DOCUMENTATIA PROIECTULUI

Mix and Match

STUDENTI

Constantinescu David

Georgescu Alexandru-Anand

1. Prezentarea proiectului

In acest proiect ne-am propus sa implementam cunoscutul joc “Mix and Match” in care se genereaza un numar de perechi de carti, cu fata in jos, jucatorul trebuind sa asocieze cartile de acelasi tip prin mai multe incercari. In functie de modul de joc selectat, jucatorul va trebui sa descopere 4, 9 sau 12 perechi de carti. Initial toate cartile sunt cu fata in jos, iar utilizatorul are voie sa aleaga, pe rand cate 2 carti. Daca acestea coincid vor ramane cu fata in sus, altfel, se vor intoarce inapoi pe spate. Jucatorul trebuie sa isi aminteasca pe ce pozitie se afla anumite carti de care are nevoie pentru a asocia fiecare carte cu perechea sa. Am optat pentru implementarea acestui joc in Android deoarece este un exercitiu interesant si interactiv prin care se poate antrena memoria de scurta durata.

1. Tehnologii folosite

Aplicatia este realizate in Android Studio 4.2.1 folosind Gradle versiunea 6.7.1. In proiectul prezentat am implementat mai multe activitati si fragmente, un RecycleView, dar si un serviciu pentru notificari asociat unui Broadcast Receiver. Am folosit iconitele default din Android pentru a crea perechile necesare jocului, care sunt automat amestecate si aranjate pe ecran. Am folosit snackbar-uri pentru About si pentru diverse mesaje de alerta in timpul jocului.

Pentru finalul jocului am dorit sa avem o animatie deosebita, si am adaugat efecte de confetti prin intermediul unei librarii independente:

<https://github.com/jinatonic/confetti>

1. Arhitectura aplicatiei si partea de backend

Aplicatia implementeaza 3 activitati principale, Menu, Main si Create. Cea Menu deserveste drept meniu al aplicatiei, activitatea Main reprezinta jocul in sine, cu toata functionalitatea sa, inclusiv cele trei moduri de dificultate, iar cea Create ofera posibilitatea crearii unui joc custom. De asemenea, am implementat un serviciu de notificari, ce se foloseste de un Serviciu si un BroadcastReceiver.

Menu Activity este activitatea desemnata sa porneasca prima si serveste drept meniu, avand in componenta doar doua butoane: Play si About. Butonul Play porneste jocul (MainActivity) iar cel de About afiseaza intr-un snackbar cateva informatii despre aplicatie.

In Main Activity am creat un RecycleView in care vom stoca imaginile. Marimea acestora, cat si pozitionarea lor pe ecran se va face automat in functie numarul de perechi, numarul de coloane pe care afisam imaginile si inaltimea ecranului. Pe baza acestor valori, cat si a adaptorului “GameBoardAdapter” se va construi un grid in celulele caruia se vor genera imaginile aleator prin metoda **DEFAULT\_ICONS.shuffled().take** ce primeste ca parametru numarul de perechi. Pentru a tine evidenta jocului am creat 2 CardViews, unul pentru numarul de miscari efectuate si celalalt pentru numarul de perechi gasite. In functie de cat de aproape suntem de terminarea jocului, CardView-ul pentru perechi se coloreaza gradual de la rosu la verde pentru a ilustra progresul jucatorului, acest proces numindu-se interpolare. Orice pereche gasita va ramane intoarsa cu fata in sus, iar pentru a le face mai usor de ignorat (deoarece jucatorul trebuie sa se concentreze pe cartile ramase neintoarse) am decis sa adaugam un layout gri deasupra lor prin metoda **ViewCompat.setBackgroundTintList.**

Pentru cartile cu care se va desfasura jocul am creat o clasa “MemoryCard” cu 3 atribute:

* Un identificator unic de tip Int
* O variabila de tip bool care sa arate daca acea carte este intoarsa sau nu
* O variabila bool care indica daca a fost gasita perechea cartii respective

In clasa “Game” retinem starea jocului, cat si anumite metode ce tin de actiunea jocului: intoarcerea unei imagini, resetarea jocului , finalul jocului, etc.

Enum-ul “BoardSize” este folosit pentru a retine informatiile generice despre joc in functie de dificultatea lui , cum ar fi numarul de perechi, numarul de coloane, inaltimea ecranului.

Conversia de la imaginile generate la obiectele de tip “MemoryCard” se face prin **cards = randomizedImages.map { MemoryCard(it) }** , unde it = identificator. Nu este necesar sa atribuim valori celorlalti membrii deoarece sunt initializati cu false din constructor.

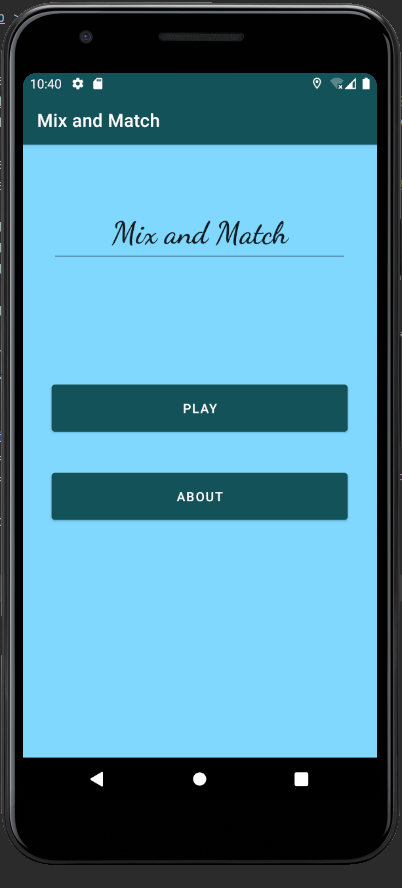
Pentru a verifica functionalitatea jocului am creat un Log pentru pozitia cartii, identificatorul ei si numarul de perechi gasite prin metoda **Log.i(TAG,message)**. Acestea sunt apelate de fiecare data cand se da click pe o carte.

Tot in Main Activity am tratat diverse mici exceptii, cat si mesajele de eroare. In partea de sus a acestei activitati avem un meniu de unde putem reseta jocul sau se poate schimba dificultatea jocului, utilizatorul avand de ales una din cele trei optiuni (Easy, Medium, Hard).

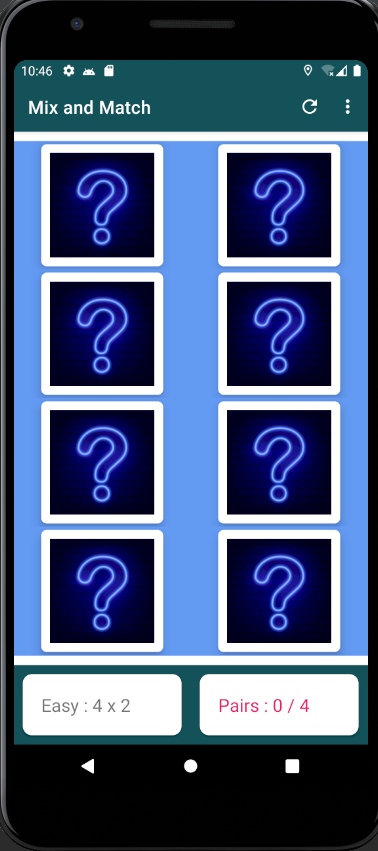
Pentru trimiterea de notificari am implementat serviciul NotificationService de tip IntentService, impreuna cu un BroadcastReceiver – MyReceiver si doua clase : NotificationUtils si ResultActivity. La pornirea aplicatiei, in activitatea de meniu se apeleaza si functia **NotificationUtils().setNotification(auxTime, this@MenuActivity)** care va porni NotificationService prin intermediul functiei **OnReceive** din MyReceiver. In NotificationService se va crea un canal pentru a putea trimite notificari sistemului, iar apoi se va construi o notificare care va fi ulterior afisata dupa 7.5 secunde dupa ce jocul a fost pornit.

1. Utilizarea aplicatiei

Cand utilizatorul deschide aplicatia, intalneste meniul de start ce are 2 optiuni, inceperea unui nou joc sau vizualizarea sectiunii de About.

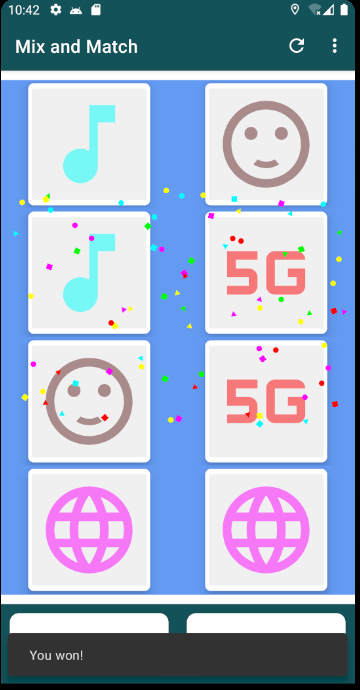


Odata inceput jocul, in functie de dificultatea jocului ce a fost aleasa, se genereaza aleator un set de perechi, initial toate cu fata in jos.



In partea de jos a ecranului se poate urmari progresul jucatorului.Dupa ce au fost descoperite toate perechile, utilizatorul este anuntat ca s-a

termanat jocul, aparand animatiile de final.



In partea de sus a ecranului avem meniul cu optiunile restart game, choose new size and create custom game.



