POLITECNICO
DI MILANO

LEGO DIGITAL SONORO 2

Documentazione Tecnica

Giacomo Bresciani

Indice

Capitolo 1

Introduzione

1.1 Scopo del documento

Lo scopo di questo documento è quello di mostrare le scelte implementative fatte durante lo sviluppo dell'applicazione *Lego Digital Sonoro 2*. Esso si divide in 3 sezioni: *Database*, *Logica di gioco*, e *Interfaccia Utente*.

1.2 Descrizione dell'applicazione

Lego Digital Sonoro 2 è un videogioco per piattaforma Android progettato esclusivamente per dispositivi tablet. Lo scopo del gioco è quello di comporre parole a partire da tessere colorate rappresentanti diverse sillabe. Ogni tessera ha un colore associato direttamente alla sillaba che rappresenta e non appena viene toccata il dispositivo riproduce (tramite il sintetizzatore vocale) il suono della sillaba. Toccando due tessere una dopo l'altra è possibile comporre una parola che, se corretta, verrà aggiunta all'elenco delle parole trovate sotto forma di immagine composta dalle due tessere. La schermata consente inoltre di riascoltare le parole trovate, sia in italiano sia in inglese. Una partita può essere composta da più schermata; ogni schermata termina quando sono state trovate tutte le parole componibili con le sillabe mostrate (2 o 4).
Durante il corso della partita l'applicazione registra i tempi in cui vengono identificate le parole, e le invia ad una mail configurabile nelle impostazioni. E' presente inoltre una modalità cooperativa a due giocatori a turni. Ogni

turno consente ad un giocatore un tentativo per indovinare una parola.
La figura 1.2 mostra l'interfaccia di gioco.

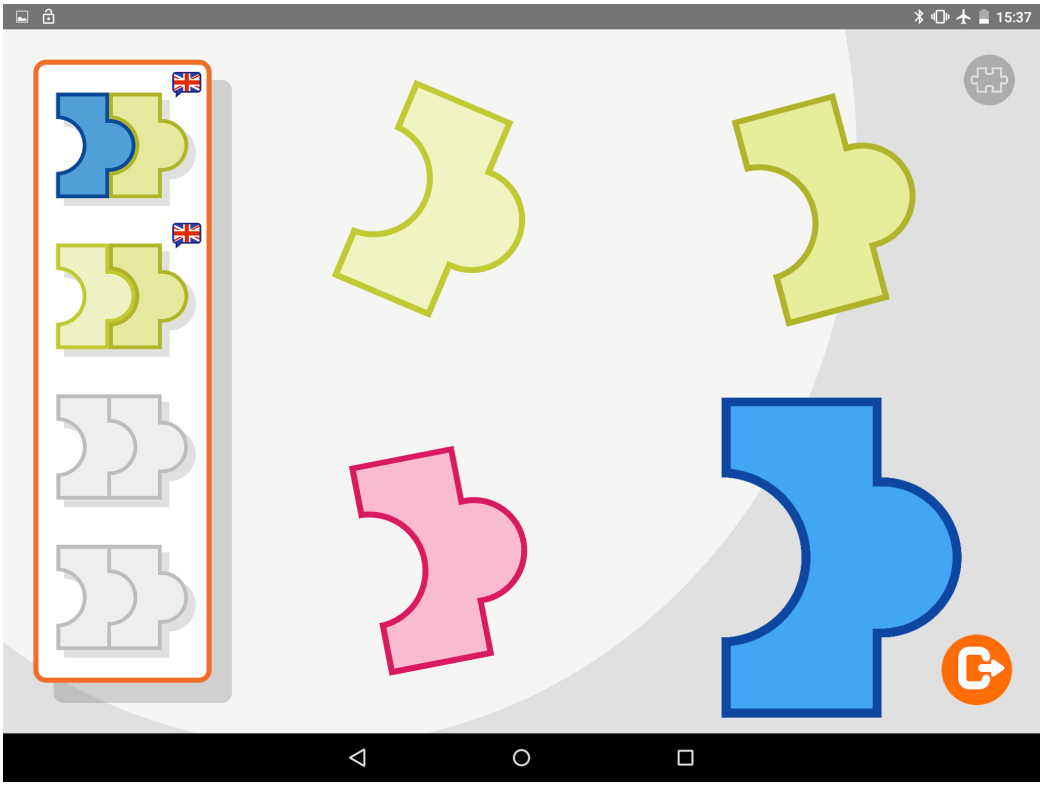

Figura 1.1: Schermata di gioco

Capitolo 2

Database

2.1 Struttura

Il database SQLite è stato implementato tramite SugarORM, un ORM (Object-Relational Mapping) sviluppato appositamente per Android. In questo modo le entità del database sono direttamente mappate su classi Java, utilizzando il pattern DAO (Data Access Object).
Il database contiene 5 entità all'interno del package `it.gbresciani.legodigitalsonoro.model`:

`Word` Contiene la parola, sia in italiano sia in inglese, e le sillabe che lo compongono.

`Syllable` Contiene la sillabla e l'esadecimale del colore associato.

`WordStat` Permette di tracciare il momento in cui una parola viene trovata.

`GameStat` Contiene le statistiche di una particolare partita

In figura 2.1 sono riportati dettagli delle entità.

2.2 Inizializzazione

Il database viene inizializato la prima volta che l'applicazione viene aperta con i dati in formato *JSON* presenti nei files `words.json` e `syllable.json` contenuti nella cartella degli `assets`.
Il listing 2.1 contiene un estratto del file `words.json`

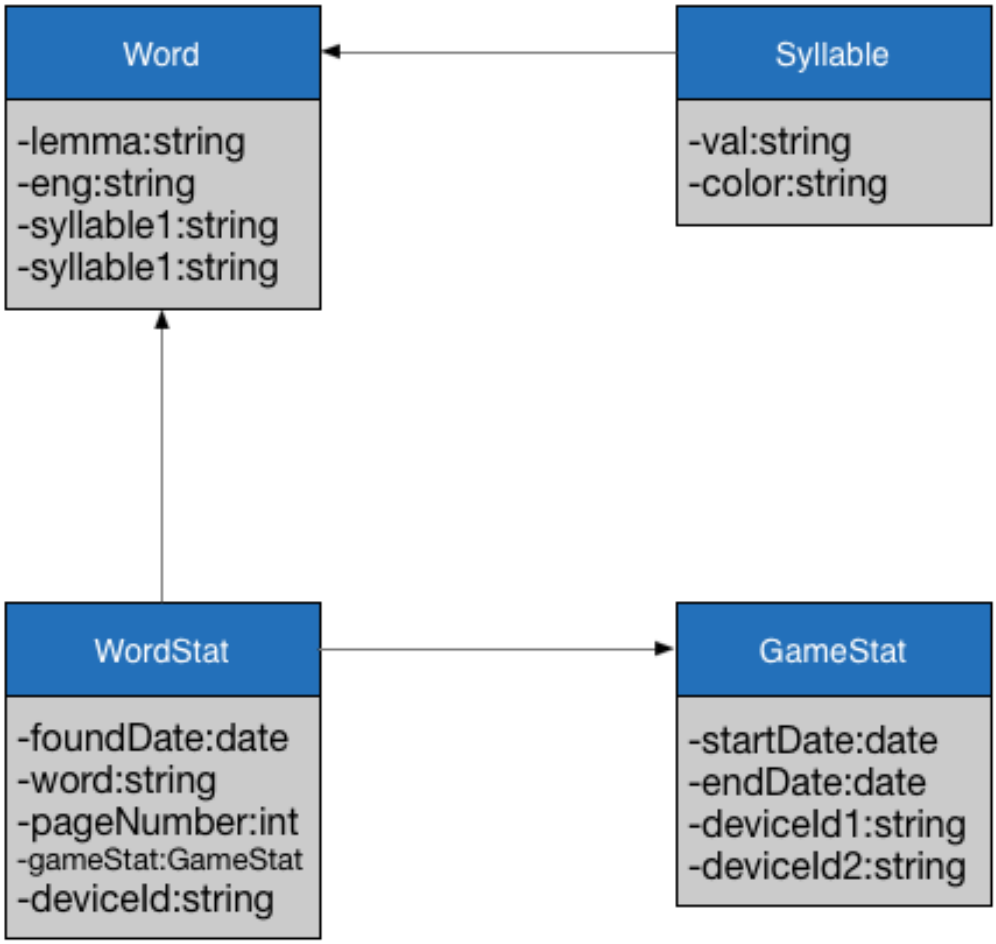

Figura 2.1: Entità presenti nel database

Il mapping degli elementi JSON con la classe Java associata avviene tramite la libreria Open Source GSON, sviluppata da Goolge, che fa uso di annotazioni per mappare gli attributi delle classi con i campi JSON.
L'operazione di conversione dei dati da JSON a classi Java e il successivo inserimento all'interno del database, in quanto operazioni relativamente lunghe, non vengono eseguite all'interno del *Thread* principale ma in uno differente grazie ad un `IntentService` contenuto in `it.gbresciani.legodigitalsonoro.services.InitDBService`.

Listing 2.1: Porzione del file `words.json`

```
[
  {
    "lemma":"arco",
    "eng":"arc",
    "syllable1":"ar",
    "syllable2":"co"
  },
  ...
```

Capitolo 3

Logica di gioco

La logica di gioco è affidata all'Activity `PlayActivity`; essa gestisce inoltre i `Fragments` (vedi capitolo 4) che visualizzano l'Intefaccia Utente.

3.1 Glossario

In questa sezione vengono descritti acuni termini utili alla comprensione delle successive sezioni.

Page Una partita è composta da una o più pagine. Una pagina può contenere 2 o 4 sillable e da 1 a 4 parole da trovare

User role Durante una partita multi player i giocatori si distinguono in "master", colui che "orchestra" la partita, e "slave", colui che vi partecipa in modo passivo

Turno Durante una partita multi player rappresenta la possibilità per un giocatore di provare a indovinare una parola. Il giocatore che non è in possesso del turno attende la mossa dell'altro

3.2 Event Bus

Per far comunicare i vari componenti dell'applicazione (`Activities`, `Fragments`, `Services`) è stato utilizzato Otto, un *Event Bus*, sviluppaato appositamente per Android, che permette ai vari componenti di postare eventi su un unico

Bus e per riceverli in modo asincrono.
La classe `BusProvider` permette, tramite un metodo statico, di accedere allo stesso Bus in ogni punto dell'applicazione.
All'interno del package `package it.gbresciani.legodigitalsonoro.events` son presenti numerose classi (ad esempio `WordClickedEvent` o `PageCompletedEvent`) che rappresentano i diversi eventi generati e "ascoltati" dai componenti dell'app.
La classe `NineBus` è una piccola estensione del Bus predefinito di Otto che permette di pubblicare tutti gli eventi sul `Main Thread`, anche se generati in un altro `Thread`.

3.3 GameState

`GameState` è la classe che rappresenta lo stato di una partita, sia essa single o multi player. Viene inizializzato da `PlayActivity`, e aggiornato durante il corso della partita in seguito ai diversi eventi prodotti dagli altri componenti dell'app.
I suoi attributi sono:

pageNumber Il numero di pagina corrente.

pages Il numero di pagine totali della partita in corso (stabilito nelle impostazioni).

pageWordsToFindNum Il numero di parole ancora da trovare nella pagina corrente.

wordsAvailable Un `ArrayList` contentente le parole ancora da trovare.

wordsFound Un `ArrayList` contenente le parole gia trovate.

syllables Un `ArrayList` contenente le syllabe presenti nella pagina corrente.

currentPlayer Se la partita è multi player contiene una stringa che rappresenta il ruolo del giocatore che detiene in quel momento il turno.

currentPlayerDeviceId Se la partita è multi player contiene una stringa che rappresenta l'id del dispositivo che detiene in quel momento il turno.

La classe `GameState` contiene attributi annotati con `@SerializedName` in modo da poter essere convertita in un oggetto JSON (tramite GSON) ed essere inviata al secondo giocatore nel caso di partita multiplayer (maggiori dettagli nella sezione 3.7).

3.4 Statistiche di gioco

Le satistiche della partita, come accennato nel capitolo sul database, vengono salvate tramite i DAO `WordStat` e `GameStat` che contengono, ad esempio, i tempi di ritrovamento delle parole, le parole stesse, la pagine su cui sono state trovate o eventualmente l'id del giocatore che le ha trovate.

3.5 Game flow

Tutta le fasi di una partita sono gestite dall'`Activity` `PlayActivity`. All'avvio, cioè durante la chiamata al metodo `onCreate()` essa esegue le seguenti operazioni:

1. Carica i settaggi della partita
2. Carica i suoni di gioco
3. Istanzia il sintetizzatore vocale `TextToSpeech`
4. In base al tipo di partita (single o multi player) inizializza o meno il Bluetooth e setta la variabile `multi` a "MASTER" o "SLAVE"

I principali metodi di `PlayActivity` sono:

`startGame()` Si occupa di inizializzare gli oggetti per le statistiche e chiama `constructPage()` and `startPage()`

`constructPage()` Inizializza lo stato della partita (`GameState`) con i dati relativi alla pagina corrente. Chiama i metodi accessori `Helper.chooseSyllables()` (sceglie in modo casuale le sillabe per la pagina) e `Helper.permuteSyllablesInWords()` (calcola le permutazione delle sillabe scelte per trovare le parole componibili)

Listing 3.1: Struttura delle satistiche inviate

```
Dispositivo N: device id
Tempo totale: mm:ss
N - word: mm:ss (# of page on which the word was found)
...
```

`startPage()` Inizializza i `Fragments` che mostreranno l'interfaccia utente

`updateState()` Si occupa di gestire la logica di gioco: determina se tutte le parole sono state trovate o se interrompere il gioco quando non è il proprio turno e posta sul Bus l'evento `StateUpdatedEvent` in modo che i `Fragments` possano reagire correttamente

L'`Activity` presenta inoltre diversi metodi annotati con `@Subscribe` che permettono di reagire agli eventi pubblicati da altri componenti.

3.6 Invio statistiche

L'invio delle statistiche avviene tramite l'invio di una mail all'indirizzo settato nelle impostazioni dell'applicazione.
L'invio è gestito dall'`IntentService` `GenericIntentService` al quale viene passato l'id corrispondente alla partita appena conclusa. Interrogando il database esso carica i dati e li inserisce nel corpo dell'email formattandoli come mostrato nel listing 3.1. Il listing 3.2 ne mostra invece un esempio.

3.7 Multiplayer

Come già detto la modalità multi player segue le stesse regole di quella single player se non per il fatto che i due giocatori provano a turno a trovare una parola.
Il tablet "master" è quello che inizia il processo di connessione scegliendo, tra i dispositivi in ascolto, il tablet "slave". I vari aspetti della partita come i settaggi, le sillabe presenti e le parole da trovare vengono tutti stabiliti dal "master" che si occupa quindi anche di gestire il flow del gioco, mentre lo "slave" partecipa in modo passivo.

Listing 3.2: Esempio di statistiche

```
- One player -

Dispositivo 1: BC:20:A4:73:6B:49
Tempo totale: 00:13
1 - fumo: 00:02 (1)
1 - muro: 00:13 (2)

- Two Player -

Dispositivo 1: BC:20:A4:73:6B:49
Dispositivo 2: D4:0B:1A:15:FD:AD
Tempo totale: 00:14
2 - ceno: 00:02 (1)
1 - noce: 00:09 (1)
2 - buco: 00:14 (2)
```

La comunicazione tra i due dispositivi avviene tramite Bluetooth. La classe che si occupa di stabilire la connessione è `BluetoothService`; essa si serve di tre `Thread` differenti (`AcceptThread`, `ConnectThread`, `ConnectedThread`) per ricevere un connessione, effettuare una connessione, e inviare e ricevere messaggi.
I messaggi scambiati sono stringhe contenenti oggetti JSON preceduti da un header che ne identifica la natura. I possibili messaggi sono 4:

GAME_STATE Contiene la rappresentazione JSON dell'oggetto `GameState` descritto nella sezione 3.3. Solo il "master" invia questo tipo di messaggio allo "slave", il quale ne ricava tutte le informazioni necessarie.

SIMPLE_TURN_PASS Lo "slave" invia questo messaggio al "master" quando non è riuscito a trovare parole e quindi sta semplicemente passando il turno.

WORD_FOUND Lo "slave" invia questo messaggio insieme alla parola trovata, in modo che il "master" possa aggiornare lo stato della partita.

GAME_END Il "master" invia questo messaggio allo "slave" al termine della partita.

Capitolo 4

User Interface

Come anticipato nel capitolo 3, l'interfaccia grafica del gioco è gestita quasi interamente da `Fragments`, che rispondono agli eventi pubblicati sul Bus dall'activity `PlayActivity` aggiornando i loro componenti.
I due principali `Fragments` sono `WordsFragment` e `SyllablesFragment` (vedi figura 4), che si occupano rispettivamente di gestire la porzione di schermo contenente le parole, e la porzione contenente le sillabe disponibili.

4.1 WordsFragment

`WordsFragment` si occupa di gestire la parte di interfaccia grafica che riguarda le parole già trovate e quella ancora da trovare.
Esso presenta una lista verticale di "slot" che possono essere "vuoti", o "riempiti" con le tessere di puzzle corrispondenti alle sillabe che formano la parola. Viene inoltre mostrata accanto ad ogni slot "pieno" una bandiera inglese che se toccata permette di ascolare il suono della parola tradotta in inglese.
Il metodo `initUI()` si occupa, dato il numero di parole da trovare e la dimensione dello schermo (calcolata a runtime), di calcolare la dimensione degli slot corretta per gli slot con lo scompo di riempire al meglio lo spazio disponibile.
Le immagini degli slot sono fornite come file PNG all'interno della cartella `assets` e vengono caricate dinamicamente come Bitmap della dimensione corretta (per risparimare memoria) dal metodo `loadWordBitmap()`.
Il `Fragment` è sottoscritto all'evento `StateUpdatedEvent` in modo da rea-

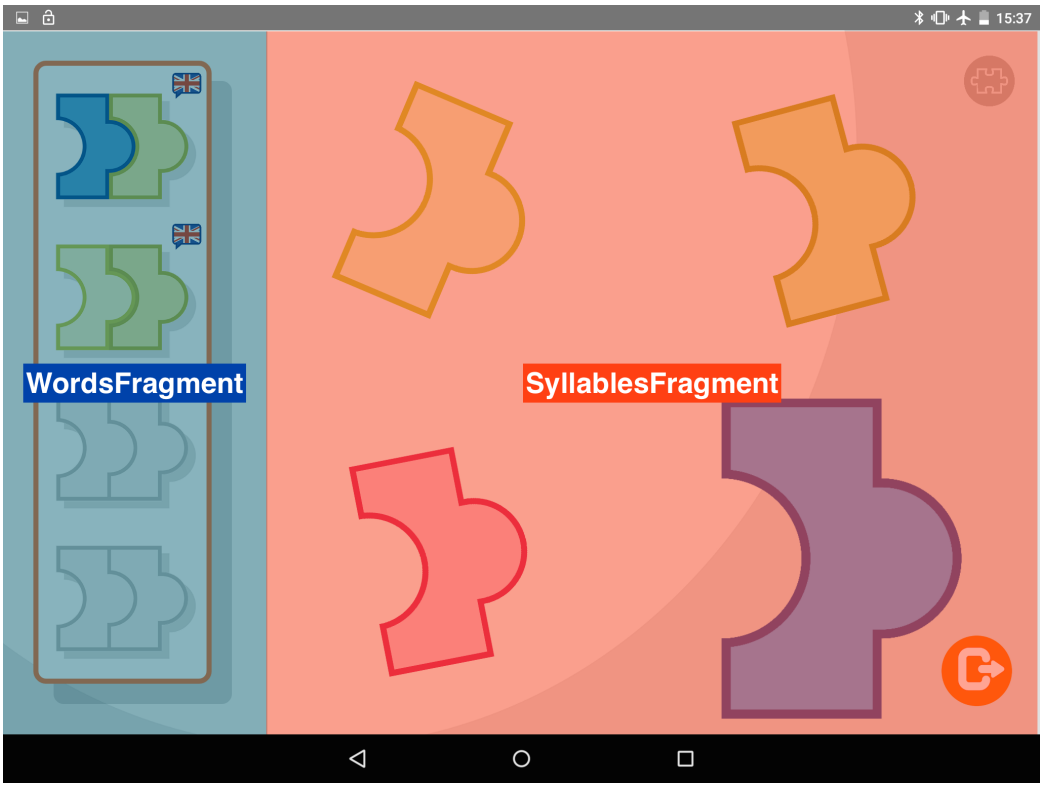

Figura 4.1: Fragments della schermata di gioco

gire ai cambiamenti di stato della partita, come il ritrovamento di una nuova parola.

4.2 SyllablesFragment

`SyllablesFragment` si occupa di gestire la parte di interfaccia grafica che rigurda le sillabe disponibili nella pagina corrente.
Esso presenta una griglia sparsa di 2 o 4 tessere di puzzle colorate rappresentanti ognuna una sillaba. Se toccate le tessere vengono selezionate e si "alzano" per restituire un feedback della selezione e riproducono il suono della sillaba a cui sono associate.
Al pari di `WordsFragment` calcola la dimensione corretta delle immagini e le carica come Bitmap.

Capitolo 5

Librerie esterne

Di seguito vengono elencate le librerie utilizzate per lo sviluppo dell'applicazione.

SugarORM
: Libreria Open Source ORM.

Gson
: Libreria Java che permette la conversione tra oggetti Java e la loro rappresentazione JSON.

Otto
: Libreria che permette di sfruttare un event bus.

ButterKnife
: Libreria per l'injection delle View.