Corso di Programmazione ad Oggetti, anno accademico 2013/2014
Corso di Laurea in Ingegneria e Scienze Informatiche – Università di Bologna

Relazione relativa all' ingegnerizzazione di Schooltrack: programma per la memorizzazione degli esami conseguiti all' interno del proprio corso di laurea.

Realizzata da: Tonini Diego

1 Analisi del problema

La realizzazione di un applicazione funzionante su SO Android che permette di memorizzare visionare, e modificare i dati relativi ai propri esami, con voti annessi, nonché di usare tali dati al fine di elaborarli per avere delle medie. L' applicazione mira a offrirsi come strumento d' uso volto a semplificare il normale procedimento che bisogna seguire per conoscere i propri voti.

L' applicazione si pone in una posizione intermedia fra due tipi di soluzioni:

A) visionare i propri voti avendoli scritti su documento cartaceo è oggettivamente più semplice ma meno funzionale, in quanto un applicazione se fatta bene è intuitiva, abbatte drasticamente i tempi di ricerca rispetto ad un documento cartaceo. Permette il back-up. Inoltre è pensabile che l' oggetto che viene più utilizzato e che quindi sia accessibili nell' arco di una giornata sia proprio lo smartphone.

B) visionare i propri voti nel dettaglio su portali on-line è comunque più complesso rispetto ad aprire un applicazione. Tale operazione potrebbe non essere possibile causa mancanza di banda, mancanza di connessione, manutenzione al portale. L'accesso al portale comporta nella maggioranza dei casi un sistema di autenticazione, che rappresenta quindi un ulteriore perdita di tempo se l'obiettivo rimane la semplice visione.

Funzionalità offerte dall' applicazione:

- **Aggiunta esame:**
  E' presente una gui dedicata all' inserimento di un esame. Gli unici campi obbligatori sono il nome dell' esame e la data (impostata al giorno corrente di default)

- **Visualizzazione carriera:**
  E' presente una gui realizzata mediante una lista, in cui viene visualizzato l' elenco di tutti gli esame conseguiti. Tale lista può essere ordinata secondo il nome, il voto, e la data, e per ogni modalità l' ordinamento viene fatto crescente/decrescente in base al click sulla voce desiderata.

- **Modifica esame:**
  Un click prolungato su un esame della lista, da all' utente la possibilità di modificare l'esame. Può essere modificato ogni campo. La modifica consente quindi di aggiornare un voto, i crediti o la data.

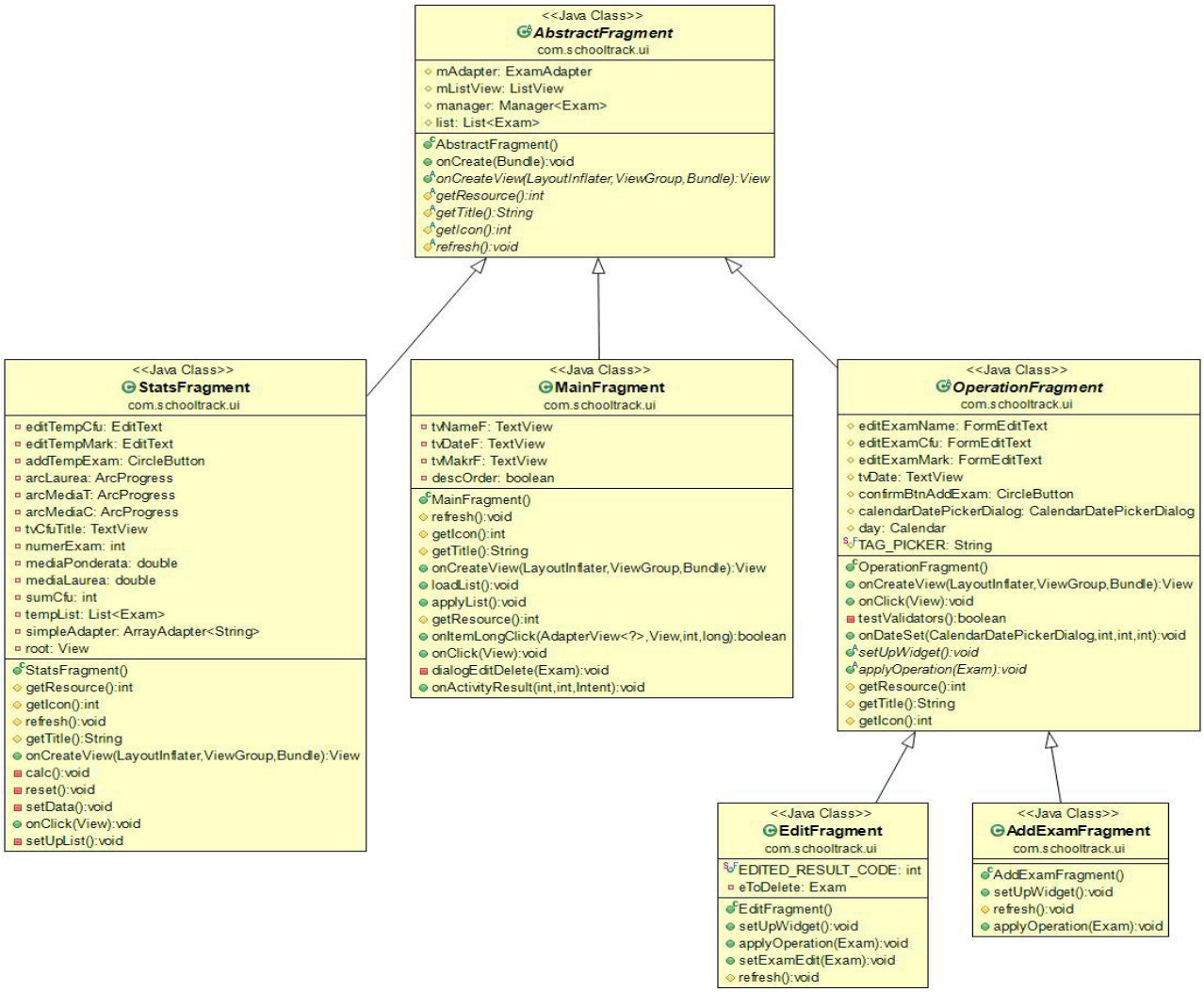
- **Eliminazione esame:**
  Sempre tramite click prolungato sulla lista, si può scegliere di eliminare definitivamente l'esame appena selezionato. Ovviamente un piccolo alert richiede la conferma.

- **Visione statistiche:**
  E' presente una gui che immediatamente calcola 3 dati fondamentali per uno studente.
  Quanto si è vicini alla laurea in termini percentuali, la media in trentesimi e quella in centesimi.
  (La percentuale di "avanzamento carriera" è calcolata sui crediti conseguiti rispetto ai crediti totali.
  Quindi 3 esami da 6 crediti generano un avanzamento maggiore rispetto a 3 esami da 12 crediti.
  Questa scelta permette di non obbligare l' utente ad impostare il numero totale di esami che potrebbe magari non conoscere al momento dell' installazione dell' applicazione

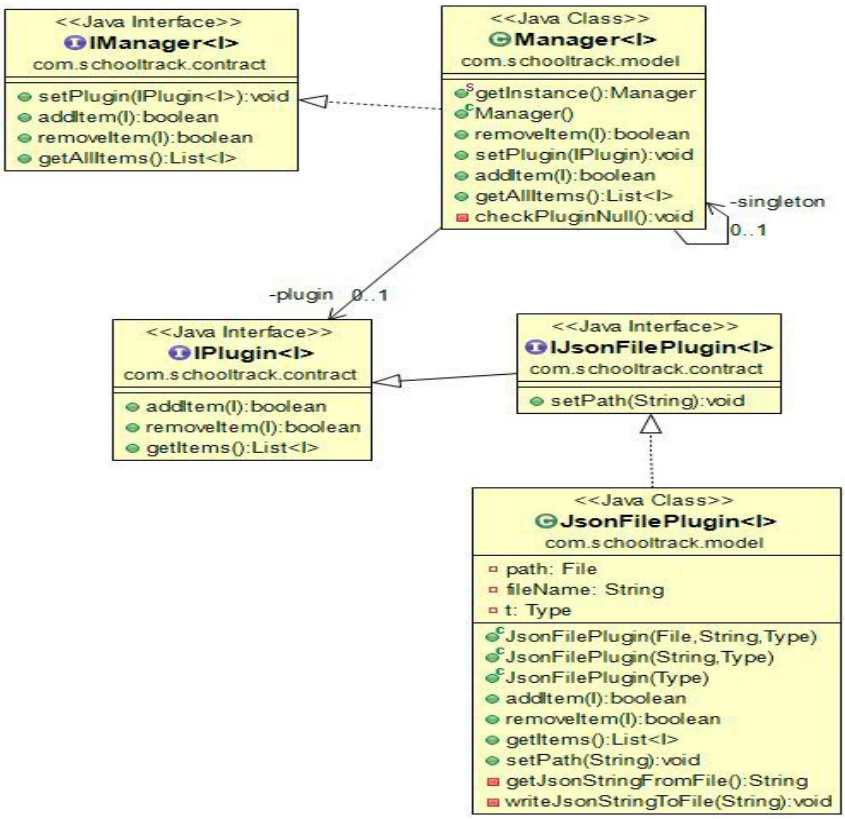
- **Aggiunta voto temporaneo – (schermata statistiche)**
  Nella schermata statistiche è possibile ricalcolare i 3 valori precedentemente descritti includendo degli esami ancora non verbalizzati. Strumento utile per fare una breve previsione dopo un esame. Tali esami non saranno scritti su file

2 Progettazione architetturale

La maggior parte delle scelte per quanto riguarda la progettazione sono state vincolate dalla struttura del framework android. Possiamo quindi definire le view tutti i file .xml. Le view sono quindi create solamente con codice xml. Activity e Fragment assumono una funzione di model-controller.
Senza entrare nei particolari, possiamo affermare che il cuore dell' applicazione è la MainActivity.
Ogni Activity ha il suo ciclo di vita, cosi come un fragment. Ogni fragment è contenuto in un Activity padre.
Una qualsiasi operazione che comporta l' aggiunta o rimozione di un esame nel file, comporta un aggiornamento del fragment corrente. Viene quindi creata una classe AbstractFrament, in quanto alcuni metodo custom e nativi di android sono identici. Possiamo astrarre un inserimento e una modifica in una classe operazione comune chiamata OperationFragment, la quale tratta i dati e le view nello stesso modo, tranne per l' uso finale di tali dati. Definiamo quindi due specializzazioni. Di seguito un digramma mostra le dipendenze fra i vari fragment.



La parte riguardante la gestione degli esami, quindi operazioni di visualizzazione inserimento e rimozione sono stati pensati come operazioni da poter effettuare sia su file, sia su server o altre piattaforme.
Nel caso specifico sapendo che tali operazioni comportano delle scritture su file, si è scelto di creare una classe Manager tramite la quale è possibile accedere da ogni punto del' applicazione. La scelta è giustificata dalla presenza di fragment, che devono compiere tali operazioni ogni volta che l'utente cambia la vista nella applicazione. Il concetto di inserimento/rimozione è stato pensato per essere usato da oggetti di ogni tipo. In questo modo abbiamo un gestore di oggetti riusabile anche in futuro per altri progetti.
Di seguito è presente il diagramma che descrive l'effettiva implementazione per operazioni su file, obiettivo principale del' applicazione.



Organizzazione dei package

- **ui**
  package con gli elementi che permettono la visualizzazione del' applicazione. Gli elementi sono visibili nei precedenti diagrammi.
- **Models**
  package con gli oggetti modello per rappresentare esami e operazioni con essi
- **contract**
  package contente le interfaccia.
- **Utils**
  package contenente classe/classi di utilità

Gestione dei esami

La scelta su come implementare un meccanismo di memorizzazione è ricaduta sull' uso di file di testo formattato per essere usato come file json. L'applicazione effettua quindi operazione di scrittura sia per aggiungere sia per eliminare un esame.
Viene caricata una lista di esami da uno specifico file. Di default viene cercato il file "SDCARD/schooltrack/carriera.json". In assenza di tale file, esso viene creato. Se un esame viene eliminato, il manager provvedere alla scrittura su file della medesima lista precedentemente caricata avendo rimosso quello selezionato.
Questa modalità è obbligatoria se si vuole usare file json.
Il concetto di gestione degli esami è implementabile anche per operazioni su server web, mediante le interfacce presenti. Nel caso specifico era richiesto di implementare una gestione su file.

VANTAGGI:

- Json è un formato usato su una vasta gamma di piattaforme e servizi.
- La dimensione di un file json per memorizzare dati, è minore di molti altri formati se usati per rappresenta le stesse informazioni.
- Json si presta per essere usato in applicazione per lo scambio dei dati tramite protocolli basati su connessione internet (ecco giustificato il punto precedente).
- un file json può rappresentare un qualsiasi oggetto custom. In questo caso rappresenta un esame che sarà trattato istanziando oggetti di tipo Exam. La possibilità di usare il Manager con ogni tipo di oggetti è garantita usando tale formato. Una memorizzazione di un esame tramite testo a piacimento prevede la costruzione di uno strumento ad-hoc solo per l'esame e non per oggetti diversi. Non sarebbe possibile riusare tali classi.
- In un ottica futura, con pochi passaggi sarà possibile aggiungere dei plugin che aggiungono rimuovono oggetti custom tramite richieste ad un server. Anche questo aspetto è reso possibile poiché è stato scelto json. Diversamente ogni server avrebbe dovuto implementare il proprio parser di un file.

SVANTAGGI:

- un utente che non riesce ad assimilare json potrebbe avere difficoltà nel leggere/scrivere il file generato dall' applicazione mediante tastiera.

PERCHE' JSON:

- I vantaggi sono numerosi rispetto agli svantaggi. Ci sono numerose librerie già pronte e collaudate da colossi del IT che manipolano file json.

Gestione delle statistiche

L' applicazione permette di effettuare alcuni calcoli in modo da avere delle statistiche minimali per quanto riguarda la carriera universitaria. Tali informazioni sono: la media in trentesimi, la media in centesimi e una percentuale di avanzamento riferita alla laurea.
Tale percentuale non è molto precisa , e tale decisione è voluta. Essa è il rapporto fra i crediti conseguiti totali e i crediti massimi (180). Per una percentuale corretta si sarebbe dovuto guardare il rapporto fra esami conseguiti e esami totali. Molti studenti (probabili utenti) avrebbero dovotu indicare al termine del' installazione tale numero. Molti di loro non sapendolo procedono con la disinstallazione del' applicazione.

Voti temporanei

Lo studente necessita spesso di calcolare le statistiche anche non avendo verbalizzato il voto. La vista statistiche aiuta in questa richiesta permettendo di inserire esami senza effettuare operazioni su file. Alla

chiusura dell'applicazione tali dati saranno quindi persi

Pattern

- **Singleton**
  Viene usato il pattern singleton per instanziare il manager. Possiamo effettuare operazioni di scrittura lettura da ogni fragment senza dovere creare nuovamente il manager.

- **Strategy**
  Viene usato il pattern strategy per la gestione dei plugin. Possiamo aggiungere e rimuovere oggetti generici dal manager. Esso permette di trattare l'oggetto aggiunto senza conoscere i dettagli implementativi del plugin. Settando quindi un plugin diverso da JsonFilePlugin, per esempio un ipotetico FTPServerPlugin, il manager è in grado ora di aggiungere o rimuovere l'oggetto trattato da un server ftp. Solo l'impostazione è stata implementata, in quanto nel caso specifico era necessario solo scrivere su file.

- **Template**
  OperationFragment implementa tale pattern, in quanto definisce un template comune per le classi che ereditano.

- **ViewHolder**
  Pattern usato e consigliato dalle guidelines di Android. Viene usato per l'implementazione delle listView. Permette di riciclare layout senza crearne tanti inutilmente.

3 Sviluppo

Testing

L' applicazione è stata testata per l' intero sviluppo su un motorola moto x 2013, dotato di android 4.4.4 (Kitkat). Ad applicazione ultimata i testi sono stati condotti anche su Huawei Ascend P6 con android 4.4.2 ed infine anche su emulatore (arm) con android 4.2

Api minime richieste: 16

Metodologia di lavoro

Ho impiegato 1 settimana almeno per cercare di produrre un diagramma il più possibile preciso. Ho cercato di creare classi comuni quando necessario. L'idea di usare il json piuttosto che file in formato diverso è maturata durante la programmazione, quindi successivamente alla stesura del diagramma. Con ciò è stato necessario rimodellare nomi di alcuni metodi per renderli più chiari e apportare modifiche per produrre classi riusabili.

Codici esterni

L' applicazione è stata sviluppata grazie all ' uso di alcune librerie

- **Gson** - https://code.google.com/p/google-gson/
  Libreria usata per trattare oggetti formattati con lo standard Json

- **supportV4, appCompat**
  Librerie già presenti nel framework android, necesarie per la gestione del' action bar e del componente che rende possibile le transizioni fra le varie viste.

- **MaterialDialogs** - https://github.com/afollestad/material-dialogs
  Libreria usata per produrre alert dialog con un interfaccia più bella rispetto a quelli nativi del framework android.
- **CircleButton -** https://github.com/markushi/android-circlebutton
  Libreria usata per produrre un bottone in stile material-design.
- **BetterPickers** - https://github.com/derekbrameyer/android-betterpickers
  Libreria usata per avere un custom dialog che permette la seleziona di una data.
- **FormEditText** - https://github.com/vekexasia/android-edittext-validator
  Libreria che estende EditText. Permette di avere una EditText con validazione tramite regularExpression e piccoli alert di errore all' interno della stessa view.

  La gestione delle librerie su l' IDE usato (AndroidStudio) permette di aggiungere dipendenze semplicemente aggiungendo una stringa all' interno di un file .gradle.
  Risulta quindi molto conveniente sviluppare su tale Ide, in quanto si riducono drasticamente i tempi per quanto riguarda l' aggiunta di codice di terze parti. Si consiglia quindi l' installazione di AndroidStudio per avere un progetto avviabile su smarphone mediante singolo click.

4 Commenti finali

La scelta di creare un applicazione da solo ha avuto lati negativi e positivi. Essendo io l'unico programmatore ho il pieno controllo su tutto quello che viene fatto. Infatti non è stato usato del tempo per concordate punti in comune con altri sviluppatori. La scelta si sviluppare da solo è stata presa in quanto già pensavo di creare un applicazione android. Ad esame conseguito non conoscevo nessuno che avesse delle basi sulla programmazione android. L'idea di programmare insieme ad altri rimane comunque l'obiettivo principale in caso di progetti futuri. Sviluppando sa solo non ho sentito la necessità di adoperare repository online. A progetto ultimato ho comunque fatto delle prove di push e commit sia da eclipse, ma anche tramite linea di comando. Ovviamente per un progetto in team dovrò apprendere a fondo l'intero uso di mercurial e/o git. L'applicazione poteva essere sviluppata in meno tempo, ma producendo classi non adatte ad essere riusate. Ho pensato che investire del tempo per renderle riusabili era più importante di finire prima il progetto.