Instituto Superior de Engenharia de Coimbra

Licenciatura em Engenharia Informática

Arquiteturas Móveis

Leandro Adão Fidalgo | a2017017144

Pedro dos Santos Alves | a2019112789

Renato Rodrigues de Oliveira | a2010011421

Laboratório P1
Trabalho Prático 1

Coimbra, 2 de janeiro de 2022

Índice

1. Introdução

O presente relatório descreve o projeto desenvolvido pelos alunos: Leandro Fidalgo, Pedro Alves e Renato Oliveira, no âmbito da disciplina de Arquiteturas Móveis da Licenciatura em Engenharia Informática do Instituto Superior de Engenharia de Coimbra.

Este trabalho consiste no desenvolvimento de uma versão do jogo do Reversi/Othello, sendo que esta versão contém algumas alterações em relação ao jogo original. Esta versão irá ter vários modos de jogo, nomeadamente offline, dois jogadores e três jogadores, em que nos modos de dois e três jogadores, todos jogarão em dispositivos diferentes (online em rede local ou rede externa). Nesta versão alternativa, também poderão ser feitas dois tipos de jogadas especiais:

- Colocação de uma peça bomba, que irá rebentar todas as peças à sua volta;
- Troca de peças, que consiste em trocar duas peças do jogador atual por uma peça do jogador adversário.

O relatório será dividido em quatro partes:

- Numa primeira parte, serão abordadas as decisões tomadas na implementação do código, relativamente à autenticação (Firebase), à base de dados na cloud (Firestore), à colocação da bomba, configurações do jogo, à comunicação entre servidor- cliente e aos callbacks usados.
- Na segunda parte, será abordado o diagrama da estrutura geral da aplicação, que contém todas as classes do programa e a descrição das classes mais relevantes.
- Numa terceira parte, será abordado o funcionamento entre servidor-cliente para o modo de dois e três jogadores.
- Por fim, numa quarta parte, será abordada a conclusão do trabalho.

O objetivo do presente trabalho é consolidar todos os conhecimentos adquiridos nas aulas teóricas e práticas ao longo de todo o semestre.

2. Decisões tomadas na implementação

2.1. Autenticação (Firebase)

Apesar de não estar explicito no enunciado, para a autenticação do utilizador optou-se por utilizar a autenticação do Firebase. O utilizador efetua a autenticação através de uma conta da Google..

2.2. Base de dados na cloud (Firestore)

Os dados do jogador são alterados e guardados na base de dados da Firebase sendo esta a Firestore, lá é guardada a informação acerca do jogador tal como nome, avatar (imagem do jogador que é guardada em Base64), email (sendo este o ID dos vários documentos e o email da conta Google), e ainda sendo guardadas as cinco melhores pontuações, estas também contêm o avatar do adversário, o seu nome e a pontuação de todos os jogadores que participaram nesse jogo.

2.3. Colocação da bomba

Após a leitura do enunciado, optou-se por definir que a bomba apenas poderia ser jogada num espaço onde esse jogador pudesse colocar uma peça e que nesse mesmo local surgiria uma peça do jogador. Depois de reler o enunciado mais algumas vezes foi possível constatar que a implementação da bomba não seria bem como teria sido pensado inicialmente, porém optou-se por ficar com a ideia inicial pois não só faz mais sentido (não faz sentido acabar o jogo logo na primeira jogada com um empate) como também já tinha sido implementada.

2.4. Configurações do jogo

No enunciado apenas estava explicito que deveria ser possível mostrar as jogadas possíveis ou não. Porém foi ainda implementado uma opção para desativar as peças especiais ou não, uma opção para poder colocar peças especiais sem serem desativadas e ainda uma outra opção que permite passar a próxima jogada automaticamente sem ter que intervir.

2.5. Comunicação entre cliente-servidor

Utilizou-se o protocolo TCP como controlo da transmissão dos dados entre o cliente e o servidor. No enunciado apenas estava explícito que deveria apenas funcionar para redes locais, porém foi implementado de modo a funcionar com redes externas também (caso o utilizador que faz de servidor tenha o porto aberto). Para a comunicação foi utilizado um ObjectInputStream/ObjectOutputStream, enviando sempre Strings em formato JSON.

2.6. Callbacks

Para manter a consistência entre os dados e a relação entre logica e interface, foram utilizados callbacks. Deste modo é possivel facilitar a atualização dos diversos passos durante o jogo. Por exemplo quando um utilizador coloca uma peça no tabuleiro do jogo podem ser chamados vários callbacks (onPlacePiece, onPlayerScoreUpdated, onNextPlayer, etc..) para atualizar a vista.

3. Estrutura geral da aplicação

3. 1. Descrição das classes

Como é possível verificar no diagrama de classes (Anexo 1), é bastante complexo, porém está bastante organizado e prevenindo assim erros no desenvolver da aplicação.

O ReversiGame é responsável por controlar o estado do tabuleiro e dos jogadores, o NetworkReversiGame é uma extensão do ReversiGame e contém o socket para enviar os dados para o servidor, porém está ainda responsável por iniciar o GameServer, iniciar o cliente e tratar de todos os comandos provenientes do servidor.

O GameServer é responsável por enviar os comandos para os clientes e tratar dos comandos provenientes dos jogadores para o servidor. Também é responsável pelo processamento do jogo no contexto online, pois é ele que trata da gestão do jogo online e envia para os seus clientes as alterações que existiram no jogo.

O Player é responsável por conter os dados sobre o jogador, a peça do jogador, a sua pontuação e ainda se pode colocar alguma bomba ou se pode trocar de peças com o adversário. Também contém a informação do jogador relativamente ao Firestore, username, top scores e avatar.

3. 2. Descrição das atividades

Relativamente às atividades da aplicação, foram criadas 6 atividades, todas elas com a opção de *portrait* e *landscape*:

- **MainActivity** – Atividade principal, responsável por tratar do registo/autenticação com uma conta Google.
- **ProfileActivity** – Atividade que mostra a informação do utilizador, nome, email, foto e as cinco melhores pontuações.
- **EditProfileActivity** – Atividade responsável pela edição da informação do utilizador, como carregar uma nova foto de perfil ou mudar o seu nome. Após guardar as alterações esta informação será atualizada na Firestore.
- **GameSetttingsActivity** – Atividade responsável por mostrar e definir as definições do jogo, quer seja offline como online.
- **WaitingRoomActivity** – Atividade que funciona como "sala de espera". É onde os utilizadores ficam à espera de que outros utilizadores se liguem antes de iniciar o jogo. É aqui que são atribuídos os jogadores ao modelo e se dá início ao jogo em si.
- **GameActivity** – Atividade onde o jogo irá decorrer. Esta é responsável por criar dinamicamente as vistas dos jogadores, o tabuleiro do jogo e também onde os callbacks são executados e os dados no modelo são atualizados.

3. 3. Relação entre servidor-cliente

A implementação do modo online (Anexo3) é idêntica á do modo offline (Anexo2), porém no modo online estão a ser enviadas constantemente mensagens para o servidor via TCP que depois serão tratadas pelo mesmo e é reenviado a resposta correspondente. Ambos os clientes são tratados da mesma forma mesmo um deles sendo o servidor.

Para existir comunicação entre o servidor e o cliente foi necessário criar um protocolo daquilo que será enviado, esse protocolo contém diversos comandos (como por exemplo: “Connect”, “Disconnect”, “ShowPlaceablePieces”) sendo que este comando é sempre constituído pelo seu nome que será o seu identificador para saber o que irá executar. Também poderá ser constituído por um payload, sendo que este é o corpo da mensagem com a informação relevante para esse comando, como é possível verificar no Anexo4.

4. Funcionalidades

Funcionalidades	Implementado	Parcialmente	Não implementado
Interface e interação com o utilizador	**X**		
Modo de jogo 1 (incluindo regras)	**X**		
Modo de jogo 2	**X**		
Modo de jogo 3	**X**		
Definição de perfil de jogador (com fotografia) e partilha do mesmo	**X**		
Gestão e listagem de histórico de resultados	**X**		
Suporte a diferentes línguas (min. PT e EN) e créditos (“about”)	**X**		
Suporte para diferentes orientações de ecrã (com layouts independentes)	**X**		
Robustez e qualidade de código (inclui tratamento de erros e exceções)	**X**		
Manual de utilizador	**X**		

5. Conclusão

A proposta do trabalho prático surgiu com o intuito de consolidar os conhecimentos de java para dispositivos móveis (Android) adquiridos ao longo do semestre na cadeira de Arquiteturas Móveis.

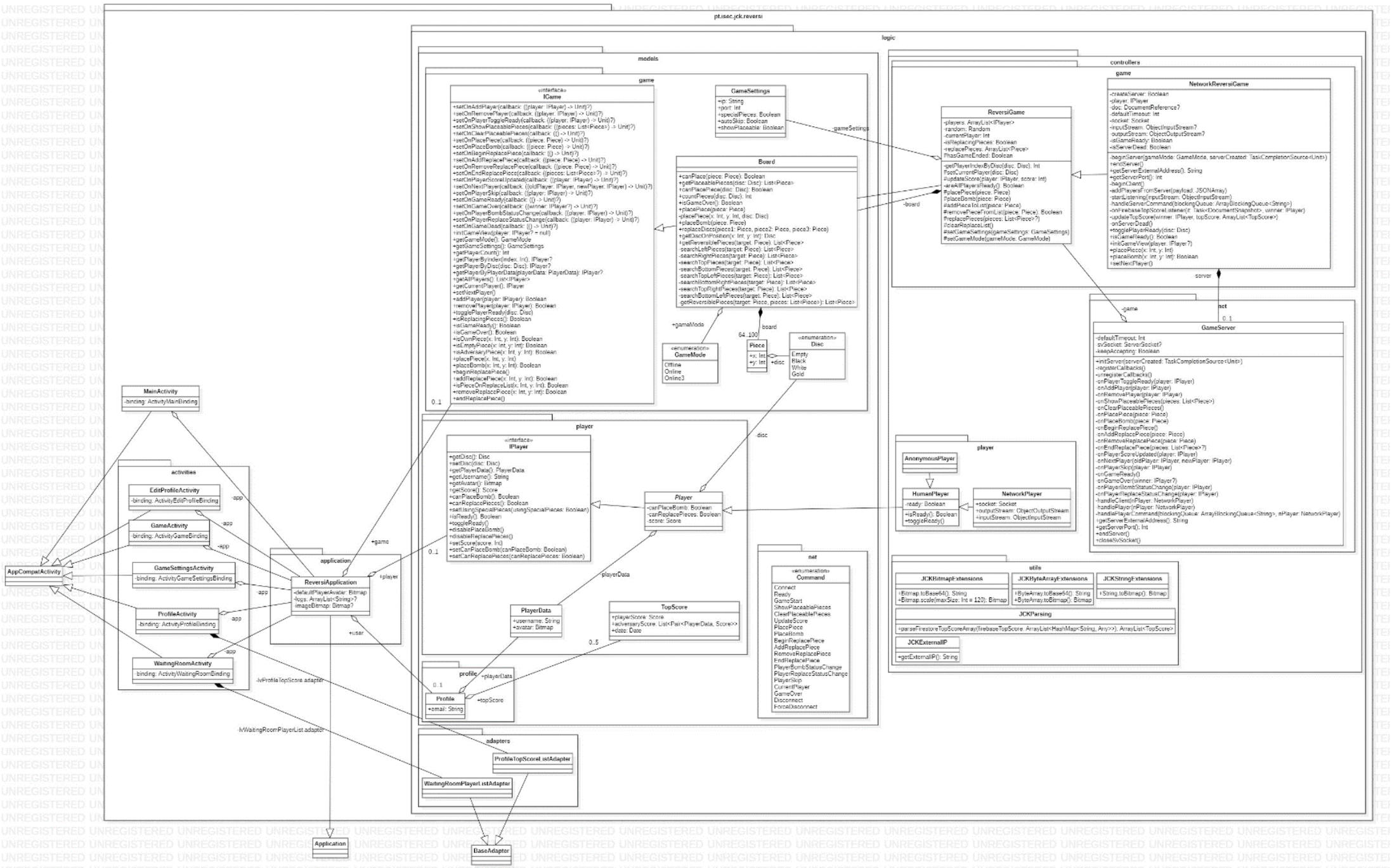
Durante o desenvolvimento deste trabalho, apesar de tudo o que foi pedido no enunciado ter sido implementado, foram surgindo dúvidas e problemas que foram superados com a ajuda dos professores da disciplina, os apontamentos por eles disponibilizados e da Internet.

Com a implementação do trabalho, pôde-se perceber que, para o desenvolvimento de um jogo para smartphone existe uma clara dependência entre os programadores e os designers gráficos para obter um bom resultado final.

O produto final desenvolvido possui elementos básicos de um jogo para smartphone e procura seguir uma linha de jogos simples.

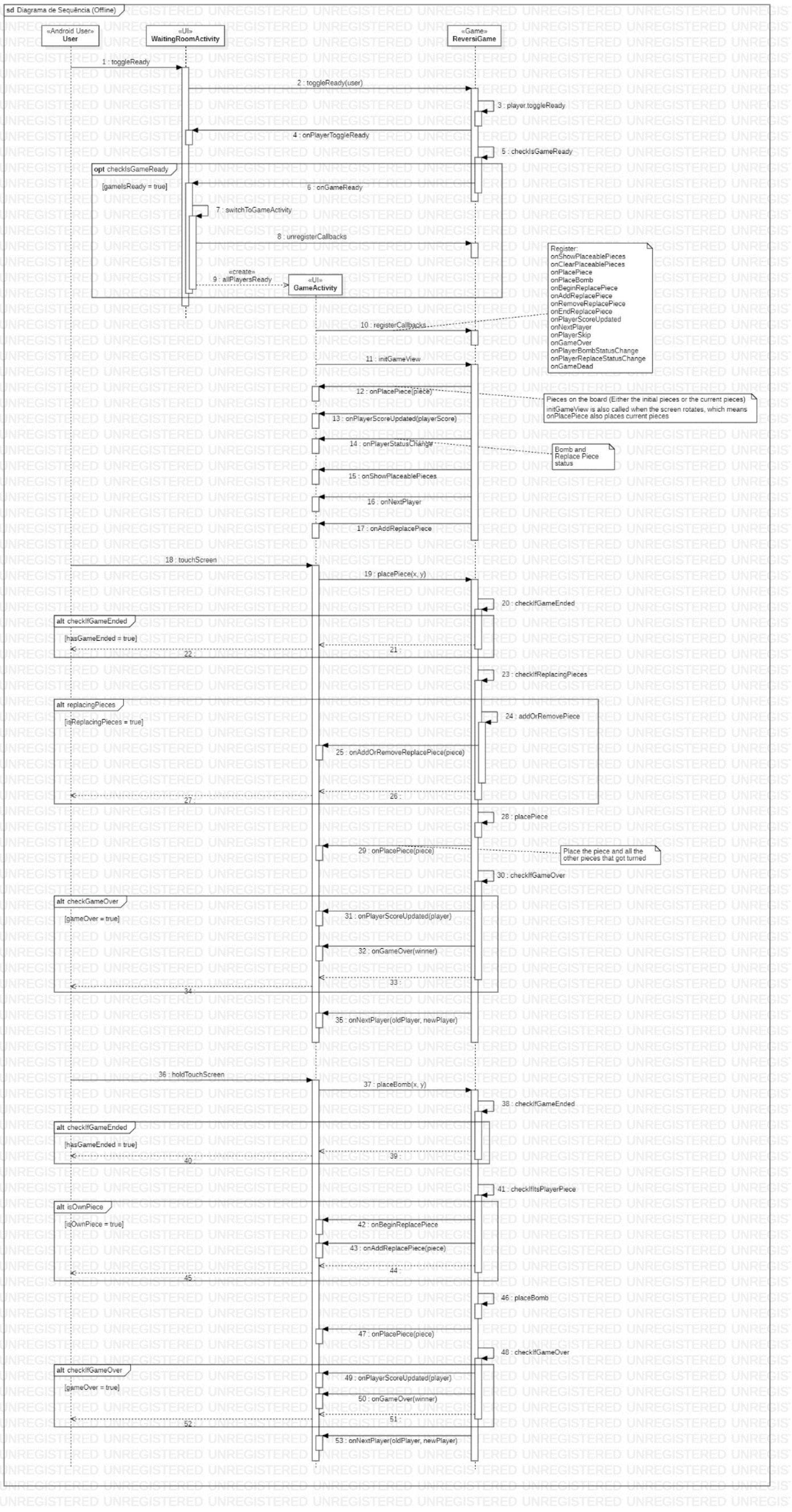
Pode concluir-se que o objetivo proposto de desenvolver o jogo Reversi/Othello para Android foi bastante enriquecedor. Pode verificar-se como funciona tudo por de trás de um smatphone e como pode ser bastante útil no futuro programar com a linguagem Kotlin, por ser uma linguagem de programação bastante utilizada quando se desenvolve para android. Tal feito oferece a possibilidade aos alunos de se especializarem cada vez mais no assunto, dando continuidade a este trabalho ou amadurecendo ideias de novos jogos. Os conceitos aprendidos com este projeto podem ser aplicados continuamente em diversos tipos de jogos para Android.

Anexo 1

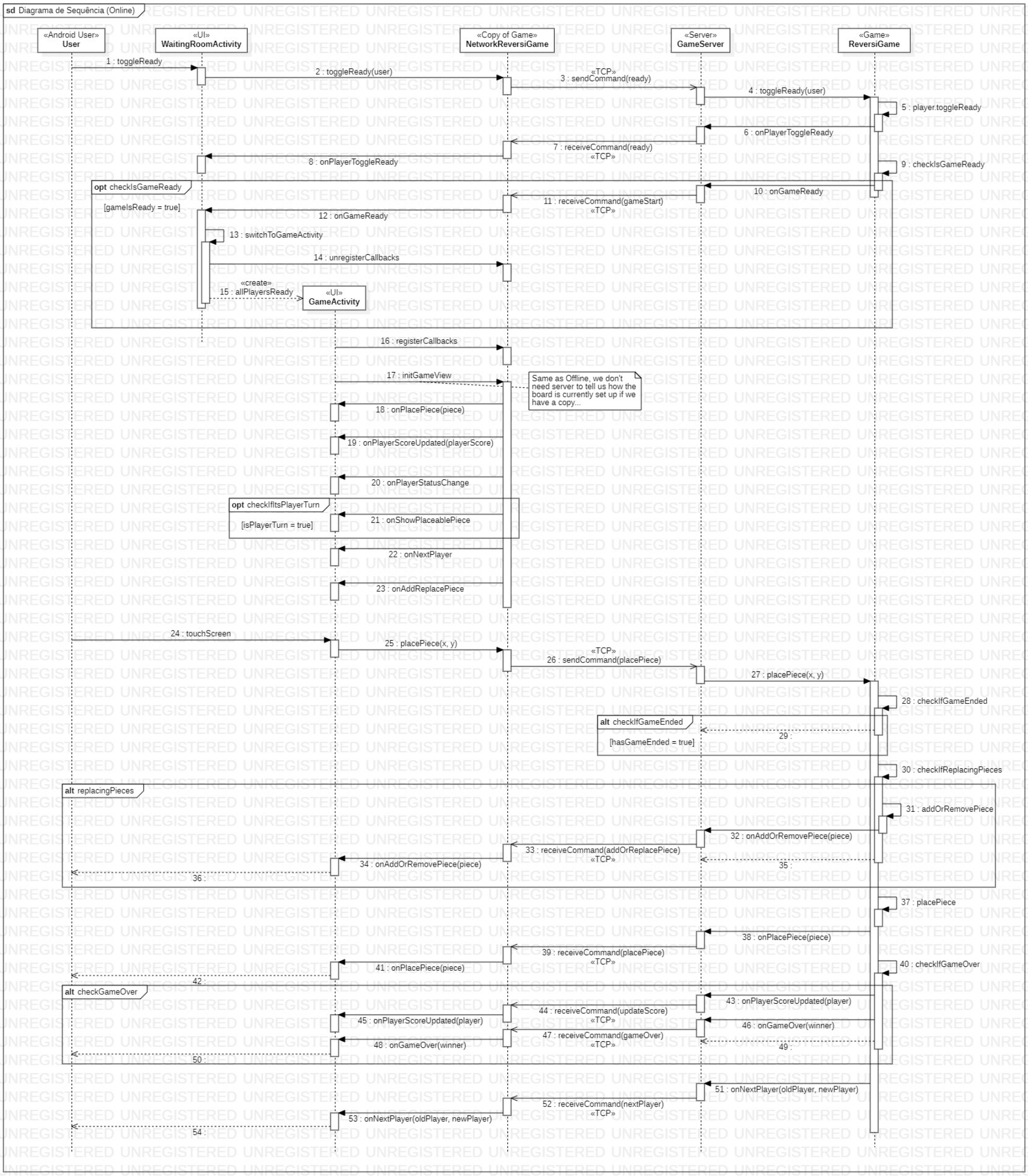

pt.isec.jck.reversi
logic
models
game
«interface»
IGame
GameSettings
Board
«enumeration»
GameMode
«enumeration»
Disc
Piece
player
«interface»
IPlayer
Player
PlayerData
TopScore
net
«enumeration»
Command
controllers
game
ReversiGame
NetworkReversiGame
GameServer
player
AnonymousPlayer
HumanPlayer
NetworkPlayer
utils
JCKBitmapExtensions
JCKByteArrayExtensions
JCKStringExtensions
JCKParsing
JCKExternalIP
MainActivity
activities
EditProfileActivity
GameActivity
GameSettingsActivity
ProfileActivity
WaitingRoomActivity
AppCompatActivity
application
ReversiApplication
profile
Profile
adapters
ProfileTopScoreListAdapter
WaitingRoomPlayerListAdapter
Application
BaseAdapter

I

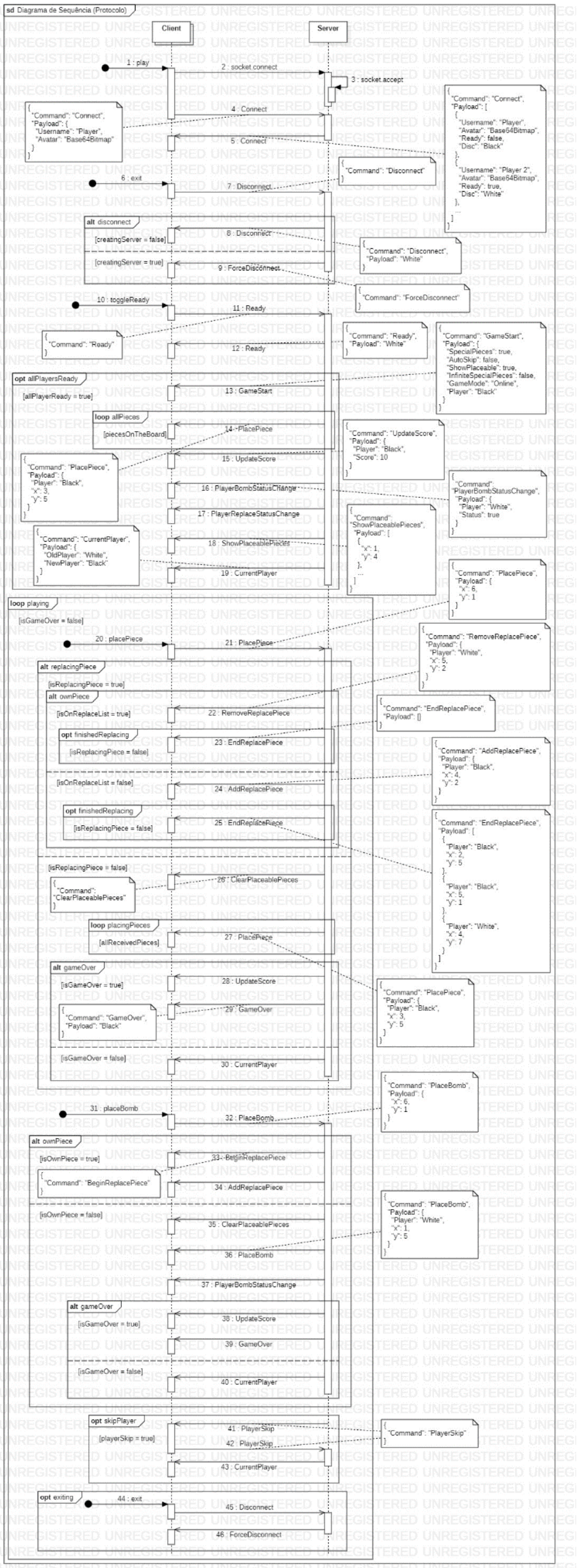
Anexo 2


sd Diagrama de Sequência (Offline)
«Android User» User
«UI» WaitingRoomActivity
«Game» ReversiGame
1 : toggleReady
2 : toggleReady(user)
3 : player.toggleReady
4 : onPlayerToggleReady
5 : checkIsGameReady
opt checkIsGameReady
[gameIsReady = true]
6 : onGameReady
7 : switchToGameActivity
8 : unregisterCallbacks
«create»
9 : allPlayersReady
«UI» GameActivity
Register:
onShowPlaceablePieces
onClearPlaceablePieces
onPlacePiece
onPlaceBomb
onBeginReplacePiece
onAddReplacePiece
onRemoveReplacePiece
onEndReplacePiece
onPlayerScoreUpdated
onNextPlayer
onPlayerSkip
onGameOver
onPlayerBombStatusChange
onPlayerReplaceStatusChange
onGameDead
10 : registerCallbacks
11 : initGameView
12 : onPlacePiece(piece)
Pieces on the board (Either the initial pieces or the current pieces)
initGameView is also called when the screen rotates, which means onPlacePiece also places current pieces
13 : onPlayerScoreUpdated(playerScore)
14 : onPlayerStatusChange
Bomb and Replace Piece status
15 : onShowPlaceablePieces
16 : onNextPlayer
17 : onAddReplacePiece
18 : touchScreen
19 : placePiece(x, y)
20 : checkIfGameEnded
alt checkIfGameEnded
[hasGameEnded = true]
22 :
21 :
23 : checkIfReplacingPieces
alt replacingPieces
[isReplacingPieces = true]
24 : addOrRemovePiece
25 : onAddOrRemoveReplacePiece(piece)
27 :
26 :
28 : placePiece
29 : onPlacePiece(piece)
Place the piece and all the other pieces that got turned
30 : checkIfGameOver
alt checkGameOver
[gameOver = true]
31 : onPlayerScoreUpdated(player)
32 : onGameOver(winner)
34 :
33 :
35 : onNextPlayer(oldPlayer, newPlayer)
36 : holdTouchScreen
37 : placeBomb(x, y)
38 : checkIfGameEnded
alt checkIfGameEnded
[hasGameEnded = true]
40 :
39 :
41 : checkIfItsPlayerPiece
alt isOwnPiece
[isOwnPiece = true]
42 : onBeginReplacePiece
43 : onAddReplacePiece(piece)
45 :
44 :
46 : placeBomb
47 : onPlacePiece(piece)
48 : checkIfGameOver
alt checkIfGameOver
[gameOver = true]
49 : onPlayerScoreUpdated(player)
50 : onGameOver(winner)
52 :
51 :
53 : onNextPlayer(oldPlayer, newPlayer)

Anexo 3


sd Diagrama de Sequência (Online)
«Android User» User
«UI» WaitingRoomActivity
«Copy of Game» NetworkReversiGame
«Server» GameServer
«Game» ReversiGame
1 : toggleReady
2 : toggleReady(user)
«TCP» 3 : sendCommand(ready)
4 : toggleReady(user)
5 : player.toggleReady
6 : onPlayerToggleReady
7 : receiveCommand(ready) «TCP»
8 : onPlayerToggleReady
9 : checkIsGameReady
opt checkIsGameReady
[gameIsReady = true]
10 : onGameReady
11 : receiveCommand(gameStart) «TCP»
12 : onGameReady
13 : switchToGameActivity
14 : unregisterCallbacks
«create» 15 : allPlayersReady
«UI» GameActivity
16 : registerCallbacks
17 : initGameView
Same as Offline, we don't need server to tell us how the board is currently set up if we have a copy...
18 : onPlacePiece(piece)
19 : onPlayerScoreUpdated(playerScore)
20 : onPlayerStatusChange
opt checkIfItsPlayerTurn
[isPlayerTurn = true]
21 : onShowPlaceablePiece
22 : onNextPlayer
23 : onAddReplacePiece
24 : touchScreen
25 : placePiece(x, y)
«TCP» 26 : sendCommand(placePiece)
27 : placePiece(x, y)
28 : checkIfGameEnded
alt checkIfGameEnded
[hasGameEnded = true]
29 :
30 : checkIfReplacingPieces
alt replacingPieces
31 : addOrRemovePiece
32 : onAddOrRemovePiece(piece)
33 : receiveCommand(addOrReplacePiece) «TCP»
34 : onAddOrRemovePiece(piece)
35 :
36 :
37 : placePiece
38 : onPlacePiece(piece)
39 : receiveCommand(placePiece) «TCP»
40 : checkIfGameOver
41 : onPlacePiece(piece)
42 :
alt checkGameOver
43 : onPlayerScoreUpdated(player)
44 : receiveCommand(updateScore) «TCP»
45 : onPlayerScoreUpdated(player)
46 : onGameOver(winner)
47 : receiveCommand(gameOver) «TCP»
48 : onGameOver(winner)
49 :
50 :
51 : onNextPlayer(oldPlayer, newPlayer)
52 : receiveCommand(nextPlayer) «TCP»
53 : onNextPlayer(oldPlayer, newPlayer)
54 :

Anexo 4


sd Diagrama de Sequência (Protocolo)
Client
Server
1 : play
2 : socket.connect
3 : socket.accept
4 : Connect
5 : Connect
6 : exit
7 : Disconnect
alt disconnect
[creatingServer = false]
8 : Disconnect
[creatingServer = true]
9 : ForceDisconnect
10 : toggleReady
11 : Ready
12 : Ready
opt allPlayersReady
[allPlayerReady = true]
13 : GameStart
loop allPieces
[piecesOnTheBoard]
14 : PlacePiece
15 : UpdateScore
16 : PlayerBombStatusChange
17 : PlayerReplaceStatusChange
18 : ShowPlaceablePieces
19 : CurrentPlayer
loop playing
[isGameOver = false]
20 : placePiece
21 : PlacePiece
alt replacingPiece
[isReplacingPiece = true]
alt ownPiece
[isOnReplaceList = true]
22 : RemoveReplacePiece
opt finishedReplacing
[isReplacingPiece = false]
23 : EndReplacePiece
[isOnReplaceList = false]
24 : AddReplacePiece
opt finishedReplacing
[isReplacingPiece = false]
25 : EndReplacePiece
[isReplacingPiece = false]
26 : ClearPlaceablePieces
loop placingPieces
[allReceivedPieces]
27 : PlacePiece
alt gameOver
[isGameOver = true]
28 : UpdateScore
29 : GameOver
[isGameOver = false]
30 : CurrentPlayer
31 : placeBomb
32 : PlaceBomb
alt ownPiece
[isOwnPiece = true]
33 : BeginReplacePiece
34 : AddReplacePiece
[isOwnPiece = false]
35 : ClearPlaceablePieces
36 : PlaceBomb
37 : PlayerBombStatusChange
alt gameOver
[isGameOver = true]
38 : UpdateScore
39 : GameOver
[isGameOver = false]
40 : CurrentPlayer
opt skipPlayer
[playerSkip = true]
41 : PlayerSkip
42 : PlayerSkip
43 : CurrentPlayer
opt exiting
44 : exit
45 : Disconnect
46 : ForceDisconnect
{ "Command": "Connect", "Payload": { "Username": "Player", "Avatar": "Base64Bitmap" } }
{ "Command": "Connect", "Payload": [{ "Username": "Player", "Avatar": "Base64Bitmap", "Ready": false, "Disc": "Black" }, { "Username": "Player 2", "Avatar": "Base64Bitmap", "Ready": true, "Disc": "White" }, ...] }
{ "Command": "Disconnect" }
{ "Command": "Disconnect", "Payload": "White" }
{ "Command": "ForceDisconnect" }
{ "Command": "Ready" }
{ "Command": "Ready", "Payload": "White" }
{ "Command": "GameStart", "Payload": { "SpecialPieces": true, "AutoSkip": false, "ShowPlaceable": true, "InfiniteSpecialPieces": false, "GameMode": "Online", "Player": "Black" } }
{ "Command": "PlacePiece", "Payload": { "Player": "Black", "x": 3, "y": 5 } }
{ "Command": "UpdateScore", "Payload": { "Player": "Black", "Score": 10 } }
{ "Command": "PlayerBombStatusChange", "Payload": { "Player": "White", "Status": true } }
{ "Command": "CurrentPlayer", "Payload": { "OldPlayer": "White", "NewPlayer": "Black" } }
{ "Command": "ShowPlaceablePieces", "Payload": [{ "x": 1, "y": 4 }, ...] }
{ "Command": "PlacePiece", "Payload": { "x": 6, "y": 1 } }
{ "Command": "RemoveReplacePiece", "Payload": { "Player": "White", "x": 5, "y": 2 } }
{ "Command": "EndReplacePiece", "Payload": [] }
{ "Command": "AddReplacePiece", "Payload": { "Player": "Black", "x": 4, "y": 2 } }
{ "Command": "EndReplacePiece", "Payload": [{ "Player": "Black", "x": 2, "y": 5 }, { "Player": "Black", "x": 5, "y": 1 }, { "Player": "White", "x": 4, "y": 7 }] }
{ "Command": "ClearPlaceablePieces" }
{ "Command": "PlacePiece", "Payload": { "Player": "Black", "x": 3, "y": 5 } }
{ "Command": "GameOver", "Payload": "Black" }
{ "Command": "PlaceBomb", "Payload": { "x": 6, "y": 1 } }
{ "Command": "BeginReplacePiece" }
{ "Command": "PlaceBomb", "Payload": { "Player": "White", "x": 1, "y": 5 } }
{ "Command": "PlayerSkip" }