Documentazione Progetto
Lorenzo il Magnifico

Benedetti Claudio

Surace Pinò

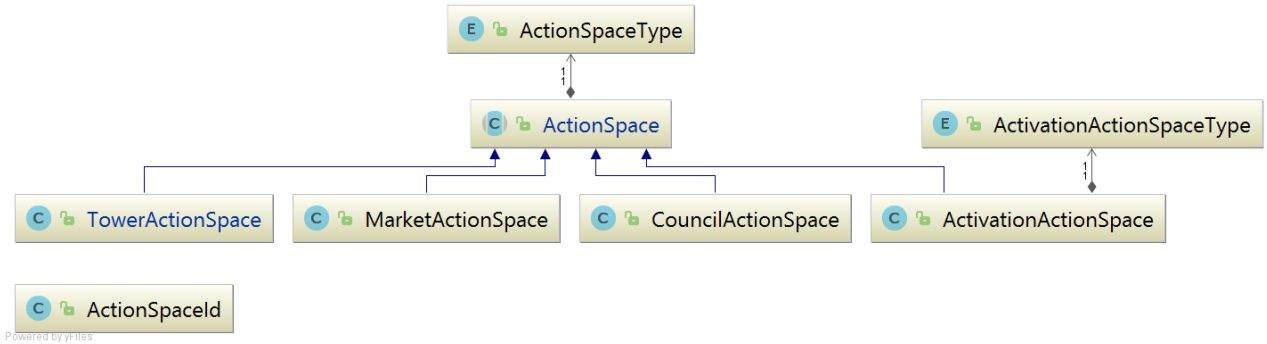
Logica del Gioco

La logica di gioco è stata implementata con il seguente schema generale:

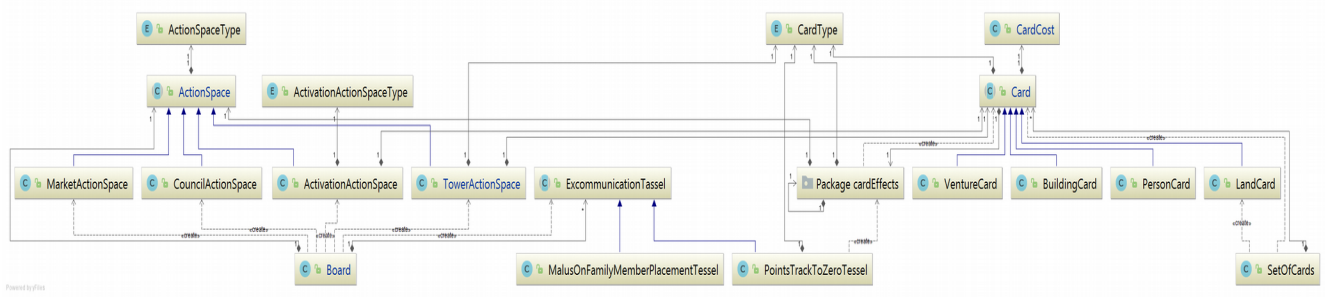
In primo luogo, la gestione del gioco “vera e propria” (ad esempio, lo scheduling dei turni o l’elezione del giocatore vincitore) sono gestite dal package gameStructure. In essa la classe principale è gameRoom. Questa si avvale delle altre classi del package principalmente come supporto. Essa svolge inoltre il compito di punto di incontro fra network e logica.

Il resto del model è composto da package svolti a modellizzare i principali componenti del gioco. In essi si è spesso ricorso al paradigma dell’ereditarietà (spesso da una superclasse astratta) e del polimorfismo, in modo da riuscire ad astrarre le funzionalità chiave ed evitando pesanti controlli “stile-switch” (esempio, Card è classe astratta da cui ereditano LandCard e PersonCard: in Player ci si avvale della classe Card sia per modellizzare LandCard che PersonCard).

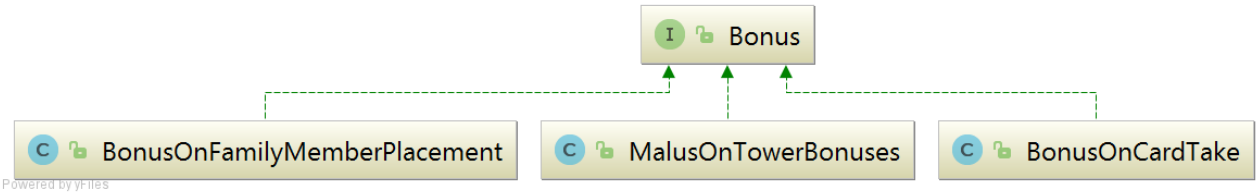
Spazi azione



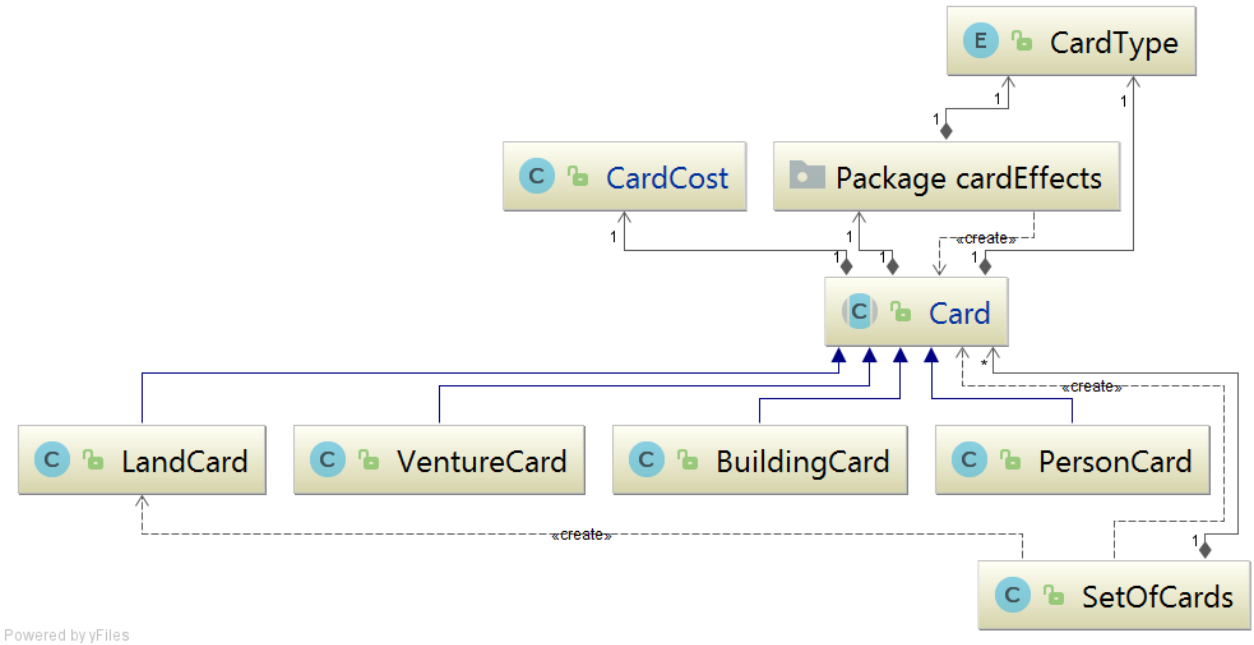
Tabellone di Gioco


ActionSpaceType
CardType
CardCost
ActionSpace
ActivationActionSpaceType
Card
MarketActionSpace
CouncilActionSpace
ActivationActionSpace
TowerActionSpace
ExcommunicationTassel
Package cardEffects
VentureCard
BuildingCard
PersonCard
LandCard
Board
MalusOnFamilyMemberPlacementTessel
PointsTrackToZeroTessel
SetOfCards

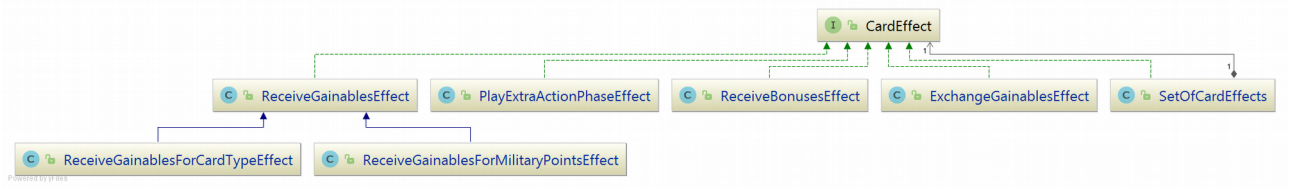
Bonus


Bonus
BonusOnFamilyMemberPlacement
MalusOnTowerBonuses
BonusOnCardTake

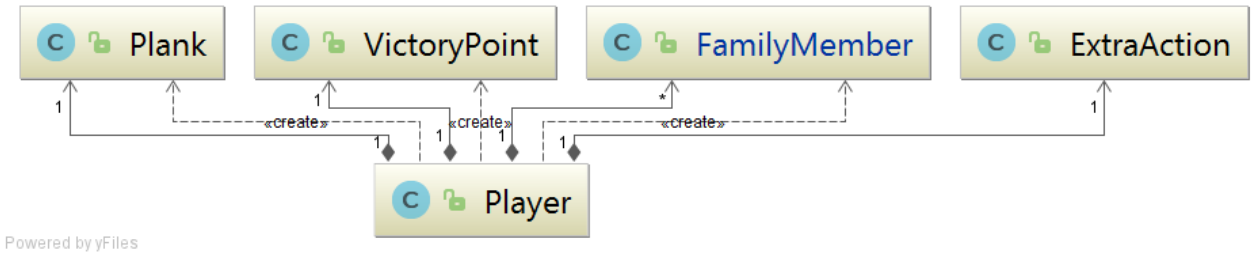
Carta



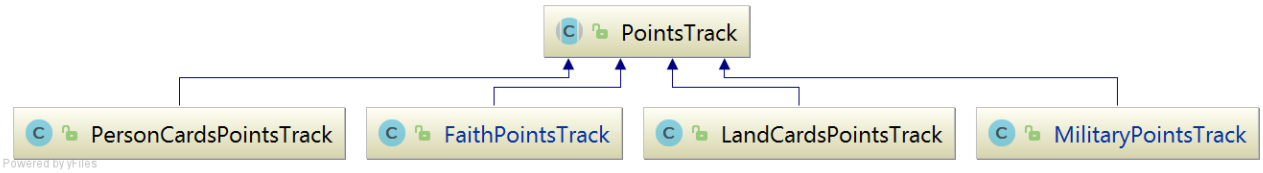
Effetti



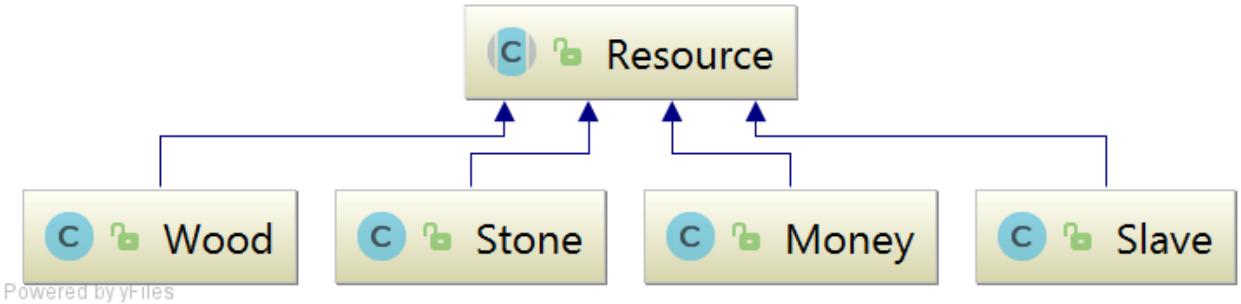
Giocatore



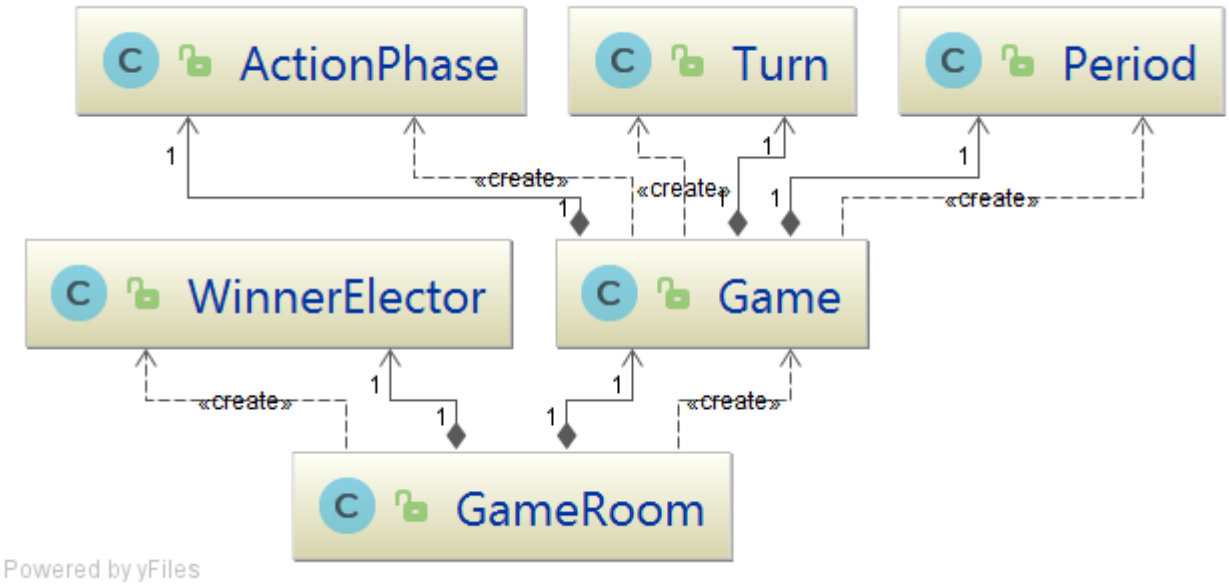
Punti



Risorse



Gioco


ActionPhase
Turn
Period
WinnerElector
Game
GameRoom
«create»
«create»
«create»
«create»
«create»
1
1
1
1
1
1
1
1
Powered by yFiles

Network

La parte network è stata implementata sì da permettere di usufruire sia della tecnologia rmi che socket, e in modo che client con tecnologie diverse potessero giocare insieme. Per far questo è stato implementato un primo “strato” di interfacce e classi (spesso astratte) con lo scopo di disaccoppiare le funzioni da implementare dalle effettive loro implementazioni, delegate ad un secondo “strato”, dipendente invece dalla tecnologia.

Esempio:

La classe RemotePlayer (primo strato) è astratta ed estesa da SocketPlayer e RMIPlayer (secondo strato).

Ad ogni client è associato un SocketPlayer o un RMIPlayer a seconda di quale tecnologia il primo abbia scelto di usufruire, che ne gestisce le richieste, “propagandole” opportunamente verso la logica, ed aggiornandolo sull’output delle richieste stesse.

In tutto ciò, SockePlayer e RMIPlayer si limitano ad usare le rispettive tecnologie per implementare le stesse funzionalità, definite in RemotePlayer. Si è dunque fatto sì che “al resto del programma” sia solo visibile l’oggetto RemotePlayer, permettendo di evitare pesanti e antiestetici controlli del tipo

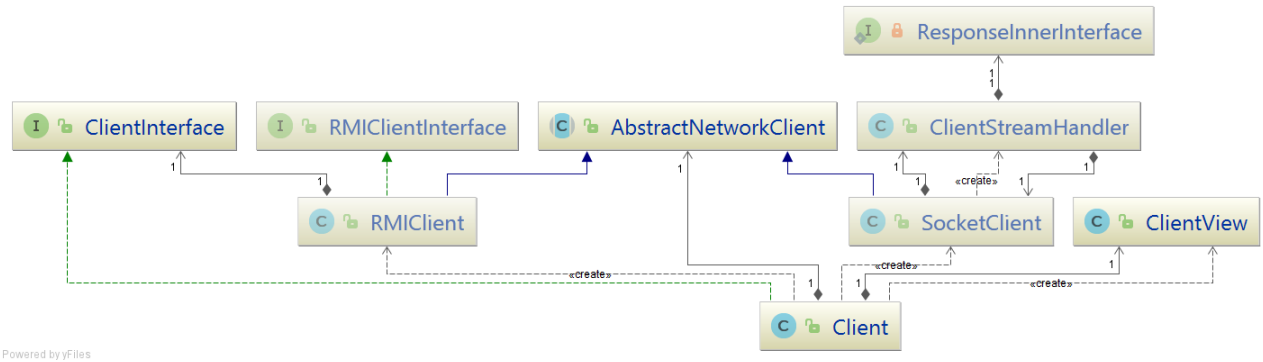
```
If tecnologia_usata = socket

then //prendi la carta usando le socket;

else

//prendi la carta usando RMI;
```

sostituendoli con codice del tipo:

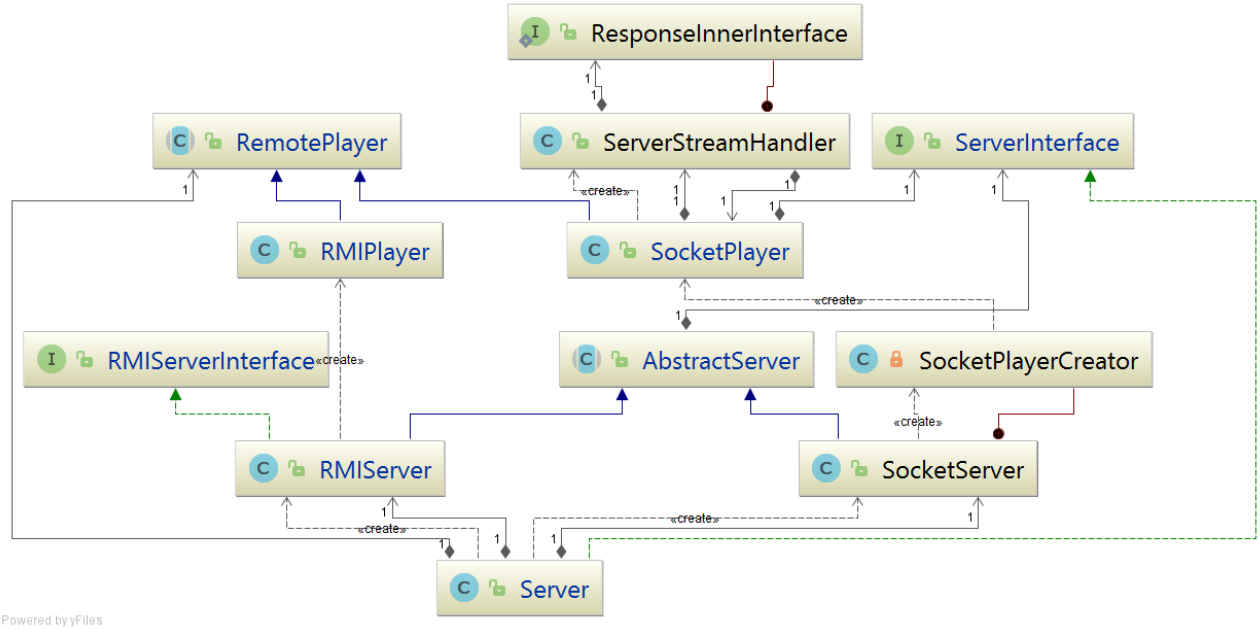
```
RemotePlayer.prendiLaCarta();
```

Si è dunque fatto largo uso dei paradigmi di ereditarietà e polimorfismo per dati.

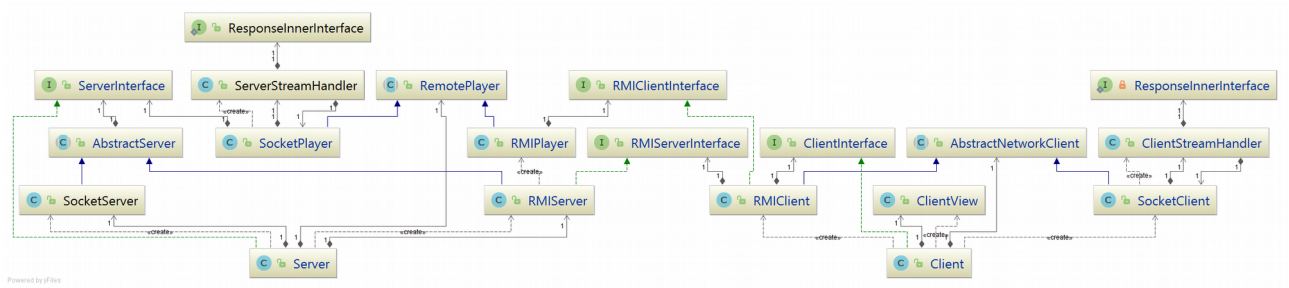
Client



Server

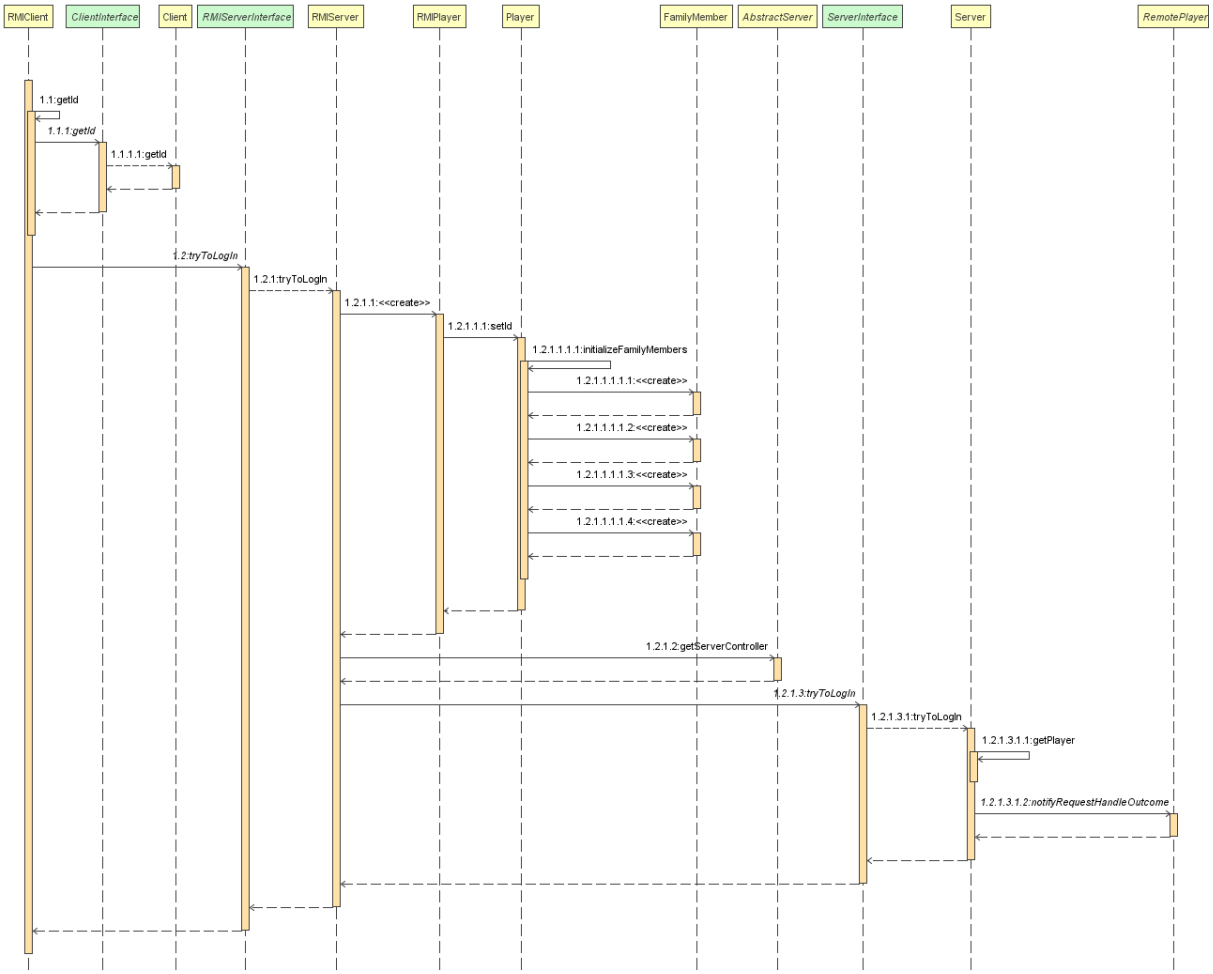


Network: overview

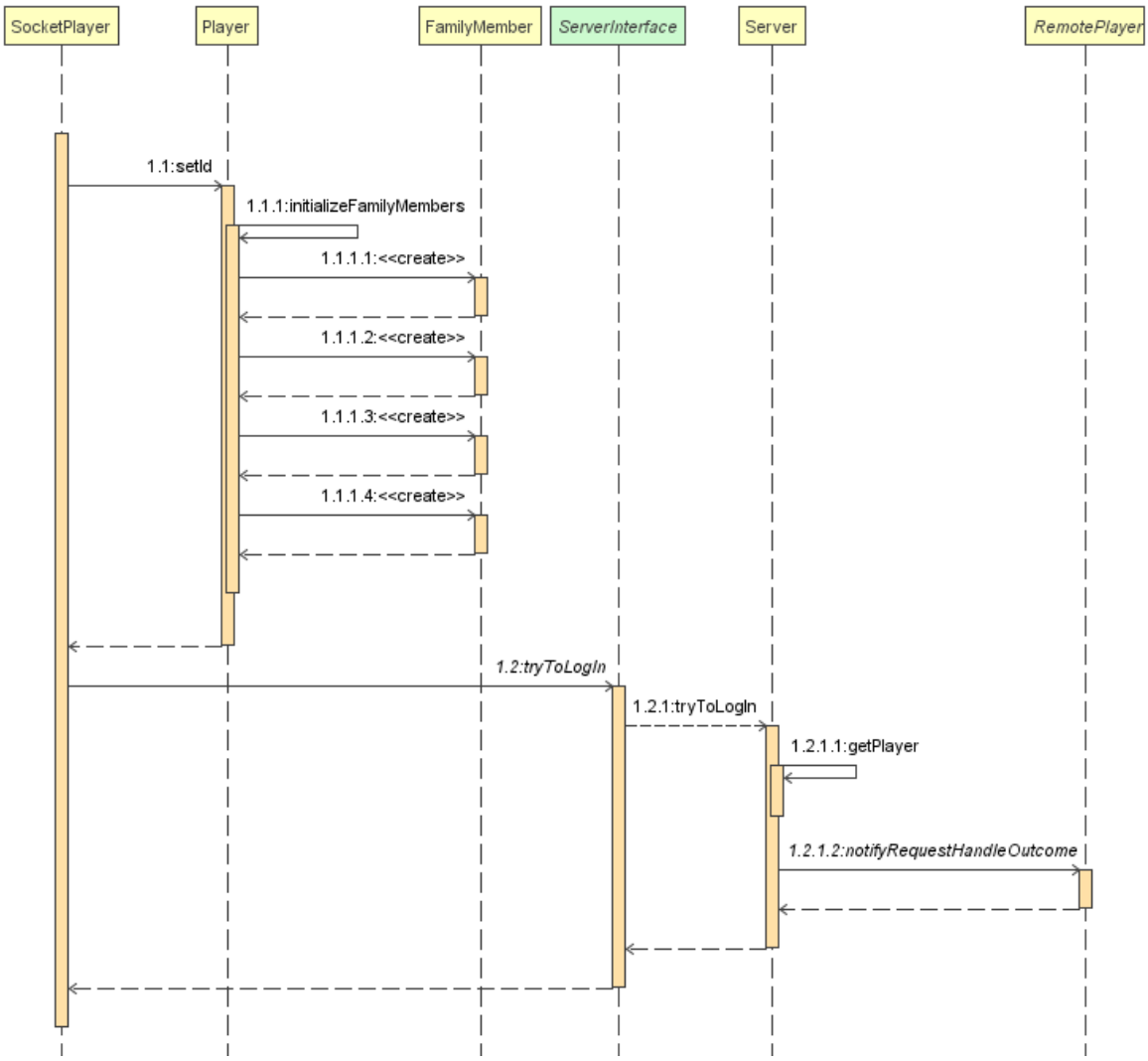


Evoluzione del Gioco

RMI



Socket (Lato Server)

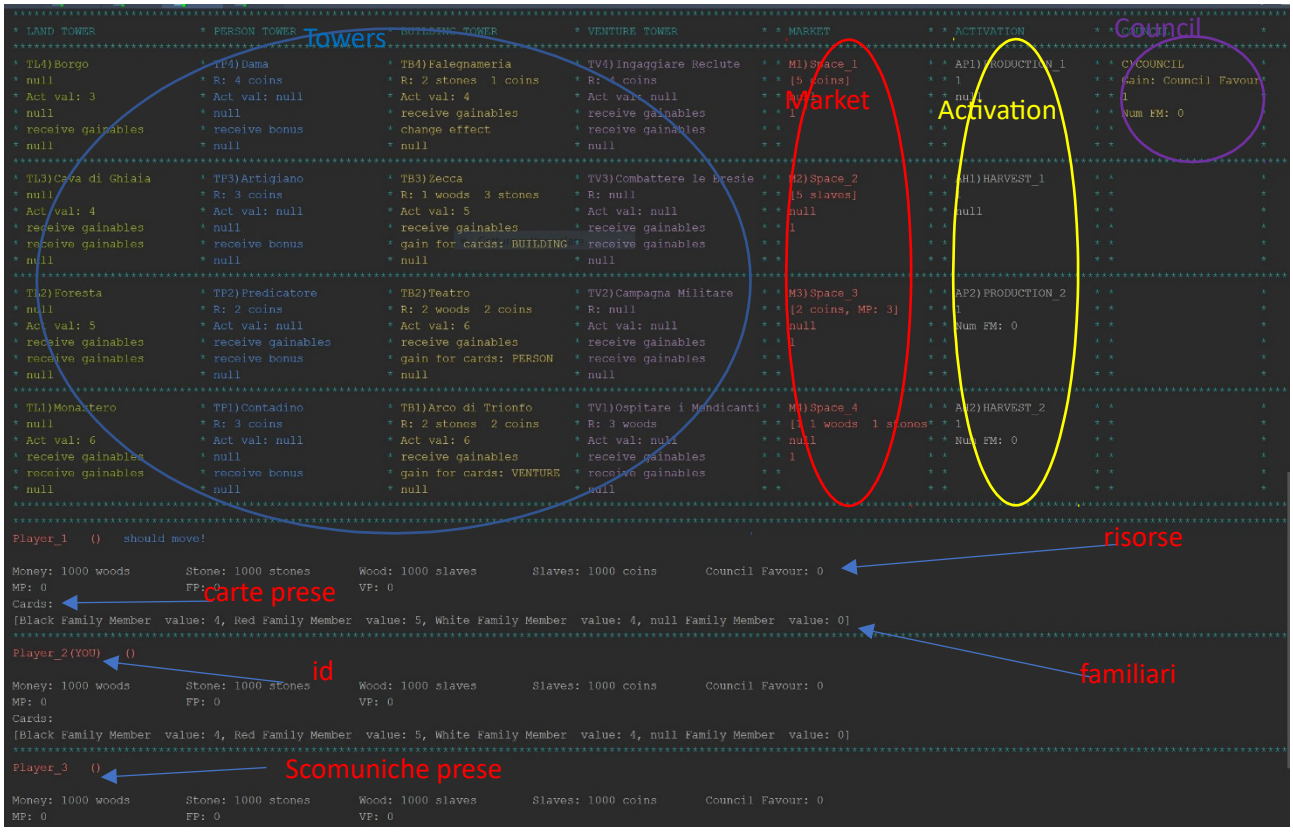


Interfaccia Utente

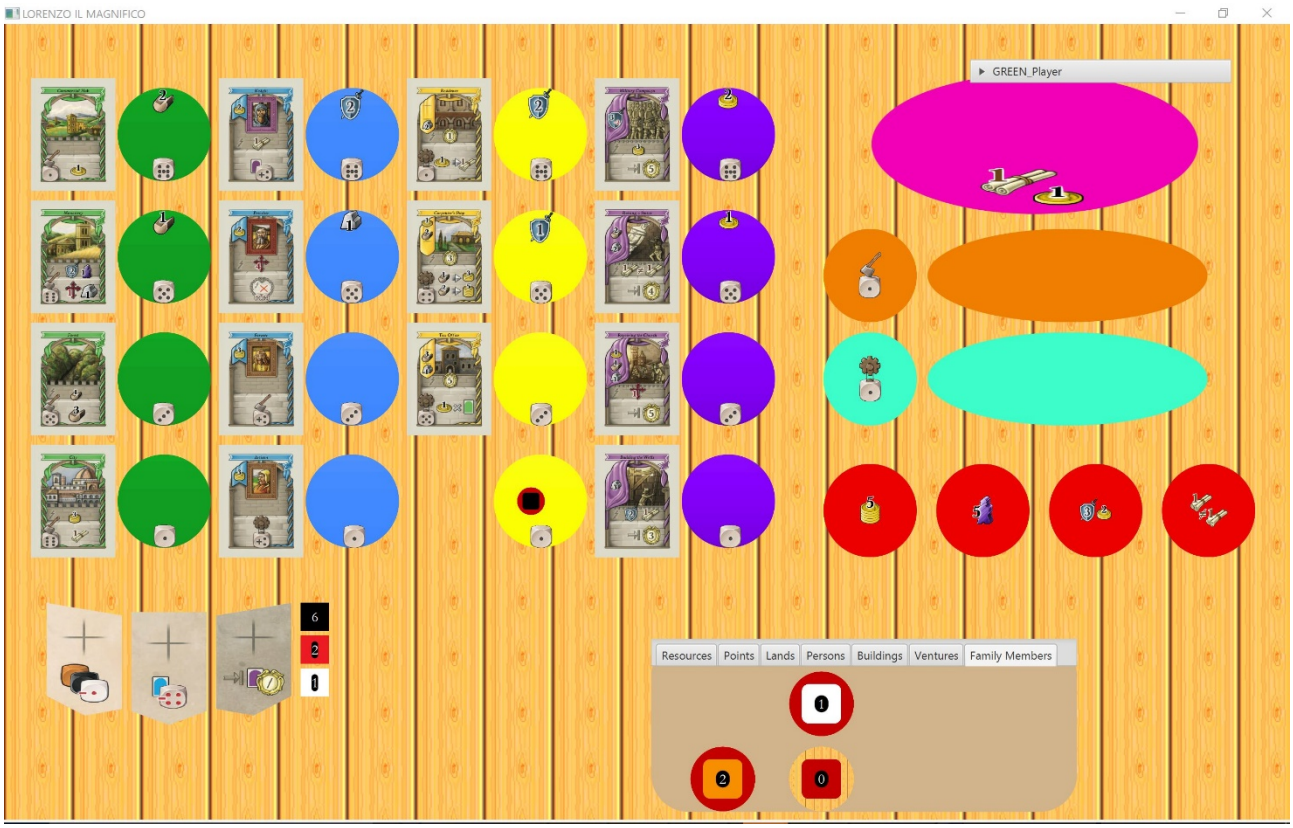
L’interfaccia utente si avvale, come il Network peraltro, di ereditarietà e polimorfismo per dati per disaccoppiare le funzioni offerte dalle loro effettive implementazioni.

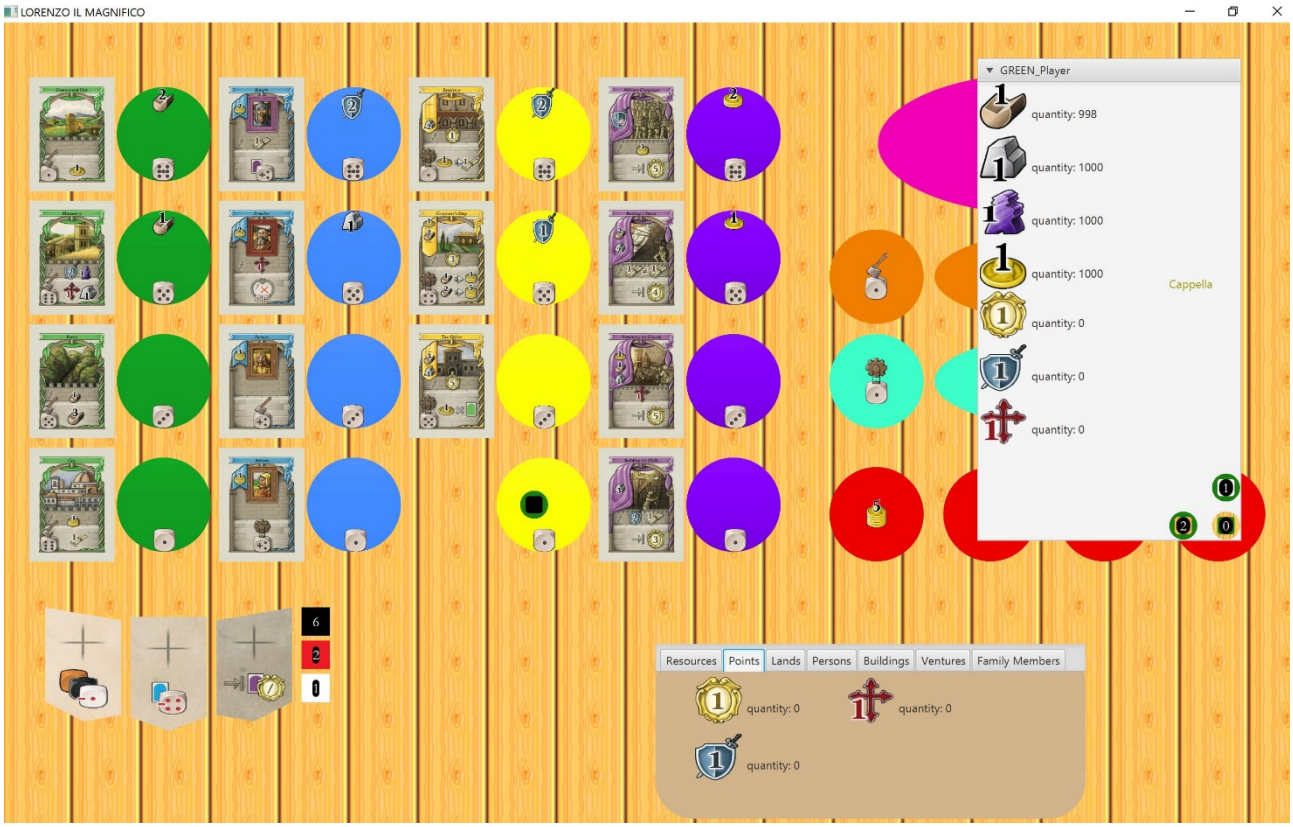
Infatti la classe astratta AbstractUserInterfaceClient definisce le funzionalità che la generica UI deve garantire, delegando l’implementazione delle stesse alle due sottoclassi CliClient per la CLI e GuiClient per la GUI.

CLI

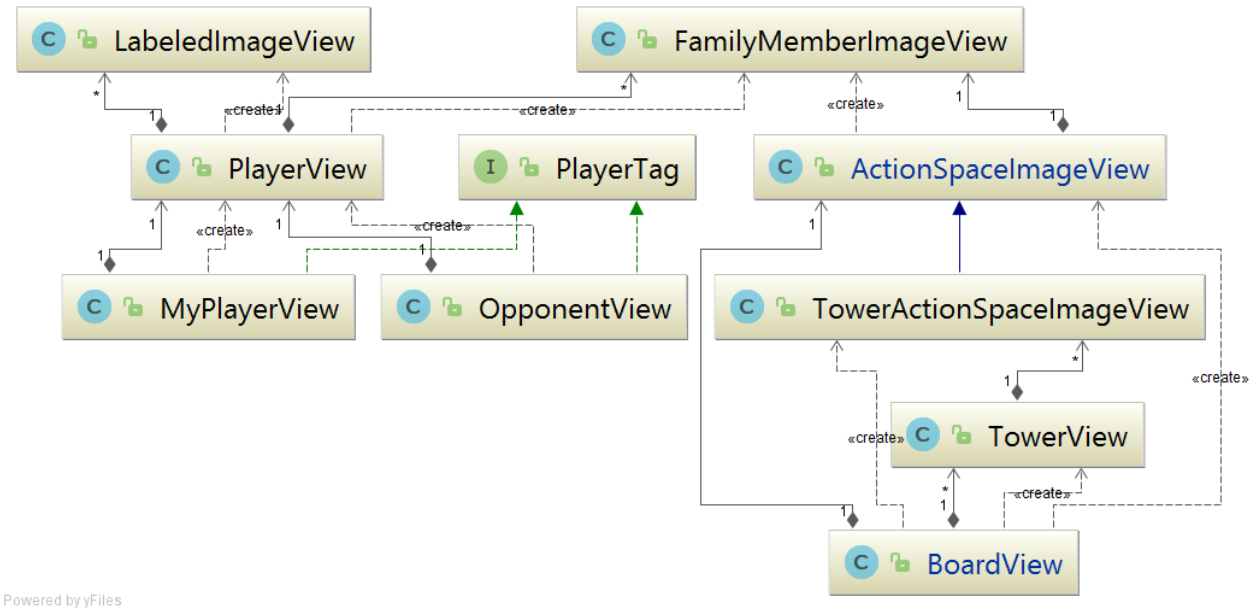


GUI

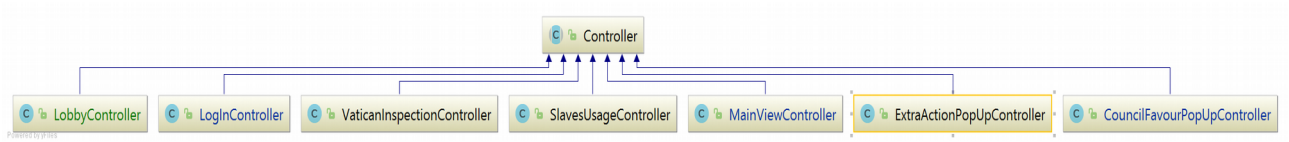

LORENZO IL MAGNIFICO
GREEN_Player
6
2
0
Resources
Points
Lands
Persons
Buildings
Ventures
Family Members
1
2
0


LORENZO IL MAGNIFICO
GREEN_Player
quantity: 998
quantity: 1000
quantity: 1000
quantity: 1000
quantity: 0
quantity: 0
quantity: 0
Cappella
6
2
0
Resources
Points
Lands
Persons
Buildings
Ventures
Family Members
quantity: 0
quantity: 0
quantity: 0

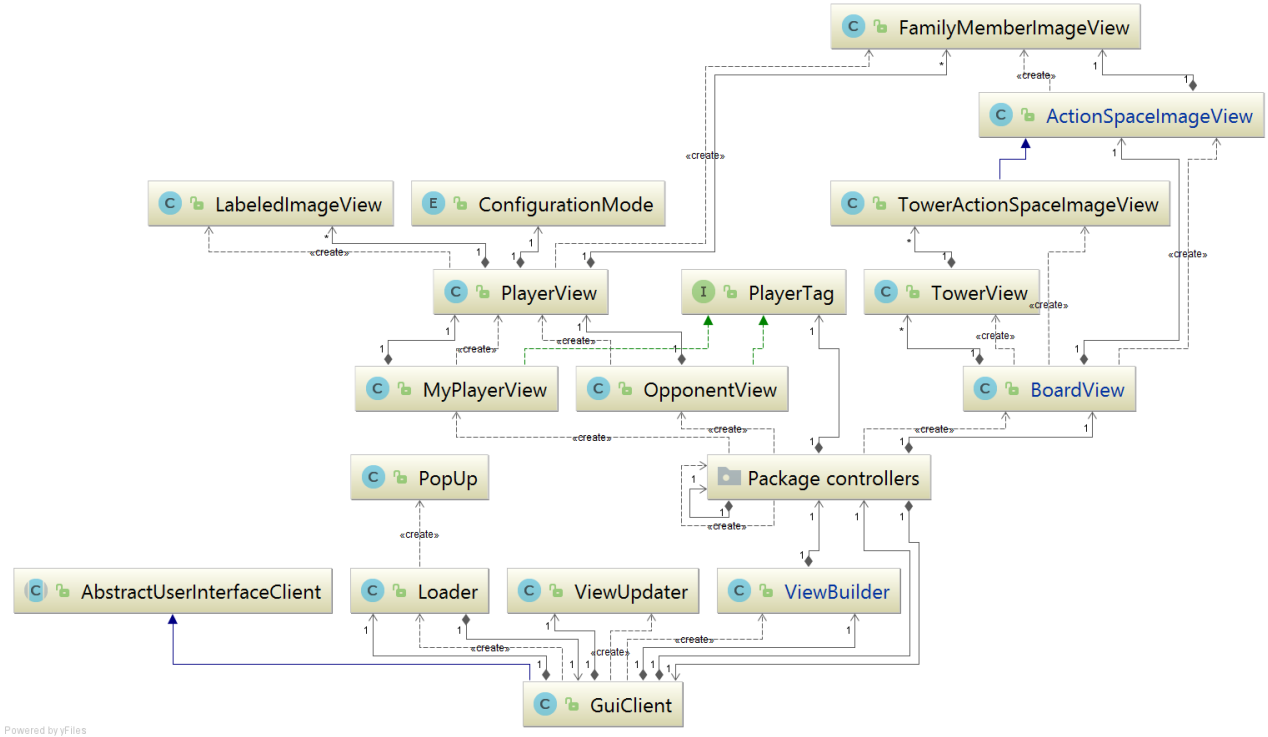
GUI: components



GUI: controllers



GUI: overview



Test

I test sono stati implementati sulla logica del gioco, in particolare sui metodi principali di ogni classe, ovvero quelli più esposti alle altre istanze.