Issue Android

ANALISI TRAMITE TOOL LINT E ARCHITETTURA DEDICATA

Buonocore Salvatore, Carotenuto Gennaro, Di Palo Arcangelo

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II, D.I.E.T.I.

1 Introduzione

Scopo del progetto è trarre informazioni inerenti i difetti più comuni che affliggono le Applicazioni Android.

L'analisi dei difetti è effettuata mediante **Lint**, tool per l'analisi statica del codice contenuto in Android Studio che può aiutare ad identificare e correggere problemi legati alla qualità strutturale dell'applicazione, e rilevare potenziali bug, senza doverla eseguire o dover scrivere casi di test.

Ogni problema (issue) riscontrato dal tool è riportato con una descrizione e un livello di severity, in modo da poter assegnare le opportune priorità ai miglioramenti da apportare. La flessibilità del tool consente di abbassare il livello di priorità di un problema per ignorare le issue che non sono rilevanti per il progetto , o aumentarlo per sottolineare problemi specifici.

Il tool Lint controlla i file sorgenti del progetto Android per potenziali bug e miglioramenti riguardanti: correttezza, security, performance, usability, accessibility e internationalization [1]. E' inoltre possibile scegliere quali check effettuare e quindi quali Issue analizzare [2] [3].

2 Procedure e modalità di misurazione

Questa sezione è dedicata all'illustrazione delle metodologie adottate per l'analisi, e più in generale al modo in cui la ricerca è stata condotta, partendo dalla fonte delle Applicazioni, dalla loro analisi per la ricerca delle Issue e la descrizione dell'Architettura con cui le issue sono state a loro volta analizzate.

2.1 Objects Selection

Sarà ora descritta la sorgente dati da cui sono state ricavate le Applicazioni.

Il campione in esame proviene da **F-Droid** [4], una repository (in gergo "app store") per applicazioni Android e progetti free software nata nel 2010.

Figura 2.1 - Logo repository F-Droid

Le Applicazioni possono essere installate o ne si può scaricare il codice sorgente sia mediante website che client app, anche senza un account di registrazione.

La repository F-Droid contiene circa 2300 applicazioni (in confronto Google Play Store ne contiene circa 1,43 milioni) tutte di dominio pubblico e liberamente scaricabili. Il progetto è gestito interamente da volontari e non è applicato un processo formale di app review se non quello dell'intera comunity.

Per l'analisi non è stata utilizzata l'intera repository ma esclusivamente un campione casuale di 844 applicazioni, quindi circa il 36%. Non essendo stato condotto uno studio statistico preciso non è possibile affermare quanto dei risultati presenti in tale analisi rispecchi l'intera popolazione di applicazioni, come ad esempio quelle presenti in Google Play Store.

2.2 Data Extraction

Tali applicazioni sono elaborate dal Tool Lint, in Figura 2.2 è mostrato come il tool processi i file sorgente:

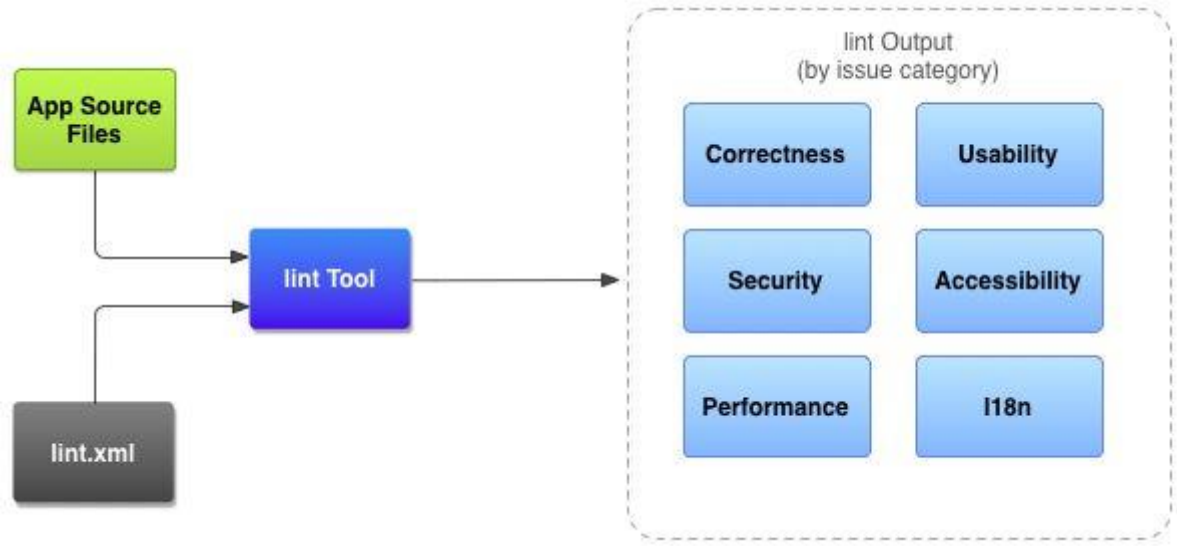


Figura 2.2 – Workflow di analisi del codice mediante Lint

INPUT

Gli **Application source files** corrisponde all'insieme di file che compongono il progetto Android, tra cui file java, XML, icone e file di configurazione. Il file **lint.xml** è un file di configurazione in cui l'utente può specificare: gli issue checks da escludere, i livelli di severity.

ELABORAZIONE

Il **Lint Tool** esamina i source files e utilizza il file di configurazione lint.xml per effettuare i checks richiesti. Se il progetto inlude build variants (per creare differenti versioni dell'applicazione da un singolo progetto) è possibile utilizzare un Gradle Wrapper per invocare Lint per tutte le varianti.

OUTPUT

Il risultato dell'intero processo consiste in una stampa, a video o su **file XML**, delle Issue dell'Applicazione in input [1]. Gli output saranno approfonditi nella Sezione 2.3.1 relativa alle fonti dati su cui è stata effettuata l'analisi. Dalle 844 Applicazioni analizzate sono state ricavate circa 78.000 issue.

2.3 Issue Analysis: Architettura

L'obiettivo della presente sezione è quello di definire **l'architettura generale del sistema utilizzato per effettuare le analisi**, partendo da quelle che sono le caratteristiche peculiari di ogni singolo tool utilizzato. Essa segue un pattern architetturale di tipo multi-layer in cui sono presenti i livelli funzionali di seguito descritti:

- **Data Source Layer**: contiene le sorgenti dati strutturate le cui informazioni dovranno essere integrate all'interno del sistema;
- **Integration Layer**: è costituito da un middleware capace, attraverso l'uso di connettori di tipo "plug and play", di interfaccairsi alle sorgenti dati ed estrarne le informazioni di interesse rappresentandole, attraverso apposite regole di mapping, in un formato coerente con il modello dei dati;
- **Knowledge Layer**: è la base di conoscenza del sistema, costituita da uno schema relazionale;

- **Analytics Layer**: consiste di una applicazione di data analytics per l'analisi e la presentazione delle informazioni;

Di seguito una semplice architettura degli strumenti utilizzati:

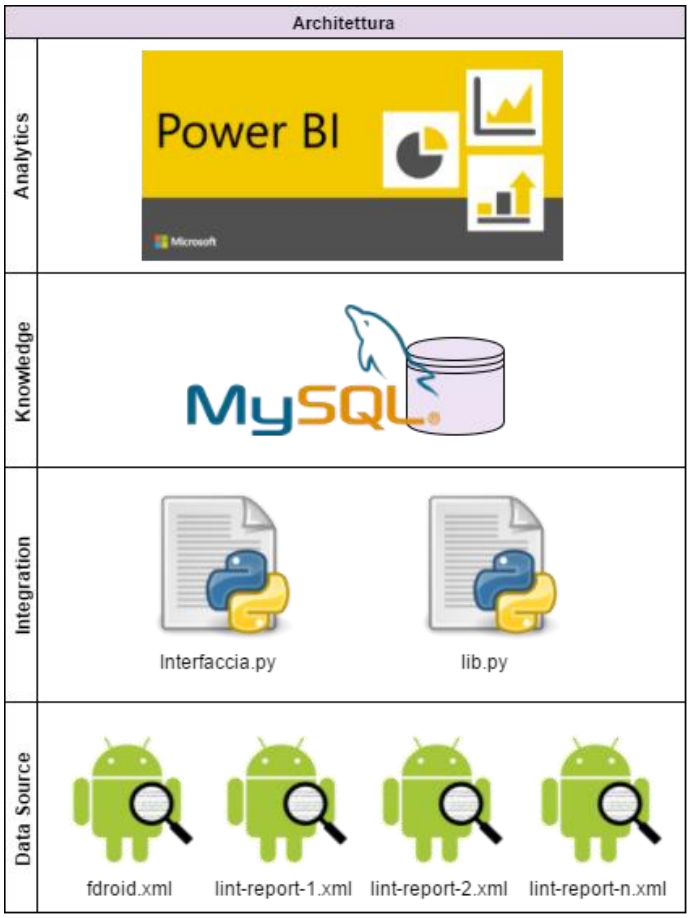


Figura 2.3 – Architettura della soluzione

Scopo dell'architettura creata è quello di analizzare in modo automatico, o comunque con piccole configurazioni, le Issue ricavate mediante Tool Lint.

Di seguito un'analisi approfondita di ogni Layer appena descritto.

2.3.1 Data Source Layer

Nella seguente sezione saranno approfondite le **sorgenti dati del sistema**, che corrispondono all'output del processo di analisi del Tool Lint. Tali sorgenti sono caratterizzate da due tipi di file XML con due strutture differenti: **file fdroid** e **file lint**.

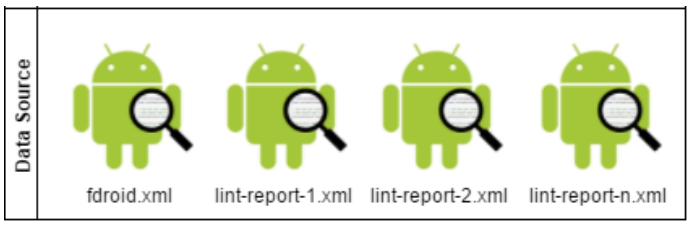


Figura 2.4 - Data Source Layer

2.3.1.1 File Fdroid

Il File F-Droid è un **unico file**, a fronte delle centinaia di file lint.xml, contenente informazioni relative alla Repository F-Droid da cui sono ricavete le applicazioni Andoid. Le informazioni riguardano le circa 2.000 applicazioni della Repository e sono così strutturate:

```
<fdroid>
      <application id="xxx">
            <package>
                  ...
            </package>
      </application>
      <application id="yyy">
            <package>
                  ...
            </package>
      </application>
</fdroid>
```

Blocco codice 2.1 - Struttura file fdroid.xml

Lo scheletro della gerarchia XML riguarda quindi le informazioni di:

- **application**: contiene vari campi con informazioni sulla singola applicazione
- **package**: memorizza informazioni relative ai vari package della singola applicazione, come ad esempio la versione.

di seguito un estratto più dettagliato del file fdroid, riguardante una singola applicazione:

```
<application id="de.j4velin.systemappmover">
      <id>de.j4velin.systemappmover</id>
      <added>2014-10-14</added>
      <lastupdated>2016-10-05</lastupdated>
      <name>/system/app mover</name>
      <summary>Add and remove system apps</summary>
      <icon>de.j4velin.systemappmover.172.png</icon>
      <desc>This app moves apps from and to the /system/app folder, making
them a system app or a user app. [...]
      </desc>
      <license>Apache-2.0</license>
      <categories>System</categories>
      <category>System</category>
      <web></web>
      <source>https://github.com/j4velin/SystemAppMover</source>
      <tracker>https://github.com/j4velin/SystemAppMover/issues</tracker>
      <marketversion>1.7.2</marketversion>
      <marketvercode>172</marketvercode>
      <requirements>root</requirements>
      <package>
            <version>1.7.2</version>
            <versioncode>172</versioncode>
            <apkname>de.j4velin.systemappmover_172.apk</apkname>
            <srcname>de.j4velin.systemappmover_172_src.tar.gz</srcname>
            <hash type="sha256">aebc378ef4a621b45f7362ab6b9[...]</hash>
            <size>1152636</size>
            <sdkver>7</sdkver>
            <targetSdkVersion>21</targetSdkVersion>
            <added>2016-10-05</added>
            <sig>59146dacaabff24fe6a051092235e78d</sig>
            <permissions>ACCESS_SUPERUSER</permissions>
      </package>
      <package>
            ...
      </package>
      <package>
            ...
      </package>
</application>f
```

Blocco codice 2.2 - Dettaglio struttura file fdroid.xml

saranno analizzate solo le informazioni principali ai fini dell'analisi delle Issue:

Nella sezione principale **application** troviamo:

- **id**: id univoco dell'applicazione all'interno della repository F-Droid;
- **added**: data di aggiunta dell'applicazione alla repository;

- **lastupdated**: data di ultimo aggiornamento dell'applicazione;
- **category**: categoria a cui appartiene l'applicazione;

Nella sottosezione **package** ritroviamo invece:

- **size**: grandezza dello specifico package dell'applicazione;
- **sdkver**: versione del Software Development Kit utilizzato per scrivere l'applicazione;
- **added**: data di aggiunta del package. La data di aggiunta dell'ultimo package corrisponde al campo lastupdated della sezione application;
- **permission**: permessi richiesti dall'applicaizione al sistema operativo Android.

2.3.1.2 File Lint

Per ogni applicazione della Repository è stato prodotto, mediante Tool Lint, un file lint con i risultati del Testing. La struttura di base è la seguente:

```
<issues format="4" by="lint 25.2.4">
    <issue>
        <location>
    </issue>
    <issue>
        <location>
        <location>
    </issue>
</issues>
```

Blocco codice 2.3 - Struttura file lint-report.xml

Le informazioni contenute nei vari campi riguardano:

- **issue**: informazioni relative al difetto trovato
- **location**: dove si è manifestato quel particolare difetto

Entrando nel dettaglio si mostra di seguito l'estrato di un'unica issue

```
<issue
      id="WrongViewCast"
      severity="Fatal"
      message="Unexpected cast to `ProgressBar`: layout tag was `SeekBar`"
      category="Correctness"
      priority="9"
      summary="Mismatched view type"
      explanation="Keeps track of the view types associated with ids and if it
finds a usage of the id in the Java code it ensures that it is treated as the
same type."
      errorLine1="    ProgressBar pbar = ((ProgressBar) findView-
ById(R.id.FREQ_input));"
      errorLine2="
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~">
      <location
            file="src\us\achromaticmetaphor\imcktg\ConfirmContacts.java"
            line="196"
            column="24"/>
</issue>
```

Blocco codice 2.4 - Dettaglio struttura file lint-report.xml

Lo schema della sezione principale "issue":

- **id**: nome dell'issue;
- **severity**: livello di severity dell'issue. Si ricorda che tale livello può essere modificato dall'utente;
- **category**: categoria dell'issue;
- **prioriy**: priorità di intervento;
- **explanation**: motivo probabile per cui si è verificata l'issue;

Scendendo nella gerarchia troviamo la sottosezione "location":

- **file**: file dell'intero progetto in cui si è riscontrata la issue.

2.3.2 Integration Layer

Il livello Integration si occupa della **migrazione da file XML a entità di tipo relazionale**, svolgendo quindi le basilari funzioni ETL.

Figura 2.5 - Integration Layer

In particolare automatizza il processo di data extraction dalle differenti data source e di data load all'interno del Layer Knowledge.

Come mostrato in Figura 2.5 è composto da due moduli software, **lib** e **interfaccia**:

- **interfaccia**: un'interfaccia testuale che consente di accedere alle funzioni di **lib** in modo rapido e ordinato.

- **lib**: si connette con il livello Data Storage ed effettua varie operazioni in modo automatico, tra cui l'import dei dati ed alcune query di verifica

Tali moduli costruiscono l'integrità dell'Integration Layer estrapolando le varie informazioni dalla gerarchia XML e traducendole in dati in formato SQL.

Per maggiori dettagli è possibile fare riferimento alla documentazione all'interno del modulo lib (consegnato con la documentazione).

2.3.2.1 Requisiti

Per la scrittura di entrambi i moduli stati utilizzati i seguenti software e librerie:

- Linguaggio di programmazione: Python version 2.7.13
- Ambiente di sviluppo: Notepad++
- Librerie aggiuntive non presenti nella distribuzione standard di Python:
  - Menu-2.0
  - prettytable-0.7.2
  - tqdm-master
  - pymysql
  - xml.etree.ElementTree

2.3.2.2 Utilizzo

All'interno del file *"lib.py"* sono indicati i vari **input da modificare** per una corretta esecuzione, per la maggiore si tratta di impostazioni tipo: nome utente, password del database e percorsi di catelle in cui reperire e salvare i dati. Una volta modificate le impostazioni è possibile interagire semplicemente lanciando *"menu.py"* da terminale (o con doppio click).

L'interfaccia, molto minimale, è composta da un Menu principale e due Sottomenu, uno per importare ed eliminare i dati dal database e uno per effettuare query in maniera automatica:

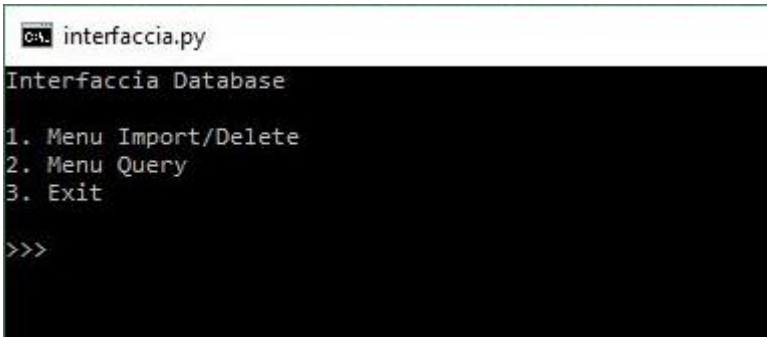


Figura 2.6 - Menu principale

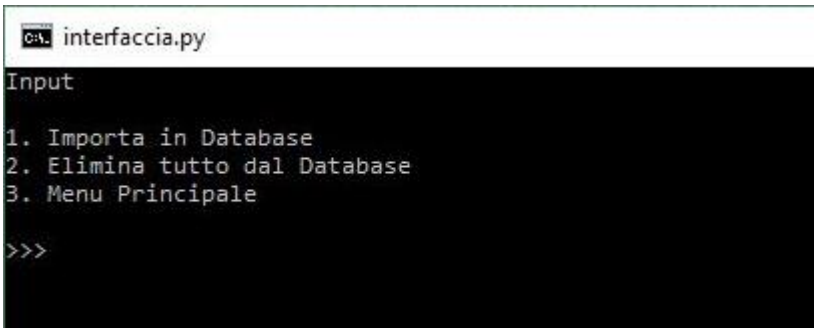


Figura 2.7 - Sottomenu per importare i dati

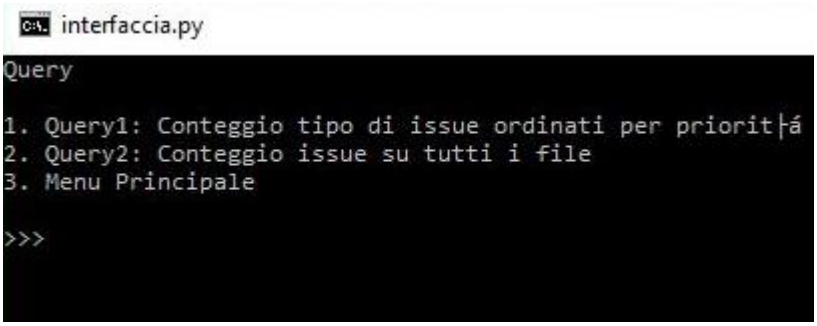


Figura 2.8 - Sottomenu Query

Le Query, dopo essere state mostrate a video, possono essere stampate in formato CSV per analisi future.

Nota: l'intera analisi è demandata all'Analytics Layer, quindi sono state create esclusivamente due query a puro scopo di esempio.

Nel **caso di errori** per la scrittura su un file di testo è presente una possibile soluzione nella documentazione interna al modulo *"lib.py",* nella funzione *"Query"*. L'errore indicato riguarda i permessi richiesti dal Database MySQL per accedere a determinati percorsi.

Si consiglia di memorizzare il file fdorid in una cartella separata dai file lint, altrimeni il programma importerà il file fdroid in modo errato. Inoltre **si consiglia** di eliminare (o spostare) i file lint dalla cartella una volta importati, in quanto non esiste alcun controllo sui file già presenti all'interno del database, quindi un import errato porterebbe a dei duplicati e quindi ad analisi errate.

2.3.3 Knowledge Layer

Il Knowledge Layer si occupa dello storage dei dati. Consiste di un Database MySQL.

Figura 2.9 - Knowledge Layer

Di seguito la struttura del Database e gli Schemi delle Tabelle:

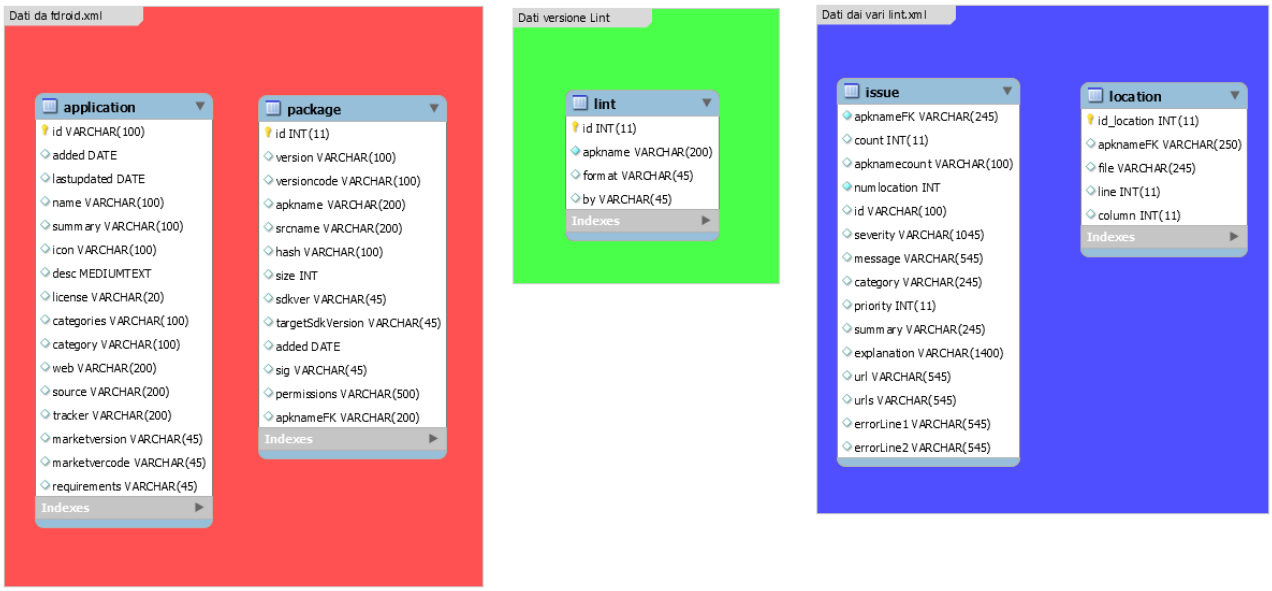


Figura 2.10 - ER Schema

Di seguito le Relazioni tra le Tabelle

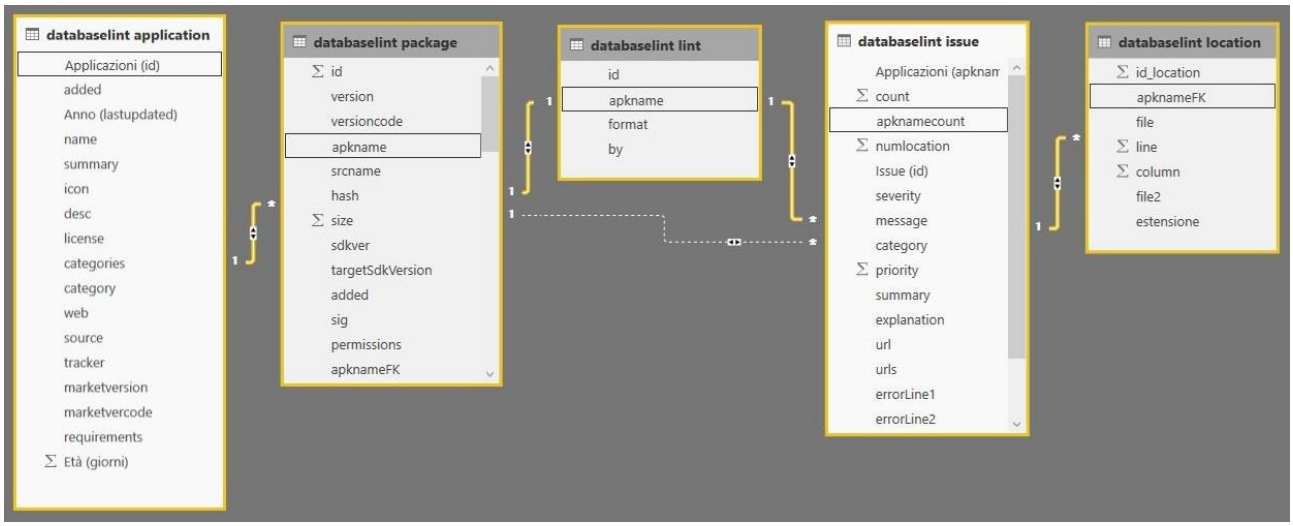


Figura 2.11 - Relazioni tra tabelle

La gestione è ovviamente rimandata al DBMS e l'import è effettuato mediante Integration Layer. Per motivi di ottimizzazione **non sono presenti vincoli di Foreign Key**, quindi le relazioni sono create ai livelli più alti dell'Architettura, in particolare nell'Analytics Layer.

2.3.3.1 Requisiti

Per il Database e la creazione degli Schemi sono state utilizzate le seguenti risorse:

- MySQL Community 5.7.17.0
- Schemi ER o Dump file

Per ricreare gli schemi è possibile importare il file *EER Schema* (consegnato con l'elaborato) all'interno del MySQL Workbench (File/Open Model). Aperto il file è possibile importare lo schema all'interno del Database (Database/Forward Engineer).

Per comodità è stato fornito anche un file, “Dump_20170430.sql” contenente una **copia dell’intero database** (strutture e dati) ed è possibile importarlo direttamente dal MySQL Workbench (Selezionare l’istanza del server dal pannello MySQL Connection, selezionare il menu Server/Data Import).

2.3.3.2 Utilizzo

Il Layer ETL ed i relativi moduli presuppongono che esistano gli Schemi tra le tabelle e che ovviamente il server MySQL sia attivo.

Per attivare il server bisogna lanciare il seguente comando da **Prompt dei comandi in modalità Amministratore:** "C:\Program Files\MySQL\MySQL Server 5.7\bin\mysqld" --defaults-file="C:\ProgramData\MySQL\MySQL Server 5.7\my.ini" --datadir="C:\ProgramData\MySQL\MySQL Server 5.7\Data" --console

Nota: “defaults-file” e “datadir” sono comandi per indicare manualmente il percorso dei file di impostazione, spesso non è necessario utilizzarli.

Non è necessario accedere direttamente al Database ma se si volesse accedere al Knowledge Layer, evitando l’interazione con l’Integration Layer, è possibile avviare la tradizionale interfaccia MySQL aprendo un normale **Prompt dei comandi** eseguendo il comando: "C:\Program Files\MySQL\MySQL Workbench 6.3 CE\mysql" -u root -p

Si fa notare che root è l’user di default creato per questa specifica istanza.

2.3.4 Analytics Layer

All’Analytics Layer è demandata l’intera analisi. A tale scopo è stato utilizzato il software di Business Intelligence Microsoft Power BI:

Figura 2.12 - Analytics Layer

Tale layer si occupa di collegarsi al Layer Data Storage e ricavare in automatico le informazioni per poter creare e aggiornare in real time i vari report.

Si sottolinea che non è necessario effettuare query di alcun tipo sul Database, le Tabelle sono importate con le relative Relazioni (se esistono, altrimenti sarà PowerBI a crearle) in modo automatico e i vari report sono realizzati “trascinando” le colonne all’interno dei grafici.

2.3.4.1 Requisiti

Per il Layer è necessaria la seguente risorsa:

- Microsoft PowerBI

Per ulteriori informazioni fare riferimento al sito ufficiale [5].

2.3.4.2 Utilizzo

Per creare nuovi report si deve inannzitutto **collegare il Layer Data Storage** (che sia MySQL o qualuque altra fonte di dati) a PowerBI (*Home/Recupera Dati/Altro/Database/Database MySQL*) inserendo semplicemente i dati per la connessione.

In caso di problemi con le credenziali del Database si può risovlere impostandole manualmetne: *File/Opzioni* e *Impostazioni/Impostazioni origine dati/Origine dati nel file corrente/Modifica autorizzazioni/Credenziali Modifica/passare da Windows a Database* e inserire le credenziali.

Terminata l'impostazine della Sorgente Dati, PowerBI si occuperà di collegarsi al Database ed importare le tabelle richieste per poter effettuare le varie analisi.

Per alcune analisi più specifiche è stato necessario normalizzare i dati. Tale processo non è immediato in PowerBI, si è dovuta quindi creare una nuova *"Misura"* come di seguito riportato:

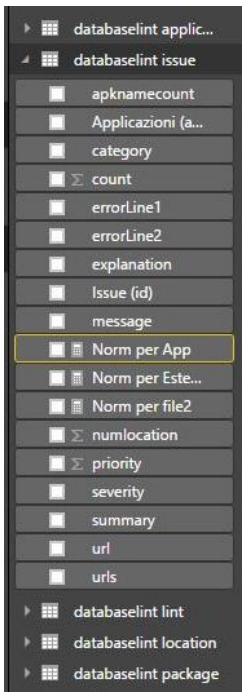


Figura 2.13 - PowerBI nuova misura

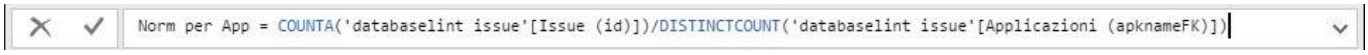


Figura 2.14 - PowerBI nuova misura, dettaglio

La nuova misura, chiamata *"Norm per App"* (Normalizzazizone per numero di Applicazioni) non fa altro che "contare il numero di issue e dividerlo per il numero di applicazioni ralazionate a quella issue". Può essere considerata come una colonna che viene creata dinamicamente al momento dell'utilizzo.

3 Casi di studio

Si ricorda che il soggetto dell'analisi sono le Issue delle applicazioni android e in questa sezione saranno descritti i vari casi di studio analizzati specificando obiettivi, metriche e risultati.

Per descrivere le metriche, e più in generale per fare riferimento ad un attributo, si utilizzerà la **notazione** *"nome tabella"."nome attributo"*: es. *issue.id* per riferirsi all'attributo id della tabella issue.

Di seguito gli obiettivi di ogni analisi:

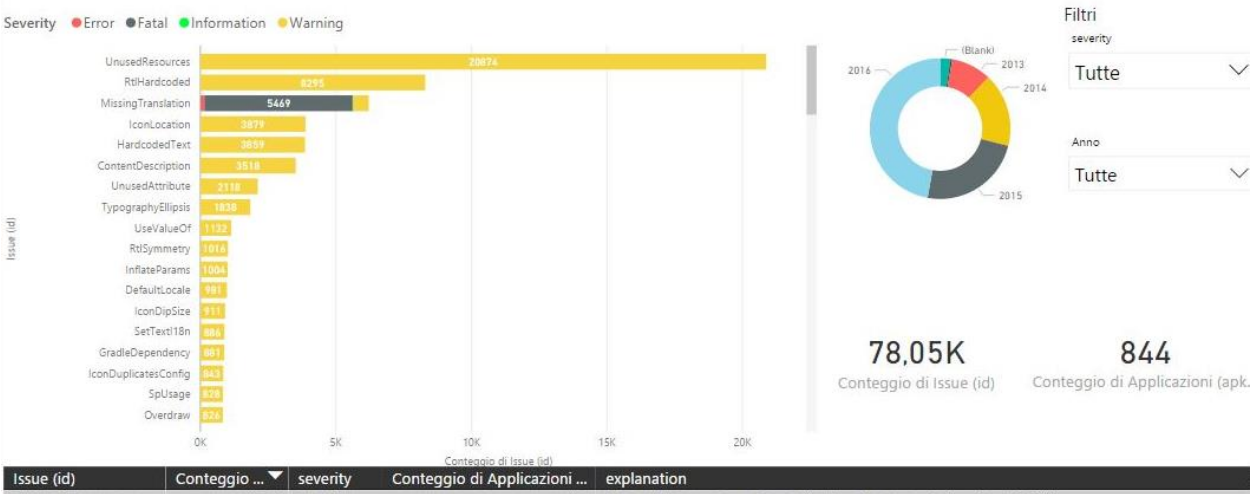
- **Analisi delle issue**: sarà effettuata una presentazione generale delle issue;
- **Andamento delle Issue negli anni**: si entrerà nel merito del trend temporale delle issue;
- **Età di una applicazione**: si terrà conto dell'andamento delle issue in base alla "maturità" di una applicazione;

- **Category**: si analizzeranno le issue considerando le varie categorie di applicazioni;
- **File ed Estensione**: si analizzerà "il luogo" in cui è stata rilevata la issue, ovvero il file del progetto Android;
- **Versione Android SDK**: analisi rivolta alle issue relazionate al framework Android SDK utilizzato per il progetto Android.

3.1 Analisi delle issue

Primo obiettivo dell'analisi è creare un **quadro generale delle Issue**, descrivendone la frequenza delle occorrenze, la categoria e la gravità, così come sono state riscontrate nelle applicazioni analizzate.

Dall'analisi sottostante è possibile ricavare alcune semplici informazioni:



Issue (id)	Conteggio ...	severity	Conteggio di Applicazioni ...	explanation
UnusedResources	20874	Warning	622	Unused resources make applications larger and slow down builds.
RtlHardcoded	7055	Warning	305	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in
MissingTranslation	5469	Fatal	145	If an application has more than one locale, then all the strings declared in one language shou
IconLocation	3879	Warning	241	The res/drawable folder is intended for density-independent graphics such as shapes define
ContentDescription	3518	Warning	342	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription`
HardcodedText	3285	Warning	278	Hardcoding text attributes directly in layout files is bad for several reasons: * When creating
UnusedAttribute	2028	Warning	325	This check finds attributes set in XML files that were introduced in a version newer than the c
TypographyEllipsis	1838	Warning	153	You can replace the string "..." with a dedicated ellipsis character, ellipsis character (..., &#823
RtlHardcoded	1240	Warning	44	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in

Figura 3.1 – Report: Issue più frequenti

E' possibile affermare che:

> dalle 844 applicazioni analizzate sono state riscontrate circa 78.050 issue, in media un centinaio di issue ad applicazione.

Entrando nel merito dell'analisi è possibile chiedersi:

3.1.1 Quali sono le issue più frequenti?

La prima informazione da apprendere riguarda proprio il **numero totale di Issue raccolte** e la loro caratterizzazione.

3.1.1.1 Metrica

Si terrà in considerazione semplicemente il numero totale di issue riscontrate all'interno delle varie applicazioni. Per ricavare il numero di issue è necessario fare riferimento ad un unico attributo:

- **Numero di issue**: informazione presente nel dato *issue.id*, ricavato dal file *lint-report.xml* come mostrato in Blocco codice 2.4. L'analisi corrisponde ad un semplice "group by and count".

Risultati più elaborati, come il numero di applicaizoni in cui è presente l'issue, sono stati ricavati tramite relazione con la tabella *application*.

3.1.1.2 Analisi

L'analisi farà riferimento al Report delle ssue più frequenti, si considerino quindi dei dettagli di Figura 3.1:

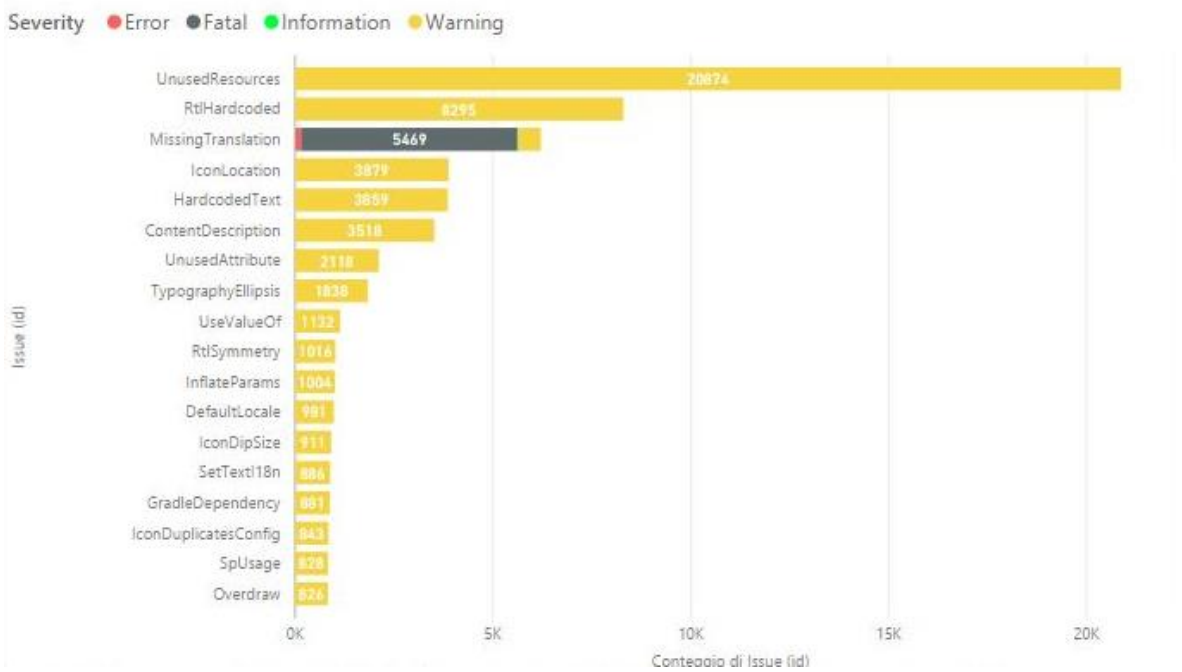


Figura 3.2 - Dettaglio Figura 3.1

L'istogramma orizzontale in figura sopra, mostra la classifica, ordinata per numero di occorrenze, delle Issue più frequenti. La tabella sottostante, invece, evidenzia le stesse informazioni ricavabili dall'istrogramma ma presenta più dettagli, tra cui: severity, numero di applicazioni e spiegazione dell'issue in questione.

Issue (id)	Conteggio ...	severity	Conteggio di Applicazioni ...	explanation
UnusedResources	20874	Warning	622	Unused resources make applications larger and slow down builds.
RtlHardcoded	7055	Warning	305	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in
MissingTranslation	5469	Fatal	145	If an application has more than one locale, then all the strings declared in one language shou
IconLocation	3879	Warning	241	The res/drawable folder is intended for density-independent graphics such as shapes defined
ContentDescription	3518	Warning	342	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription`
HardcodedText	3285	Warning	278	Hardcoding text attributes directly in layout files is bad for several reasons: * When creating
UnusedAttribute	2028	Warning	325	This check finds attributes set in XML files that were introduced in a version newer than the c
TypographyEllipsis	1838	Warning	153	You can replace the string "..." with a dedicated ellipsis character, ellipsis character (..., &#823
RtlHardcoded	1240	Warning	44	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in

Figura 3.3 - Dettaglio Figura 3.1

A valle dell'analisi si può quindi affermare che

> l'issue in assoluto con più occorrenze è **UnusedResources** con **20.874** issue in **622** applicazioni. Quindi circa il **27%** delle issue appartengono a questa tipologia e circa il **74%** delle applicazioni analizzate presentano questo difetto.

Seguono in classifica:

- **RtlHardoded** con 7.055 occorrenze per 305 applicaizoni, circa il 9% delle issue totali, di gran lunga inferiore al primo;
- **MissingTranslation** con 5.469 per 145 applicaizoni.

Si nota quindi l'enorme distacco tra la prima e la seconda issue più frequenti.

Per approfondimenti sulle tipologie di issue si faccia riferimento alla Tabella 5.1.

3.1.1.3 Conclusioni

Da una prima e semplice analisi è evidente che la maggioranza delle applicazioni sono affette da problemi che tendono a causare un **sovradimensionamento dell'applicazione** (UnusedResource) e altri problemi che riguardano la visualizzazione del testo (RtlHardcoded) e la mancata traduzione di alcune stringhe in varie lingue (MissingTranslation).

3.1.2 Quali sono le severity più frequenti?

Altra informazione interessante riguarda la Severity delle varie Issue raccolte.

3.1.2.1 Metrica

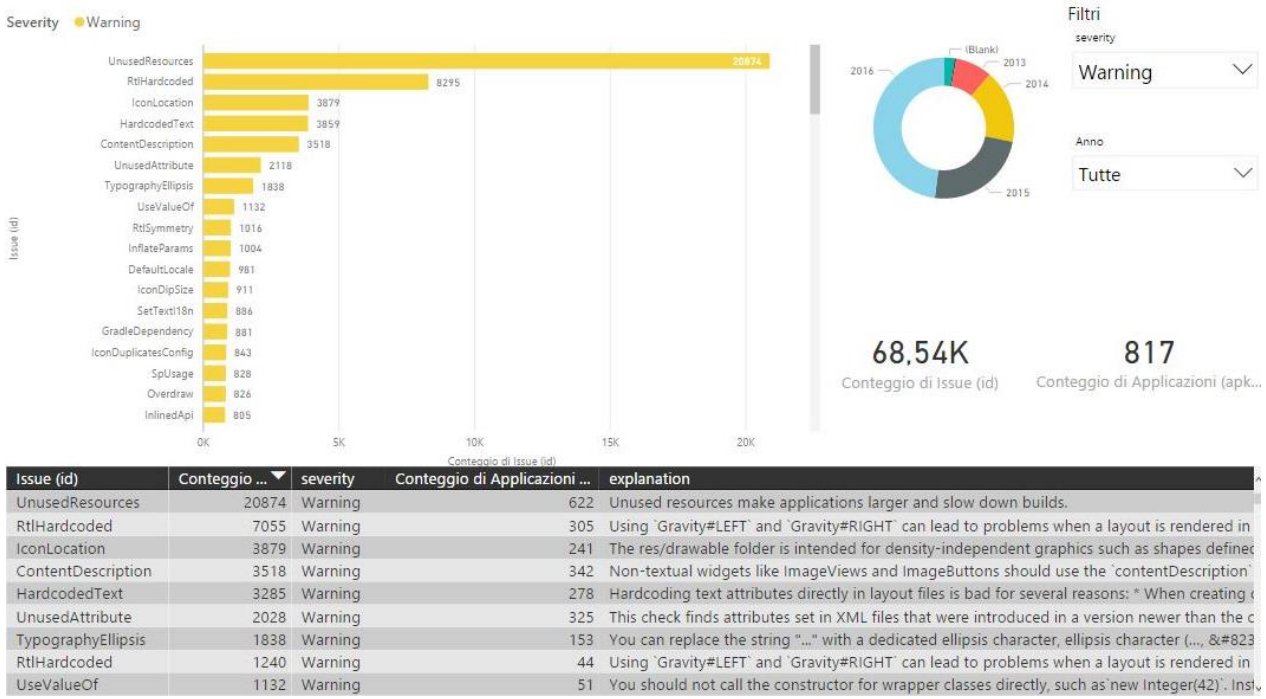
La metrica utilizzata in questo caso è la stessa della Sezione 3.1.1.1, si differenziano per l'applicazione del filtro Severity, in modo da visualizzare esclusivamente le issue più frequenti del tipo di interesse per l'analisi.

3.1.2.2 Analisi

L'analisi è stata condotta allo stesso modo per tutte le Severity, si ritiene necessaria una introduzione esplicativa del Report generale:

Il Report, molto simile a quello mostrato in Figura 3.1, mostra esclusivamente le Issue appartenenti ad un determinato tipo di *Severity*: ad esempio nell'immagine sottostante sono evidenziate solamente le Issue di tipo **Warning**, in giallo nell'istrogramma orizzontale. Il grafico ad anello, sulla destra, mostra la quantità della specifica Severity verificatasi negli anni (analisi che sarà approfondita nella Sezione 3.2) e immediatamente sotto sono mostrati due valori che rappresentano: il numero di issue appartententi a quella Severity; il numero di applicazioni che presentano almeno una issue di quel tipo di Severity. Infine è mostrata la tabella con le informazioni supplementari per l'analisi.

Di seguito le analisi:



Issue (id)	Conteggio ...	severity	Conteggio di Applicazioni ...	explanation
UnusedResources	20874	Warning	622	Unused resources make applications larger and slow down builds.
RtlHardcoded	7055	Warning	305	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in
IconLocation	3879	Warning	241	The res/drawable folder is intended for density-independent graphics such as shapes definec
ContentDescription	3518	Warning	342	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription`
HardcodedText	3285	Warning	278	Hardcoding text attributes directly in layout files is bad for several reasons: * When creating (
UnusedAttribute	2028	Warning	325	This check finds attributes set in XML files that were introduced in a version newer than the c
TypographyEllipsis	1838	Warning	153	You can replace the string "..." with a dedicated ellipsis character, ellipsis character (..., &#823
RtlHardcoded	1240	Warning	44	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in
UseValueOf	1132	Warning	51	You should not call the constructor for wrapper classes directly, such as`new Integer(42)`. Inst

Figura 3.4 - Report: Severity Warning

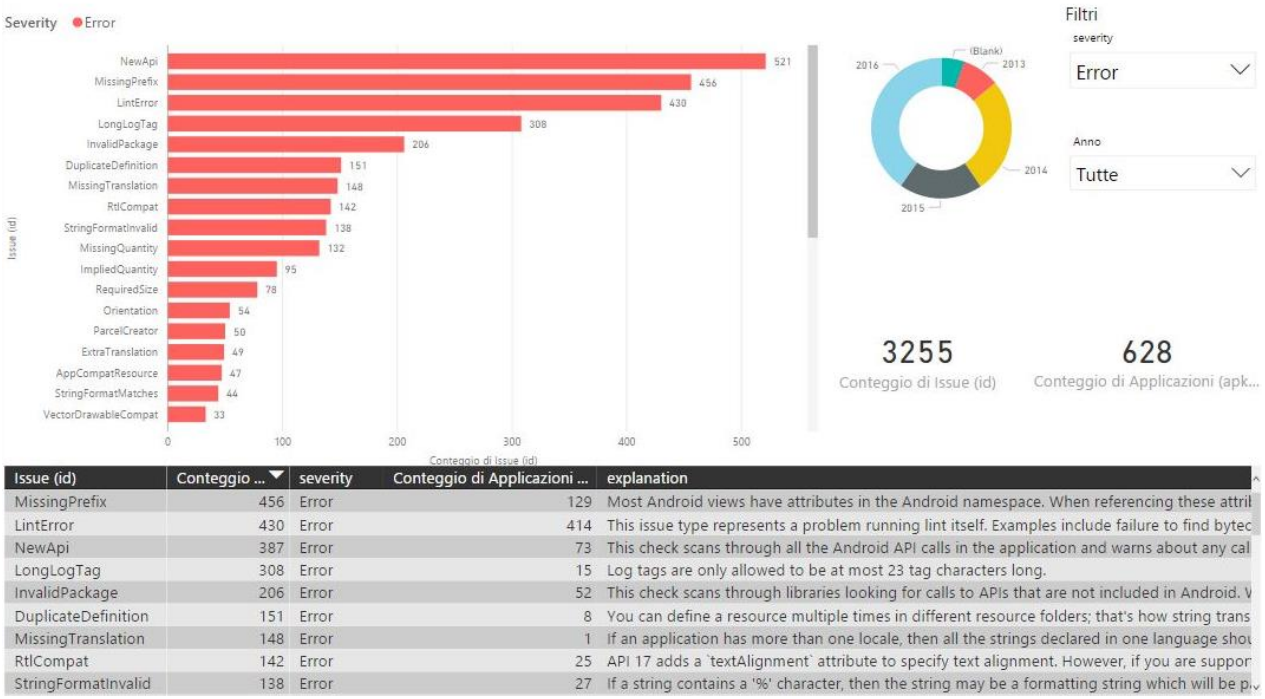
L'analisi mostra che l'unica differenza con la Figura 3.1 è nell'assenza di **MissingTraslation** dalla classifica, che è principalmente di tipo *Fatal* ed *Error*, quindi probabilmente si trova più in basso e non è visualizzabile nell'immagine.

Data la somiglianza con un'analisi già effettuata non si approfondirà la descrizione delle issue. Si ritengono, invece, rilevanti gli indici statistici di seguito riportati:

> E' possibile riscontrare che circa 68.540 issue appartengono alla severity Warning, ovvero circa l'88% del totale delle issue. Inoltre 817 delle applicazioni analizzate presentano questo tipo di severity, ovvero il 97%.

Questa informazione fa concludere che 27 applicazioni (3%) contengono esclusivametne severity di tipo Error, Fatal o Information.

La prossima severity analizzata è Error:



Issue (id)	Conteggio ...	severity	Conteggio di Applicazioni ...	explanation
MissingPrefix	456	Error	129	Most Android views have attributes in the Android namespace. When referencing these attril
LintError	430	Error	414	This issue type represents a problem running lint itself. Examples include failure to find bytec
NewApi	387	Error	73	This check scans through all the Android API calls in the application and warns about any cal
LongLogTag	308	Error	15	Log tags are only allowed to be at most 23 tag characters long.
InvalidPackage	206	Error	52	This check scans through libraries looking for calls to APIs that are not included in Android. V
DuplicateDefinition	151	Error	8	You can define a resource multiple times in different resource folders; that's how string trans
MissingTranslation	148	Error	1	If an application has more than one locale, then all the strings declared in one language shou
RtlCompat	142	Error	25	API 17 adds a `textAlignment` attribute to specify text alignment. However, if you are suppor
StringFormatInvalid	138	Error	27	If a string contains a '%' character, then the string may be a formatting string which will be p.

Figura 3.5 - Report: Severity Error

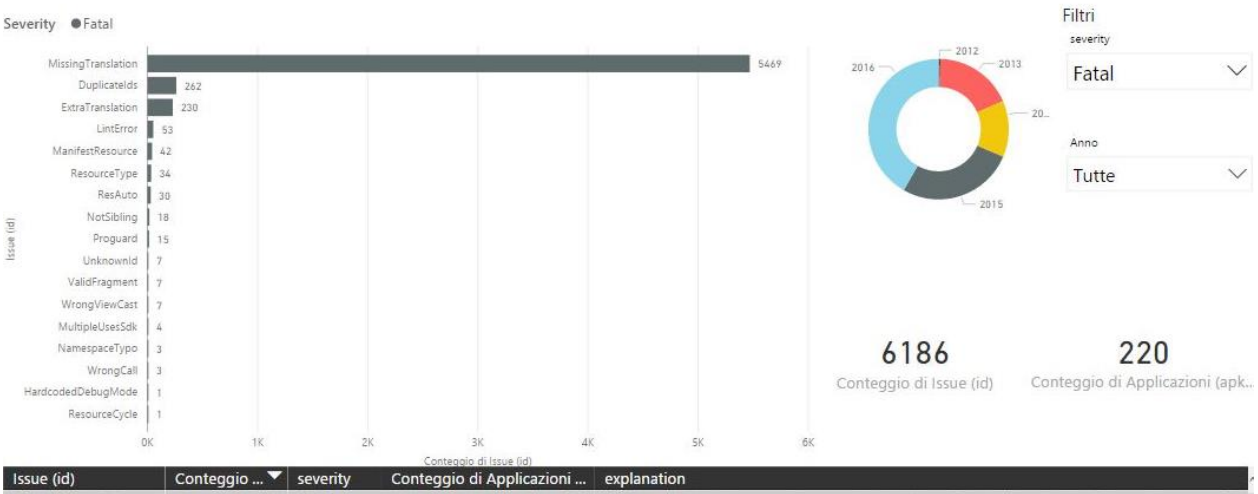
La Severity Error riguarda 3.255 issue (4%) e 628 applicazioni (74%), valori di gran lunga inferiori alla Severity di tipo Warning, almeno per quanto riguarda il numero di Issue.

A differenza dei Warning, in questo caso è interessante entrare nel dettaglio delle singole issue, guardando l'istogramma orizzontale (in alto a sinistra) si può riscontrare che:

> L'Error più comune è NewApi con 521 issue (16% degli Error e 0,6% delle issue totali) presenti in 73 applicazioni, seguito da MissingPrefix con 456 issue in 129 applicazioni e LongLogTag con 308 issue in 15 applicazioni.

Si nota una incongruenza tra istogramma orizzontale e tabella (in basso), probabilmente dovuta al fatto che alcune Issue hanno Explanation diversa, quindi sono raggruppate in modo diverso.

Il prossimo Report riguarderà le Severity di tipo Fatal:



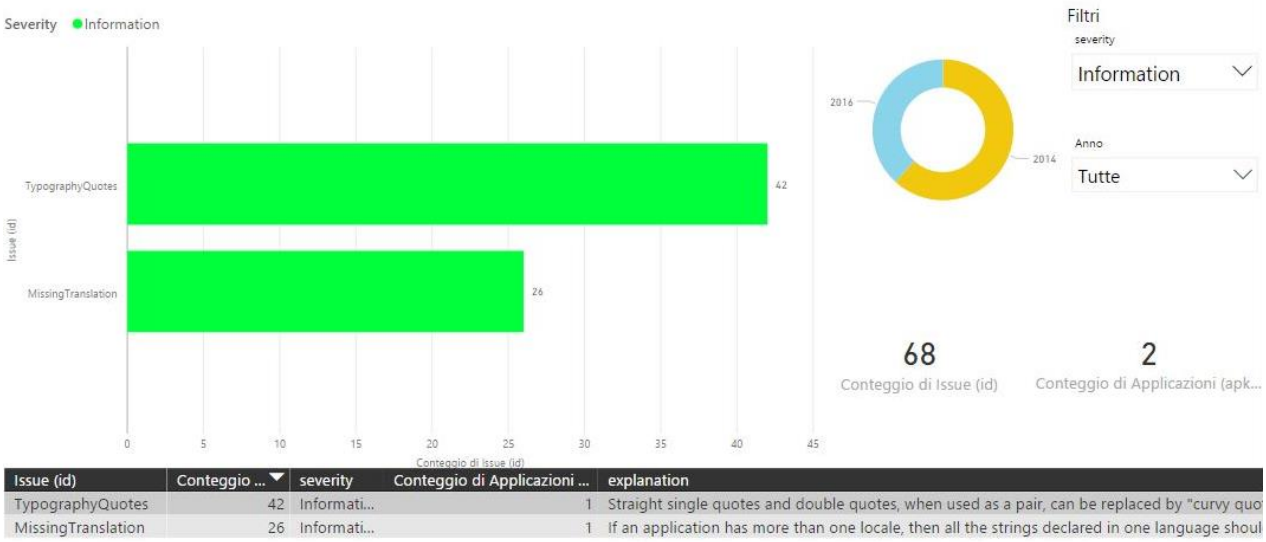
Issue (id)	Conteggio ...	severity	Conteggio di Applicazioni ...	explanation
MissingTranslation	5469	Fatal	145	If an application has more than one locale, then all the strings declared in one language shou
DuplicateIds	262	Fatal	21	Within a layout, id's should be unique since otherwise `findViewById()` can return an unexpec
ExtraTranslation	230	Fatal	35	If a string appears in a specific language translation file, but there is no corresponding string
LintError	53	Fatal	30	This issue type represents a problem running lint itself. Examples include failure to find bytec
ManifestResource	42	Fatal	5	Elements in the manifest can reference resources, but those resources cannot vary across cor
ResourceType	34	Fatal	11	Ensures that resource id's passed to APIs are of the right type; for example, calling `Resource
ResAuto	30	Fatal	2	In Gradle projects, the actual package used in the final APK can vary; for example,you can ad
NotSibling	18	Fatal	10	Layout constraints in a given `RelativeLayout` should reference other views within the same r
Proguard	15	Fatal	14	Using `-keepclasseswithmembernames` in a proguard config file is not correct; it can cause s

Figura 3.6 - Report: Severity Fatal

Si riscontra, sorprendentemente, un significativo aumento nel numero di Issue di tipo Fatal rispetto alle Issue di tipo Error, sono infatti circa il doppio, localizzate però in un minor numero di applicazioni.

La Severity di tipo Fatal in assoluto con più occorrenze è **MisingTranslation** con 5.469 issue ovvero l'88% di tutti i Fatal e il 7% di tutte le issue. In classifica si trovano anche **DuplicateIds** con 262 occorrenze ed **ExtraTranslation** con 230 occorrenze.

L'ultima Severity da analizzare riguarda le issue di tipo Information



Issue (id)	Conteggio ...	severity	Conteggio di Applicazioni ...	explanation
TypographyQuotes	42	Informati...	1	Straight single quotes and double quotes, when used as a pair, can be replaced by "curvy quo
MissingTranslation	26	Informati...	1	If an application has more than one locale, then all the strings declared in one language shoul

Figura 3.7 - Report: Severity Information

Le uniche due issue presenti in questa categoria sono TypographyQuotes e MissingTranslation che probabilmente è presente in quanto alcune di queste issue non sono propriamente dei Fatal, sarebbe interessante visualizzare l'applicazione in esame e vedere il comportamento di tali linee di codice.

3.1.2.3 Conclusioni

A valle dell'analisi è possibile affermare che le Severity più frequenti sono quelle di tipo Warning, seguite da Fatal, Error ed Information. In particolare:

- I Warning corrispondono praticametne alle issue viste nella sezione 3.1.1 riguardanti le Issue più frequenti;
- Error, essendo in netta minoranza, non sono presenti nella sezione 3.1.1, infatti si ritrovano issue inerenti possibili malfunzionamenti su versioni di android non più supportate dalle API all'interno dell'applicazione (NewApi) o difetti di visualizzazione a causa di mancati namespace utilizzati per gli attributi (MissingPrefix).;
- I Fatal, invece, riguardano possibili crash dell'applicazione, in particolare problemi di traduzione con stringhe non tradotte in altre lingue (MissingTraslation) o addirittura stringhe tradotte ma non presenti nella lingua originale (ExtraTranslation) che se visualizzate nelle altre lingue causerebbero un crash, in quanto non esiste fisicamente la linea di codice a esse collegata, oppure id non univoci per l'identificazione delle view che provocano ambiguità nelle chiamate (DuplicateIds).

3.1.3 Quali sono le categorie di Issue più frequenti?

Il tool Lint suddivide le Issue per *Category*, tale suddivisione consente di riassumere la tipologia dell'issue identificandola, ad esempio, come una problematica di Prestazione, piùttosto che di Correttezza o Sicurezza.

Obiettivo dell'analisi è apprendere quali siano le Categorie di Issue più frequenti all'interno delle applicazioni Android.

3.1.3.1 Metrica

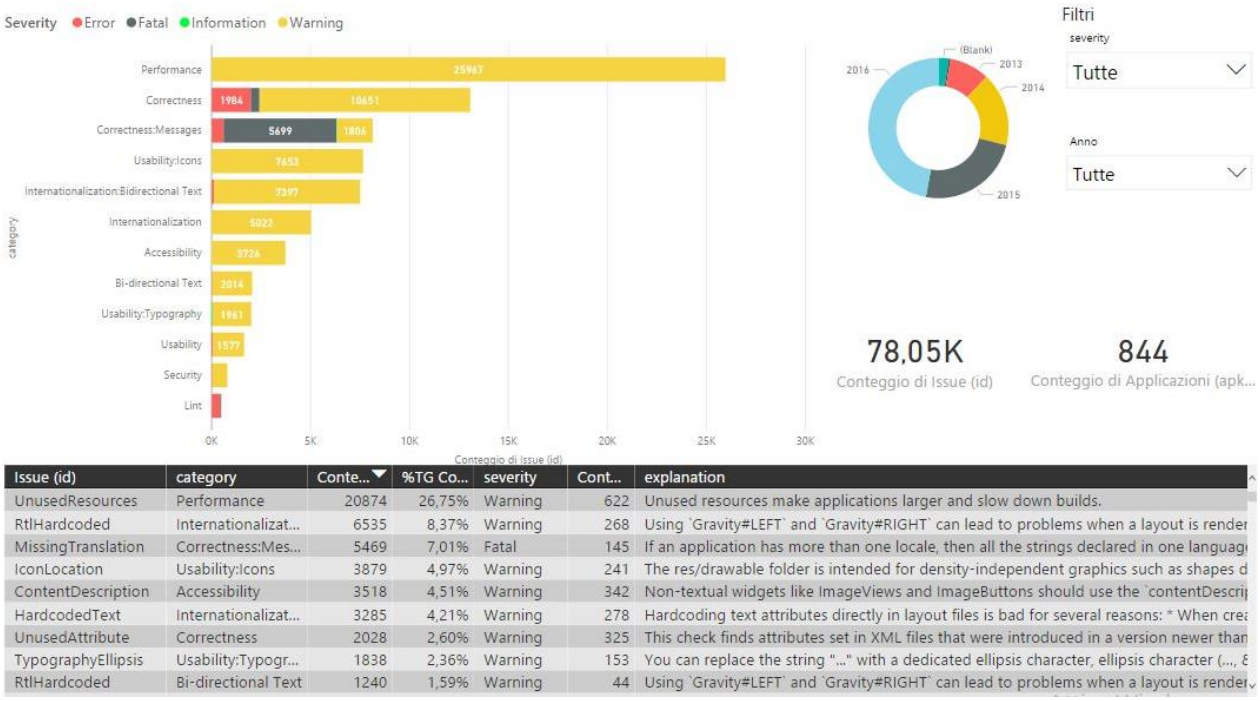
In questo caso è stata utilizzata una metrica molto simile a quella già vista in per 3.1.1.1. Per ricavare il numero di issue e la categoria si è fatto riferimento ai seguenti dati:

- **Categoria delle issue**: informazione presente nel dato issue.category, ricavato dal file lint-report.xml come mostrato in Blocco codice 2.4.
- **Numero di issue**: informazione presente nel dato issue.id, ricavato dal file lint-report.xml come mostrato in Blocco codice 2.4. L'analisi corrisponde ad un semplice "group by and count".

Risultati più elaborati, come il numero di applicaizoni in cui è presente l'issue, sono stati ricavati tramite relazione con la tabella application.

3.1.3.2 Analisi

Come mostrato dalla Figura sottostante, la tipologia di Report resta fondamentalmente la stessa per questa prima parte dei casi di studio.



Issue (id)	category	Conte...	%TG Co...	severity	Cont...	explanation
UnusedResources	Performance	20874	26,75%	Warning	622	Unused resources make applications larger and slow down builds.
RtlHardcoded	Internationalizat...	6535	8,37%	Warning	268	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is render
MissingTranslation	Correctness:Mes...	5469	7,01%	Fatal	145	If an application has more than one locale, then all the strings declared in one languag
IconLocation	Usability:Icons	3879	4,97%	Warning	241	The res/drawable folder is intended for density-independent graphics such as shapes d
ContentDescription	Accessibility	3518	4,51%	Warning	342	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescrip
HardcodedText	Internationalizat...	3285	4,21%	Warning	278	Hardcoding text attributes directly in layout files is bad for several reasons: * When crea
UnusedAttribute	Correctness	2028	2,60%	Warning	325	This check finds attributes set in XML files that were introduced in a version newer than
TypographyEllipsis	Usability:Typogr...	1838	2,36%	Warning	153	You can replace the string "..." with a dedicated ellipsis character, ellipsis character (..., &
RtlHardcoded	Bi-directional Text	1240	1,59%	Warning	44	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is render

Figura 3.8 - Report: Category

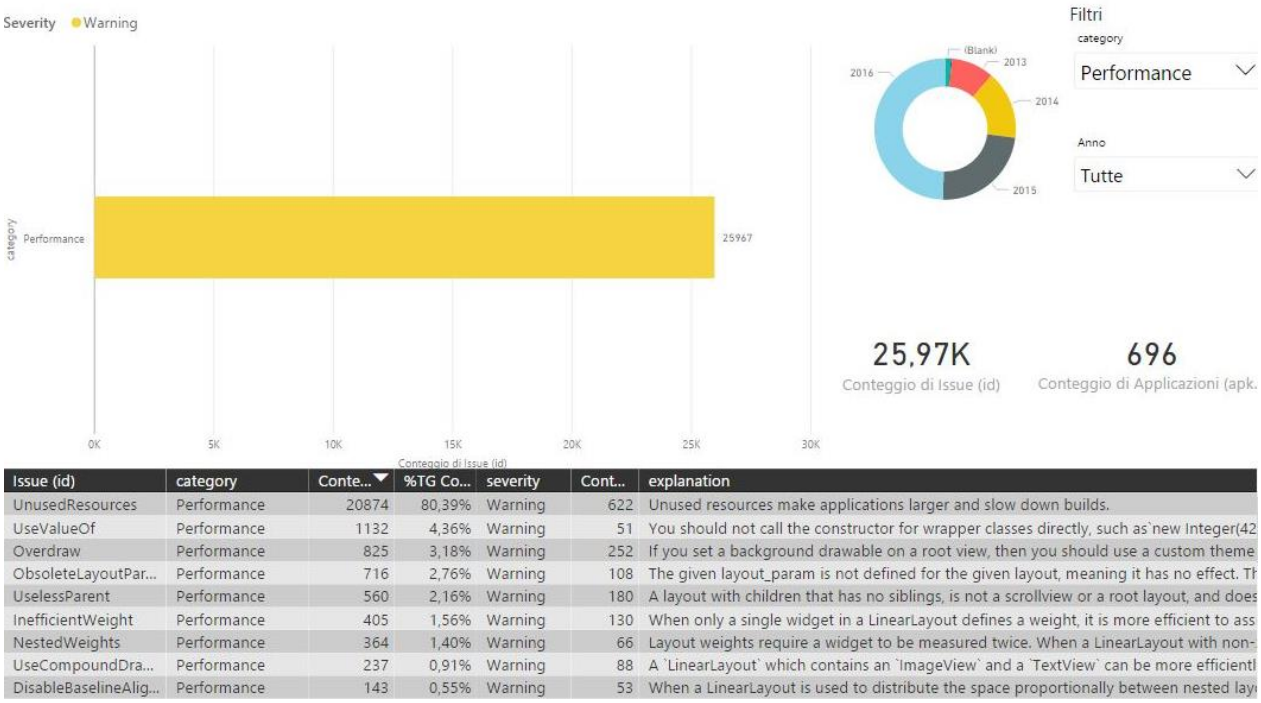
Dalla figura è possibile apprendere che

> la Categoria di issue più diffusa è **Performance** con 25.967 issue (33%), seguita da **Correctness** con 13.070 issue (17%) e da **Correctness: Messages** (10%)

quindi le Applicazioni Android soffrono principalmente di problemi legati alla mancata ottimizzazione del codice o problemi di correttezza legati alla particolare programmazione.

L'anlisi entrerà ora nel merito delle Categoria di Issue più diffuse per degli approfondimenti. I Report mostrati di seguito sono ricavati dal Report in Figura 3.8 applicando i filtri appropriati ai dati.

Il primo dettaglio fa riferimento alle Issue di tipo Performance:



Issue (id)	category	Conte...	%TG Co...	severity	Cont...	explanation
UnusedResources	Performance	20874	80,39%	Warning	622	Unused resources make applications larger and slow down builds.
UseValueOf	Performance	1132	4,36%	Warning	51	You should not call the constructor for wrapper classes directly, such as`new Integer(42
Overdraw	Performance	825	3,18%	Warning	252	If you set a background drawable on a root view, then you should use a custom theme
ObsoleteLayoutPar...	Performance	716	2,76%	Warning	108	The given layout_param is not defined for the given layout, meaning it has no effect. Th
UselessParent	Performance	560	2,16%	Warning	180	A layout with children that has no siblings, is not a scrollview or a root layout, and does
InefficientWeight	Performance	405	1,56%	Warning	130	When only a single widget in a LinearLayout defines a weight, it is more efficient to ass
NestedWeights	Performance	364	1,40%	Warning	66	Layout weights require a widget to be measured twice. When a LinearLayout with non-
UseCompoundDra...	Performance	237	0,91%	Warning	88	A `LinearLayout` which contains an `ImageView` and a `TextView` can be more efficientl
DisableBaselineAlig...	Performance	143	0,55%	Warning	53	When a LinearLayout is used to distribute the space proportionally between nested lay

Figura 3.9 - Report: Category Performance

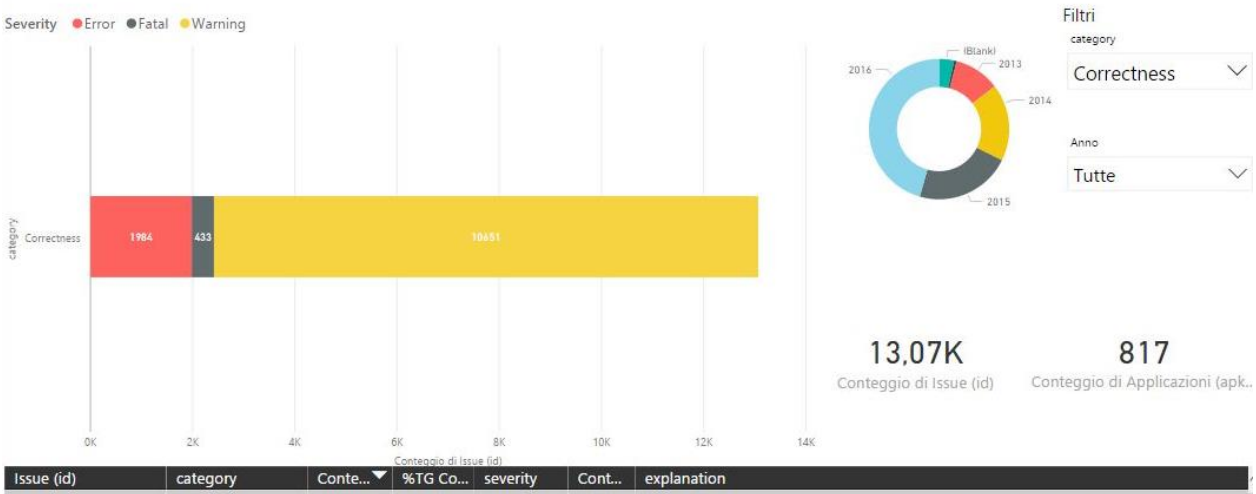
In figura è possibile riscontrare i calssigi oggetti visivi utilizzati fino ad ora: l'istogramma mostra la quantità di issue di tipo Performance presenti nel datasset, mentre dalla tabella è possibile ricavare utleriorio approfondimenti.

A valle dell'analisi si può affermare che

> le issue più diffuse della Categoria Performance sono: **UnusedResources** con 20.874 issue (27%), **UseValueOf** con 1.132 issue (1,4%), **Overdraw** con 825 issue (1%), **ObsoleteLayoutParam** con 716 issue (0,9%)

inoltre è interessante notare che sono coinvolte 696 applicazioni, pari all'82%! Quindi **la maggior parte delle Applicazioni analizzate soffre di problemi legati alle Performance**. Per approfondimenti sul tipo di Issue è possibile fare riferimento alla Tabella 5.1.

Si entrerà ora nel merito delle Issue di tipo Correctness,



Issue (id)	category	Conte...	%TG Co...	severity	Cont...	explanation
UnusedAttribute	Correctness	2028	15,52%	Warning	325	This check finds attributes set in XML files that were introduced in a version newer than
InflateParams	Correctness	1004	7,68%	Warning	246	When inflating a layout, avoid passing in null as the parent view, since otherwise any la
DefaultLocale	Correctness	980	7,50%	Warning	152	Calling `String#toLowerCase()` or `#toUpperCase()` *without specifying an explicit locale
GradleDependency	Correctness	881	6,74%	Warning	322	This detector looks for usages of libraries where the version you are using is not the cu
SpUsage	Correctness	828	6,34%	Warning	121	When setting text sizes, you should normally use `sp`, or "scale-independent pixels". Th
InlinedApi	Correctness	804	6,15%	Warning	122	This check scans through all the Android API field references in the application and flag
OldTargetApi	Correctness	734	5,62%	Warning	667	When your application runs on a version of Android that is more recent than your `targ
MissingPrefix	Correctness	456	3,49%	Error	129	Most Android views have attributes in the Android namespace. When referencing these
CommitPrefEdits	Correctness	432	3,31%	Warning	113	After calling `edit()` on a `SharedPreference`, you must call `commit()` or `apply()` on the

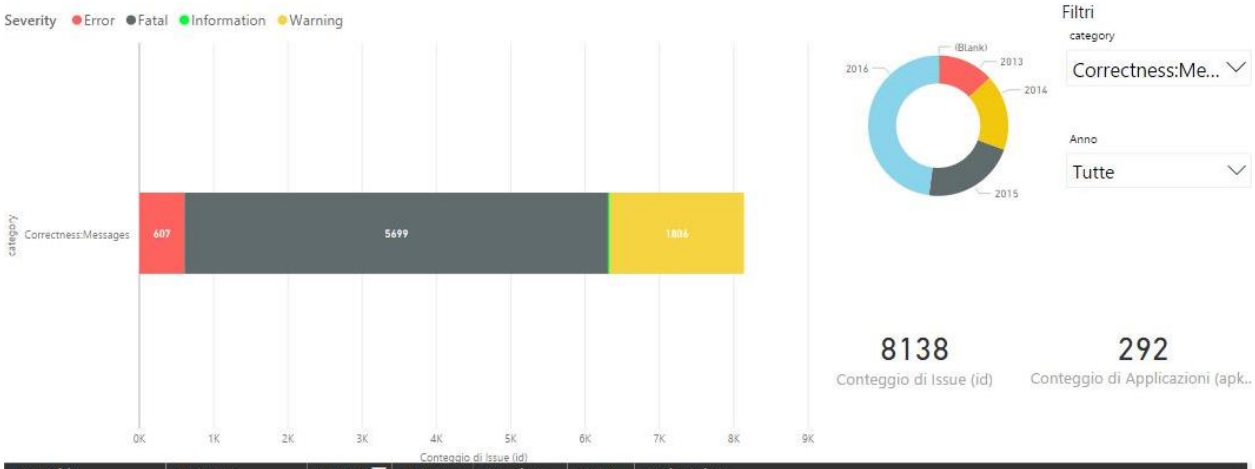
Figura 3.10 - Report: Category Correctness

Dall'analisi soprastante è possibile ricavare alcune informazioni,

> le issue più diffuse della Categoria Correctness sono: **UnusedAttribute** con 2.028 issue (3%), **InflateParams** con 1.004 issue (1,3%), **DefaultLocale** con 980 issue (1,2%)

e anche in questo caso le applicazioni coinvolte sono tantissime, pari al 97%!

Di seguito le issue di tipo Correctness:Messages, ovvero una sottocategoria di Issue del tipo Correctness.



Issue (id)	category	Conte...	%TG Co...	severity	Cont...	explanation
MissingTranslation	Correctness:Mes...	5469	67,20%	Fatal	145	If an application has more than one locale, then all the strings declared in one languag
MissingTranslation	Correctness:Mes...	568	6,98%	Warning	8	If an application has more than one locale, then all the strings declared in one languag
Typos	Correctness:Mes...	367	4,51%	Warning	125	This check looks through the string definitions, and if it finds any words that look like li
LogNotTimber	Correctness:Mes...	323	3,97%	Warning	11	Since Timber is included in the project, it is likely that calls to Log should instead be go
ExtraTranslation	Correctness:Mes...	230	2,83%	Fatal	35	If a string appears in a specific language translation file, but there is no corresponding
UnusedQuantity	Correctness:Mes...	187	2,30%	Warning	11	Android defines a number of different quantity strings, such as `zero`, `one`, `few` and `r
PluralsCandidate	Correctness:Mes...	157	1,93%	Warning	71	This lint check looks for potential errors in internationalization where you have translat
StringFormatCount	Correctness:Mes...	152	1,87%	Warning	40	When a formatted string takes arguments, it usually needs to reference the same argur
MissingTranslation	Correctness:Mes...	148	1,82%	Error	1	If an application has more than one locale, then all the strings declared in one languag

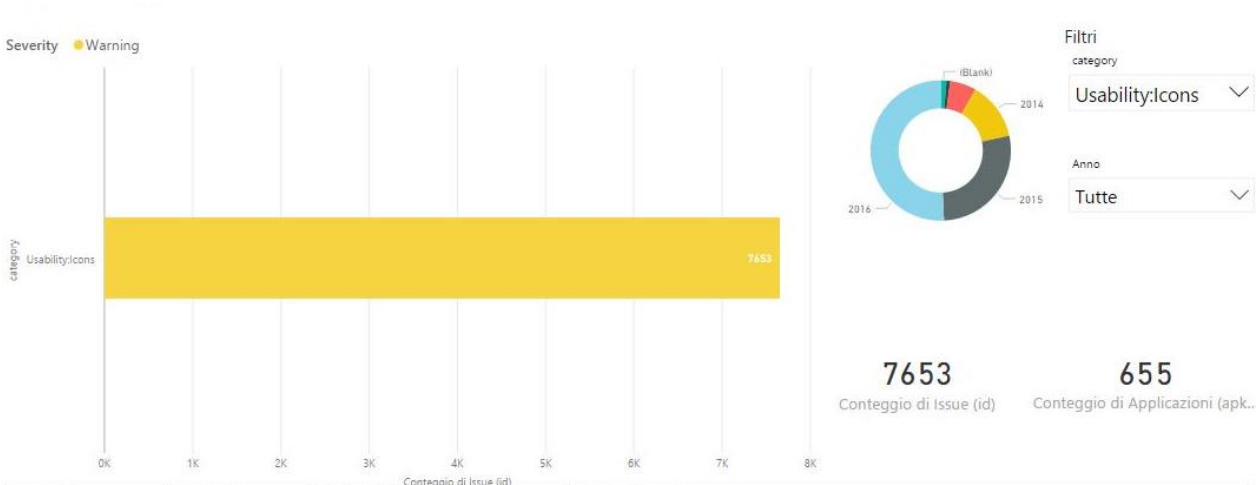
Figura 3.11 - Report: Category Correctness:Messages

dalla figura è possibile dedurre che

> le issue più diffuse della Categoria Correctness:Messages sono: **MissingTranslation** con 6.037 issue (8%), **Typos** con 367 issue (0,5%), **LogNotTimber** con 323 issue (0,4%).

In questo caso il numero di applicazioni è di gran lunga ridotto, pari circa a 300, ma bisogna sottolineare ce si tratta pur sempre di una categoria di Correctness e che, volendo sommare i valori alla categoria principale, si potrebbe addirittura superare la tipologia Performance.

L'ultima analisi riguarda la Category Usability, riferita principalmente all'esperienza dell'utente:



Issue (id)	category	Conte...	%TG Co...	severity	Cont...	explanation
IconLocation	Usability:Icons	3879	50,69%	Warning	241	The res/drawable folder is intended for density-independent graphics such as shapes d
IconDipSize	Usability:Icons	911	11,90%	Warning	115	Checks the all icons which are provided in multiple densities, all compute to roughly th
IconDuplicatesConf...	Usability:Icons	843	11,02%	Warning	85	If an icon is provided under different configuration parameters such as `drawable-hdpi`
IconDensities	Usability:Icons	639	8,35%	Warning	247	Icons will look best if a custom version is provided for each of the major screen density
IconDuplicates	Usability:Icons	555	7,25%	Warning	68	If an icon is repeated under different names, you can consolidate and just use one of th
IconMissingDensity...	Usability:Icons	408	5,33%	Warning	393	Icons will look best if a custom version is provided for each of the major screen density
IconColors	Usability:Icons	165	2,16%	Warning	64	Notification icons and Action Bar icons should only white and shades of gray. See the A
IconLauncherShape	Usability:Icons	97	1,27%	Warning	46	According to the Android Design Guide (http://developer.android.com/design/style/icc
GifUsage	Usability:Icons	71	0,93%	Warning	9	The `.gif` file format is discouraged. Consider using `.png` (preferred) or `.jpg` (acceptab

Figura 3.12 - Report: Usability:Icons

Come mostrato nell'analisi

> le issue più diffuse della Categoria Usability:icons sono:**IconLocation** con 3.879 issue (5%), **IconDipSize** con 911 issue (1,2%), **IconDuplicatesConfig** con 843 issue (1%)

3.1.3.3 Conclusioni

La Categoria di Issue in assoluto più diffusa è Performance, il che singifica che molte applicazioni (più dell'80%) possono ulteriormente essere ottimizzate.

3.2 Andamento delle Issue negli anni

Mostrato il quadro generale delle tipologie di Issue raccolte, è ora di interesse apprendere l'andamento, se esiste, delle Issue nei vari anni.

3.2.1 Il numero di issue dimininuisce con gli anni?

La prima domanda che ci si pone è **se i tool di testing siano davvero utilizzati per rendere le applicazioni più stabili**. Ci si aspetta quindi che le applicazioni più recenti abbiano un numero di issue inferiore rispetto alle applicazioni degli anni passati o che ci sia almeno un trend di decrescita del numero di issue.

3.2.1.1 Metrica

Per effettuare un confronto tra gli anni è necessario **normalizzare** il totale delle issue per le applicazioni rilasciate in quell'anno.

Per ricavare tali informazioni è necessario fare riferimento a più tabelle, in particolare:

- **Anno di rilascio del package**: informazione presente nel dato application.lastupdate, ricavato dal file fdroid.xml come mostrano in Blocco codice 2.2. Si precisa che è possibile ricavare i dati da application.lastupdate e non da package.added perché nell'analisi sono state considerate solo le ultime versioni delle applicazioni, quindi application.lastupdate = package.added;
- **Tipo di issue**: informazione presente nel dato issue.id, ricavato dal file lint-report.xml come mostrato in Blocco codice 2.4.

Per un esempio pratico di come sia stata effettuata la normalizzazione è possibile fare riferimento alla Figura 2.14.

Altre informazioni di contorno possono essere comunque ricavate dalla tabella issue, come la spiegazione della issue e il livello di severity.

3.2.1.2 Analisi

Le analisi sono presentate in una forma simile a quelle viste finora, si differenziano gli istogrammi sulla sinistra che rappresentano l'andamento delle issue negli anni (sopra) e l'andamento delle severity negli anni (sotto). Su tali dati si baserà l'intera analisi.

Anno	Issue (id)	Conteggio di Issu...	Conteggio di App...	severity	explanation
2016	UnusedResources	11043	245	Warning	Unused resources make applications larger and slow down builds.
2015	UnusedResources	4644	173	Warning	Unused resources make applications larger and slow down builds.
2016	RtlHardcoded	3608	164	Warning	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is
2014	UnusedResources	2729	142	Warning	Unused resources make applications larger and slow down builds.
2016	IconLocation	2525	87	Warning	The res/drawable folder is intended for density-independent graphics such as sh
2016	MissingTranslation	2380	40	Fatal	If an application has more than one locale, then all the strings declared in one la
2013	UnusedResources	2138	60	Warning	Unused resources make applications larger and slow down builds.
2014	RtlHardcoded	1719	73	Warning	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is

Figura 3.13 – Report: Issue per Anno

Facendo riferimento alla figura sopra è possibile analizzare nel dettaglio il trend delle difettosità negli anni.

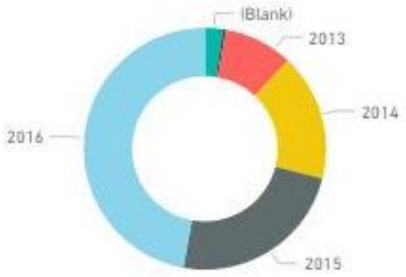


Figura 3.14 - Dettaglio Figura 3.13

Il grafico ad anello, un dettaglio della Figura 3.13, evidenzia che poco meno della metà delle applicazioni della Repository F-Droid sono state aggiornate al 2016, le restanti presentano aggiornamenti più datati.

Tale informazione non è però significativa ai fini di un confronto, il dato delle Issue va infatti normalizzato per nuemro di applicaizoni rilasciate per il particolare anno di riferimento. Così facendo si ricava la seguente informazione:

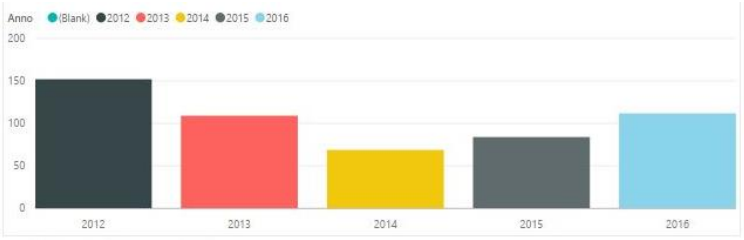


Figura 3.15 - Dettaglio Figura 3.13

L'istogramma mostra l'andamento delle issue nei vari anni ed è possibile affermare che

> i dati **non mostrano un trend generico**, è solo possibile affermare che fino al 2014 si assisteva ad una diminuzione sostanziale delle difettosità in rapporto al numero di applicazioni rilasciate. Trend che invece è andato ad aumentare negli ultimi due anni dell'analisi.

3.2.1.3 Conclusioni

In conclusione **non è possibile affermare che le issue diminuiscano con gli anni**, anzi ultimamente si assiste ad un trend di crescita.

L'analisi non consente però di sostenere delle ipotesi significative riguardo la natura dell'attuale tendenza, probabilmete entrando nel merito di ogni anno sarà possibile capire quali issue ne sono la causa, in ogni caso non si può "incolpare" né gli strumenti di testing (diventati più precisi) né i programmatori che potrebbero ignorarli deliberatamente (diventati più pigri).

3.2.2 Il livello di severity diminuisce con gli anni?

Si entrerà ora nel merito delle Severity delle Issue, in particolare si vuole appredere se le maggiori issue che si presentano negli ultimi anni abbiano un livello più basso di severity: ci sono più issue ma almeno sono meno gravi?

3.2.2.1 Metrica

In aggiunta alle metriche della Sezione 3.2.1.1 ritroviamo la seguente informazione:

- **Severity delle issue**: informazione presente nel dato issue.severity, ricavato dal file lint-report.xml come mostrato in Blocco codice 2.4

3.2.2.2 Analisi

Il numero e il tipo di Severity sono state già trattate nella Sezione 3.1.2, in questa sezione sarà evidenziato esclusivamente il confronto negli anni.

Le seuenti analisi sono state ricavate dallo stesso report di Figura 3.13, in particolare dall'istogramma immediatamente sopra la tabella, utilizzando il Filtro Severity in modo da visualizzare solo ed esclusivamente una categoria di Severity.

Effettuando pirma un confronto generale è possible ricavare alcune informazioni e già in parte rispodnere alla domanda posta:

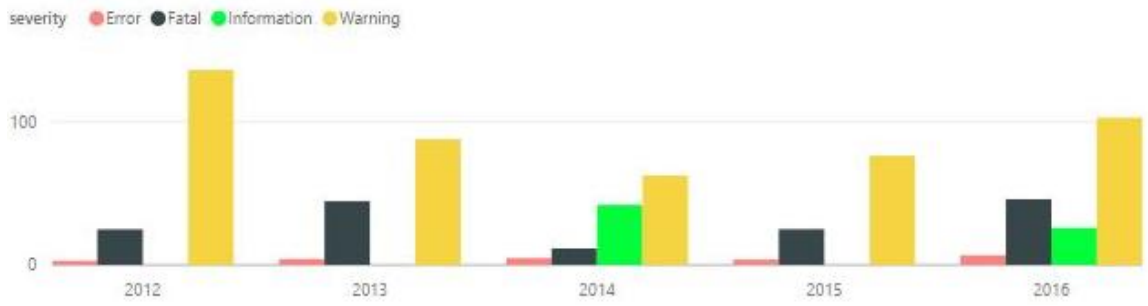


Figura 3.16 - Dettaglio Figura 3.13

Prima di tutto è possibile evidenziare che

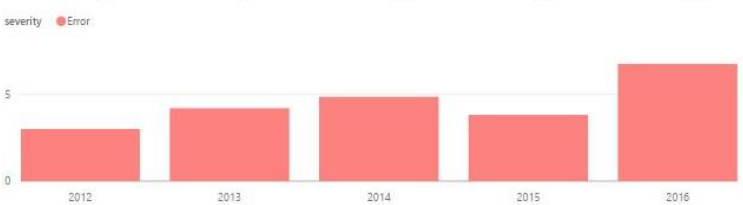
> la Severity di tipo Warning è sempre la più diffusa

ed è proprio quella che tende a polarizzare l'andamento complessivo del trend della Sezione 3.2.1. Dall'analisi si può notare che non solo i Warning aumentano, si ha una stessa corrispondenza nell'aumento delle Severity di tipo Error e Fatal, ciò porta ad affermare che

> con l'avanzare degli anni non solo ci sono più issue ma la loro **gravità aumenta**!

Nelle successive analisi si entrerà nel merito di ogni Severity.

La prima severity analizzata è Error:



3255
Conteggio di Issue (id)

628
Conteggio di Applicazioni (apknameF..

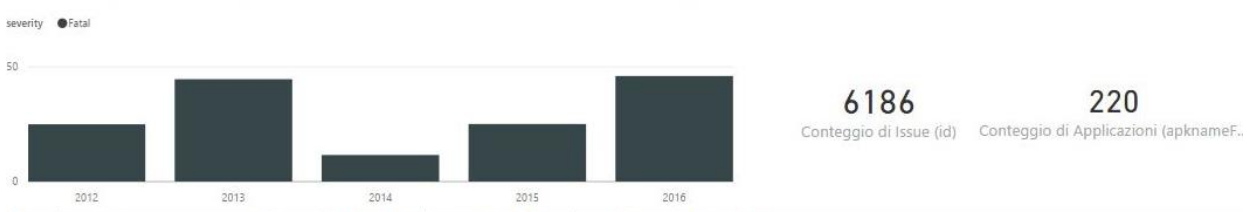
Anno	Issue (id)	Conteggio di Issu...	Conteggio di App...	severity	explanation
2016	MissingPrefix	248	64	Error	Most Android views have attributes in the Android namespace. When referencin
2016	NewApi	248	43	Error	This check scans through all the Android API calls in the application and warns al
2014	LintError	176	176	Error	This issue type represents a problem running lint itself. Examples include failure t
2014	MissingTranslation	148	1	Error	If an application has more than one locale, then all the strings declared in one la
2014	DuplicateDefinition	142	4	Error	You can define a resource multiple times in different resource folders; that's how
2014	LongLogTag	126	2	Error	Log tags are only allowed to be at most 23 tag characters long.
2015	LintError	114	111	Error	This issue type represents a problem running lint itself. Examples include failure t
2016	StringFormatInvalid	114	20	Error	If a string contains a '%' character, then the string may be a formatting string wh

Figura 3.17 - Dettaglio Figura 3.13

Per la Severity Error è possibile riscontrare un trend pressappoco crescente,

in contrasto con il trend complessivo evidenziato nella Sezione 3.2.1 che mostrava una crescita delle Issue solo dall'anno 2014 in poi.

L'analisi successiva mostra invece il trend delle Issue di tipo Fatal:



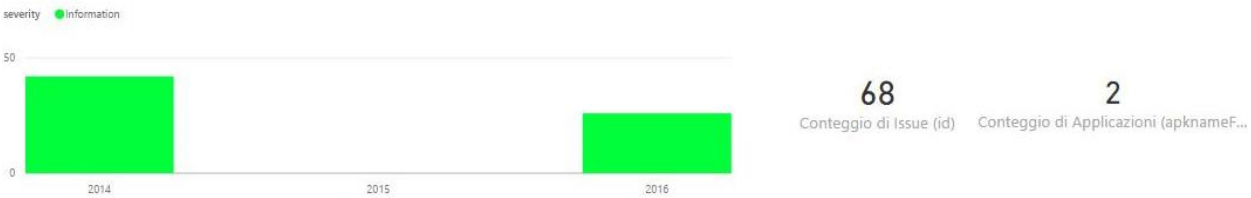
Anno	Issue (id)	Conteggio di Issu...	Conteggio di App...	severity	explanation
2016	MissingTranslation	2380	40	Fatal	If an application has more than one locale, then all the strings declared in one la
2015	MissingTranslation	1502	45	Fatal	If an application has more than one locale, then all the strings declared in one la
2013	MissingTranslation	856	21	Fatal	If an application has more than one locale, then all the strings declared in one la
2014	MissingTranslation	715	38	Fatal	If an application has more than one locale, then all the strings declared in one la
2013	DuplicateIds	165	7	Fatal	Within a layout, id's should be unique since otherwise `findViewById()` can returr
2016	ExtraTranslation	145	11	Fatal	If a string appears in a specific language translation file, but there is no correspo
2015	DuplicateIds	64	4	Fatal	Within a layout, id's should be unique since otherwise `findViewById()` can returr
2013	ManifestResource	39	3	Fatal	Elements in the manifest can reference resources, but those resources cannot va

Figura 3.18 - Dettaglio Figura 3.13

Questa categoria di Issue, proprio come il trend generale presentato nella Sezione 3.2.1, dimostra un andamento crescente dall'anno 2014 in poi, quindi

La Severity Fatal presenta un trend crescente dal 2014 in poi

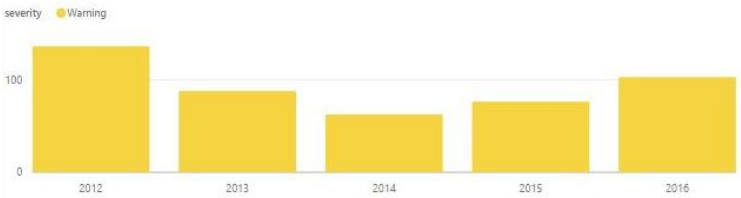
Sotto sono mostrate le issue della categoria Information. Per tale categoria è possibile notare che sono esclusivamente 2 le applicaizoni coinvolte per un totale di 68 Issue. L'anomalia dell'analisi non consente di pervenire a precise conclusioni.



Anno	Issue (id)	Conteggio di Issu...	Conteggio di App...	severity	explanation
2014	TypographyQuotes	42	1	Information	Straight single quotes and double quotes, when used as a pair, can be replaced b
2016	MissingTranslation	26	1	Information	If an application has more than one locale, then all the strings declared in one lan

Figura 3.19 - Dettaglio Figura 3.13

Per l'ultima categoria di Severity saranno analizzate le issue di tipo Warning che sono senza dubbio le più frequenti:



Anno	Issue (id)	Conteggio di Issu...	Conteggio di App...	severity	explanation
2016	UnusedResources	11043	245	Warning	Unused resources make applications larger and slow down builds.
2015	UnusedResources	4644	173	Warning	Unused resources make applications larger and slow down builds.
2016	RtlHardcoded	3608	164	Warning	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is
2014	UnusedResources	2729	142	Warning	Unused resources make applications larger and slow down builds.
2016	IconLocation	2525	87	Warning	The res/drawable folder is intended for density-independent graphics such as sh
2013	UnusedResources	2138	60	Warning	Unused resources make applications larger and slow down builds.
2014	RtlHardcoded	1719	73	Warning	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is
2016	ContentDescription	1409	151	Warning	Non-textual widgets like ImageViews and ImageButtons should use the `content

Figura 3.20 - Dettaglio Figura 3.13

Essendo le applicazioni con maggiore influenza, presentano lo stesso andamento generale presentato nella Sezione 3.2.1 e anche per tale categoria è possibile affermare che

> la Severity Warning presenta un trend crescente con origine nell'anno 2014

3.2.2.3 Conclusioni

Trend diversi per ogni tipo di Severity rendono **impossibile generalizzare le conclusioni**, ognuna presenta un andamento preciso ma riducendo l'intervallo di analisi tra il 2014 e il 2016 è possibile affermare che il trend di Error, Fatal e Warning è in crescita, anche se con velocità differenti, è quindi possibile affermare che il livello di Severity non diminuisce con gli anni.

3.2.3 Ogni anno ha una sua issue di tendenza?

In questa sezione si apprenderà se ogni anno ha una sua particolare issue di tendenza.

L'analisi si discosta parecchio da quella effettuata nella Sezione 3.1.1 dove si è concentrata maggiore attenzione sulle issue più frequenti in assoluto, mentre nella sezione attuale si farà riferimento a quelle più frequenti divise nei vari anni.

3.2.3.1 Metrica

La metrica utilizzata coincide con quella illustrata nella Sezione 3.2.1.1, le informazioni saranno però presentate in modo differente e con diverse finalità di analisi.

Si sottolinea che, a differenza dello studio nella Sezione 3.2.1.1, in questo caso non è necessario il passo di normalizzazione perché il confronto non è tra anni diversi, bensì nello stesso anno. Si vuole infatti stilare una classifica, anno per anno e apprendere le issue con più occorrenze.

3.2.3.2 Analisi

Considerando la Figura 3.13 ed estranendone il dettaglio di seguito mostrato è possibile anticipare un'analisi generica, introducendo il quadro generale della situazione:

Anno	Issue (id)	Conteggio di Issu...	Conteggio di App...	severity	explanation
2016	UnusedResources	11043	245	Warning	Unused resources make applications larger and slow down builds.
2015	UnusedResources	4644	173	Warning	Unused resources make applications larger and slow down builds.
2016	RtlHardcoded	3608	164	Warning	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is
2014	UnusedResources	2729	142	Warning	Unused resources make applications larger and slow down builds.
2016	IconLocation	2525	87	Warning	The res/drawable folder is intended for density-independent graphics such as sh
2016	MissingTranslation	2380	40	Fatal	If an application has more than one locale, then all the strings declared in one la
2013	UnusedResources	2138	60	Warning	Unused resources make applications larger and slow down builds.
2014	RtlHardcoded	1719	73	Warning	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is

Figura 3.21 – Dettaglio Figura 3.13

La figura sopra mostra le Issue raggruppate per tipo, issue (id) in tabella, e divise per Anno di rilascio del package. Ordinandole per Conteggio decrescente è possibile ricavare l'informaizone relativa alle issue più frequenti.

Si fa notare che, a differenza della Figura 3.1, nella Figura 3.21 le varie issue sono ulteriormente divise per Anno, quindi non rappresentano la stessa informazione.

Entrando nel merito dell'analisi si deduce che

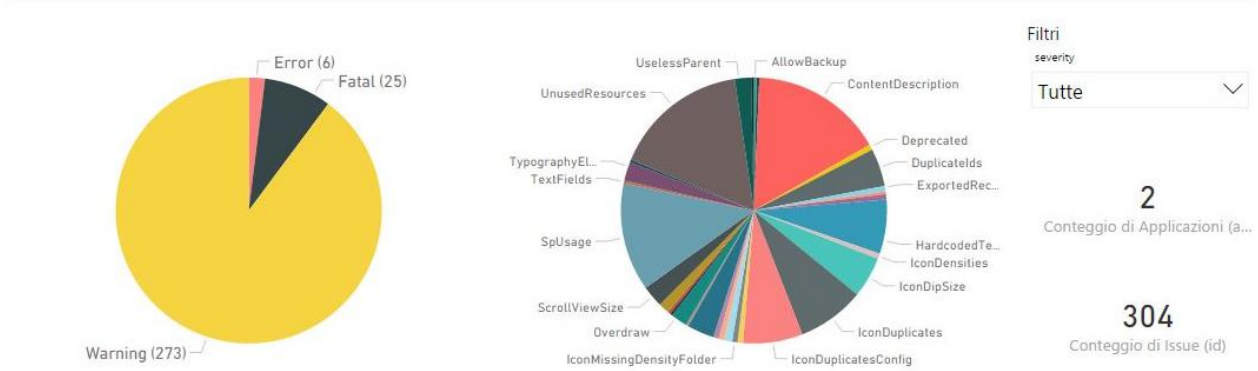
> la Issue più frequente in assoluto è **UnusedResources** con 11.043 occorrenze solo nell'anno 2016 (14% di tutte le issue), inoltre anche nel 2015, 2014 e 2013, resta sempre in testa.

Quindi è possibile affermare che **la Issue più frequente in ogni anno è UnusedResources**, anche senza approfondimenti.

In ogni caso tale difettosità è catalogata come "warning" e come da didascalia "rende le applicaizoni più grandi e ne rallenta la costruzione", quindi non è una Issue pericolosa, al contrario di **MissingTranslation**, l'unica issue della classifica ad essere catalogata come "Fatal", questo significa che ha una priorità di intervento maggiore.

Per un'analisi completa si entrerà ora nel dettaglio di ogni singolo anno:

Di seguito si analizzano le Issue appartenenti alle Applicazioni rilasciate nell'Anno 2012:



Anno	Issue (id)	severity	Conteggio di Issue (id)	Applicazioni (id)	explanation
2012	ContentDescription	Warning	49	apps.babycaretimer	Non-textual widgets like ImageViews and ImageButtons should use the `contentDes
2012	SpUsage	Warning	40	apps.babycaretimer	When setting text sizes, you should normally use `sp`, or "scale-independent pixels".
2012	UnusedResources	Warning	39	apps.babycaretimer	Unused resources make applications larger and slow down builds.
2012	IconDuplicates	Warning	25	apps.babycaretimer	If an icon is repeated under different names, you can consolidate and just use one c
2012	IconDuplicatesConfig	Warning	21	apps.babycaretimer	If an icon is provided under different configuration parameters such as `drawable-h
2012	HardcodedText	Warning	20	apps.babycaretimer	Hardcoding text attributes directly in layout files is bad for several reasons: * When
2012	DuplicateIds	Fatal	14	apps.babycaretimer	Within a layout, id's should be unique since otherwise `findViewById()` can return an
2012	IconDipSize	Warning	14	apps.babycaretimer	Checks the all icons which are provided in multiple densities, all compute to roughly
2012	UnusedResources	Warning	11	com.gpstether	Unused resources make applications larger and slow down builds.

Figura 3.22 – Report: Anno 2012

L'esigua quantità di applicazioni dell'Anno 2012, appena 2, non consente di effettuare analisi sulle Issue ma potrebbe essere utilie per analisi selle singole applicazioni.

Entrando nel merito delle Issue, si nota che

> l'issue più diffusa nel 2012 è **ContentDescription** con 49 occorrenze (circa il 16%),

tutte concentrate esclusivamente in una delle due applicazioni e come da didascalia “mancata descrizione testuale per contenuti visivi come ImageView e ImageButton”. A giudicare dall’elevato numero, possiamo dedurre che il creatore dell’app abbia completametne omesso di descrivere alcuni oggetti visivi.

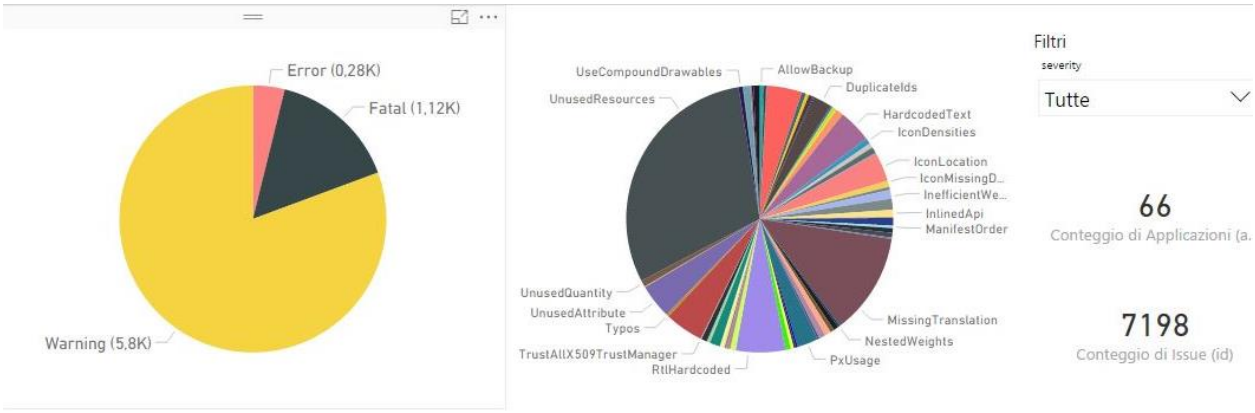
Al secondo posto troviamo **SpUsage** con 40 istanze e al terzo **UnusedResource** con 39 che rappresentano in ordine “resize del testo in base alla dimensione e alla densità di pixels del dispositivo” e “risorse non utilizzate che rallentano l’applicazione”.

E’ già stata evidenziata la presenza di poche app risalenti al 2012 ma tale mancanza sottolinea che

> delle circa 800 applicazioni analizzate della Repository F-Droid la quasi totalità è stata aggiornata almeno nel 2013, quindi è possibile affermare di essere in presenza di una Repository alquanto attiva.

Inoltre trovandoci nella situazione di sole due app in fase di analisi si può effettuare uno studio specifico ed affermare che sicuramente l’applicazione con id “apps.babcaretimer” non è stata adeguatamente ottimizzata né per essere utilizzata su dispositivi con diversa risoluzione né per ottimizzazione di prestazioni (date le varie risorse non utilizzate).

Si passerà ora all’analisi dell’anno 2013:



Anno	Issue (id)	severity	C...	explanation
2013	UnusedResources	Warning	2138	Unused resources make applications larger and slow down builds.
2013	MissingTranslation	Fatal	856	If an application has more than one locale, then all the strings declared in one language should also be translated in all other l
2013	RtlHardcoded	Warning	427	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales where text flows from righ
2013	TypographyEllipsis	Warning	332	You can replace the string "..." with a dedicated ellipsis character, ellipsis character (..., …). This can help make the text m
2013	ContentDescription	Warning	303	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription` attribute to specify a textual desc
2013	UnusedAttribute	Warning	295	This check finds attributes set in XML files that were introduced in a version newer than the oldest version targeted by your ap
2013	HardcodedText	Warning	283	Hardcoding text attributes directly in layout files is bad for several reasons: * When creating configuration variations (for exam
2013	IconLocation	Warning	251	The res/drawable folder is intended for density-independent graphics such as shapes defined in XML. For bitmaps, move it to
2013	PxUsage	Warning	179	For performance reasons and to keep the code simpler, the Android system uses pixels as the standard unit for expressing din
2013	DuplicateIds	Fatal	165	Within a layout, id's should be unique since otherwise `findViewById()` can return an unexpected view.

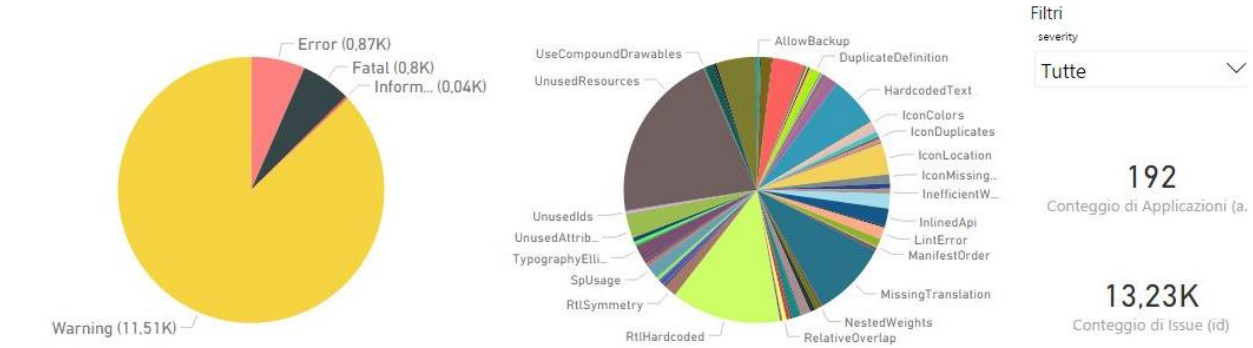
Figura 3.23 – Report: Anno 2013

Nel 2013 c’è stato un aumento di app aggiornate, si passa infatti dalle 2 Applicazioni del 2012 alle 66 del 2013, ed ovviamente ad un incremento sostanziale delle Difettosità, anche se, come si nota dalla Top 10 mostrata in Figura 3.23,

> nel 2013 restano in testa Issue molto simili al 2012: UnusedResources (circa 30%), MissingTranslation (circa 12%) e RtlHardcoded (circa 6%).

Avendo un maggior nuemro di applicazioni è possibile notare Issue non presenti nel 2012, come: **TypographyEllispsis**, **UnusedAttribute**, **IconLocation** e **PxUsage**. Tali Issue sono però presenti in Tabella 5.1, quindi fanno parte delle issue più frequenti. Si faccia riferimento alla suddetta tabella per approfondimenti.

L’analisi sucessiva riguarderà l’anno 2014:



Anno	Issue (id)	severity	C...	explanation
2014	UnusedResources	Warning	2729	Unused resources make applications larger and slow down builds.
2014	RtlHardcoded	Warning	1719	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales where text flows from rig
2014	HardcodedText	Warning	807	Hardcoding text attributes directly in layout files is bad for several reasons: * When creating configuration variations (for exam
2014	MissingTranslation	Fatal	715	If an application has more than one locale, then all the strings declared in one language should also be translated in all other
2014	UseValueOf	Warning	627	You should not call the constructor for wrapper classes directly, such as`new Integer(42)`. Instead, call the `valueOf` factory me
2014	ContentDescription	Warning	517	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription` attribute to specify a textual desc
2014	IconLocation	Warning	495	The res/drawable folder is intended for density-independent graphics such as shapes defined in XML. For bitmaps, move it to
2014	UnusedAttribute	Warning	389	This check finds attributes set in XML files that were introduced in a version newer than the oldest version targeted by your ap
2014	MissingTranslation	Warning	350	If an application has more than one locale, then all the strings declared in one language should also be translated in all other

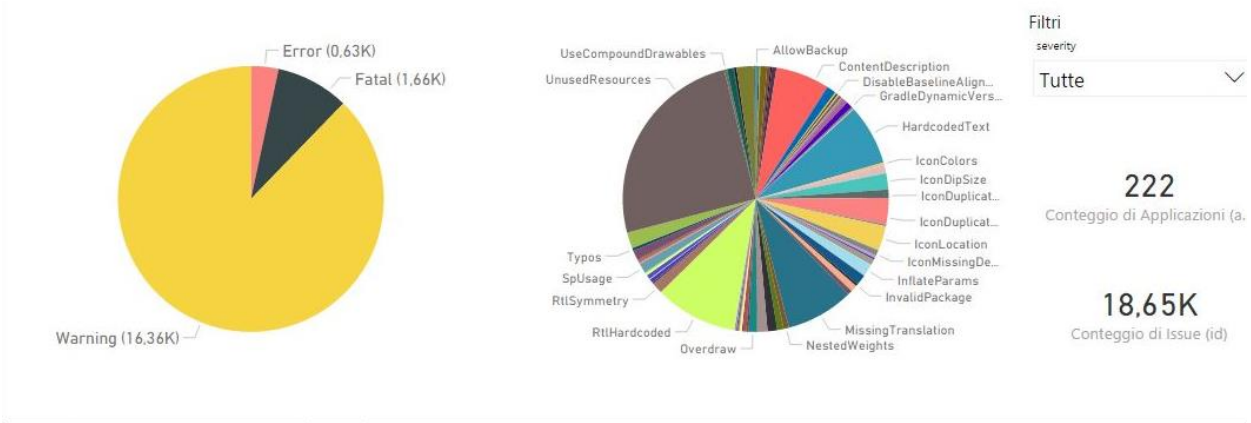
Figura 3.24 – Report: Anno 2014

Con l'andare avanti degli anni si riscontrano più applicazioni aggiornate e ovviamente più issue, nonostante questo aumento il 2014 vede una classifica poco dissimile al 2013,

> al primo posto c'è sempre UnusedResources (20%), seguito da RtlHardcoded (13%) e HardcodedText (6%).

MissingTranslation, come già evidenziato in precedenza, risulta sia in quarta che nona posizione, a causa della sua natura duale di Warning e Fatal a seconda del contesto.

Il prossimo Report riguarderà l'anno 2015:

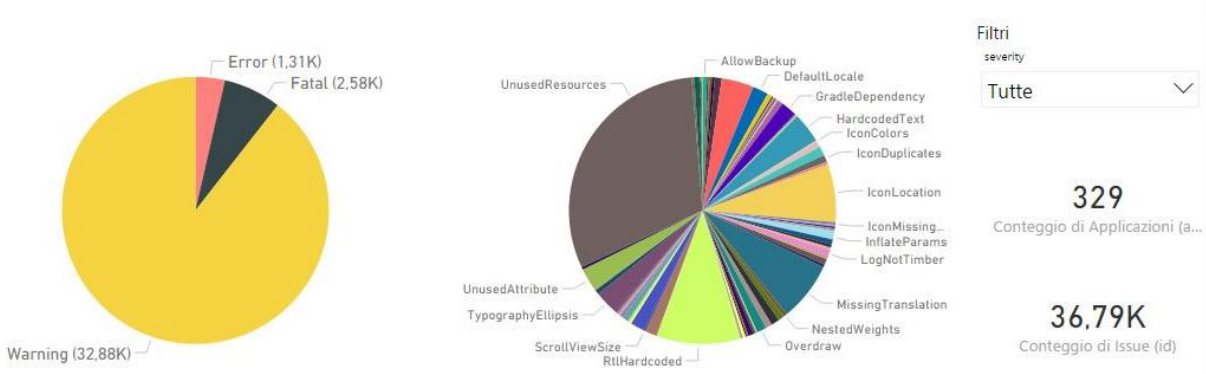


Anno	Issue (id)	severity	C...	explanation
2015	UnusedResources	Warning	4644	Unused resources make applications larger and slow down builds.
2015	MissingTranslation	Fatal	1502	If an application has more than one locale, then all the strings declared in one language should also be translated in all other l
2015	HardcodedText	Warning	1353	Hardcoding text attributes directly in layout files is bad for several reasons: * When creating configuration variations (for exam
2015	ContentDescription	Warning	1197	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription` attribute to specify a textual desc
2015	RtlHardcoded	Warning	1094	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales where text flows from rigl
2015	RtlHardcoded	Warning	755	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales where text flows from rigl
2015	IconDuplicatesConfig	Warning	591	If an icon is provided under different configuration parameters such as `drawable-hdpi` or `-v11`, they should typically be diffe
2015	IconLocation	Warning	554	The res/drawable folder is intended for density-independent graphics such as shapes defined in XML. For bitmaps, move it to
2015	UseValueOf	Warning	383	You should not call the constructor for wrapper classes directly, such as`new Integer(42)`. Instead, call the `valueOf` factory me

Figura 3.25 – Report: Anno 2015

Il 2015 offre in classifica: UnusedResources (25%), MissingTranslation (8%), HardcodedText (7%).

Infine l'anno 2016:



Anno	Issue (id)	severity	C...	explanation
2016	UnusedResources	Warning	11043	Unused resources make applications larger and slow down builds.
2016	RtlHardcoded	Warning	3608	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales where text flows from rigl
2016	IconLocation	Warning	2525	The res/drawable folder is intended for density-independent graphics such as shapes defined in XML. For bitmaps, move it to
2016	MissingTranslation	Fatal	2380	If an application has more than one locale, then all the strings declared in one language should also be translated in all other l
2016	ContentDescription	Warning	1409	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription` attribute to specify a textual desc
2016	TypographyEllipsis	Warning	1024	You can replace the string "..." with a dedicated ellipsis character, ellipsis character (..., …). This can help make the text r
2016	UnusedAttribute	Warning	1008	This check finds attributes set in XML files that were introduced in a version newer than the oldest version targeted by your ap
2016	HardcodedText	Warning	759	Hardcoding text attributes directly in layout files is bad for several reasons: * When creating configuration variations (for exam
2016	GradleDependency	Warning	691	This detector looks for usages of libraries where the version you are using is not the current stable release. Using older versior

Figura 3.26 – Report: Anno 2016

L'analisi consente di apprendere che

nel 2016 le issue più diffuse sono state: **UnusedResources** (30%), **RtlHardcoded** (10%), **IconLocation** (7%)

3.2.3.3 Conclusioni

Purtroppo non è possibile riscontrare tendenze diverse nei vari anni, anzi, esistono in generale issue più frequenti come UnusedResource, MissingTranslation e RtlHardcoded che dominano l'intero periodo in esame.

E' però possibile affermare, data la quasi totalità di Warning in ogni anno, che le issue "prioritarie" sono quasi sempre risolte per prime, lasciando trascurate ottimizzazioni minori.

3.2.4 Esiste un trend nelle issue?

è interessante notare se con il tempo ci sono difettosità che scompaiono, magari diventano problematiche note e i tool di testing aiutano i programmatori ad evitarle.

Verranno cosiderate esculicamente le Top 10 dei vari anni.

3.2.4.1 Metrica

Si farà riferimento alla metrica utilizzata nella Sezione 3.2.3.1 e fondamentalmente alle stesse informazioni, cambierà il modo di presentarle per ottenere un riscontro immediato di quanto appreso.

3.2.4.2 Analisi

Di seguito una tabella riepilogativa per il confronto tra gli anni. Ogni icona rappresenta se la Issue, rispetto all'anno precedente, ha subito o meno variazioni di posizione in classifica. Scopo dell'analisi è scoprire Issue che entrano o escono dalle classifiche, quindi issue *prive di icona*:

2012	2013	2014	2015	2016
UnusedResources	UnusedResources	UnusedResources	UnusedResources	UnusedResources
ContentDescription	MissingTranslation	RtlHardcoded	RtlHardcoded	RtlHardcoded
HardcodedText	RtlHardcoded	MissingTranslation	MissingTranslation	MissingTranslation
MissingTranslation	TypographyEllipsis	HardcodedText	HardcodedText	IconLocation
TypographyEllipsis	ContentDescription	UseValueOf	ContentDescription	ContentDescription
IconLocation	UnusedAttribute	ContentDescription	IconLocation	HardcodedText
	HardcodedText	IconLocation	UseValueOf	TypographyEllipsis
	IconLocation	UnusedAttribute	UnusedAttribute	UnusedAttribute
	UseValueOf	TypographyEllipsis	RtlSymmetry	RtlSymmetry
	RtlSymmetry	RtlSymmetry	TypographyEllipsis	UseValueOf

Figura 3.27 - Trend nelle Issue

La prima evidenza è che

dal 2013 in poi non ci sono nuove entrate in classifica,

infatti si riscontrano esclusivamente cambi di posizione. L'unica calssifica che evidenzia delle entrate è quella del 2013, ma tale fenomeno è dovuto all'esigua quantità di informazioni ricavate per l'anno 2012, in cui mancavano molte tipologie di Issue.

3.2.4.3 Conclusioni

Scopo dell'analisi era di riscontrare variazioni delle tipologie di issue nelle varie classifiche annuali, l'analisi ha di contro dimostrato che tale variazione non esiste, in quanto le Top 10 sono dominate dalle stesse tipologie di issue, semplicemetne con frequnze diverse e quindi in posizioni diverse.

E' inoltre possibile notare che negli ultimi due anni si assiste alla medesima Top 3.

3.3 Età di una applicazione

Terminata l'analisi per Anno di aggiornamento di un'applicaizone si vuole ora spostare l'attenzione su obiettivi differenti. In questa sezione si definirà, in modo del tutto fantasioso, *l'età di una applicazione* e si cercheranno delle relazioni con il numero di issue.

3.3.1 Applicazioni più mature presentano meno Issue?

Proprio come per un essere umano, che più matura meno errori commette (o almeno così dovrebbe essere), ci si aspetta che applicaioni più "mature", quindi in età più avanzata rispetto ad altre, abbiano un numero minore di issue.

3.3.1.1 Metrica

La metrica utilizata fa riferimento a ciò che è stata chiamata "età di una applicazione", definita come la differenza tra l'anno del suo ultimo aggiornamento e l'anno del suo inserimento nella Repository F-Droid. Volendo indicare il dato di riferimento:

- **Data di upload nella repository**: informazione presente nel dato application.added, ricavato dal file fdroid.xml come mostrato in Blocco codice 2.2.
- **Data di ultimo aggiornamento**: informazione presente nel dato application.lastupdated, ricavato dal file fdroid.xmo come mostrato in Blocco codice 2.2.

E' possibile quindi definire l'età di una applicazione come mostrato di seguito:

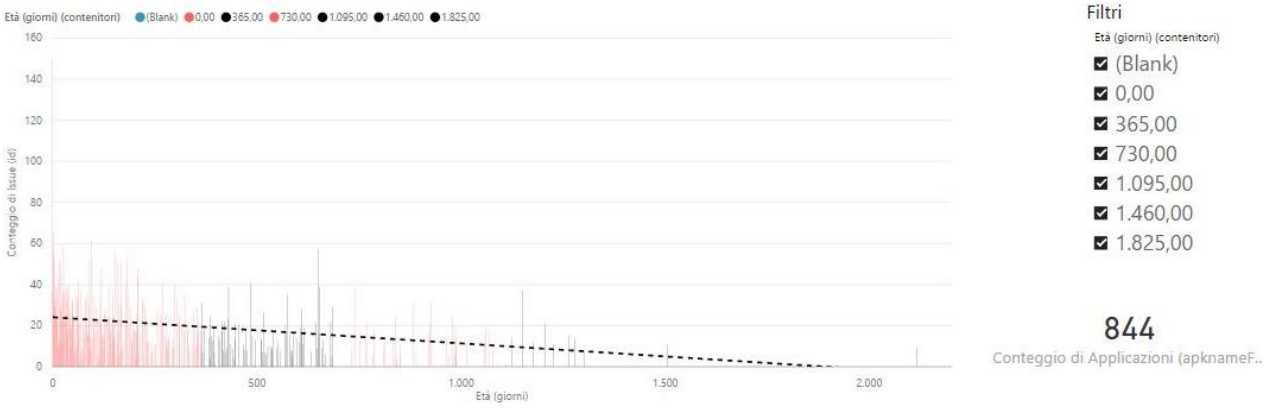
$$Età = data\ ultimo\ aggiornamento - data\ upload$$

La misura è stata crata come "nuova colonna" all'interno dell'Analytics Layer, non sono quindi stati modificati i dati originali presenti nel Knowledge Layer.

Creata tale misura è quindi possibile effettuare nuove analisi, come di seguito riportato.

3.3.1.2 Analisi

Di seguito è mostrato un grafico dell'andamento delle issue rispetto all'età di una applicazione:



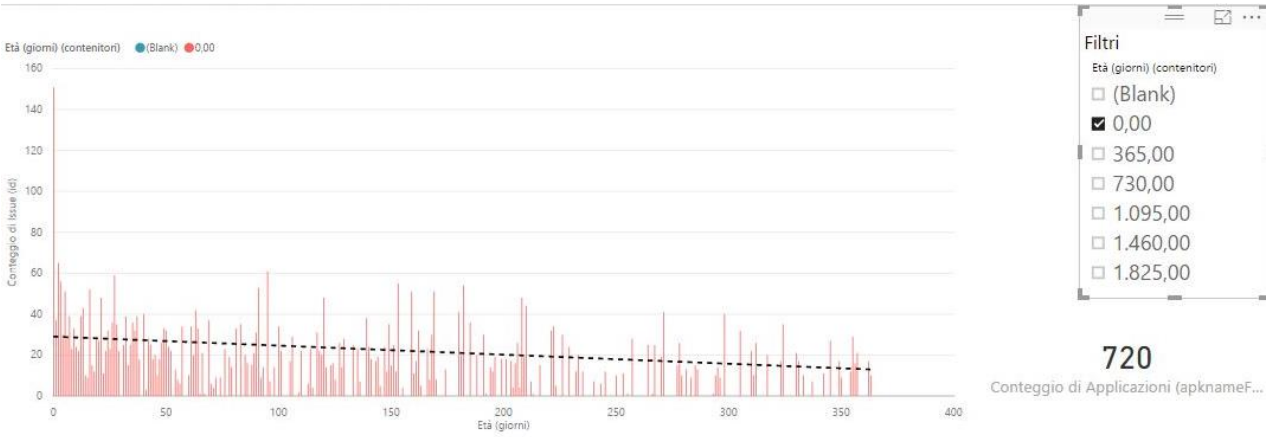
name	added	Anno (lastupdated)	Età (giorni)	Conteggio di Issue (id)
Sokoban	mercoledì 5 gennaio 2011	giovedì 20 ottobre 2016	2.115,00	23
Compass Keyboard	lunedì 8 ottobre 2012	lunedì 21 novembre 2016	1.505,00	12
Multi Sms	mercoledì 7 novembre 2012	martedì 31 maggio 2016	1.301,00	10
GCstar Viewer	venerdì 7 settembre 2012	lunedì 7 marzo 2016	1.277,00	1058
aMetro	venerdì 28 settembre 2012	martedì 15 marzo 2016	1.264,00	80
USB Device Info	mercoledì 17 luglio 2013	mercoledì 23 novembre 2016	1.225,00	27
SliderSynth	sabato 9 marzo 2013	sabato 25 giugno 2016	1.204,00	313
SuperGenPass	martedì 25 settembre 2012	martedì 15 dicembre 2015	1.176,00	35
Music	lunedì 24 giugno 2013	giovedì 18 agosto 2016	1.151,00	547
Totale				78046

Figura 3.28 - Report: Età delle applicazioni

Sull'asse delle ascisse è presente l'Età delle applicazioni (in giorni), sulle ordinate il numero di issue. E' evidente che

applicazioni più mature presentano meno issue rispetto ad applicazioni più giovani.

Di seguito è stato effettuato uno “zoom”, andando ad analizzare prettametne le applicazioni con al massimo un anno di età:



name	added	Anno (lastupdated)	Età (giorni)	Conteggio di Issue (id)
Episodes	mercoledì 11 marzo 2015	martedì 8 marzo 2016	363,00	27
Tint Browser	giovedì 10 gennaio 2013	martedì 7 gennaio 2014	362,00	49
RMLL/LSM	venerdì 4 luglio 2014	venerdì 26 giugno 2015	357,00	48
OSMBugs	lunedì 2 marzo 2015	domenica 21 febbraio 2016	356,00	260
Intent Intercept	sabato 15 agosto 2015	giovedì 4 agosto 2016	355,00	26
OSRS Helper	lunedì 7 dicembre 2015	sabato 26 novembre 2016	355,00	234
Smashwords Access	mercoledì 29 gennaio 2014	domenica 18 gennaio 2015	354,00	27
Headphone detector for Vanilla Music	mercoledì 10 giugno 2015	mercoledì 25 maggio 2016	350,00	11
Pixel Dungeon	lunedì 20 ottobre 2014	lunedì 5 ottobre 2015	350,00	4
Totale				64260

Figura 3.29 – Dettaglio Figura 3.28

E’ possibile osservare che

le applicaizoni nel loro primo anno di vita, presentano un trend nettamente decrescente nel numero di issue,

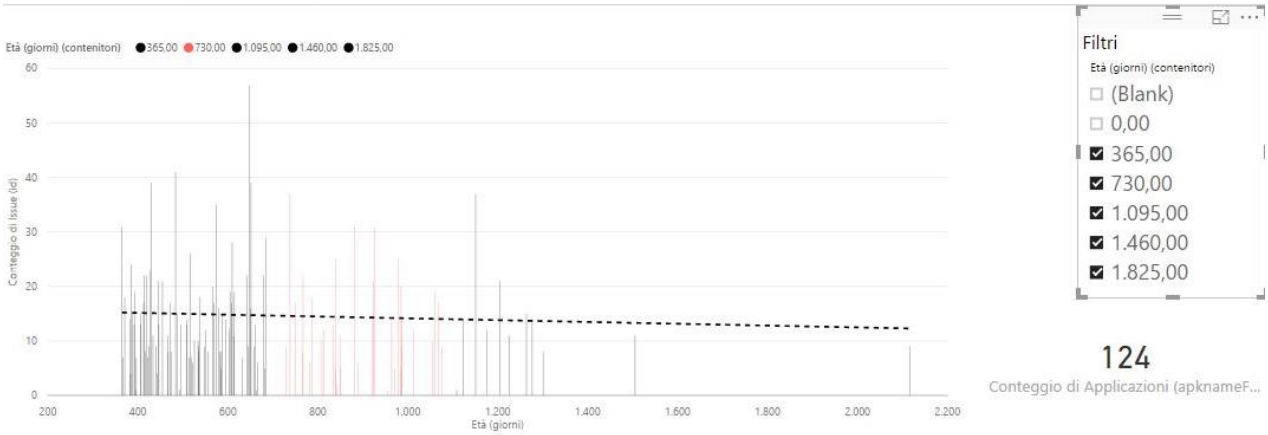
c’è infatti una discesa di circa 20 punti.

Una informazione “secondaria” che è possibile ricavare dall’analisi è che

720 app su 844 (85,3%) hanno meno di un anno di vita,

ciò attesta che **la repository è composta da applicazioni molto giovani**, cioè che sono state aggiornate entro un anno dal loro rilascio.

L’immagine sottostante, invece, **mostra tutte le applicaizoni con più di un anno di vita**, fino ad arrivare a 2000 giorni circa, quindi a 5 anni e mezzo:



name	added	Anno (lastupdated)	Età (giorni)	Conteggio di Issue (id)
Sokoban	mercoledì 5 gennaio 2011	giovedì 20 ottobre 2016	2.115,00	23
Compass Keyboard	lunedì 8 ottobre 2012	lunedì 21 novembre 2016	1.505,00	12
Multi Sms	mercoledì 7 novembre 2012	martedì 31 maggio 2016	1.301,00	10
GCstar Viewer	venerdì 7 settembre 2012	lunedì 7 marzo 2016	1.277,00	1058
aMetro	venerdì 28 settembre 2012	martedì 15 marzo 2016	1.264,00	80
USB Device Info	mercoledì 17 luglio 2013	mercoledì 23 novembre 2016	1.225,00	27
SliderSynth	sabato 9 marzo 2013	sabato 25 giugno 2016	1.204,00	313
SuperGenPass	martedì 25 settembre 2012	martedì 15 dicembre 2015	1.176,00	35
Music	lunedì 24 giugno 2013	giovedì 18 agosto 2016	1.151,00	547
Totale				**13786**

Figura 3.30 - Dettaglio Figura 3.28

> Le applicazioni con più di 1 anno di vita dimostrano comunque un trend decrescente, anche se meno marcato delle compagne più giovani.

3.3.1.3 Conclusioni

E' possibile concludere, a valle delle analisi effettuate, che **applicazioni più mature presentano una tendenza ad avere un numero minore di Issue**, quindi, supponendo che i programatori abbiano utilizzato tool di testing o i feedback della community, si può affermare che il testing serve!

3.4 Category dell'Applicazione

Altro obiettivo interessante è comprendere che legame ci sia tra le issue e la **categoria a cui appartiene un'applicazione**, cercando di capire sia se esistano categorie con più alto rischio di issue, sia se ci siano issue particolari per una determinata categoria.

3.4.1 Esiste una category con più Issue?

E' opportuno chiedersi se esistono categorie di applicazioni maggiormente soggette a presentare un qualche tipo di problematica.

3.4.1.1 Metrica

Per effettuare un confronto tra categorie diverse è necessario normalizzare il totale delle issue per le applicaizoni appartenenti a quella categoria.

Per ricavare tali informazioni è necessario fare riferimento a più tabelle, in particolare:

- **Categoria dell'applicazione**: informazione presente nel dato application.category, ricavato dal file fdroid.xml come mostrano in Blocco codice 2.2.
- **Tipo di issue**: informazione presente nel dato issue.id, ricavato dal file lint-report.xml come mostrato in Blocco codice 2.4.

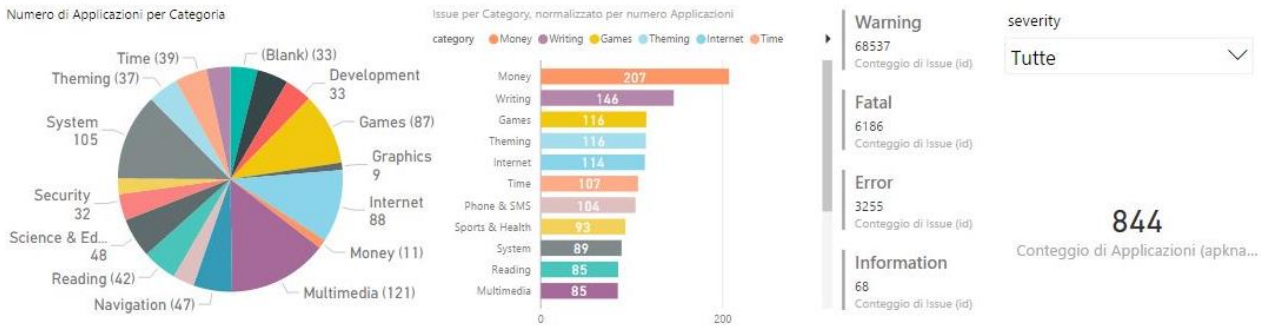
Per un esempio pratico di come sia stata effettuata la normalizzazione è possibile fare riferimento alla Figura 2.14.

Si fa notare che la metrica è simile a quella utilizzata nella Sezione 3.2.1.1, la differenza è fatta nel Layer Analytics in cui le issue sono raggruppate per category e non per anno, rendendo la misura adattabile al contesto.

Altre informazioni di contorno possono essere comunque ricavate dalla tabella issue, come la spiegazione della issue e il livello di severity.

3.4.1.2 Analisi

Di seguito l'analisi effettuata:



Issue (id)	severity	Conteggio di Issue (id)	explanation
UnusedResources	Warning	20874	Unused resources make applications larger and slow down builds.
RtlHardcoded	Warning	7055	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales whe
MissingTranslation	Fatal	5469	If an application has more than one locale, then all the strings declared in one language should also be t
IconLocation	Warning	3879	The res/drawable folder is intended for density-independent graphics such as shapes defined in XML. Fc
ContentDescription	Warning	3518	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription` attribute tc
HardcodedText	Warning	3285	Hardcoding text attributes directly in layout files is bad for several reasons: * When creating configuratic
UnusedAttribute	Warning	2028	This check finds attributes set in XML files that were introduced in a version newer than the oldest versic
TypographyEllipsis	Warning	1838	You can replace the string "..." with a dedicated ellipsis character, ellipsis character (..., …). This car
RtlHardcoded	Warning	1240	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales whe
UseValueOf	Warning	1132	You should not call the constructor for wrapper classes directly, such as`new Integer(42)`. Instead, call th
Totale		**78046**	

Figura 3.31 – Report: Category

Dall'intero Report è possibile estrarre varie informazioni, come l'immagine seguente:

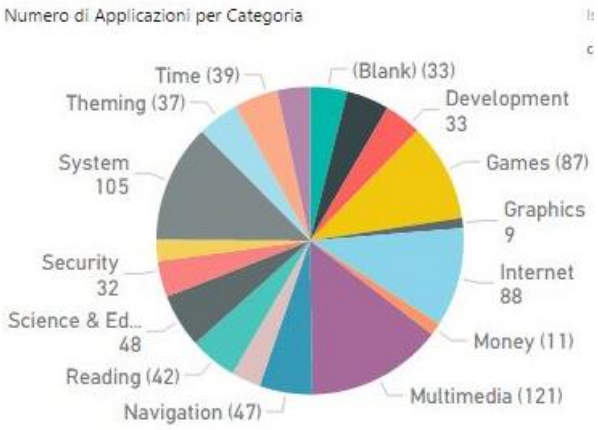


Figura 3.32 – Dettaglio Figura 3.31

Dal dettaglio sopra si apprende che la maggior parte delle applicazioni appartengono alle category: **Multimedia**, **System**, **Internet**, **Games**. Tali informazioni sono però assolute, quindi non rappresentano un confronto valido per trarre delle conclusioni appropriate.

Si faccia riferiemento all'analisi sottostante per il confronto:

Figura 3.33 – Dettaglio Figura 3.31

Le informazioni provengono dai dati normalizzati, è quindi evidente che

> la category con più issue è **Money**, seguita da **Writing** e, a pari merito, **Games** e **Theming**.

3.4.1.3 Conclusioni

Si può semplicemente concludere che si, esiste una category con più Issue ed è Money.

Nella Sezione successiva saranno analizzate nel dettaglio le category e si potranno trarre conclusioni più specifiche.

3.4.2 Esistono Issue peculiari per ogni categoria?

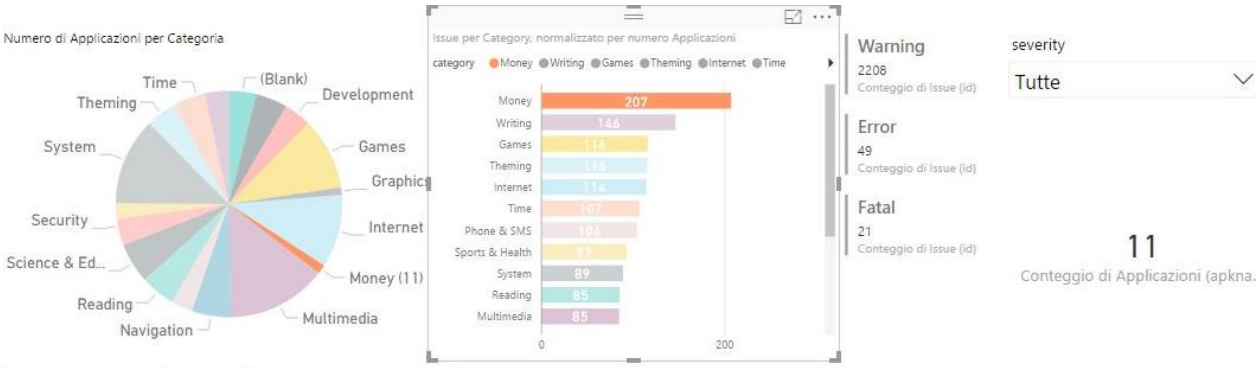
Si vuole ora entrare nello specifico delle category con più issue, cercando apprendere **se ci siano issue specifiche per alucne category piuttosto che per altre**. Per tale analisi ci si limiterà alla Top 3 delle category che presentano più Issue.

3.4.2.1 Metrica

La metrica utilizzata è la stessa della Sezione 3.4.1.1, sono però stati applicati dei filtri all'interno del Report in modo da rappresentare le informazioni nel formato più consono all'analisi.

3.4.2.2 Analisi

La prima analisi riguarda la category Money:



Issue (id)	severity	Conteggio di Issue (id)	explanation
RtlHardcoded	Warning	585	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales whe
UnusedResources	Warning	357	Unused resources make applications larger and slow down builds.
IconLocation	Warning	198	The res/drawable folder is intended for density-independent graphics such as shapes defined in XML. Fo
HardcodedText	Warning	143	Hardcoding text attributes directly in layout files is bad for several reasons: * When creating configuratio
ContentDescription	Warning	108	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription` attribute to
TypographyEllipsis	Warning	94	You can replace the string "..." with a dedicated ellipsis character, ellipsis character (..., …). This can
RtlSymmetry	Warning	84	If you specify padding or margin on the left side of a layout, you should probably also specify padding o
SpUsage	Warning	67	When setting text sizes, you should normally use `sp`, or "scale-independent pixels". This is like the `dp`
NestedWeights	Warning	53	Layout weights require a widget to be measured twice. When a LinearLayout with non-zero weights is ne
SetTextI18n	Warning	49	When calling `TextView#setText` * Never call `Number#toString()` to format numbers; it will not handle f
Totale		**2278**	

Figura 3.34 - Report: Category Money

La category

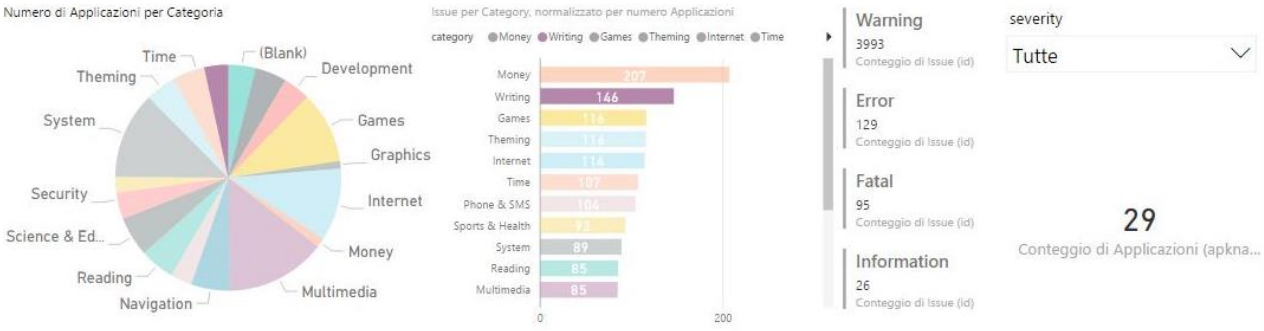
> Money presenta 2.278 issue di cui 2.208 Warning (97%), 49 Error (2%) e 21 Fatal (0,9%) per 11 applicazioni.

Di cui le issue più frequenti sono:

> RtlHardcoded con 585 occorrenze (26%), UnusedResources con 357 (16%),

le altre, meno frequenti, sono **IconLocation**, **HardcodedText** e **ContentDescription**. Volendo trarre una conclusione specifica è possibile affermare che la maggior parte delle issue riguardano problemi di layout sui vari dispositivi; risorse non utilizzate e quindi mancata ottimizzazione; problemi di risoluzione delle icone.

La seguente analisi riguarda invece la categoria Writing:



Issue (id)	severity	Conteggio di Issue (id)	explanation
UnusedResources	Warning	1601	Unused resources make applications larger and slow down builds.
IconDuplicatesConfig	Warning	498	If an icon is provided under different configuration parameters such as `drawable-hdpi` or `-v11`, they
IconDipSize	Warning	298	Checks the all icons which are provided in multiple densities, all compute to roughly the same density-
TypographyEllipsis	Warning	221	You can replace the string "..." with a dedicated ellipsis character, ellipsis character (..., …). This c
RtlHardcoded	Warning	140	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales wl
UnusedAttribute	Warning	135	This check finds attributes set in XML files that were introduced in a version newer than the oldest vers
ContentDescription	Warning	70	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription` attribute
ExtraTranslation	Fatal	70	If a string appears in a specific language translation file, but there is no corresponding string in the def
RtlHardcoded	Warning	69	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales wl
IconLocation	Warning	62	The res/drawable folder is intended for density-independent graphics such as shapes defined in XML.
Totale		**4243**	

Figura 3.35 - Report: Category Writing

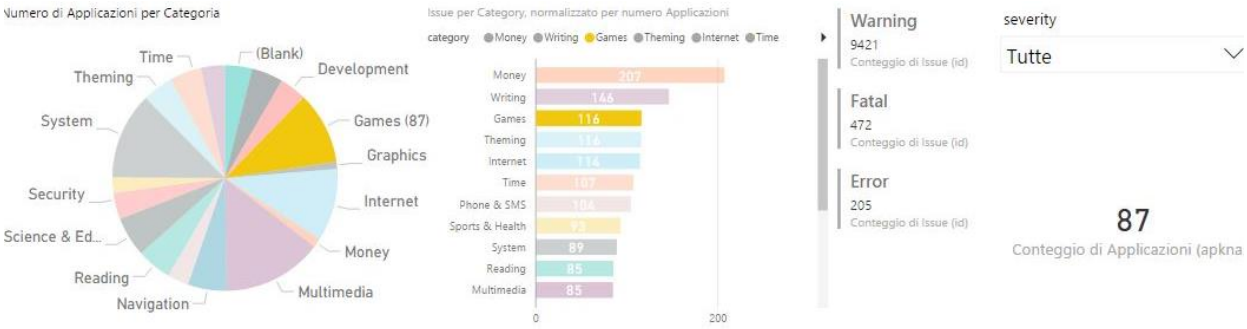
Writing mostra un totale di 4.243 issue di cui 3.993 Warning (94%), 129 Error (3%), 95 Fatal (2%) per un totale di 29 applicazioni coinvolte.

In tale categoria si evidenziano le problematiche del tipo

UnusedResource con 1.601 occorrenze (38%), **IconDuplicatesConfig** con 498 (12%), **IcondDipSize** con 298 (7%),

In particolare le issue **IconDuplicatesConfig** e **IconDipSie** sono riscontrare per la prima volta in tutte le analisi svolte. Tali Issue riguardano problemi con le icone, rispettivamente: icone utilizzate erroneamente, icone con densità diverse. Entrambi i problemi riguardano aspetti di Usability dell'applicazione.

L'ultima analisi riguarda la categoria Games:



Issue (id)	severity	Conteggio di Issue (id)	explanation
UnusedResources	Warning	2490	Unused resources make applications larger and slow down builds.
ContentDescription	Warning	980	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription` attribute
HardcodedText	Warning	816	Hardcoding text attributes directly in layout files is bad for several reasons: * When creating configura
IconLocation	Warning	736	The res/drawable folder is intended for density-independent graphics such as shapes defined in XML.
RtlHardcoded	Warning	636	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales w
UnusedAttribute	Warning	422	This check finds attributes set in XML files that were introduced in a version newer than the oldest ver
MissingTranslation	Fatal	417	If an application has more than one locale, then all the strings declared in one language should also b
RtlHardcoded	Warning	389	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales w
UseValueOf	Warning	308	You should not call the constructor for wrapper classes directly, such as`new Integer(42)`. Instead, call
HardcodedText	Warning	260	Hardcoding text attributes directly in layout files is bad for several reasons: * When creating configura
Totale		**10098**	

Figura 3.36 - Report: Category Games

La category

> **Games** mostra un totale di 10.098 (13%) issue di cui 9.421 Warning (93%), 472 Fatal (5%), 205 Error (2%) distribuite in 87 applicazioni

La catgory è caratterizzata dalle issue di tipo:

> **UnusedResources** (25%), **ContentDescription** (10%), **HardcodedText** (8%),

Games è quindi caratterizzata da problematiche di performance, accessibilità nel caso un'immagine non venga visualizzata correttametne e temi riguardanti i campi testuali.

3.4.2.3 Conclusioni

La maggior parte delle Issue riguardano comunque problemi di ottimizzazione e usabilità dell'applicazione ma, a valle dell'analisi è possibile affermare che, eccetto per UnusedResource, **dierse category presentano delle tipologie peculiari di issue**.

3.5 File ed Estensione

Un'altra analisi di interesse riguarda il luogo in cui si è verificata l'Issue, ovvero il file dell'intero progetto dell'applicazione Android in cui è presente il difetto.

3.5.1 Le issue sono concentrate maggiormente in determinati file?

Dato che un file è presente nel Knowledge Layer se e solo se ha almeno una issue al suo interno, si vuole comprendere sia quali file occorrono più spesso, sia se le issue sono concentrate maggiormente in alcuni file.

3.5.1.1 Metrica

Per effettuare un confronto tra file diversi non è necessario normalizzare per tipologia di file, anzi risulta dannoso in quanto ogni progetto android potrebbe avere file denominati in modo diverso, inoltre si pone il solo obiettivo di verificare i file con più issue, quindi in ogni caso non sarebbe necessaria.

Per ricavare le informazioni utili per l'analisi è necessario fare riferimento a più tabelle, in particolare:

- **File in cui è stata rilevata la issue**: informazione presente nel dato location.file2, ricavato dal file lint-report.xml come mostrato in Blocco codice 2.4. Il dato è stato inoltre modificato all'interno del Layer Analytics per ricavare dall'intero percorso il solo nome del file.
- **Tipo di issue**: informazione presente nel dato issue.id, ricavato dal file lint-report.xml come mostrato in Blocco codice 2.4.

3.5.1.2 Analisi

Di seguito è mostrato il report principale dell'analisi, da cui saranno ricavati i vari dettagi:

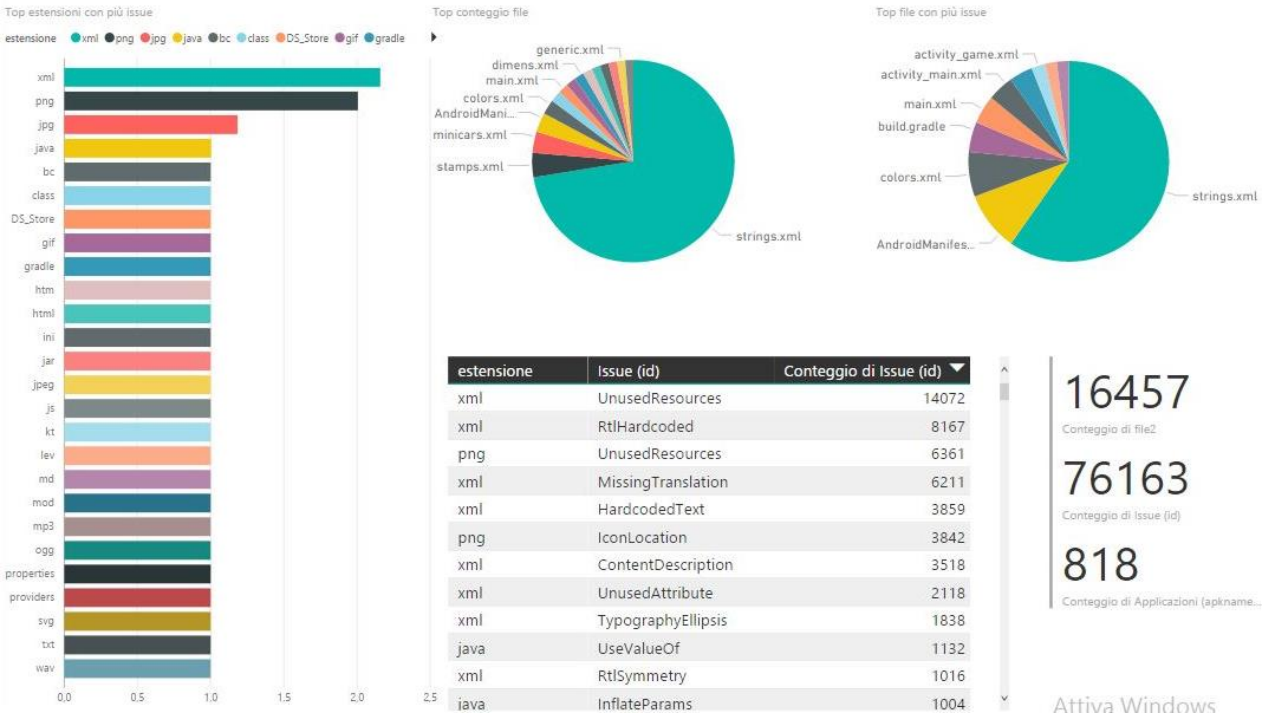


Figura 3.37 – Report: File ed Estensioni

E' evidente, nell'analisi generale sopra riportata, che il numero di issue e il numero di applicaizoni non è coerente con il resto delle analisi. La discordanza è causata dal filtro utilizzato per rendere i grafici comparabili.

Consideriamo un dettaglio di Figura 3.37:

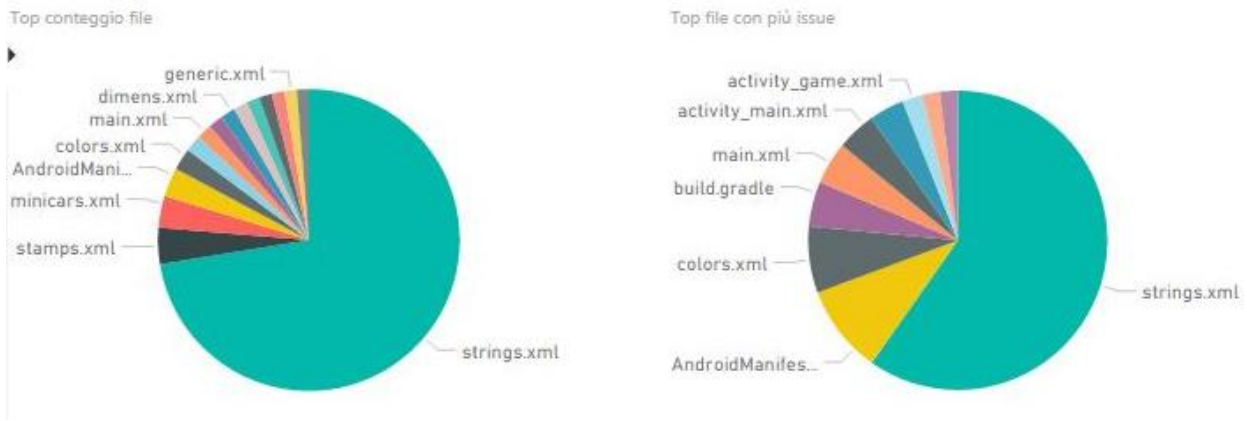


Figura 3.38 – Dettaglio Figura 3.37: (a) Numero di file, (b) Numero di issue per file

Dalla Figura 3.38 (a) si riscontra che

> i **file che occorrono più spesso** sono: strings.xml, stamps.xml e AndroidManifest.xml.

Tale informazione indica solo che sono i file più frequenti, non quelli con più issue.

Volendo invece contare il numero di issue per ogni file ritroviamo una classifica leggermetne diversa, infatti

> **le issue sono maggiormente concentrate** in: strings.xml, AndroidManifest.xml, colors.xml, build.gradle, main.xml

3.5.1.3 Conclusioni

Non è possibile entrare nel merito di ogni file in quanto dipendono dal nome assegnatogli dal programmatore ma è possibile affermare che la maggior parte degli errori sono localizzati nel file **strings.xml**.

L'analisi ha portato alla luce che la maggior parte degli errori rilevati dal tool Lint sono localizzati nei file con estensione XML, tale analisi sarà approfondita nella Sezione 3.5.3.

3.5.2 File diversi presentano issue simili?

Bisogna precisare che il nome di un file dipende dalla "fantasia" del programmatore, quindi si analizzeranno esclusivamente i file "standard" di un generico progetto Android ed in particolare i file "AndroidManifest.xml" e "strings.xml", mentre nella Sezione 3.5.3 sarà approfondita la trattazione del file "build.gradle" in quanto unico file con tale estensione.

3.5.2.1 Metrica

La metrica utilizzata fa riferimento alla Sezione 3.5.1.1, l'unica differenza è l'utilizzo del filtro per la selezione dei dati correlati ai file "AndroidManifest.xml" e "strings.xml".

3.5.2.2 Analisi

La prima analisi di riferimento sarà quella sul file "strings.xml":

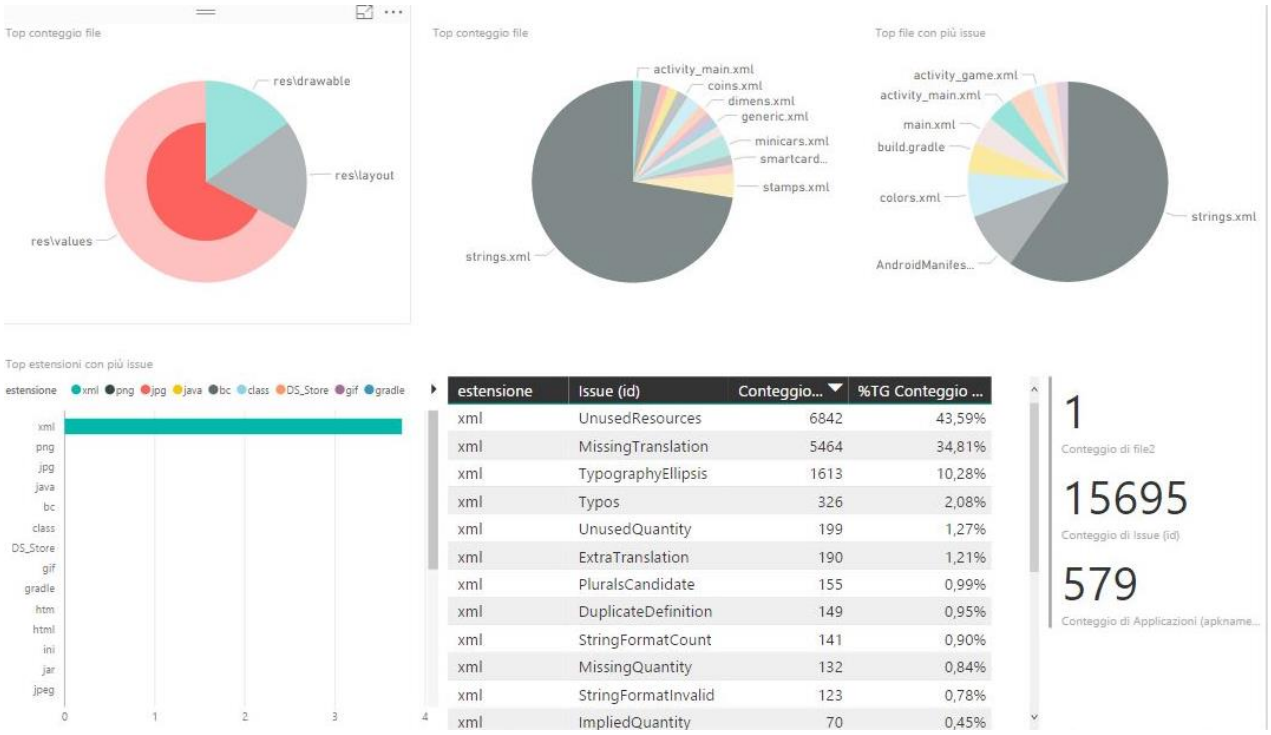


Figura 3.39 - Report: File strings.xml

Dall'analisi è possibile apprendere che

> il file **strings.xml** presenta issue in **579** applicazioni (69%) con un totale di **15.695** issue (20%).

Volendo entrare nel merito delle singole issue del file si afferma che

> il file strings.xml presenta come issue più popolari: **UnusedResources** con 6.842 issue (9%), **MissingTranslation** con 5.464 issue (7%), **TypographyEllipsis** con 1.613 issue (2%)

Per approfondimenti sulle tipologie di issue è possibile fare riferimento alla Tabella 5.1.

La seguente figura mostra le informazioni relative al file "AndroidManifest.xml":

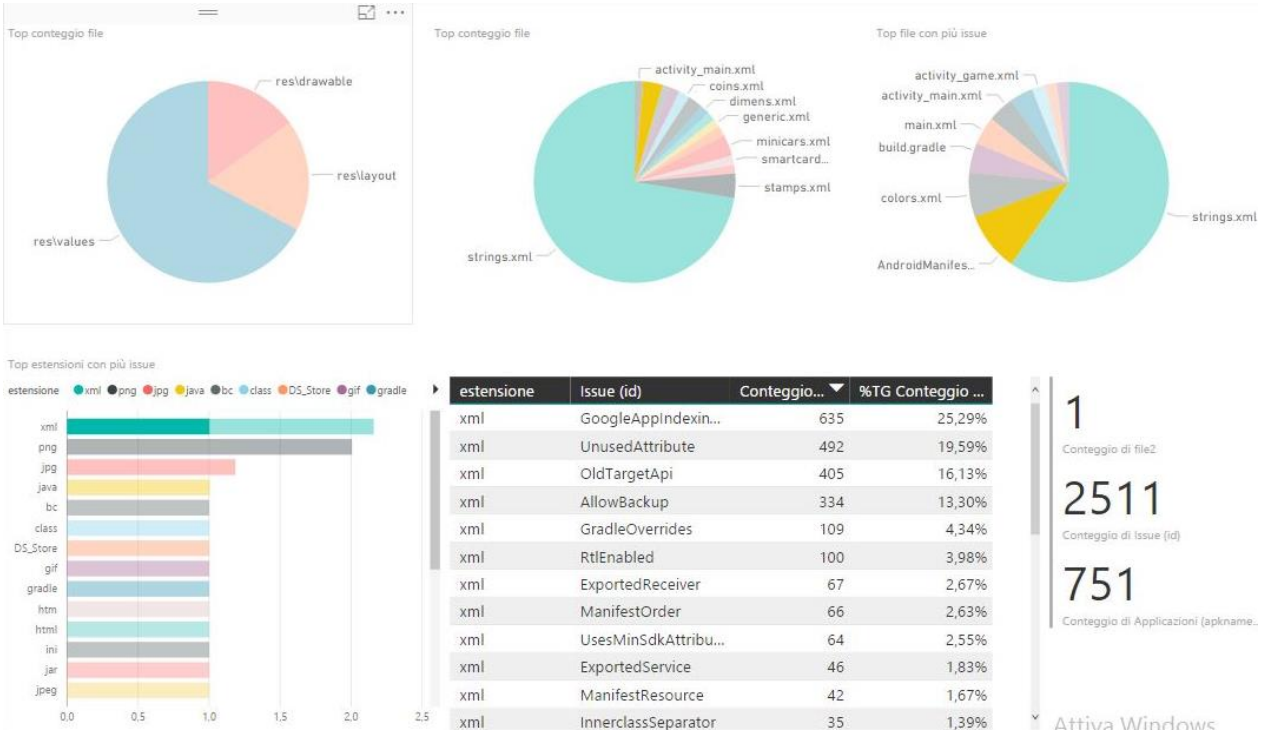


Figura 3.40 - Report: File AndroidManifest.xml

Dall'analisi è possibile apprendere che

> il file **AndroidManifest.xml** presenta degli arrori in 751 applicazioni (89%) con un totale di 2.511 issue (3%).

Volendo entrare nel merito delle singole issue del file si afferma che

> le tipologie di issue maggiormente presenti file AndroidManifest.xml sono: **GoogleAppIndexingWarning** con 635 issue (0,8%), **UnusedAttribute** con 492 issue (0,6%), **OldTargetApi** con 405 issue (0,5%), **AllowBackup** con 334 issue (0,4%)

Per approfondimenti sulle tipologie di issue è possibile fare riferimento alla Tabella 5.1.

3.5.2.3 Conclusioni

Rispondendo precisamene alla domanda si afferma che, **si, file diversi presentano issue simili**. Si precisa però che l'analisi è limitata ai file di tipo XML perché sono i file più comuni rilevati al momento, qiundi è ovvio che in file diversi, ma con la stessa estensione, possano esserci issue simili.

Nella Sezione 3.5.4 sarà appunto svolta una indagine mirata alle issue presenti nelle varie estensioni dei file.

3.5.3 Quali estensioni hanno più issue?

L'analisi della sezione precedente ha evidenziato che i file che occorrono più spesso hanno estensione XML. L'obiettivo di questa sezione è invece quello di studiare i file raggruppati per estensione e apprendere in che tipo di file siano principalmente concentrate le issue.

3.5.3.1 Metrica

Per confrontare le varie estensioni è necessario effettuare un passo di normalizzare per il numero di file appartenenti a quel tipo di estensione, in particoalre le informazioni sono ricavabili da:

- **Estensione del file**: informazione presente nel dato location.estensione, ricavato dal file lint-report.xml come mostrato in Blocco codice 2.4. Il dato è stato inoltre modificato all'interno del Layer Analytics per ricavare dall'intero percorso il solo l'estensione del file.
- **Tipo di issue**: informazione presente nel dato issue.id, ricavato dal file lint-report.xml come mostrato in Blocco codice 2.4. Tale dato va normalizzato per estensione, a tal fine è stata creata la misura "Norm per Estensione", come mostrato nell'immagine sottostante

```
Norm per Estensione = COUNTA('databaselint location'[estensione])/COUNTA('databaselint issue'[Issue (id)])
```

Si fa notare che la normalizzazione è stata effettuata al contraro in quanto l'Analytics Layer tendeva ad approssimare i valori, troppo piccoli, a zero rendendo l'analisi impossibile.

E' ovvio immaginare che il numero di estensioni sia abbastanza elevato, per questo è stato applicato un filtro in modo da considerare solo le estensioni più frequenti.

3.5.3.2 Analisi

Di seguito un dettaglio di Figura 3.37:

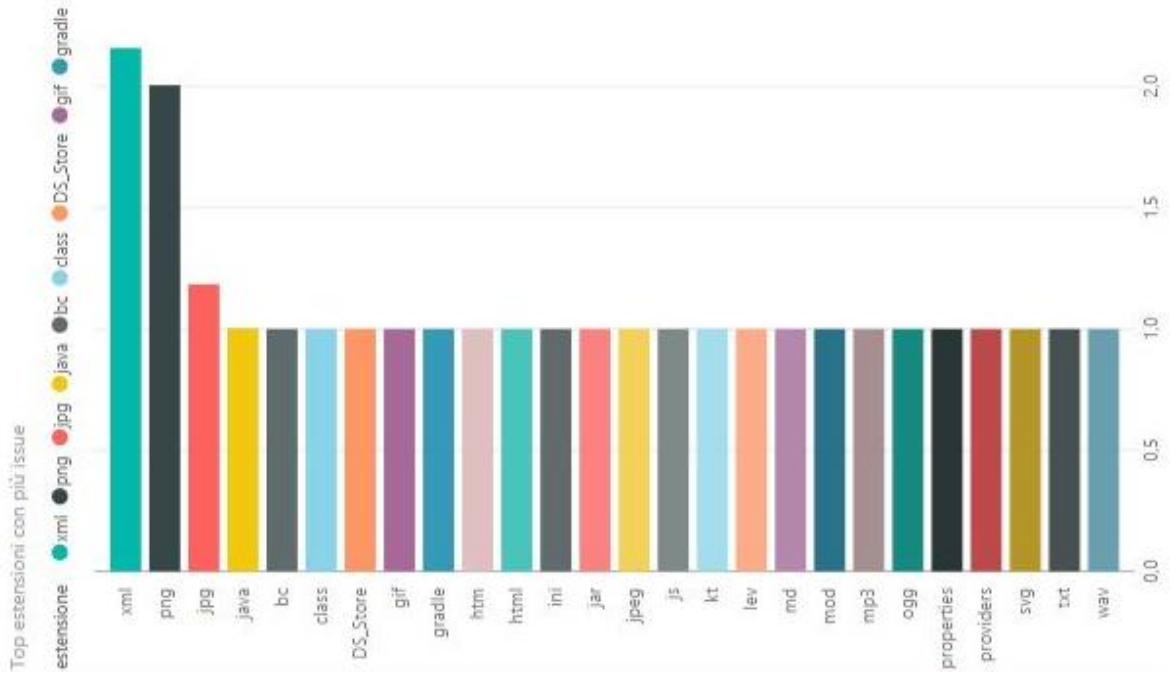


Figura 3.41 – Dettaglio Figura 3.37, Estensioni con più issue

L'analisi evidenzia, a valle della normalizzazione per numero di file, che

> i file con più issue sono quelli con estensione XML, PNG, JPG e Java,

L'analisi seguente entrerà nel merito di ogni file per evidenziare issue peculiari.

3.5.3.3 Conclusioni

I file con più issue sono quelli con estensione XML, PNG, JPG e Java.

3.5.4 Esistono issue peculiari per ogni estensione o file?

Appresa che esiste una sostanziale differenza nel numero di issue presenti in ogni estensione, si vuole ora comprendere che tipo di issue partecipano ad ogni tipologia di file e se ci sono o meno degli elementi in comune.

3.5.4.1 Metrica

La metrica utilizzata rispecchia quella evidenziata nella Sezione 3.5.3.1, l'unica differenza st nel filtro applicato nell'Analytics Layer.

Le percentuali di “file”, “issue” e “applicazioni” saranno mostrate rispetto ai totali mostrati in Figura 3.37, mentre la percentuale di dettaglio delle singole tipologie di issue sarà mostrato in relazione al totale in Figura 3.1.

3.5.4.2 Analisi

Le analisi di seguito si concentrano esclusivamente sulle estensioni più interessanti, tra cui: XML, PNG, JPG, Java, Gradle.

Nella prima analisi sarà studiata l’estensione XML:

estensione	Issue (id)	Conteggio di Issue (id)
xml	UnusedResources	14072
xml	RtlHardcoded	8167
xml	MissingTranslation	6211
xml	HardcodedText	3859
xml	ContentDescription	3518
xml	UnusedAttribute	2118
xml	TypographyEllipsis	1838
xml	RtlSymmetry	1016
xml	SpUsage	828
xml	Overdraw	826
xml	ObsoleteLayoutParam	716
xml	GoogleAppIndexingW...	635

Figura 3.42 - Report: Estensione XML

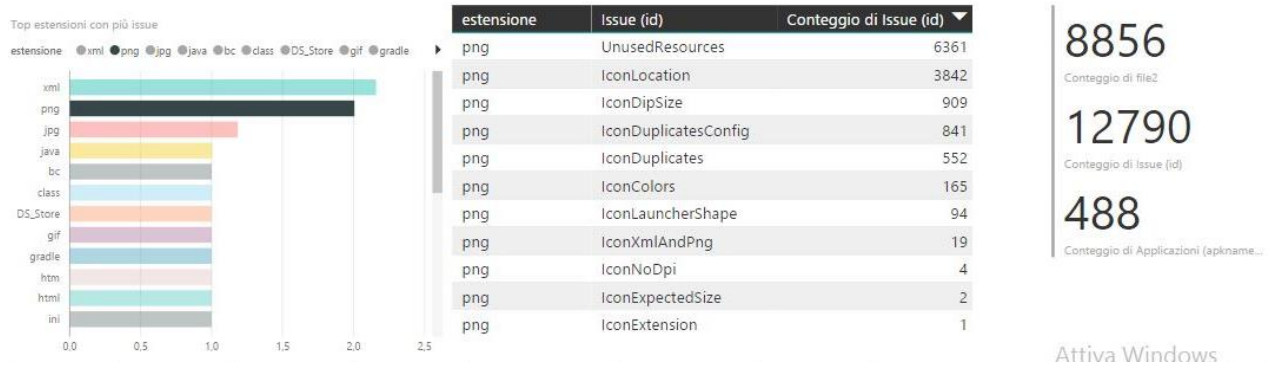
La figura sopra sottolinea che

> i file con estensione XML sono 4.724 (29%) e presentano 53.157 (70%) issue appartenenti a 802 (95%) applicazioni

Le tipologie di issue più comuni in tali file sono

> UnusedResources 14.072 (18%), RtlHardcoded 8.167 (10%), MissingTranslation 6.211 (8%), HardcodedText 3.859 (5%)

in pratica si tratta delle issue più frequenti dell’intero caso di studio. Si evidenzia che le proporizoni sono calcolate rispetto alla stessa tipologia di issue e non rispetto al totale delle issue.



estensione	Issue (id)	Conteggio di Issue (id)
png	UnusedResources	6361
png	IconLocation	3842
png	IconDipSize	909
png	IconDuplicatesConfig	841
png	IconDuplicates	552
png	IconColors	165
png	IconLauncherShape	94
png	IconXmlAndPng	19
png	IconNoDpi	4
png	IconExpectedSize	2
png	IconExtension	1

Figura 3.43 - Report: Estensione PNG

L’analisi evidenzia che

> i file con estensione PNG sono 8.856 (29%) e presentano 12.790 (17%) issue appartenenti a 488 (60%) applicazioni

Le tipologie di issue più comuni in tali file sono

> UnusedResources 6.361 (8%), IconLocation 3.842 (5%), IconDipSize 909 (1,16%), IconDuplicatesConfig 841 (1,07%), IconDuplicates 552 (0,7%)

Si riscontrano tipologie interessanti di issue, non evidenziate da altre analisi: IconDipSize, IconDuplicatesConfig, IconDuplicates. Per approfondimenti fare riferimento alla Tabella 5.1.

La prossima analisi ha come scopo approfondire l'estensione JPG:

Figura 3.44 - Report: Estensione JPG

Si apprende che

> i file con estensione JPG sono 63 (0,38%) e presentano 76 (0,1%) issue appartenenti a 27 (3,3%) applicazioni

Le tipologie di issue più comuni in tali file sono

> IconLocation e UnusedResources, rispettivamente con 35 e 30 occorrenze

Da tali pochi dati si può desumere che in alcuni casi si utilizzano come icone dei file con estensioneJPG anziché PNG. Inoltre si apprende che, nonostante siano poche le applicazioni ad avere questi file, sono comunque moltissimi gli errori commessi nell'utilizzarli.

Le prossime due analisi si concentreranno su due tipologie particolari di file: Java e Gradle.

Figura 3.45 - Report: Estensione Java

Per quanto riguarda la prima estensione possiamo dire che

> i file con estensione **Java** sono 2.487 (15%) e presentano 8.456 (11%) issue appartenenti a 574 (70%) applicazioni

Le tipologie di issue più comuni in tali file sono

> **UseValueOf** 1.132 (1,4%), **InflateParams** 1.004 (1,3%), **DefaultLocale** 981 (1,2%), **SetTextI18n** 886 (1,1%), **InlinedApi** 805 (1%)

L'estensione di tipo Gradle, invece, presenta le seguenti caratteristiche:

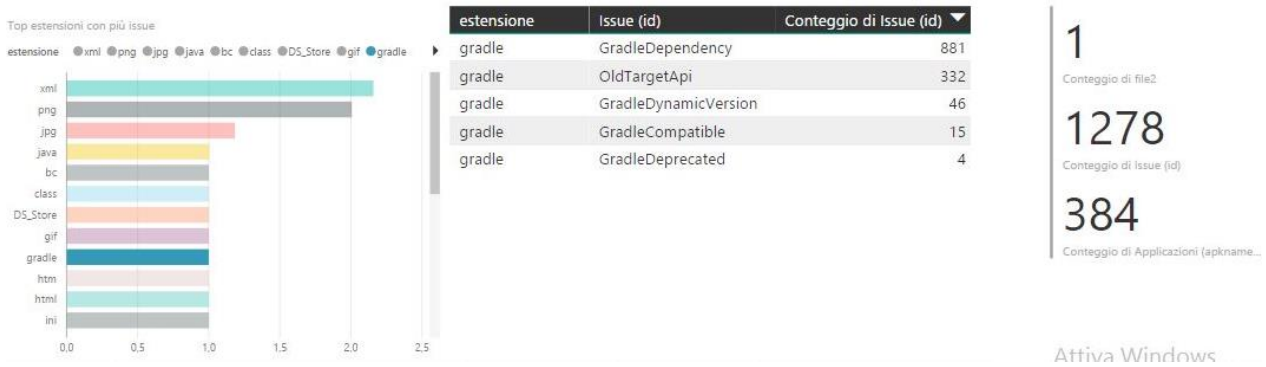


Figura 3.46 - Report: Estensione Gradle

> c'è un unico tipo di file con estensione **Gradle**, ovviamente si fa riferimento al file "build.gradle"

che presenta 1.278 (1,6%) issue appartenenti a 384 (47%) applicazioni

Le tipologie di issue più comuni in tali file sono

> **GradleDependency** 881 (1%), **OldTargetApi** 332 (0,4%), **GradleDynamicVersion** 46 (0,06%)

L'analisi ha ovviamente portato alla luce tipologie di issue non presenti in altre casistiche. Per approfondimenti fare riferimento alla Tabella 5.1.

3.5.4.3 Conclusioni

In conclusione è possibile affermare sia che esistono Issue comuni a più estensioni (es. UnusedResource) sia Issue specifiche per ogni tipologia di estensione, come era ovvio immaginare.

3.6 Versione Android SDK

3.6.1 Qual è la versione SDK con più issue?

L'attuale obiettivo dell'analisi è apprendere se esiste una Versione SDK di Android che presenta più issue rispetto alle altre.

3.6.1.1 Metrica

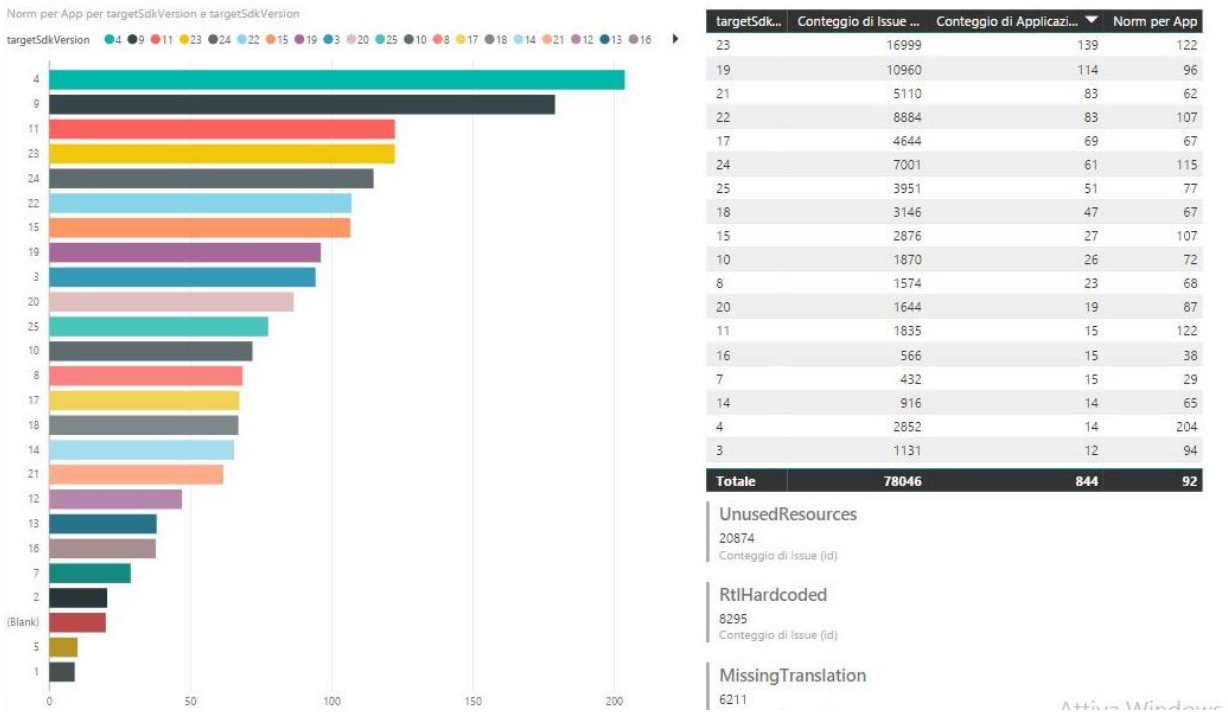
Per effettuare un confronto tra SDK diverse è necessario normalizzare per numero di applicaizioni sviluppate con quella precisa SDK.

Per ricavare le informazioni utili ai fini dell'analisi è necessario fare riferimento a più tabelle, in particolare:

- **SDK utilizzata per sviluppare l'applicazione**: informazione presente nel dato package.targetSdkVersion, ricavato dal file lint-report.xml come mostrato in Blocco codice 2.4. Il dato è stato inoltre modificato all'interno del Layer Analytics per ricavare dall'intero percorso il solo nome del file.
- **Tipo di issue**: informazione presente nel dato issue.id, ricavato dal file lint-report.xml come mostrato in Blocco codice 2.4.

3.6.1.2 Analisi

Di seguito l'analisi effettuata:



targetSdk...	Conteggio di Issue ...	Conteggio di Applicazi...	Norm per App
23	16999	139	122
19	10960	114	96
21	5110	83	62
22	8884	83	107
17	4644	69	67
24	7001	61	115
25	3951	51	77
18	3146	47	67
15	2876	27	107
10	1870	26	72
8	1574	23	68
20	1644	19	87
11	1835	15	122
16	566	15	38
7	432	15	29
14	916	14	65
4	2852	14	204
3	1131	12	94
Totale	**78046**	**844**	**92**

Figura 3.47 - Report: Versione SDK

Volendo entrare nel dettaglio possiamo immediatamente constatare che

> l'SDK con più issue è la 4, seguita dalla 9 e dalla 11.

Mentre, volendo valutare la "popolarità" di una SDK è possibile affermare che

> l'SDK più utilizzata è la 23 con cui sono state sviluppate 139 applicazioni (16%).

E' interessante notare come l'SDK 23 sia la quarta SDK con più issue.

3.6.1.3 Conclusioni

Effettivamente è possibile affermare che l'SDK Versione 4 è quella che presenta il maggior numero di issue.

4 Conclusioni e Sviluppi futuri

In conclusione è possibile affermare che tramite Lint è possibile ottenere una conoscenza concreta delle problematiche del framework Android SDK e degli errori più comuni commessi dai programmatori.

Inoltre sono stati riscontrati failure comuni a più applicazioni e a più versioni del framework che possono essere risolti grazie all'aiuto dello strumento Lint.

Per quanto riguarda gli sviluppi futuri del progetto, un primo intervento da effettuare consiste nell'aggiornare le analisi con tutte le applicazioni della Repository F-Droid e magari estendere le considerazioni a diverse repository.

5 Appendice

Si elencheranno di seguito le Issue incontrate nel corso delle varie analisi, indicando Severity ed Explanation:

Issue	Severity	Explanation
AllowBackup	Warning	The allowBackup attribute determines if an application's data can be backed up and restored.
ContentDescription	Warning	Non-textual widgets like ImageViews and ImageButtons should use the `contentDescription` attribute to specify a textual description of the widget such that screen readers and other accessibility tools can adequately describe the user interface.
DefaultLocale	Warning	Calling `String#toLowerCase()` or `#toUpperCase()` *without specifying an explicit locale* is a common source of bugs. The reason for that is that those methods will use the current locale on the user's device, and even though the code appears to work correctly when you are developing the app, it will fail in some locales. For example, in the Turkish locale, the uppercase replacement for `i` is *not* `I`.
DuplicateIds	Fatal	Within a layout, id's should be unique since otherwise `findViewById()` can return an unexpected view
GoogleAppIndexingWarning	Warning	Adds URLs to get your app into the Google index, to get installs and traffic to your app from Google Search.
GradleDependency	Warning	This detector looks for usages of libraries where the version you are using is not the current stable release. Using older versions is fine, and there are cases where you deliberately want to stick with an older version. However, you may simply not be aware that a more recent

		version is available, and that is what this lint check helps find.
GradleDependency	Warning	This detector looks for usages of libraries where the version you are using is not the current stable release. Using older versions is fine, and there are cases where you deliberately want to stick with an older version. However, you may simply not be aware that a more recent version is available, and that is what this lint check helps find.
GradleDynamicVersion	Warning	Using `+` in dependencies lets you automatically pick up the latest available version rather than a specific, named version. However, this is not recommended; your builds are not repeatable; you may have tested with a slightly different version than what the build server used. (Using a dynamic version as the major version number is more problematic than using it in the minor version position.)
HardcodedText	Warning	Hardcoding text attributes directly in layout files is bad for several reasons. […] In Android Studio and Eclipse there are quickfixes to automatically extract this hardcoded string into a resource lookup.
IconDipSize	Warning	Checks the all icons which are provided in multiple densities, all compute to roughly the same density-independent pixel (`dip`) size. This catches errors where images are either placed in the wrong folder, or icons are changed to new sizes but some folders are forgotten.
IconDuplicates	Warning	If an icon is repeated under different names, you can consolidate and just use one of the icons and delete the others to make your application smaller. However, duplicated icons usually are not intentional and can sometimes point to icons that were

		accidentally overwritten or accidentally not updated.
IconDuplicatesConfig	Warning	If an icon is provided under different configuration parameters such as `drawable-hdpi` or `-v11`, they should typically be different. This detector catches cases where the same icon is provided in different configuration folder which is usually not intentional.
IconLocation	Warning	The res/drawable folder is intended for density-independent graphics such as shapes defined in XML. For bitmaps, move it to `drawable-mdpi` and consider providing higher and lower resolution versions in `drawable-ldpi`, `drawable-hdpi` and `drawable-xhdpi`. If the icon *really* is density independent (for example a solid color) you can place it in `drawable-nodpi`.
InflateParams	Warning	When inflating a layout, avoid passing in null as the parent view, since otherwise any layout parameters on the root of the inflated layout will be ignored.
InflateParams	Warning	When inflating a layout, avoid passing in null as the parent view, since otherwise any layout parameters on the root of the inflated layout will be ignored.
InlinedApi	Warning	This check scans through all the Android API field references in the application and flags certain constants, such as static final integers and Strings, which were introduced in later versions. These will actually be copied into the class files rather than being referenced, which means that the value is available even when running on older devices. In some cases that's fine, and in other cases it can result in a runtime crash or incorrect behavior.

MissingTranslation	Fatal	If an application has more than one locale, then all the strings declared in one language should also be translated in all other languages.
ObsoleteLayoutParam	Warning	The given layout_param is not defined for the given layout, meaning it has no effect. This usually happens when you change the parent layout or move view code around without updating the layout params. This will cause useless attribute processing at runtime, and is misleading for others reading the layout so the parameter should be removed.
OldTargetApi	Warning	When your application runs on a version of Android that is more recent than your `targetSdkVersion` specifies that it has been tested with, various compatibility modes kick in. This ensures that your application continues to work, but it may look out of place. For example, if the `targetSdkVersion` is less than 14, your app may get an option button in the UI.
OldTargetApi	Warning	When your application runs on a version of Android that is more recent than your `targetSdkVersion` specifies that it has been tested with, various compatibility modes kick in. This ensures that your application continues to work, but it may look out of place. For example, if the `targetSdkVersion` is less than 14, your app may get an option button in the UI. To fix this issue, set the `targetSdkVersion` to the highest available value. Then test your app to make sure everything works correctly.
Overdraw	Warning	If you set a background drawable on a root view, then you should use a custom theme where the theme background is null. Otherwise, the theme background will be painted

		first, only to have your custom background completely cover it; this is called ""overdraw".
PxUsage	Warning	For performance reasons and to keep the code simpler, the Android system uses pixels as the standard unit for expressing dimension or coordinate values. That means that the dimensions of a view are always expressed in the code using pixels, but always based on the current screen density.
RtlHardcoded	Warning	Using `Gravity#LEFT` and `Gravity#RIGHT` can lead to problems when a layout is rendered in locales where text flows from right to left. Use `Gravity#START` and `Gravity#END` instead. Similarly, in XML `gravity` and `layout_gravity` attributes, use `start` rather than `left`
SetTextI18n	Warning	When calling `TextView#setText` * Never call `Number#toString()` to format numbers; it will not handle fraction separators and locale-specific digits properly. Consider using `String#format` with proper format specifications (`%d` or `%f`) instead. * Do not pass a string literal (e.g. ""Hello"") to display text. Hardcoded text can not be properly translated to other languages. Consider using Android resource strings instead. * Do not build messages by concatenating text chunks. Such messages can not be properly translated.
TypographyEllipsis	Warning	You can replace the string ""..."" with a dedicated ellipsis character, ellipsis character (…, …). This can help make the text more readable.
UnusedAttribute	Warning	This check finds attributes set in XML files that were introduced in a

		version newer than the oldest version targeted by your application (with the `minSdkVersion` attribute)
UnusedResources	Warning	Unused resources make applications larger and slow down builds
UseValueOf	Warning	You should not call the constructor for wrapper classes directly, such as`new Integer(42)`. Instead, call the `valueOf` factory method, such as `Integer.valueOf(42)`. This will typically use less memory because common integers such as 0 and 1 will share a single instance.

Tabella 5.1 - Issue più frequenti

6 Riferimenti

[1] «Intro Android Lint,» [Online]. Available: https://developer.android.com/studio/write/lint.html#overview.

[2] «Android studio project,» [Online]. Available: http://tools.android.com/tips/lint.

[3] «Android studio project, lint check completi,» [Online]. Available: http://tools.android.com/tips/lint-checks.

[4] «F-Droid,» [Online]. Available: https://f-droid.org/.

[5] «PowerBI,» [Online]. Available: https://powerbi.microsoft.com/it-it/.