Analisi Progetto

sudoku by Daloma, Mobile Programming 2020

Specifiche

- Il progetto deve sviluppare una interfaccia per il gioco del Sudoku.
- L' App dovrà prevedere:
    - Un generatore di schemi (vedi librerie su internet)
    - Una interfaccia utente
    - Una fase di gioco
    - Un database dei risultati precedenti
        - Partite giocate
        - Partite vinte (terminate)
        - Miglior Tempo
- L'interfaccia dovrà prevedere il massimo degli aiuti possibili all'utente:
    - Suggerisci mossa (intelligente)
    - Appunti utente per ognuna delle 81 celle
    - Fare riferimento ad app già esistenti
    - Gli schemi presentati all'utente dovranno essere
        - Automaticamente generati da un motore API
        - https://sugoku.herokuapp.com/board?difficulty=medium
        - Da una banca online sviluppata dal gruppo

Si è scelto di adoperare come generatore di puzzle l'API presentata nelle specifiche.

Si è scelto di NON consentire all'utente la rotazione dello schermo, anche facendo riferimento ad app già esistenti.

Classi

NOME	DESCRIZIONE	IN RELAZIONE
Cell	Elemento base della tavola del sudoku.	Board, SudokuBoardView, Game Activity
Board	Logica della tavola del Sudoku.	Cell, SudokuBoardView, SudokuModel, Game Activity
SudokuModel	Gestisce la selezione delle celle nella view.	Board, Cell, Game Activity
SudokuBoardView	View della tavola del sudoku.	Board, Cell, SudokuModel, Game Activity
Pausable Chronometer	Estende il cronometro base di android.	Game Activity

SudokuGenerator	Si occupa di generare un nuovo puzzle.	Main Activity, Game Activity, Board.
Utils	Contiene metodi utili ad altre classi.	Game Activity, Board
SudokuSolver	Si occupa della risoluzione del Sudoku.	Game Activity, Board, Main Activity
Dialog Classes: · InfoDialog · LoadingDialog · NewGameErrorDialog · ResumingGameDialog	Dialog che appaiono all'occorrenza nel ciclo di vita dell'app.	Game Activity, Main Activity
Globals	Contiene valori statici per le difficoltà.	Game Activity ,Main Activity

Activities

NOME	DESCRIZIONE	IN RELAZIONE CON
Main Activity	Activity principale dell'app da cui si accede alle altre.	Game Activity, Settings Activity, Stats Activity.
Game Activity	Activity in cui è presente la fase di gioco.	Main Activity. Settings Activity, Stats Activity
Settings Activity	Activity che si occupa di gestire le impostazioni dell'app.	Main Activity, Game Activity
Stats Activity	Activity che si occupa di mostrare le statistiche per le partite giocate.	Game Activity, Main Activity.

Strategie di sviluppo e librerie

Volley

Volley viene usata nella classe SudokuGenerator, che si occupa di generare un nuovo schema per il sudoku. È stato scelto di usarla solo nella generazione e non nella risoluzione (anche se l'API aveva questa possibilità) per non far dipendere troppo l'app da una connessione ad internet.

La richiesta per un nuovo puzzle viene implementata aggiungendo alla coda di Volley una richiesta per un file json, che sarebbe la tavola. Una volta ricevuto correttamente , nella onResponse(), di questa tavola json ne viene fatto il parsing e le celle vengono aggiunte alla classe Board creandole ad una ad una usando il valore iniziale ricevuto dall'API. Se il valore iniziale è 0 la cella è vuota altrimenti la cella in questione non solo ne viene inizializzato il valore ma viene anche settata come cella iniziale, in modo che non sia poi modificabile dall'utente durante la partita.

New game

Quando viene cliccato sul pulsante new game nella main activity, viene richiamata la classe SudokuGenerator e quindi Volley. Viene creato quindi un ArrayList di Cell che poi viene passato al costruttore della classe Board, creando

così una tavola da gioco iniziale.

Sempre nella onResponse() di Volley, viene poi lanciata l'activity per la fase di gioco, passando questi parametri nell'Intent. Viene fatto questo procedimento nella onResponse() per essere sicuri che non venga mai lanciata l'activity senza che Volley abbia restituito una tavola valida per il gioco. A tal proposito viene lanciata una finestra di attesa e/o di errore nel caso in cui Volley dia una risposta non positiva.

Shared Preferences

Nell'intera app viene fatto un uso massiccio delle shared preferences. In primo luogo vengono usate per salvare le impostazioni dell'applicazione, usando il file xml "preferences.xml". Sono presenti inoltre 3 diverse shared preferences, una per ogni grado di difficoltà. Questo per permettere all'utente di salvare una partita in corso per ogni difficoltà e per mostrare le statistiche relative agli stessi. Spesso in una activity (Main o Game) ci si ritrova a dover capire di quale difficoltà la partita è in corso o quale difficoltà è stata selezionata per avviarne una nuova, o nel caso disponibile di riprenderne una in corso; per far ciò viene mantenuta una variabile shared preferences che all'occorrenza viene settata prendendo quella della difficoltà relativa.

Resume game

Nel caso di riprendere una partita in corso ad esempio, una volta capito a quale difficoltà stiamo facendo riferimento, vengono caricati i dati relativi a quella

partita, ovvero la board e il tempo. Quest'ultimo però viene preso dalle shared preferences direttamente nella Game Activity, al momento di settare il cronometro. All'activity di gioco poi vengono passati nell'Intent:

- La board presa dalle shared preferences
- La board risolta (vedi dopo)
- La difficoltà della partita

La board risolta viene passata all'intent sotto forma di array monodimensionale, dato che android mette a disposizione questo metodo, usando la classe Utils come convertitore, sfruttando il fatto che in un array monodimensionale possiamo risalire alla posizione di riga e colonna come:

POS: RIGA*9 + COLONNA

Gson

Gson è un'altra libreria utilizzata nel progetto. Viene usata per inserire nelle shared preferences l'oggetto Board. Quest'ultima viene convertita in un file json e passata come stringa nella shared preferences, per poi essere ripresa e ritrasformata in un oggetto Board sempre usando gson.

Sudoku Board View

La view personalizzata della tavola viene disegnata attraverso alcuni metodi presenti nel metodo onDraw(). E' stato scelto di disegnare un quadrato avente per lato le dimensioni dello schermo[1] meno un valore di padding[2]. Nel metodo init()

[1] Abbiamo considerato solo la larghezza dello schermo nel codice, avendo impedito la rotazione, ma nel caso bastava settare questo valore cioè il lato del quadrato al minimo tra larghezza e altezza.
[2] Anche il padding potrebbe essere settato dinamicamente a seconda della grandezza dello schermo del device, andando a usare valori predefiniti per certi tipi di grandezza.

vengono inizializzati i numerosi Paint per le celle, inoltre viene calcolata la dimensione di una cella come ***(width - 2*padding)/ SIZE.*** La board da disegnare viene presa dal Sudoku Mode, essa contiene tutte le celle della tavola numerate da 0 a 80 e distingue se una cella è una iniziale (passata cioè dal generatore) oppure no, questo per usare un diverso tipo di Paint su celle iniziali, rendendole più marcate e visibili, oltre che per non permetterne la modifica. Per le annotazioni viene disegnato il numero da inserire partendo dallo spigolo in alto a sinistra di una cella, spostandosi lungo l'orizzontale di un terzo di cella e poi lungo la verticale sempre di un terzo di cella. Ogni qual volta che viene selezionata una cella la view viene ridisegnata tenendo conto della cella che deve essere selezionata e dell'eventuale riga e colonna (dipende dalle impostazioni).

Modello Sudoku

Cell

La classe principale del modello è la classe Cell che descrive appunto una cella della tavola. Questa può essere "iniziale" oppure no; viene deciso al momento della generazione della tavola (vedi New Game): creando una nuova cella, se questa contiene un valore diverso da zero allora è una cella iniziale. Inoltre essendo poi queste celle immagazzinate nella Board (vedi Board) hanno come attributi la riga e la colonna che ne identificano la posizione. In aggiunta ogni cella ha un ArrayList di Interi che rappresentano le annotazioni (appunti utente).

Board

Board è la classe che modella tutto il puzzle, raggruppando le 81 celle in un ArrayList. Non contiene altri attributi né metodi ma serve più che altro ad avere un modello ordinato su cui "poggiare" le celle e sui cui salvare lo stato di una partita.

Sudoku Model

La classe Sudoku Model si occupa di gestire la selezione e l'input delle celle. Il model, che fa riferimento ad una Board specifica (infatti deve essere passata nel costruttore) tiene traccia delle celle selezionate. Di default, -1 per riga e/o colonna significa cella NON selezionata.

User input

La selezione di una cella, l'inserimento di un valore, una nota ecc. sono gestite dal metodo della view onTouchEvent(). Il parametro di questo metodo (MotionEvent event) è fondamentale in quanto ci permette di risalire a quale punto dello schermo è stato cliccato dall'utente. Sapendo la posizione x e y dello schermo cliccata possiamo risalire alla riga e colonna selezionate e quindi alla cella in questione. Per far questo usiamo la relazione:
possible Selected Row = (event.getY() - *padding*)/ cell Size Pixels);

possible Selected Col = (event.getX() - *padding*)/ cell Size Pixels);

Dove cell Size Pixels è la dimensione di una cella. Se i due valori sono entrambi diversi da -1, allora invalidiamo la vista in modo da poter rendere visibile la cella selezionata. Dopodiché possiamo gestire l'input dell'utente nella Game Activity, cioè se stiamo prendendo appunti andiamo ad inserire appunti, altrimenti andiamo a inserire il valore del pulsante cliccato, nella cella selezionata.