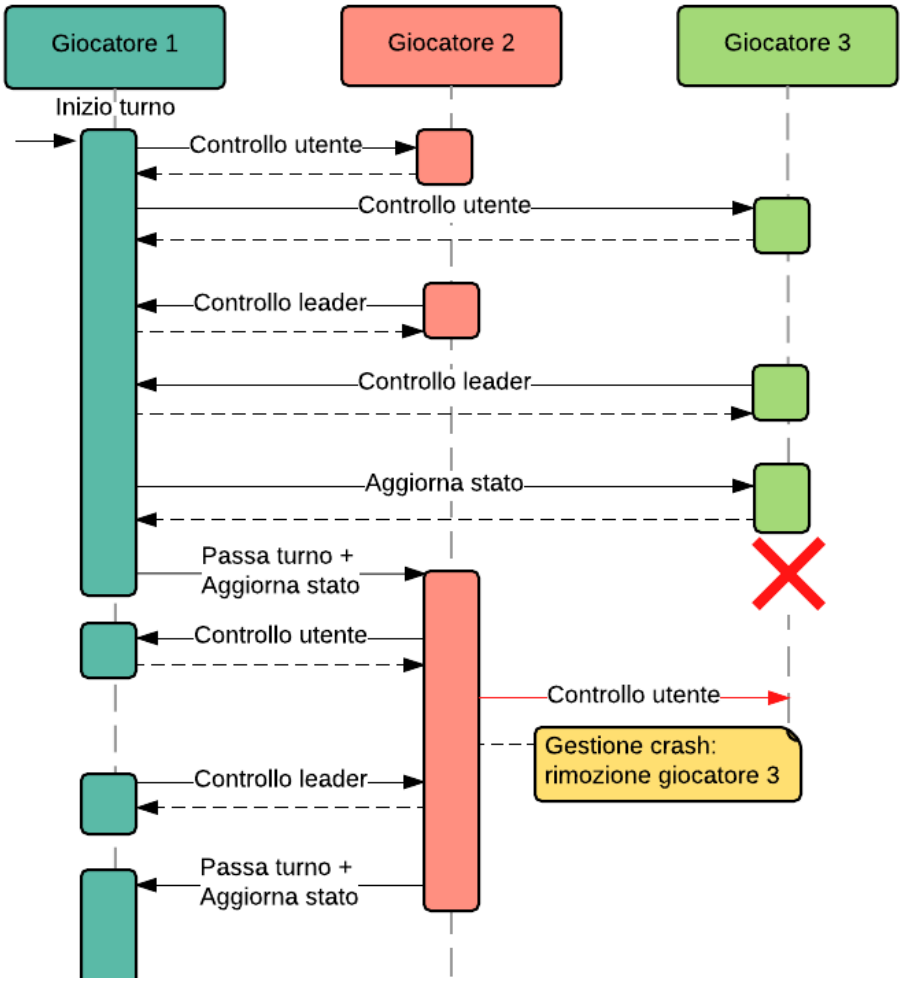


Figura 7. Interazioni UML

4 Conclusioni

In questo progetto si è realizzato un semplice sistema distribuito per comprendere al meglio i meccanismi di comunicazione e la tolleranza ai guasti di questi sistemi. Il gioco di carte UNO realizzato in questa forma ha permesso di valutare le più importanti caratteristiche dei sistemi distribuiti.

La prima parte del progetto ha riguardato lo studio e la realizzazione della struttura di rete distribuita tra i giocatori. In questa parte si è cercato di dare uguale importanza a tutti i processi anche durante la fase di matchmaking, evitando quindi di utilizzare un server centralizzato per la creazione della stanza di gioco.

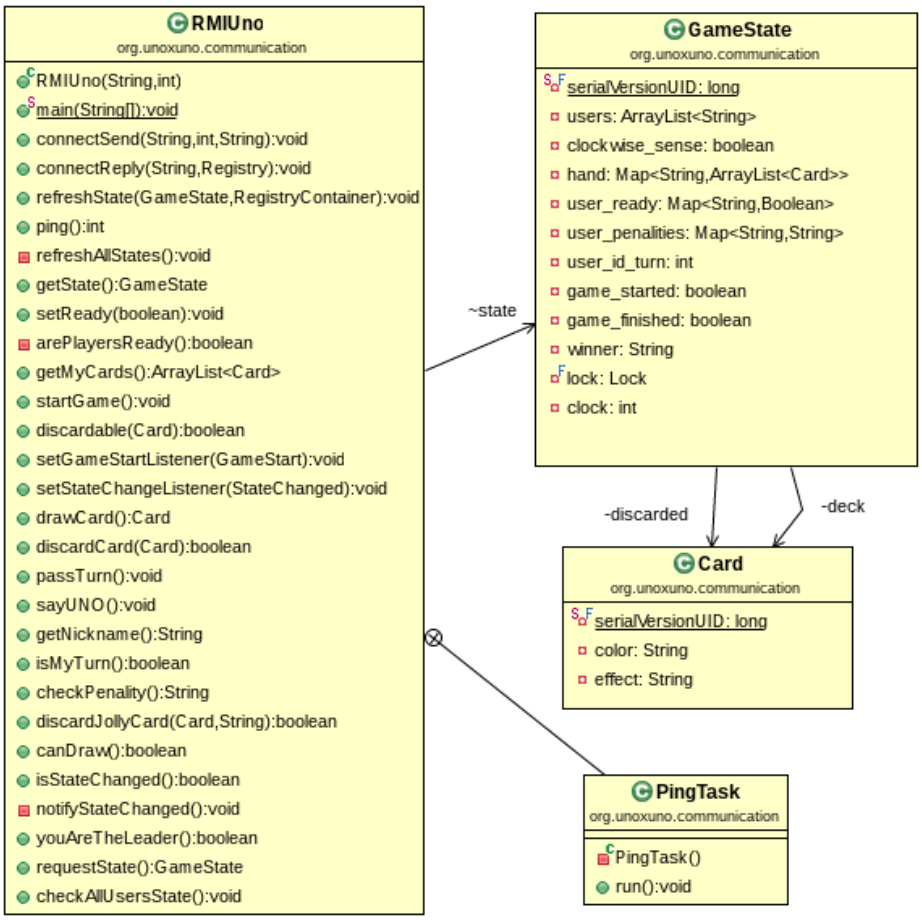
Successivamente lo sviluppo è stato indirizzato verso l'implementazione delle meccaniche principali di gioco e la gestione dei crash sul Leader o su altri nodi [3]. L'interfaccia grafica è stata realizzata contemporaneamente all'implementazione di queste meccaniche, anche per permettere il testing di ogni nuova funzionalità.

L'obiettivo del progetto è stato raggiunto e il testing sulla rete locale ha soddisfatto i requisiti posti inizialmente.

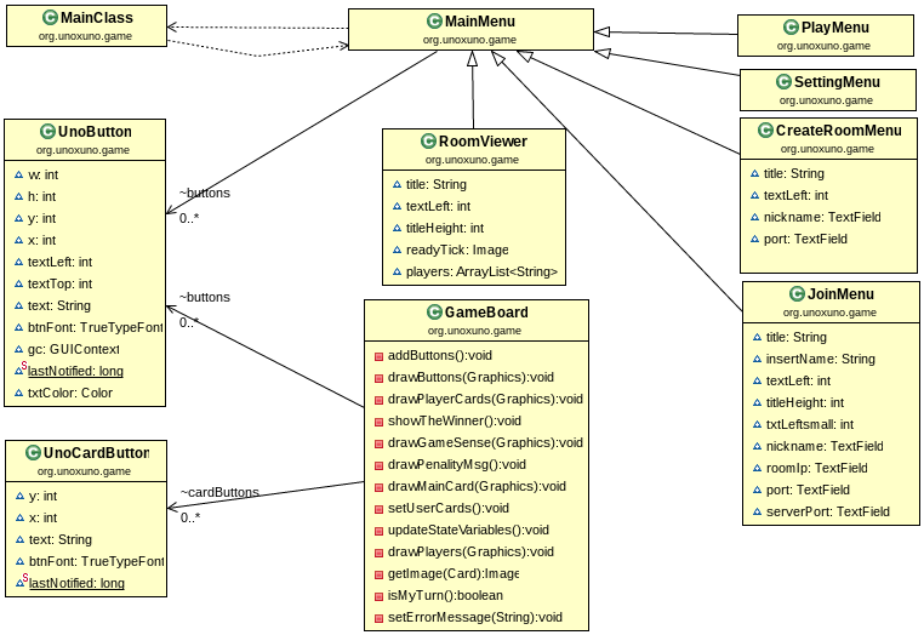
Riferimenti bibliografici

[1] Regole di UNO: https://it.wikipedia.org/wiki/UNO_(gioco_di_carte)
[2] Slick2D: http://slick.ninjacave.com/

[3] Vengono gestiti da 1 a N-1 crash, dove N è il numero di giocatori



.a Struttura del codice: comunicazione



.b Struttura del codice: GUI

Figura 8. Struttura del codice