```gleam
/// Creates the Actor
pub fn start() -> Subject(DirectorActorMessage) {
  let assert Ok(actor) =
    actor.start(DirectorActorState(dict.new()), handle_message)
  actor
}
/// Handles messages from other actors
///
fn handle_message(
  message: DirectorActorMessage,
  state: DirectorActorState,
) -> Next(DirectorActorMessage, DirectorActorState) {
  case message {
    EnqueueParticipant(game_code, player, participant_subject) -> {
      let participant = #(player, participant_subject)
      let new_queue = case state.games_waiting |> get(game_code) {
        Ok(first_participant) -> {
          //They are joining a Game
          game.start([participant, ..first_participant])
          state.games_waiting |> drop([game_code])
        }
        _ -> {
          //They created the game
          state.games_waiting |> insert(game_code, [participant])
        }
      }
      let new_state = DirectorActorState(games_waiting: new_queue)
      new_state |> actor.continue
    }
    DequeueParticipant(game_code) -> {
      let new_queue = state.games_waiting |> drop([game_code])
      let assert Ok(waiting_games) = table.ref("waiting_games")
      waiting_games |> table.delete(game_code)
      let new_state = DirectorActorState(games_waiting: new_queue)
      new_state |> actor.continue
    }
  }
}
```