

Message Class: Message

Message

-messageType: String
-senderNickName: String
-messageContent: String
-warehouseConfiguration: Warehouse
-resourcesToDistribute: ResourceType[2]
-leaderCardPositions: int[2]
-choosableLeaderCards: LeaderCard[4]
-chosenLeaderCards: LeaderCard[2]
-actualCurrentPlayer: String
-levelsToSwitch: int[2]
-marketPosition: int[2]
-temporaryResources: HashMap <ResourceType,Integer>
-resourceToInsert: ResourceType
-quantityToInsert: int
-intoExtraDeposit: boolean
-newFaithPoints: int
-developmentCardConfiguration: DevelopmentCard [3]
-extraDepositConfiguration: ExtraDeposit[2]
-strongboxConfiguration: HashMap <ResourceType,Integer>
-payUsingExtraDeposit: int[2]
-remainingWhiteMarbles: int
-actionCardConfig: ActionCardStack
-deckgrid: Deckgid

+getMessageType(): String
+getSenderNickname(): String
+getMessageContent(): String
+getWarehouseConfiguration(): Warehouse

NOTE GENERALI:

1: Prima di ogni interazioni ci sarà un "ping-pong" tra client e server per verificare che il client sia ancora connesso
2: Il protocollo si basa sulla classe Message dove ogni messaggio setterà solo alcuni attributi della classe. L'attributo fondamentale è il messageType, attraverso il quale il controller saprà leggere solo gli attributi necessari dell'oggetto Message e tradurlo in invocazioni sul model. Allo stesso modo la view tradurrà i messaggi che riceve in modifiche su quello che mostra all'utente.
3: Abbiamo deciso di comunicare tra client e server attraverso messaggi di tipo JSON, quindi sia controller che view avranno un parser che deserializzerà i messaggi JSON in un oggetto Message. Allo stesso modo avranno un metodo che serializzerà in messaggi JSON (per esempio per tradurre sulla view i click dell'utente).

sd Adding player

Client

Server

1 : Create Socket

2 : Connection request

3 : On ok - connection accepted ask login

4 : On error- invalid nickname or reached four players

5 : Login(nickname)

6 : On ok - player added to the game

7 : On error- invalid nickname or reached four players

8 : Notify all clients a player has been added

2: ["hostName", "portNumber"]

CONNECTION REPLY(SERVER)

3: {"messageType":"Simple message", "messageContent":"Connection accepted, please login with the nickname \n"}

4: {"messageType":"Simple message", "messageContent":"Connection refused, too many clients connected\n"}

LOGIN REQUEST(CLIENT)

5: {"messageType":"AddPlayer", "senderNickname":"Player's nickname"}

LOGIN REPLY(SERVER)

6: {"messageType":"Simple message", "messageContent":"Player added to the game\n"}

7: {"messageType":"Simple message", "messageContent":"Error: Your nickname is invalid or too many players have logged to the game\n"}

8: {"messageType":"AddPlayer", "senderNickname":"Player's nickname"}

sd Start single player

Client

Server

1 : Creating single player request

2 : On ok - notify single player creation

3 : On error - notify too many players added or game already started

SINGLE PLAYER CREATION(CLIENT)

1: {"messageType" : "Start single player"}

SINGLE PLAYER CREATION REPLY(SERVER)

2: {"messageType":"Simple message", "messageContent":"Single player game started succesfully\n"}

3:{"messageType":"Simple message", "messageContent":"Error: Too many players added to the game or a game has already started\n"}

sd Start multiplayer

Client

Server

1 : Creating multiplayer request

2 : On ok - notify multiplayer creation update GUI

3 : On error - notify only one player added to the game or game already started

MULTI PLAYER CREATION(CLIENT)

1:{"messageType" : "Start multiplayer"}

MULTI PLAYER CREATION REPLY(SERVER)

2: {"messageType" : "multiplayerCreation", "messageContent":"Multi player game started succesfully\n", "playerNumber: int, "choosableLeaderCards":LeaderCards[4]}

3: {"messageType" : "Simple message", "messageContent":"Error: Not enough players added to the game or a game has already started\n"}

sd FirstResourceDistribution

Client

Server

1 : First distribution request

2 : On ok- Notify action completed succesfully - GUI changed

3 : On error- Notify wrong action

```
1: {"messageType": "distributionSecondThird", "senderNickname": "Player's nickname", "ResourceToDistribute": "ResourceType"}
**{"messageType": "distributionFourth", "senderNickname": "Player's nickname", "ResourceToDistribute": "ResourceType", "SecondResourceToDistribute": "ResourceType"} //The fourth player has to choose 2 resources
2: {"messageType": "notifyWareHouseChange", "warehouseConfiguration": Warehouse }
3: {"messageType": "simpleMessage", "messageContent": "You don't have permissions for a free resource distribution\n"}
```

sd LeaderCardsChoice

Client

Server

1 : Selection of 2 leader cards from his choosable cards

2 : On ok- Notify choice completed - update GUI

3 : On error- Notify selection failed

```
1:{"messageType":"leaderCardSelection","senderNickname":"Player's nickname", leaderCardPosition1:int, leaderCardPosition2:int}  
2:{"messageType":"chosenLeaderCards", "messageContent":"Leader card choice completed succesfully\n", "chosenLeaderCards": leaderCards[2]}  
3:{"messageType":"simpleMessage", "messageContent":"Selection failed\n"}
```

sd endTurn

Client

Server

1 : End turn request

2 : On ok - notify all players current player changed - update GUI

3 : On error - notify end turn fail

4 : Optional - notify faith points added if discarded resources

5 : Optional- notify all players the end of the game

6 : Optional- The winner is...

- 1: {"messageType": "End turn request", "senderNickName": "Player's nickname"}
- 2: {"messageType": "endTurnNotification", "actualCurrentPlayer": "Player's nickname"}
- 3: {"messageType": "simpleMessage", "messageContent": "You can't end your turn now\n"}
- 4: {"messageType": "earnedFaithPoints", "messageContent": "Player current player discarded n resources\n", "newFaithPoints": "int"}
- 5: {"messageType": "simpleMessage", "messageContent": "at the end of the turn victory points will be calculated and game will end"}
- 6: {"messageType": "simpleMessage", "messageContent": "The winner is game.getWinnerPlayer.getNickname(). Congrats!"}

sd SwitchLevels

Client

Server

1 : SwitchLevel request

2 : On ok - notify action complete - GUI changed

3 : On error - notify switch failed

```
1:{"messageType":"SwitchLevel","senderNickname":"Player's Nickname","levelsToSwitch":[int level1, int level2]}
2: {"messageType": "notifyWareHouseChange", "warehouseConfiguration": Warehouse }
3: {"messageType": "simpleMessage", "messageContent":"You can't switch those levels\n"}
```


sd takeResourcesFromMarket

Client

Server

1 : takeResourcesFromMarket request

2 : On ok - notify GUI changed

3 : On ok - notify all GUI changed

4 : On error - notify invalid action

```
1:{"messageType":"takeResourceType","marketPosition":[int row,int column]}
2: {"messageType": "notifyTemporaryResourcesChanged", "temporaryResourcesConfiguration": HashMap<ResourceType,Integer> }
3:{"messageType":"notifyMarketboardChanged","marketboard" : "Marketboard"}
4: {"messageType": "simpleMessage", "messageContent":"Invalid action\n"}
```

sd insertResourcesIntoWarehouse

Client

Server

1 : insertResourcesIntoWarehouse request

2 : On ok - notify GUI changed

3 : On error - notify invalid action

```
1:{"messageType":"insertResourceType","resourceToInsert":"ResourceType","quantityToInsert":"int","intoExtraDeposit":"boolean"}
2: {"messageType": "notifyInsertedOk", "temporaryResourcesConfiguration": HashMap<ResourceType,Integer>,"warehouseConfiguration","Warehouse"}
3: {"messageType": "simpleMessage", "messageContent":"Invalid action, can't insert resources\n"}
```

sd buyDevelopmentCard

Client

Server

1 : buyDevelopmentCard request

2 : On ok - notify card inserted - GUI changed

3 : On ok- notify all GUI changed

4 : On error - notify invalid action

1:{"messageType":"buyDevelopmentCard","level":"int","colour":"Colour","slot":"int", "payUsingExtraDeposit":[int payUsingExtraDeposit1,int payUsingExtraDeposit2]}
2:{"messageType":"notifyDevelopmentCardInsertedOk","developmentCardConfiguration":"developmentCard[3]","warehouseConfiguration":"Warehouse","depositConfiguration":"ExtraDeposit[2]","strongboxConfiguration","HashMap<ResourceType,Integer>"}
3:{"messageType":"notifyDeckgridChanged","deckgrid":"Deckgrid"}
4:{"messageType": "simpleMessage", "messageContent":"You can't buy this card\n"}

sd activateProduction

Client

Server

1 : activateProductionRequest

2 : On ok - notify production successfully activated - GUI changed

3 : Optional- Vatican Report

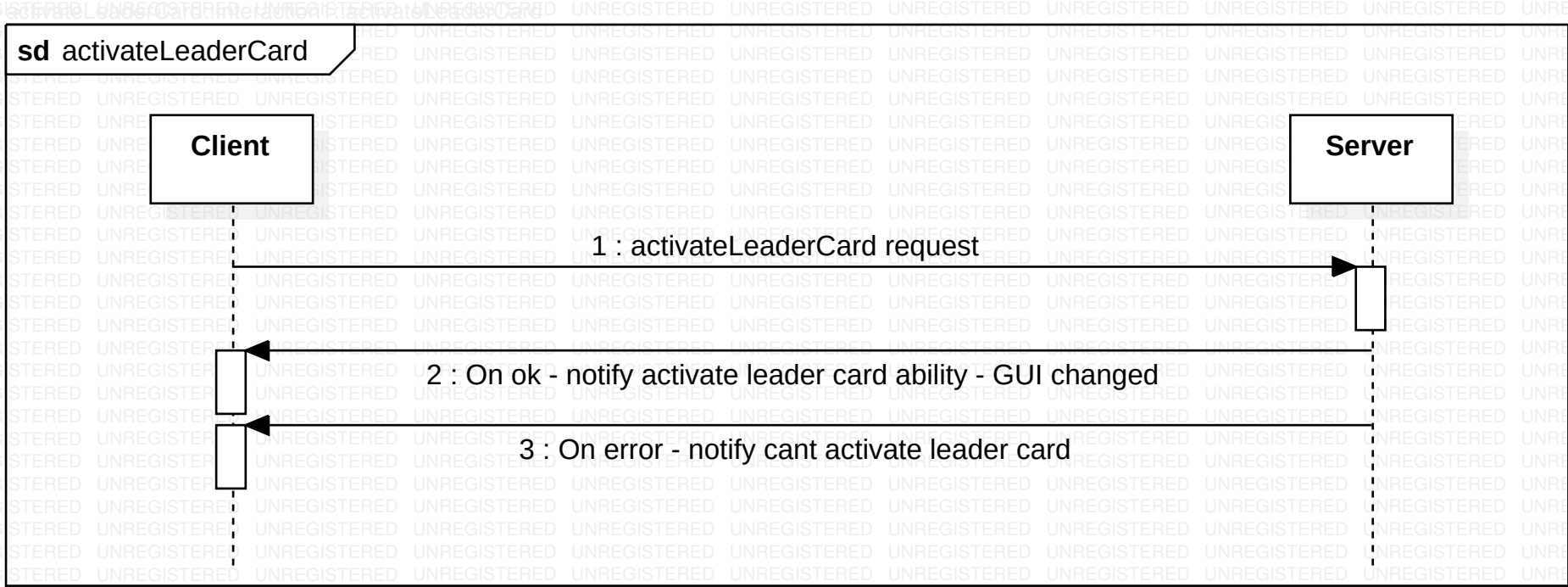
4 : On error - notify cant activate production

Il controller guarderà l'array di faith cards del player per vedere quando viene chiamata la prima, seconda o terza udienza salvando su un array di booleani quando una faith card viene posta a zero (corrispondenza 1 a 1 posizioni faithCards-booleano)

1:{"messageType":"activateProductionMessage","senderNickname":"playerNickname","whichDevCardSlot":"boolean[3]","fromPersonalBoard":"boolean","whichLeaderCard":"boolean[3]","resourceBaseProduction":{"ResourceType[3]","resourceFromLeader":{"ResourceType[2]","payUsingExtraDeposit":{"int[2]}}

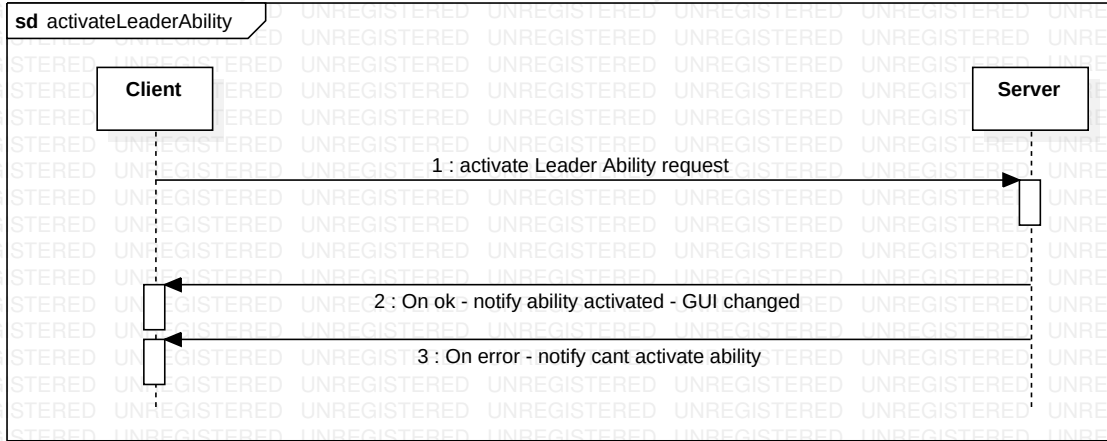
2:{"messageType":"notifyActivateProduction","warehouseConfiguration":"Warehouse","depositConfiguration":"ExtraDeposit[2]","strongboxConfiguration","HashMap<ResourceType,Integer>","newFaithPoints":{"int}}

3:{"messageType":"simpleMessage","messageContent":"Vatican Report occurred!(first, second or third managed by controller)"}
4:{"messageType":"simpleMessage","messageContent":"You can't activate production\n"}



1:{"messageType":"activateLeaderCardMessage","senderNickname":"playerNickname","position","int"}
2:{"messageType":notifyActivateLeaderCard,"chosenLeaderCard":"LeaderCard"[int]}
3:{"messageType": "simpleMessage", "messageContent":"You can't activate leader card\n"}

sd activateLeaderAbility



```
1:{"messageType":"activateLeaderAbilityMessage","senderNickname":"playerNickname","position","int"}
2:
-(case deposit){{"messageType":"notifyActivateLeaderAbilityDeposit","position","int", "extraDepositConfiguration":["extraDeposit1", "extraDeposit2"]}}
-(case discount){{"messageType":"notifyActivateLeaderAbilityDiscount","position","int" }}
-(case whiteTransformation){{"messageType":notifyActivateLeaderAbilityWhiteTransformation","position","int", "temporaryResources": HashMap <ResourceType,Integer>,
remainingWhiteMarbles: int}
-(case production){{"messageType":notifyActivateLeaderAbilityProduction","position","int" }}
3:{"messageType": "simpleMessage", "messageContent":"You can't activate this leader card\n"}
```

sd ActionCardActivation

Client

Server

1 : Activate action card request

2 : On ok- activate action card- GUI changed

3 : On error- action card has been already activated

```
1:{"messageType":"actionCardActivation"}
2:
-(case move){("messageType":"moveLorenzo","newFaithPoints":"int")}
-(case moveAndShuffle){("messageType":"moveAndShuffle","newFaithPoints":"int", "actionCardConfig":"ActionCardStack")}
-(case RemoveDevCard){("messageType":"notifyDeckgridChanged","deckgrid":"Deckgrid")}
3:{"messageType":"simpleMessage","messageContent":"action card has been already activated"}
```