ANÀLISI I MILLORA TÈCNICA D'UNA APLICACIÓ ANDROID

Luis Felipe Dias da Costa Olmo, Universitat Autònoma de Barcelona

Resum—Aquest projecte consisteix en l'anàlisi i la millora tècnica d'una aplicació Android existent del propi alumne Luis Felipe Dias da Costa Olmo. Aquesta aplicació està publicada al Play Store de Google des de fa més de dos anys, compta amb més de 80.000 descàrregues totals, així com més de 15.000 usuaris actius, i li està generant certa quantitat d'ingressos. Aquesta aplicació té una sèrie de problemes que estan afectant a usuaris reals i el codi no és tot lo correcte que hauría de ser. La proposta del projecte es dur a terme una sèrie d'accions sobre el codi per a reduir els errors, la complexitat ciclomàtica dels mètodes que sobrepassen els llindars estàndard així com millorar la qualitat del producte; l'objectiu final és aconseguir més descàrregues de l'aplicació, millor valoració en el Play Store i més ingressos. Per dur a terme el projecte i poder demostrar els resultats, s'ha fet un anàlisi inicial i final tant de l'estat al Play Store com del codi.

Paraules clau—Android, aplicació, refactor, complexitat ciclomàtica, anàlisi, IDE, Android Studio, plugins, Fabric.

Abstract—This project consists in the analysis and technical improvement of an existing Android application from the student Luis Felipe Dias da Costa Olmo. This application has been published in the Google's Play Store for two years, counts with more than 80.000 total downloads, as well as more than 15.000 active users, and it's giving him some earnings. This application has a series of roblems affecting real users and the code is not very correct. The proposal of this project is to do some actions in the code in order to reduce the number of errors, the Cyclomatic Complexity of the methods that are above the standards and also improve the quality of the product; the final objective is to achieve more downloads, a better score in the Play Store and more earnings. To make the project successful and being able to demonstrate the results, an initial and final analysis of the Play Store and code status have been made.

Index Terms—Android, application, refactor, cyclomatic complexity, analysis, IDE, Android Studio, plugins, Fabric.

1 INTRODUCCIÓ

DEGUT a l'extens ús dels dispositius mòbils amb sistema operatiu Android i al meu interès per la programació en general, vaig començar a dedicar el meu temps lliure a la creació de petites aplicacions per Android i publicar-les al Play Store. Degut a la nostàlgia i com a excusa per a aprendre i practicar el desenvolupament d'una aplicació una mica més complexa, fa qüestió de dos anys vaig crear i publicar una aplicació que és un joc anomenat Random Adventure Roguelike, que va néixer com a tribut dels clàssics *Dungeon Crawl* i *MUD*, gèneres que tenien molt d'èxit als inicis dels jocs d'ordinador, quan gairebé no hi havia res de gràfics. Amb el temps, el projecte ha anat creixent, afegint millores en quant a jugabilitat i usabilitat i ha anat creixent en nombre d'usuaris i d'ingressos (per publicitat i per ventes). El problema és que en el moment de crear aquesta aplicació, no vaig seguir tots els processos d'Enginyeria del Software que hagués hagut de seguir i, per tant, cada vegada costa més d'afegir noves funcionalitats, i hi ha més fonts de problemes. És per això que vaig decidir fer aquest projecte per tal d'analitzar i millorar la qualitat del codi, per tal de convertir el projecte en quelcom més mantenible, escalable i re aprofitable.

En un inici, junt amb el tutor del projecte Rodolfo Guichón, vam estar pensant quina hauria de ser la mesura principal sobre el codi per a saber per on començar i vam decidir que aquesta hauría de ser la Complexitat Ciclomàtica (*CC* a partir d'ara). També havíem de trobar la forma de monitoritzar els *bugs* que hi havia en el joc, i com estaven afectant als usuaris reals. Seguint una metodologia *agile*, al llarg del desenvolupament del projecte, hem anat definint i variant els objectius sobre els que treballar per tal d'adaptar-nos a les necessitats del mateix.

Gràcies a l'ús del *IDE Android Studio* i alguns *plugins*, hem pogut analitzar el codi de manera automàtica, extraient les dades de *CC*, entre d'altres. També hem pogut afegir un framework anomenat *Fabric* que envia dades sobre *crashes* automàticament, permetent veure tot l'*stacktrace* (traça de la pila), així com veure el nombre d'usuaris que tenen certa actualització, usuaris en temps real, etc…

També, mitjançant la consola de desenvolupador del Play Google, es poden veure estadístiques de descàrregues, usuaris, valoracions, puntuació de l'aplicació i errors, entre d'altres.

Amb tot això, al llarg de l'article es podrà veure quin era l'estat inicial de l'aplicació al començar el projecte, quines accions s'han realitzat i quin ha sigut l'estat de l'aplicació al finalitzar el projecte, i quins seran els passos a seguir.

2 ESTAT DE L'ART

2.1 Estat inicial del projecte

Les dades que es mostren a continuació són extretes de la consola de desenvolupador del Play Store, on es publiquen les aplicacions i es permet veure estadístiques i molta informació relacionada amb l'aplicació, així com dades extretes mitjançant l'ús d'alguna eina de l'Android Studio.

A l'inici del projecte, les xifres sobre descàrregues de l'aplicació eren les següents:

- Descàrregues totals **64.000**
- Descàrregues actives **12.000**

Descàrregues totals (Instal·lacions totals per usuari)

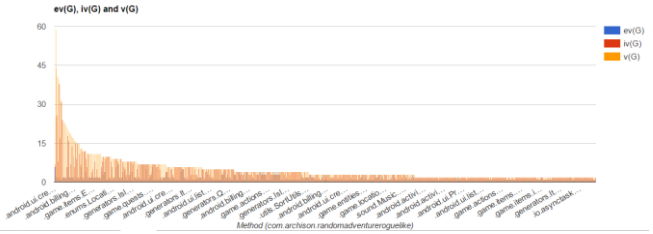
Descàrregues actives (Instal·lacions actuals per dispositiu)

Les dades sobre valoració de l'aplicació i ressenyes eren les següents:

- Puntuació mitja: **4'208** (sobre 5)
- Puntuacions totals: **3.747**
- Nombre de ressenyes: **1.013**

Valoració mitja, puntuacions totals i nombre de ressenyes

Mitjançant l'ús del *plugin* per a Android Studio *MetricsReloaded*, es poden extreure dades sobre el codi com per exemple nombre de línies, percentatge de Javadoc en el codi, o CC, per exemple. A l'executar-lo sobre el projecte, en quant a les dades sobre CC, les mesures es poden veure a nivell de package, classe o mètode. En quant a la complexitat dels mètodes, es pot veure en la següent gràfica quina era la seva CC:



Complexitat ciclomàtica dels mètodes del projecte

Tot i que no es pot observar al detall, es pot veure com hi ha certa quantitat de mètodes que tenen una CC superior a l'estàndard que és 10. En total, a l'inici del projecte hi havia més de 50 mètodes amb CC major de 10.

En la següent taula faig una petita mostra dels mètodes amb més complexitat ciclomàtica extrets de l'anàlisi (no en poso més per qüestions d'espai a l'article).

Method	CC
.android.ui.creators.ImageViews.getItemIconDrawable	59
.io.Printer.getPrintMapSize	44
.android.ui.listeners.impl.ItemUsageButtonListener.onClick	41
.game.options.OptionManager.performOption	40
.game.options.OptionManager.setAllOptions	38
.generators.LocationGenerator.configureLocation	32
.io.Printer.printLocationThings	31
.generators.IslandGenerator.createMinimumLocationTypesMap	24
.android.ui.creators.LinearLayouts.createItemInfoLayout	23
.utils.ColorUtils.getItemTypeColor	23

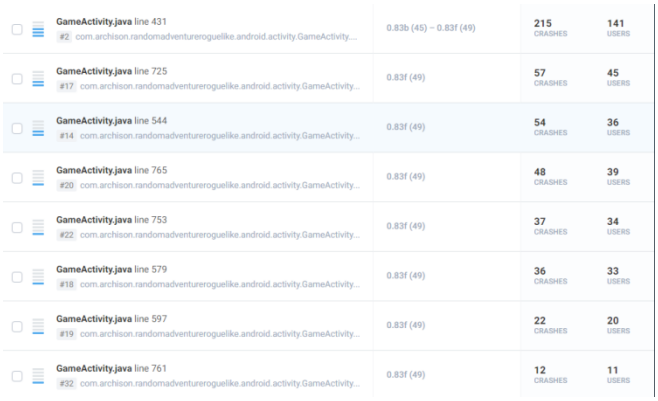
A nivell d'errors i bugs, per una banda, es podia observar a la consola de desenvolupador errors relacionats amb el guardat i càrrega de dades, així com d'altres errors que feien tancar-se l'aplicació.

Amb la integració de *Fabric*, vam poder començar a observar a producció quins eren els errors que estaven tenint els usuaris.

Estat global sobre Crashes de l'aplicació.

L'estat de l'aplicació a l'inici del projecte no el vam poder veure amb aquesta eina ja que no estava encara imple-

mentat, però a partir de maig, quan vam fer l'actualització de l'aplicació, ja vam començar a rebre dades sobre crashes.



Mostra de dades sobre crashes.

Els crashes més importants que tenia l'aplicació eren deguts a un problema amb el guardat i càrrega de dades a l'aplicació. El problema provocava que els jugadors no poguessin recuperar la seva partida guardada i haguessin de començar a jugar des del principi quan això passava.

3 OBJECTIUS

Els objectius de projecte són els següents:

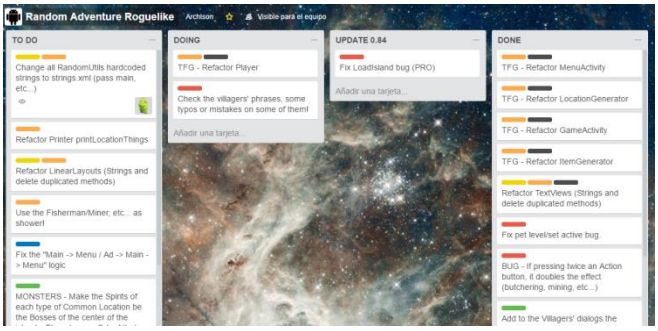
- Reducció dels bugs i crashes de l'aplicació
- Millorar l'escalabilitat de l'aplicació
- Fer que el codi sigui mantenible i extensible
- Augmentar les descàrregues de l'aplicació
- Millorar la valoració de l'aplicació

4 METODOLOGIA

Degut a la naturalesa del projecte, i seguint la forma de treball en la que s'ha estat treballant sobre l'aplicació anteriorment, amb el tutor vam decidir fer servir una metodologia *Agile* tipus *Scrum* en la que definíem els sprints a cada reunió de seguiment. A cada reunió hem definit la planificació, i, utilitzant el *Trello* com a eina de gestió de les tasques a realitzar hem pogut anar fent el seguiment de les mateixes.

Cada etapa del projecte vam definir els objectius de cara a la següent entrega i així successivament fins al final. A la reunió i als informes de seguiment vaig anar explicant l'estat de les tasques planificades, així com els canvis pertinents en la planificació i la definició dels següents passos a realitzar.

Cal destacar que a més del refactor i millores sobre el codi de cara al projecte, també he seguit el desenvolupament de l'aplicació en paral·lel.



Exemple de les tasques al Trello.

5 PLANIFICACIÓ I EVOLUCIÓ

5.1 Planificació inicial (març)

- Anàlisi de l'estat de l'aplicació
- Treballar en la refactorització dels següents quatre packages de major a menor complexitat ciclomàtica.
- Anàlisi de la complexitat ciclomàtica al finalitzar la refactorització inicial
- Corregir els bugs existents a producció
- Desplegar a producció l'aplicació
- Lliurament Informe de Progrés I

El resultat de la primera planificació va ser regular. Per una banda, sí que es va dur a terme l'anàlisi de l'estat de l'aplicació (realitzada per a l'informe inicial). En quant a corregir els bugs existents a producció, només es van poder resoldre alguns dels errors menors detectats mitjançant la consola de desenvolupador de la Play Store, perquè encara no teníem visibilitat de tots els errors existents (la implementació de *Fabric* vingué després). Per altre banda, tot i que es va estar fent refactor per a reduir CC, ens vam adonar que havíem plantejat erròniament el criteri a l'hora de decidir què refactoritzar. No ens havíem de fixar en la CC total dels package o classes, sino en la Complexitat Ciclomàtica dels mètodes individualment. A partir d'aquest punt vam veure que havíem de treballar sobre els mètodes amb CC superior a 10 i no tenir en compte les sumes totals de la mateixa. Per últim, es va desplegar l'aplicació a producció amb certes millores a nivell de codi i alguns errors corregits.

5.2 Segona planificació (abril)

- Implementació de SQLITE per al sistema de guardat i càrrega de les partides

- Un cop solucionats els problemes de carregar i guardar partida, continuar amb el refactor d'aquells mètodes de complexitat ciclomàtica superior a 10, en ordre de major a menor

A l'hora de planificar aquest *Sprint* vam decidir que realment, a més de seguir millorant la qualitat del codi reduint la complexitat dels mètodes de l'aplicació, havíem de solucionar urgentment els problemes més greus que hi havia a producció. El més greu en aquells moments era el fet de que alguns usuaris estaven reportant errors i problemes a l'hora de carregar la partida guardada. Això es va convertir en la prioritat principal, ja que això podia provocar mala imatge, mals comentaris i puntuacions, menys descàrregues, etcètera. La proposta inicial era fer una conversió del sistema de guardar i carregar les dades de l'actual conversió a JSON a fitxer, a un sistema que fes servir el sistema de base de dades SQLite.

SQLite és una base de dades pròpia de les aplicacions mòbils, que es crea dins l'aplicació i serveix per a persistir informació de forma local. És més eficient, més ràpid, etc… però a l'hora, és més complexa de desenvolupar, mantenir, ampliar, etcètera.

Vaig estar investigant sobre *ORM* (*Object Relational Mapping*) i, després de trobar diverses opcions, vaig trobar un en concret que semblava que podia servir, anomenat *Sugar ORM*.

Amb aquest ORM vaig estar intentant aplicar-lo en alguna petita part del projecte a mode de prova però em va donar molts problemes ja que les relacions 1:n no les suportava. A més, donat el elevadíssim nombre de classes a persistir, l'elevada dependència entre les diferents classes… em vaig adonar de que convertir el sistema de guardat i càrrega a SQLite era més aviat una tasca faraònica i que, en tot cas, valdria més la pena començar una nova aplicació des de zero ja plantejada amb aquest sistema.

Per no perdre el fil, vaig tornar a la motivació inicial, que era corregir el problema de les partides guardades. Entre d'altres coses, vaig contactar amb diversos usuaris que havien fet comentaris sobre el problema, i els hi vaig demanar més feedback sobre quina casuística podia provocar aquests problemes. A més a més, vaig seguir investigant sobre el tema.

Després de dies de rebre respostes, i d'analitzar l'aplicació, me'n vaig adonar i vaig ser capaç de reproduir una casuística que fa que el guardat de la partida es corrompís.

Explicat resumidament, el sistema de guardat es pot executar en diferents ocasions:

- L'usuari prem el botó d'anar enrere, apareix el menú de pausa, i té dos botons per a guardar la partida, "SAVE" i "SAVE AND EXIT". Els dos botons criden al sistema de guardat i guarden la partida. L'única diferència és que el segon fa que l'aplicació torni al menú principal de l'aplicació.

- L'usuari prem el botó "HOME" del mòbil (el del mig d'Android). Aquí, el procés de guardat s'executa en background i també guarda la partida.

- L'usuari prem el botó "SHORTCUTS" del mòbil (el que obre la pantalla de totes les aplicacions que estan executant-se al mòbil). En aquest cas també s'executa el procés de guardat en background.

El problema venia donat en els dos últims casos. Per una banda comentar que, de cara a l'aplicació i el sistema Android, la crida al procés de guardat es fa tant en un cas com en l'altre. Tant li fa si és un botó o l'altre. Per altre banda (i aquí crec que està el problema), quan l'usuari prem el botó de "SHORTCUTS", pot tancar aplicacions desplaçant-les una a una. Doncs bé, aquí, si l'usuari està dins la meva aplicació, fa clic a aquest botó i tanca l'aplicació fent aquest gest, el sistema de guardat, que encara s'està executant, no finalitza el guardat correctament, i per tant, les dades queden corrompudes, sobreescrites sense finalitzar i deixa de servir.

La solució aleshores per al problema, en comptes de fer el canvi faraònic (per no dir que ho vaig veure inviable o que em veia incapaç de realitzar) vaig veure que traient aquesta funcionalitat d'auto guardat al sortir pel botó "HOME" i el de "SHORTCUTS", potser hauría solucionat el problema. No estava segur de si hi hauria alguna altre casuística que pogués provocar el problema. Ho veuria un cop hagués actualitzat l'aplicació.

A destacar, a més de la planificació realitzada, vaig implementar *Fabric* i pujar a producció l'aplicació actualitzada. Amb aquesta actualització vaig començar a rebre estadístiques d'usuaris actius en temps real, així com dades sobre els *crashes* que succeïen als jugadors.

Exemple de visualització d'usuaris en temps real al Fabric

5.3 Tercera planificació – planificació final (maig)

- Acabar el refactor dels mètodes de Complexitat Ciclomàtica superiors a 10.
- Haver pujat actualització de l'aplicació i poder demostrar que s'han reduït els errors i els casos de pèrdua de dades de la partida.
- Continuar fent millores en el codi i bugfixing.
- Preparar la proposta d'article / article final.

He deixat gairebé tots els mètodes amb una CC igual o inferior a 10. Comentat a la secció *Estat final de l'aplicació*.

He realitzat tots els canvis que tenia pendents de cara a la internacionalització, de manera que ara només quedarà pendent traduir els textos del fitxer **strings.xml** per tal de poder gaudir de l'aplicació en altres idiomes.

També han sigut reduïts els bugs i crashes de l'aplicació, explicat també a la secció *Estat final de l'aplicació*.

Redacció de la proposta d'article i article final realitzats.

6 REFACTOR

El gruix de refactorització del projecte ha sigut realitzat en base a la complexitat ciclomàtica dels mètodes. Així doncs, abans de comentar el refactor realitzat, faig una breu explicació sobre la Complexitat Ciclomàtica.

McCabe Cyclomatic Complexity és una mètrica de qualitat de software que quantifica la complexitat d'un programa. Es realitza mesurant el nombre de camins linearment independents a través del programa. Com més elevat és el nombre, més complex és el codi.

Gràcies a aquest nombre, els desenvolupadors saben que com més elevada és la complexitat ciclomàtica, més difícils d'entendre són i més probabilitat de que provoquin problemes hi ha. També es pot fer servir per a saber el nombre de test que s'haurien d'escriure per a executar tots els camins de l'aplicació.

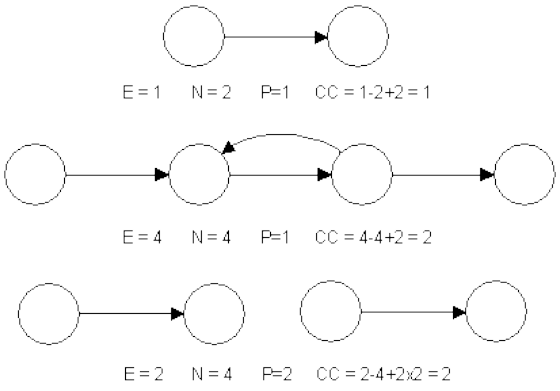
Per a calcular el nombre, en cas de no tenir cap plugin o eina que ens ho faci automàticament (com és el cas del meu projecte, fent servir el plugin d'Android Studio MetricsReloaded), es pot calcular de la següent forma:

$$CC = E - N + 2P$$

On:

CC = Complexitat Ciclomàtica
P = Nombre de parts no connectades en el flux del programa (subrutines o fils (threads))
E = Nombre de passos entre instruccions
N = Nombre de nodes

Exemples:



Refactor per a reduir la Complexitat Ciclomàtica:

He realitzat refactorització de gairebé tots els mètodes amb CC superior a 10. Hi ha molts motius pels que aquests mètodes eren de tant elevada complexitat i en els informes de seguiment hi ha alguns exemples, però a continuació faig un llistat d'alguns dels motius principals i quines solucions hi he aplicat.

- Mètodes amb switch-case sobre enums que tenen molts possibles valors
  - Afegint a cada enum un valor associat, vaig fer que, mitjançant reflexion, a partir del nom de la classe filla de Item, busqués al fitxer strings.xml del projecte, el text a retornar.
- Codi repetit
  - Hi havía molts mètodes que feien el mateix. Vaig crear submètodes als que cridar per a reaprofitar codi i reduïr longitud dels mètodes amb CC superior.
- Mètodes massa llargs
  - Vaig reduïr el volum de molts mètodes creant mètodes més petits que tinguessin funcionalitats atòmiques, en comptes de que un mètode sol fagi tota la lògica.
- Lògica dels botons massa elevada dins dels generadors de diàlegs de forma dinàmica
  - Per a separar la lògica, vaig crear una classe pare OnClickListener de la que heredéssin totes les classes que implementessin la lògica de prémer un botó. D'aquesta forma es té més localitzada la lògica dels botons.

7 EINES UTILITZADES

A continuació esmentaré les eines més rellevants que he fet servir tant per a la gestió com per el desenvolupament del projecte.

Android Studio

IDE (Integrated Development Environment) utilitzat per al desenvolupament d'aplicacions Android. Molt còmoda integració amb GIT, així com un ampli ventall de *plugins* i eines necessàries per el desenvolupament.

MetricsReloaded

Plugin instal·lat a l'Android Studio que m'ha permès analitzar, entre d'altres, la Complexitat Ciclomàtica del codi.

Bitbucket / SourceTree

Repositori de tipus GIT i eina per a gestionar-ho, que facilita la sincronització de fitxers amb el repositori remot.

Trello

Eina de gestió de tasques molt dinàmica que m'ha permès definir i gestionar la feina a fer a cada Sprint.

Google Drive / Google Docs

Disc dur compartit en el núvol en el que hem compartit tots els documents del projecte amb el tutor, permetent afegir comentaris i correccions, entre d'altres, a distancia.

Microsoft Office

Per a la creació i edició de l'article final, he fet servir el Microsoft Word i per a la conversió dels fitxers CSV que exportava amb el Plugin d'Android Studio Metrics Reloaded a fitxers en columnes he fet servir el Microsoft Excel.

GMAIL / Whatsapp / Google Calendar

Han sigut les eines utilitzades per a la comunicació, així com de la planificació de les reunions amb el tutor.

Play Google Developer Console

Plataforma per a gestionar les aplicacions publicades al Play Google, així com d'anàlisi d'estadístiques i valoracions sobre les mateixes, entre d'altres.

Fabric

Framework que he integrat a l'aplicació per a poder monitoritzar el nombre d'usuaris en temps real i els tipus i quantitat de crashes que l'aplicació té en producció.

8 ESTAT FINAL DE L'APLICACIÓ

A data 25/06/16 les dades de l'aplicació son les que es mostren a continuació.

A nivell de descàrregues de l'aplicació:

- Descàrregues actives: **15.200**
- Descàrregues totals: **85.250**

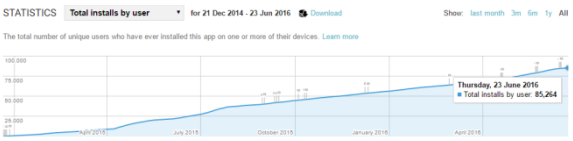
Diferència amb l'estat inicial:

- **+ 3.200** descàrregues actives
- **+ 21.250** descàrregues totals

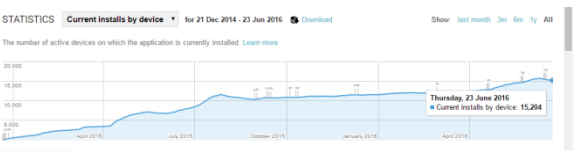
Descàrregues actives actuals

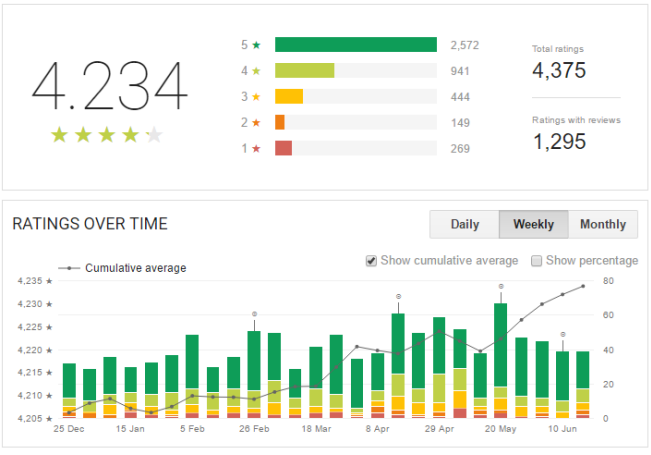
Descàrregues totals actuals

Valoració de l'aplicació:



- Puntuació mitja: **4'234** (sobre 5)
- Puntuacions totals: **4.375**
- Nombre de ressenyes: **1.295**

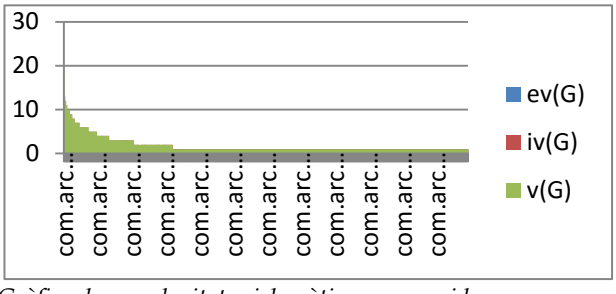




Captura de pantalla de l'estat actual al Play Google

Complexitat Ciclomàtica:

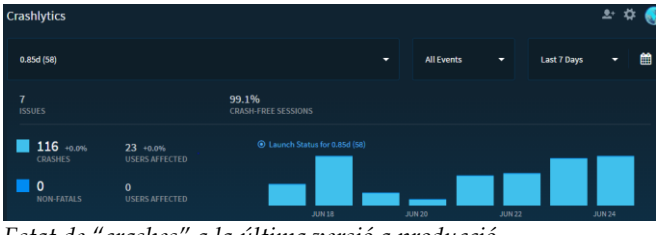
En el moment de redactar l'article final, la CC dels mètodes de l'aplicació és gairebé per a tots els mètodes de l'aplicació igual o menor a 10. Només queden tres mètodes amb CC de 20 i 19, 3 amb CC de 13, 6 amb CC de 12, 8 amb CC de 11 i la resta són 2393 mètodes amb CC igual o menor a 10.



Gràfica de complexitats ciclomàtiques resumida

Estat dels errors al Fabric:

A la versió actual (0.85d) hi ha un 99.1% de usuaris sense crashes, tot i que encara queda algun crash per corregir, les dades son prou bones. Això sí, l'ideal seria arribar al 100%, però per sobre d'un 99% ja és un molt bon resultat.



Estat de "crashes" a la última versió a producció

9 Conclusions

Hi ha una sèrie d'aprenentatges que he fet durant el projecte que estic ben segur em serviran en un futur (de fet ja m'estan servint a l'àmbit professional).

Primer de tot, adonar-me de que començar a treballar en un projecte sense una planificació prèvia, ni un disseny previ, et porta a tenir un projecte ple de defectes, gens escalable ni mantenible i que a mig-llarg termini haurà de ser refet o refactoritzat, amb el cost de temps (i econòmic en alguns casos) que això comporta.

Per altre banda, descobrir que hi ha moltes eines que et faciliten la feina a l'hora d'analitzar codi, així com per poder monitoritzar errors per a poder treballar en millorar la qualitat del producte. Gràcies al plugin de càlcul de complexitats ciclomàtiques, he pogut treballar sobre els punts més crítics de l'aplicació, obligant-me a refactoritzar el codi. Ara aquest codi és més mantenible, llegible i ampliable. Gràcies a la inclusió de Fabric, vaig poder veure quins eren tots els errors que els usuaris estaven tenint i vaig poder començar a corregir-los, millorant l'experiència dels usuaris al reduir el nombre de crashes que estaven tenint.

No només amb refactorització i correcció d'errors n'hi ha prou, hi ha molt més a fer per a millorar i mantenir una aplicació, però com a conclusió final dir que, si el codi està ben fet i tenim controlats i testejats tots els punts crítics de l'aplicació, de ben segur que aquesta tindrà més opcions de tenir èxit que una altre que no ho estigui.

10 Agraïments

En primer lloc, agrair la plena col·laboració, predisposició i ajuda del meu tutor Rodolfo Guichón durant el període de treball en el projecte. Les nostres converses i reunions han permès orientar i a trobar el camí a seguir.

Agrair també a la meva parella per el recolzament durant tota aquesta època en la que he estat treballant en el projecte.

Per últim, agrair a la Universitat Autònoma de Bellaterra per permetre'm escollir el meu propi projecte com a base sobre la que fer el TFG. He pogut fer el Treball de Fina de Grau alhora que millorava el meu propi projecte en el que porto treballant tot aquest temps.

11 Referències

[1] Google Inc., "Play Google Developer Console", Consola desenvolupador d'Android. https://play.google.com/apps/publish/. (URL link *2016)

[2] Twitter Inc., "Fabric", Framework per a detectar errors, així com analítiques d'ús de la app. https://www.fabric.io/. (URL link *2016)

[3] Trello Inc., "Trello", Eina Agile per a fer el seguiment de les tasques. https://www.trello.com/. (URL link *2016)

[4] Wikipedia.org, "Cyclomatic Complexity", Definició de què és la CC. https://en.wikipedia.org/wiki/Cyclomatic_complexity. (URL link *2016)

[5] Chambers.com.au, "Mc Cabe Cyclomatic Complexity", explicació i fórmules de la Complexitat Ciclomàtica. http://www.chambers.com.au/glossary/mc_cabe_cyclomatic_complexity.php (URL link *data desconeguda)

[6] Google Inc., "Developer Android", documentació, tutorials, exemples i tot el necessari per al desenvolupament per Android. http://developer.android.com/ (URL link *2016)

[7] SitePoint, "5 Best Android ORMs", una de les pàgines consultades en el moment del projecte en el que cercava un ORM per a transformar l'aplicació. http://www.sitepoint.com/5-best-android-orms/ (URL link *2014)

[8] Satya, developer, "Sugar ORM", un dels ORM que vaig intentar integrar a l'aplicació. Documentació i codi d'ús. http://satyan.github.io/sugar/index.html (URL link *data desconeguda)

[9] StackOverFlow, consultat en moltes ocasions per a trobar solucions a problemes, bugs i crashes, així com possibles refactor a realitzar per a millorar el codi. http://stackoverflow.com/ (URL link *2016)