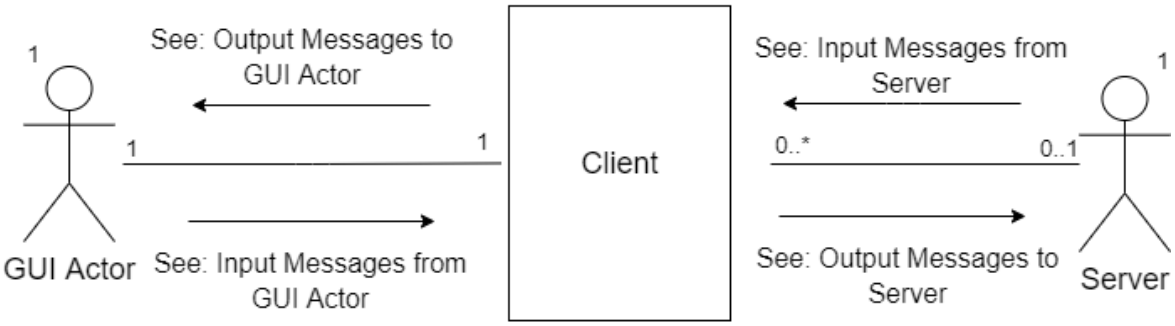
Architectural Decisions

The distributed architecture will be a client/server architecture. There will be two executables, one for the server and one for the client. At run time, there will be multiple client instances and one server instance. The game state, game behaviour, network interface and the main loop (wait for move from player, verify move, apply move to game state, send updated game to clients) will reside in the server executable and the client executables will have the graphical user interface, the network interface. The client uses long polling to receive updates from the server whenever the game state is changed and to send requests to the server corresponding to the move gotten from the GUI (if it is the turn of the local player). The server sends a response to the client only when the game state changes. The executable for the Client will have an “intelligent” GUI, so it will have some local state, will be able to do a verification of correctness of move without contacting the server, based on local information and can inform the user of which moves can be made. The local state of this “intelligent” GUI will be modified accordingly to the new game states that the server sends.

Environment Models

Client diagram:



Input Messages from Server

- readyToPlay()
- regenerateBoard()
- promptRideOn()
- initialPosition(p: Position)
- pickACharacter(options: Set{Bandit})
- currentGameboard(g: Gameboard)
- roundStarted(r: Round)
- schemingStarted()
- schemingEnded()
- stealingStarted()

- stealingEnded()
- roundEnded()
- promptCardToKeep()
- yourTurn()
- promptSchemingAction()
- cardTurned(c : Card)
- promptStealingAction
    - promptRob(options: Set{Loot})
    - promptRideAction(options: Set{Position})
    - promptMove(options: Set{Position})
    - promptMarshal(options: Set{Position})
    - promptShoot(options : Set{Bandit})
    - promptPunch(options : Set{Punchable})
- loginSuccessful()
- loginUnsuccessful()
- listExistingGames(game: Set{Session})
- gameJoined()
- gameDeleted()
- playerRemovedFromSession()
- listSessionPlayers(Set{Player})
- startGame()
- gameSaved()
- listSaveGames(Set{SaveGame})
- gunslingerRecipient(c: Bandit)
- gameWinner(c: Bandit)

<u>Output Messages to Server</u>

- characterPicked(c: Bandit, token: String)
- rideOnChoice(choice: Boolean, token: String)
- cardToKeep(c: Sequence{Card}, token: String)
- SchemingPhaseActions
    - drawCards(token: String)
    - playCard(c: Card, token: String)
    - playWhiskey(w: Whiskey, token: String)
- StealingPhaseActionChoice
    - rob(l: Loot, token: String)
    - rideAction(Position, token: String)
    - move(p: Position, token: String)
    - marshal(p: Position, token: String)
    - shoot(c: Bandit, token: String)
    - punch(c: Punchable, p: Position, l: Loot, token: String)
- login(username: String, password: String)
- requestExistingGames(token: String)
- createNewGame(token: String, username: String, saveGameID: String)
- joinGame(token: String, sessionID: String, user: String)
- deleteGameSession(sessionID: String, token: String)

- leaveGame(sessionID: String, token: String, user: String)
- updateSessionPlayers(sessionID: String, hash: String)
- launchGame(SessionID: String, token: String)
- saveGame(sessionID: String, token: String)
- requestSavedGames(token: String)

Input Messages from GUI Actor

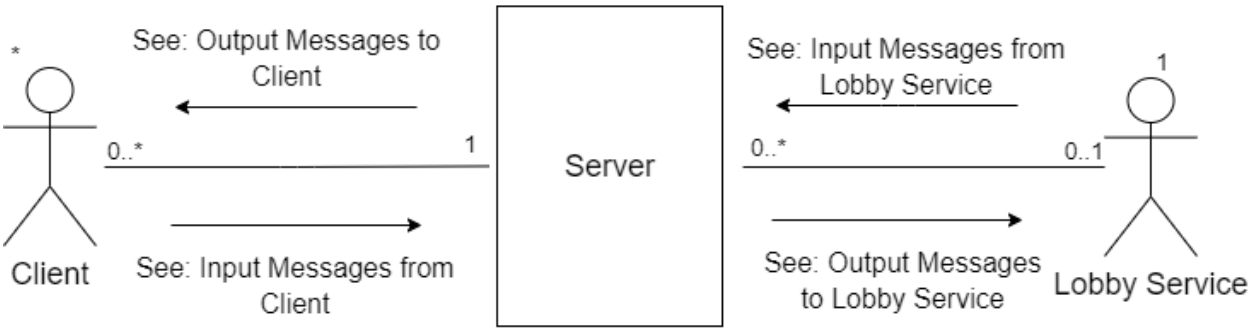
- characterPicked(c: Bandit)
- rideOnChoice(choice: Boolean)
- SchemingPhaseActions
    - drawCards()
    - playCard(c: Card)
    - playWhiskey(w: Whiskey)
- StealingPhaseActionChoice
    - rob(l: Loot)
    - rideAction(Position)
    - move(p: Position)
    - marshal(p: Position)
    - shoot(c: Bandit)
    - punch(c: Punchable, p: Position, l: Loot)
- login(username: String, password: String)
- createNewGame()
- joinGame(sessionID: String)
- deleteGameSession()
- leaveGame()
- launchGame()
- loadGame(saveGameID: String)
- saveGame()
- requestSavedGames()

Output Messages to GUI Actor

- pickACharacter(options: Set{Bandit})
- displayGameboard(g: GameBoard)
- promptRideOn()
- roundStarted(r: Round)
- schemingStarted()
- schemingEnded()
- stealingStarted()
- stealingEnded()
- roundEnded()
- turnType(t: TurnType)
- yourTurn()
- possibleActions(a: Set{Actions})
- promptSchemingAction()
- cardTurned(c : Card)
- promptStealingAction

    - promptRob(options: Set{Loot})
    - promptRideAction(options: Set{Position})
    - promptMove(options: Set{Position})
    - promptMarshal(options: Set{Position})
    - promptShoot(options : Set{Bandit})
    - promptPunch(options : Set{Punchable})
- endOfRoundEvent(e : EndofRoundEvent)
- loginSuccessful()
- loginUnsuccessful()
- listExistingGames(game: Set{Session})
- gameJoined()
- gameDeleted()
- playerRemovedFromSession()
- listSessionPlayers(Set{Player})
- startGame()
- gameSaved()
- listSaveGames(Set{SaveGame})
- gunslingerRecipient(c: Bandit)
- gameWinner(c: Bandit)
- gameOver()

Server diagram:



Input Messages from Client

- characterPicked(c: Bandit, token: String)
- rideOnChoice(choice: Boolean, token: String)
- cardToKeep(c: Sequence{Card}, token: String)
- SchemingPhaseActions
    - drawCards(token: String)
    - playCard(c: Card, token: String)
    - playWhiskey(w: Whiskey, token: String)
- StealingPhaseActionChoice
    - rob(l: Loot, token: String)
    - rideAction(Position, token: String)
    - move(p: Position, token: String)
    - marshal(p: Position, token: String)
    - shoot(c: Bandit, token: String)
    - punch(c: Punchable, p: Position, l: Loot, token: String)

- login(username: String, password: String)
- requestExistingGame()
- createNewGame(token: String, username: String, saveGameID: String)
- joinGame(token: String, sessionID: String, user: String)
- deleteGameSession(sessionID: String, token: String)
- leaveGame(sessionID: String, token: String, user: String)
- updateSessionPlayers(sessionID: String, hash: String)
- launchGame(SessionID: String, token: String)
- saveGame(SessionID: String, token: String)
- requestSavedGames(token: String)

Output Messages to Client

- pickACharacter(options: Set{Bandit})
- currentGameboard(g: GameBoard)
- promptRideOn()
- roundStarted(r: Round)
- schemingStarted()
- schemingEnded()
- stealingStarted()
- stealingEnded()
- roundEnded()
- promptCardToKeep()
- yourTurn()
- promptSchemingAction()
- cardTurned(c : Card)
- promptStealingAction
    - promptRob(options: Set{Loot})
    - promptRideAction(options: Set{Position})
    - promptMove(options: Set{Position})
    - promptMarshal(options: Set{Position})
    - promptShoot(options : Set{Bandit})
    - promptPunch(options : Set{Punchable})
- loginSuccessful()
- loginUnsuccessful()
- listExistingGames(game: Set{Session})
- gameJoined()
- gameDeleted()
- playerRemovedFromSession()
- listSessionPlayers(Set{Player})
- startGame()
- gameSaved()
- listSaveGames(Set{SaveGame})
- gunslingerRecipient(c: Bandit)
- gameWinner(c: Bandit)

Input Messages from Lobby Service

- OAuthToken(token: String)
- gameRegistered()
- nameAlreadyExists()
- loginUnsuccessful()
- listExistingGames(game: Set{Session})
- newGameCreated(sessionID: String)
- gameJoined()
- gameDeleted()
- playerRemovedFromSession()
- listSessionPlayers(Set{Player})
- startGame()
- loadedGameSuccessfully()
- gameSaved()
- listSaveGames(Set{SaveGame})
- userFromTokenAnswer(user: String)
- gameServiceUnregistered()

Output Messages to Lobby Service

- requestOAuthToken(username: String, password: String)
- registerGameService(token: String, location: String, name: String)
- createAccount(token: String, username: String, password: String, color: String)
- requestExistingGame()
- createNewGame(token: String, gameService: String, creator: String, saveGameID: String)
- joinGame(token: String, sessionID: String, user: String)
- deleteGameSession(sessionID: String, token: String)
- removePlayerFromSession(sessionID: String, token: String, user: String)
- updateSessionPlayers(sessionID: String, hash: String)
- launchGame(SessionID: String, token: String)
- saveGame(SessionID: String, token: String, players: Set{Player}, gameService: String)
- requestSavedGames(token: String, gameService: String)
- getUserFromToken(token: String)
- unregisterGameService(token: String)