***Elaborato di Ingegneria del Software 2***

***Galli Antonio M63000721***

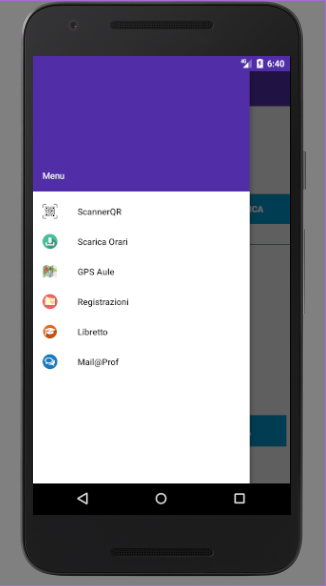
***Gravina Michela M63000708***

***Valletta Paolo M63000723***

1. ***Parte 1: Unina Campus***
   1. ***Requisiti***

Unina Campus è una applicazione Android che permette ad uno studente che la utilizza di gestire la propria carriera universitaria. Offre le seguenti funzionalità:

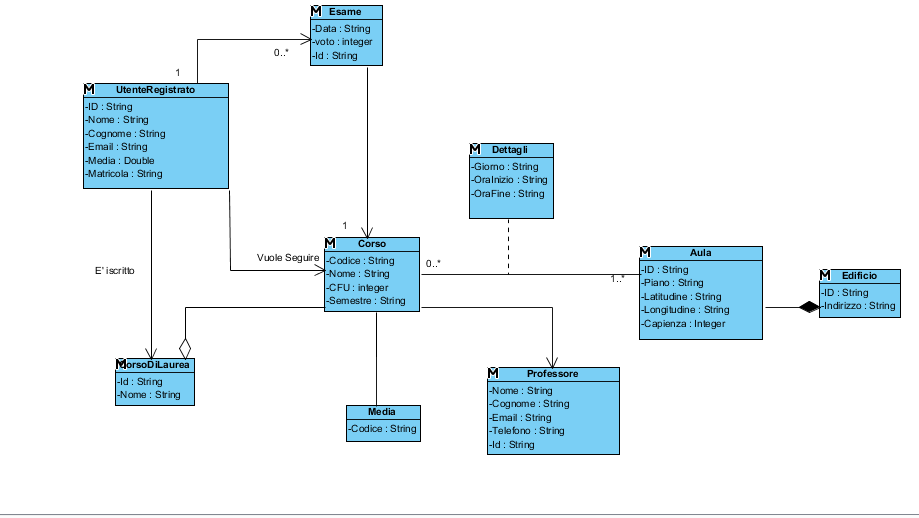
* **ScannerQR:** uno studente può inquadrare il codice QR posto fuori ogni aula al fine di vedere, nel giorno corrente, in quali ore e da quali lezioni l’aula è occupata.
* **Selezione corsi da seguire:** lo studente può selezionare i corsi del suo Corso di laurea da seguire.
* **Scarica Orari:** lo studente può scaricare gli orari dei corsi che ha scelto di seguire. Tali orari verranno sincronizzati nel calendario del dispositivo. È necessario, quindi, avere un calendario sul dispositivo per poter usufruire di tale funzionalità.
* **GPS Aule:** lo studente può visualizzare sulla mappa la posizione propria e di un’aula scelta.
* **Gestione Registrazioni:** lo studente può effettuare l’upload della registrazione di una lezione, oppure effettuare il download delle registrazioni caricate da altri studenti.
* **Libretto:** l’applicazione tiene traccia degli esami sostenuti dallo studente, il quale può aggiornare la sua carriera. È possibile aggiungere un esame o eliminarlo. In entrambi casi verrà aggiornata la media dello studente e di conseguenza la lista dei corsi che può seguire.
* **Mail@Prof:** Lo studente può inviare un’email ad un professore selezionando un corso dalla lista dei corsi da lui seguiti, oppure specificando l’email di destinazione. In entrambi i casi l’invio dell’email viene demandata ad un client esterno.
* **Registra lezione:** lo studente può selezionare un corso e quindi registrare la lezione. Il file viene memorizzato sulla memoria del dispositivo.



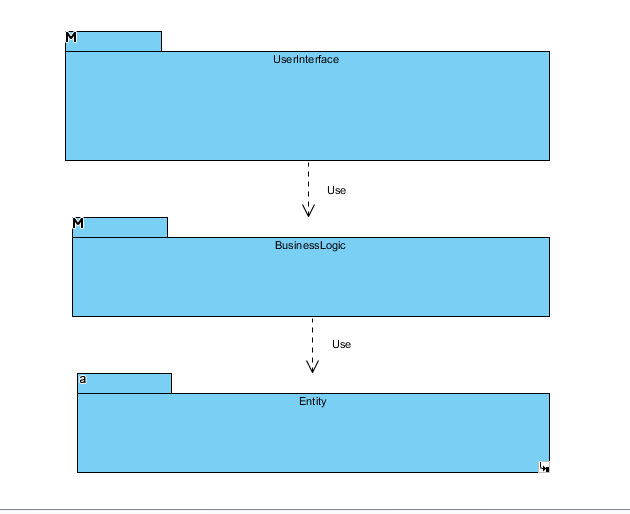
* 1. ***Struttura dell’applicazione***

L’applicazione consta sostanzialmente di 3 package UI, Business, Entity. Il primo è costituito dalle activity e ha lo scopo di interfacciarsi con l’utente e di cogliere tutti gli eventi esterni. Il secondo package è responsabile della logica dell’applicazione e del collegamento a Firebase, per la persistenza dei dati. Il package Entity, infine, contiene le classi che rappresentano le entità del dominio dell’applicazione. Ha lo scopo di mappare le tabelle del database in oggetti java.

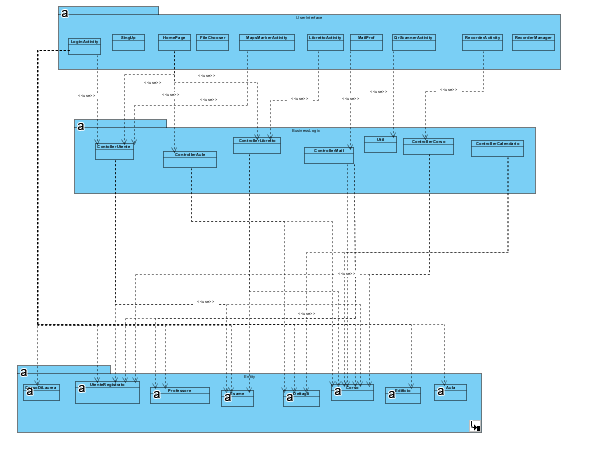
*Class Diagram Entity.*



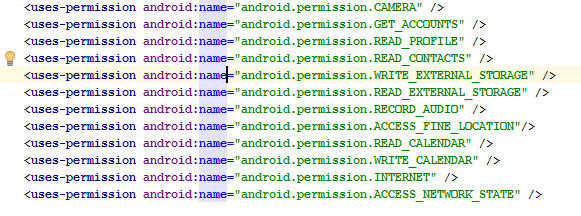
*Package Diagram Generico*



*Package Diagram nel dettaglio*



Per il corretto funzionamento, l’applicazione necessita dei seguenti permessi, riportati nel Manifest:



1. ***Parte 2: Unit Testing Robolectric.***

***2.1 Introduzione***

Il testing di una applicazione Android può essere effettuato grazie all’utilizzo di strumenti e librerie che consentono una scrittura semplice di test eseguibili che “guidano” direttamente la GUI.

Tra gli strumenti più comuni per la scrittura di test, sono ricordati:

* Android Espresso
* Robotium
* Robolectric

Android Espresso è una libreria sviluppata da Google nel 2014 che permette di scrivere test Junit eseguibili tramite l’esecutore AndroidJUnitRuner. Con Android Espresso è possibile guidare l’esecuzione di eventi direttamente su oggetti della GUI, riferire un oggetto della GUI, eseguire asserzioni su proprietà degli elementi dell’interfaccia. Espresso Test Recorder, gratuito ed integrato con Android Studio, è uno strumento che consente di generare un test per un’applicazione Android con il vantaggio di non scrivere nemmeno una riga di codice, in quanto memorizza le interazioni con un dispositivo e aggiunge asserzioni per verificare gli elementi grafici.

Robotium è un framework a supporto del testing di unità delle activity che estende e potenzia JUnit. In particolare rende più semplice la scrittura di test che riguardano più Activity, Dialog, Toast, Menu, migliorando la leggibilità dei test case. Il Funzionamento di Robotium è tutto basato sull’utilizzo di un oggetto denominato “solo”, tramite cui è possibile interrogare e modificare i widget della UI, eventualmente anche senza conoscerne l’identificativo (ad esempio è possibili selezionare un widget in base al testo che mostra).

I casi di test di sistema o della GUI scritti con Android Espresso sono eseguibili solo su emulatori o dispositivi reali. Una soluzione alternativa che permette di eseguire test senza utilizzare emulatori è rappresentata dal framework Robolectric. In pratica esso mette a disposizione implementazioni alternative (mock) di alcune versioni del framework Android. Scegliendo Robolectric come esecutore di test, questi verranno eseguiti in un ambiente mock, che fa uso di una macchina virtuale locale anziché di emulatori o dispositivi. Il codice viene eseguito su una macchina virtuale Android, che in qualche modo sostituisce automaticamente con mock tutte le classi necessarie all’esecuzione del test.

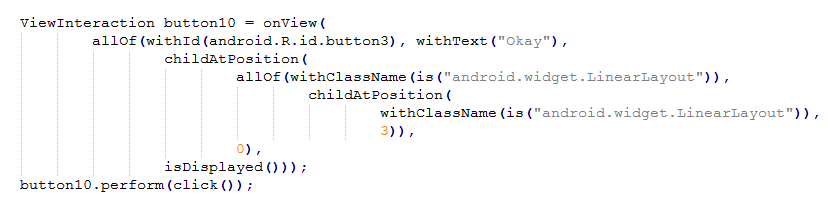
* 1. ***Catalogo di eventi di test base***

I tre framework introdotti precedentemente sono stati usati per scrivere dei test per due applicazioni semplici: MunchLife e TestApplicazione. In effetti considerando i test esistenti per l’applicazione MunchLife e scrivendone dei nuovi sia per MunchLife che per TestApplicazione è stato possibile catalogare gli eventi di test base al variare dei tre strumenti Espressso, Robotium, Robolectric. In particolare TestApplicazione è un progetto Android nato esclusivamente con lo scopo di cogliere quanti più eventi possibili, mentre MunchLife è un’applicazione reale.

Gli eventi catalogati e il modo in cui questi sono gestiti nei tre strumenti sono riportati nel file *MappeNuove.xlsx*. In questo file sono riportati degli esempi di codice di tali eventi ottenuti considerando tutti e tre gli strumenti. Sulla base di quanto riportato in tale file, è stato possibile effettuare una sorta di generalizzazione al fine di creare una corrispondenza tra le funzioni utilizzate da Espresso, Robolectric e Robotium. La generalizzazione è riportata nel file *MappaGenerica.xlsx.* Da come si può notare, Robolectric, a differenza di Espresso e Robotium, ricerca gli elementi tramite il relativo ID e tramite la funzione findViewById() e inoltre, in modo esplicito deve essere specificato se tali elementi si trovano in una activity, dialog o alert dialog. Ciò dipende dal fatto che Espresso e Robotium sono strumenti di testing dinamico e vedono cambiare il contesto di esecuzione, attendendo che il cambiamento sia compiuto prima di cercare eseguire gli eventi. Inoltre Robotium consente la ricerca di elementi anche avendo a disposizione solo il testo.

Con l’ausilio delle corrispondenze trovate è stato possibile generare dei test in Espresso e Robotium ed effettuare le traduzioni in Robolectric. In particolare per l’applicazione MuchLife è stato creato il test Prova.java tramite il recorder automatico di Espresso e successivamente tale test è stato tradotto al fine di verificare le corrispondenze in Robolectric, creando quindi il test ProvaRobolectric.java. Di seguito sono riportati alcuni esempi di corrispondenze dei due test:

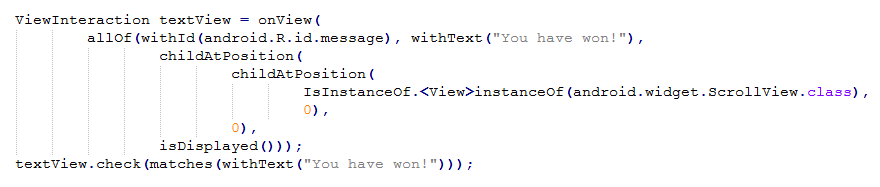
* *Click su bottone Okay Espresso*



*Click su bottone Okay Robolectric*



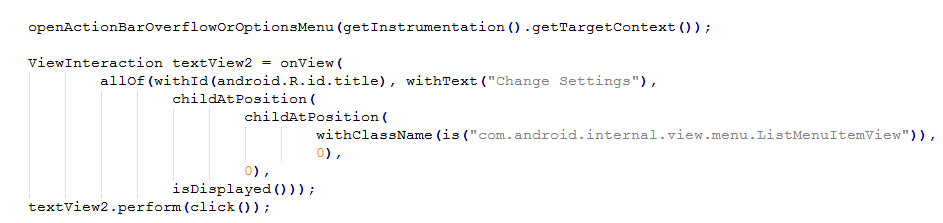
* *Verifica testo Espresso*



*Verifica testo Robolectric*



* *Click su Menu Option Espresso*

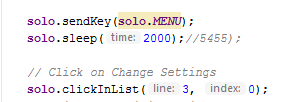


*Click su Menu Option Robolectric*



Per l’applicazione MunchLife è presente un test in Robotium, pertanto è possibile vedere come tale strumento gestisce i vari eventi. Di seguito sono riportate le corrispondenze tra Robotium e Robolectric di alcuni eventi:

* *Click su option Menu Robotium*



*Click su option Menu Robolectric*



* *Click su bottone Robotium*

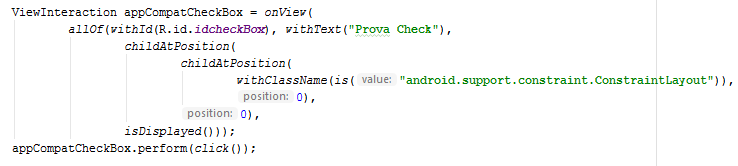


*Click su bottone Robolectric*



Per il progetto ApplicazioneTest è stato generato tramite il reconder automatico il test in Espresso ProvaTest.java. Successivamente tale test è stato tradotto al fine di utilizzare lo strumento Robolectric, ottenendo quindi ExampleUnitTest.java. Di seguito si vedono le corrispondenze di alcuni eventi:

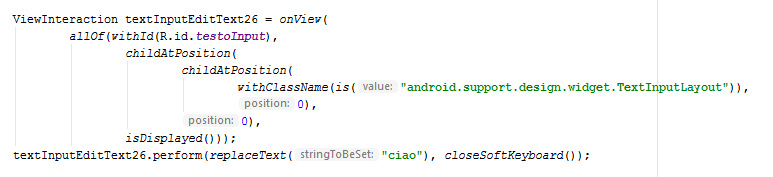
* *Click su CheckBox Espresso*



*Click su CheckBox Robolectric*



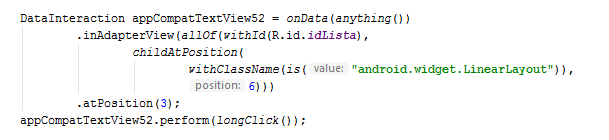
* *Verifica testo Espresso*



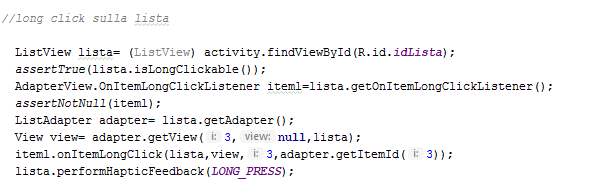
*Verifica Testo Robolectric*



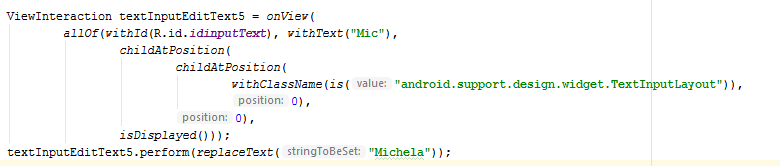
* *LongClick su item di lista Espresso*



*LongClick su item lista Robolectric*



* *Scrittura in un campo di testo Espresso*



*Scrittura di un testo Robolectric*

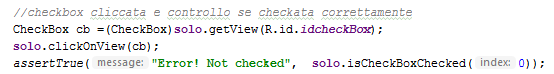


Robotium viene, invece, utilizzato per generare il test specificato nel file RobotiumTest.java, in cui sono riportati gli eventi visti per i test generati per Espresso e Robolectric. Di seguito sono riportati alcuni esempi:

* *Long Click su un elemento di una lista, il corrispondente in Robolectricè analogo a quanto visto prima*



* *Click su checkbox, il corrispondente in Robolectric è analogo a quanto visto prima*



* *Scrittura in un campo di testo, il corrispondente in Robolectric è analogo a quanto visto prima*

**solo**.enterText((EditText) **solo**.getView(R.id.***testoInput***), **"idinputText!"**);

E’ possibile, infine, fare una osservazione riguardante la corrispondenza degli eventi tra i tre strumenti. In effetti, in alcuni casi non è semplice effettuare una generalizzazione in quanto i tre strumenti, pur svolgendo la stessa azione, ricercano l’elemento in maniera diversa. In particolare Robolectric necessita sempre dell’identificativo della risorsa, mentre, Espresso e Robotium possono ricavare un elemento anche dal testo che compare nell’elemento stesso. Un esempio di quanto detto è proprio l’evento di click su un elemento del menu. Quindi, al fine di rendere possibile una traduzione automatica da Espresso/Robotium in Robolectric è necessario inserire informazioni aggiuntive.

* 1. ***Progetto Traduttore Automatico***