Reversi

Relatório Técnico

Ângelo Paiva, 2019129023
Jan Frank, 2017009793
Pedro Henriques, 2019129770

Arquiteturas Móveis
Licenciatura de Engenharia Informática, Ramo de Desenvolvimento de Aplicações
1 de janeiro de 2022

Conteúdo

1 Introdução

Neste trabalho foi implementada uma versão do clássico jogo reversi para Android. Esta aplicação foi escrita em Kotlin com recurso ao Android Studio.

Este relatório irá servir para explicar a implementação e a razão por trás das decisões que foram tomadas.

2 Manual de Utilizador

Este relatório técnico é também acompanhado por um manual de utilizador. Como manda a tradição, este foi escrito em **markdown**, tendo sido usada uma ferramenta online para converter **markdown** num **ficheiro PDF** que copia o aspeto que estes costumam ter no **GitHub**.

Neste manual de utilizador estão descritas as regras de jogo, os diferentes modos que se podem jogar e como se pode editar o perfil pessoal!

3 Guardar Dados Localmente

Todos os dados desta aplicação são guardados no diretório privado da mesma.

Para além da imagem, todas as outras informações são guardadas num ficheiro JSON, achamos que não faz sentido guardar informação em ficheiros binários ou de difícil acesso.

4 Captura de Imagem para Avatar

Era pedido que houvesse a opção de o jogador ter uma imagem associada, mas a única maneira de associar uma imagem seria capturá-la na hora. Ao nível da aplicação, esta captura teria que ser realizada pela própria aplicação, não podendo recorrer a aplicações terceiras.

Assim, respeitando a recomendação feita na documentação oficial, foi usada a biblioteca **CameraX** para a captura da imagem.

Capturamos uma imagem com rácio 1:1 e resolução 1000x1000, o rácio achamos que um avatar deveria ser quadrado e a resolução achamos que chega e sobra para um simples avatar.

Esta imagem, como referido anteriormente, é guardada no diretório privado da aplicação.

5 Comunicação Servidores - Clientes

Toda a comunicação feita entre dispositivos é feita com recurso a ficheiros JSON. Para garantir que a informação é sempre mandada corretamente e que, em caso de necessidade, se possam refatorizar propriedades, foi criado o **object JSONHelper** e, dentro deste, o **object JSONFields**.

O primeiro contém todos os possíveis valores que as respetivas propriedades podem ter. O segundo contém o nome das propriedades.

```
object JSONHelper {
   // Type of JSON
   const val PLAYER_INFO = "PLAYER_INFO"
   const val GAME_INFO = "GAME_INFO"
   const val MOVE = "MOVE"

   // Types of Piece
   const val NORMAL_PIECE = "normalPiece"
   const val BOMB_PIECE = "bombPiece"
   const val SWAP_PIECE = "swapPiece"
   const val SKIP_PIECE = "skip"

   // State of Game
   const val GAME_STATE_PLAYING = "gameStatePlaying"
   const val GAME_STATE_FINISHED = "gameStateFinished"

   object JSONFields {
      const val TYPE_NAME = "type" // String

      const val GAME_STATE_NAME = "gameState" // String

      const val PLAYER_NAME_NAME = "playerName" // String
      const val AVATAR_NAME = "avatar" // Bitmap Encoded to String
      const val SCORE_NAME = "score" // Int

      const val TYPE_PIECE_NAME = "typePiece" // String
      const val POSITION_NAME = "position" // Int
      const val SWAP_POSITIONS_ARRAY_NAME = "swapPositionsArray" // Array of Int

      const val PLAYERS_ARRAY_NAME = "playersArray" // Array of Player
      const val CURRENT_PLAYER_NAME = "currentPlayer" // Int (position in
          playersArray)
      const val BOARD_NAME = "board" // Array of Int
   }
}
```

Código 1: object JSONHelper

Para o envio dos avatares transformou-se o *bitmap* numa string **codificada em Base64**, tirando partido do **Base64.URL_SAFE** para poder enviar pelo **JSON** sem qualquer tipo de problemas e incompatibilidades.

Durante o desenvolvimento também notámos que as classes **JSON** que vêm por defeito no **Kotlin** têm algumas interações peculiares. Por exemplo, se tentarmos inserir um **JSONObject** num **JSONArray**, e for chamado o **.toString()** desse objeto, na verdade será introduzida uma string cheia de *escape characters*, impossibilitando depois o uso e manipulação do mesmo.

6 Conclusão

Para além de ter sido um verdadeiro desafio, este trabalho foi uma excelente oportunidade para conhecer melhor **Kotlin** e todas as vantagens e *syntactic-sugar* que nos proporciona.

7 Anexos

Lista de Figuras

Pedaços de Código