Hit Code: Optimitzant el mastermind per a dispositius mòbils.

Jordi Vila Xicola

Resum— Hi ha diferents videojocs del mastermind arreu de les diferents plataformes, però no hi ha cap aplicació que permeti jugar a l'usuari contra la màquina i viceversa per a dispositius mòbils. En aquest projecte, s'analitzen diferents algorismes d'intel·ligència artificial per tal d'afrontar el repte d'elaborar un joc per a jugar al mastermind que ofereixi la possibilitat d'encertar el codi de l'usuari mitjançant algorismes, i que a l'hora sigui capaç de poder-se executar en un mòbil de gamma baixa amb sistema operatiu android per a fer-lo més accessible a la majoria dels usuaris.

En aquest TFG, es reflecteixen totes les etapes de desenvolupament d'un joc per a dispositius Android, des de la planificació del projecte fins a la finalització del mateix, passant per les etapes de decisió, de disseny i d'implementació.

Paraules clau— Videojocs, Mastermind, Intel·ligencia artificial, algorismes, dispositius mòbils, Android

Abstract—There are a lot of videogames for playing mastermind games throughout the diferent platforms, but there isn't any application that allows users the possibility to play against the computer and vice versa for mobile devices. This project analyze various artificial intelligence algorithms in order to face the challenge of develop a game to play mastermind offering the possibility of hitting the user code using algorithms, and at the same time, it was being able to run it on a low-end mobile with android operative system to make it more accessible to most users.

This TFG, reflect all stages of developing a game for Android devices, from project Planning to completion of the project, seeing the decission stages, design stages and finally the implementation process.

Index Terms— Videogames, Mastermind, Artificial intelligence, algorithms, Mobile devices, Android

1 INTRODUCCIÓ

EL Mastermind[1], és un joc de taula abstracte multi jugador que combina enginy i reflexió.

Els jocs abstractes són un estil de jocs de taula en els que la sort juga un paper mínim enfront de la lògica o l'estratègia.

Com la majoria de jocs abstractes, el funcionament del joc del Mastermind pot semblar complex, però un cop s'entén la dinàmica de la partida, es torna bastant fluid.

L'objectiu del joc resta en encertar el codi de l'adversari abans que aquest encerti el teu.

Al principi de la partida, cada jugador pensa el seu codi i el marca a una part del tauler que no pot veure l'oponent.*(Figura 1)* A partir d'aquí, un dels jugadors proposa un codi que creu que pot ser el del contrincant, i aquest s'encarrega de respondre-li quantes xifres del codi són correctes, és a dir, estan al codi a endevinar just a la mateixa posició, i quantes xifres hi ha regulars, que equivalen a aquelles xifres que apareixen al codi però no estan en la mateixa posició.

Una vegada s'ha anotat la resposta, es canvien els rols i el jugador que pregunta passa a ser l'altre.

Figura 1. Joc de taula del Mastermind

A part del joc de taula, hi ha variants del mateix joc per a jugar amb l'ordinador o amb el mòbil, però no he trobat cap aplicació en la qual el mòbil intenti encertar el teu codi, només pots intentar encertar tu el codi de la màquina.

Aquest fet que no hi hagi una aplicació per a mòbil que permeti jugar a l'usuari contra la màquina i en la qual ambdues parts intentin encertar el codi del contrincant,

ha sigut la motivació principal per a triar aquest projecte com a treball de final de grau.

En aquest projecte realitzaré una aplicació que mitjançant torns, l'usuari haurà d'encertar el codi de la màquina abans que aquesta li encerti el seu.

Per a realitzar-ho, he de tenir en compte que l'aplicació final està dirigida per a tecnologies mòbils, i per tant, l'algorisme a aplicar a l'hora de predir el codi de l'usuari ha de ser un algorisme eficient en quant a recursos consumits de l'aparell que l'executa i ha de ser eficient en quant a temps d'execució. Per tant, la primera part d'aquest treball la dedicaré a buscar aquest algorisme òptim per a que l'experiència d'usuari de l'aplicació final també sigui òptima, i posteriorment intentaré convertir aquest algorisme en una aplicació amb tots els elements necessaris per a poder-hi jugar.

Així doncs, la resta del document l'estructuraré de la següent manera. A continuació presentaré els objectius plantejats per al TFG, a la secció 2 exposo els requeriments i la metodologia genèrica. A la secció 3 mostro els anàlisis dels algorismes. A la secció 4 hi ha els detalls de la implementació en Android. A la secció 5 aparèixen els resultats obtinguts amb aquest treball, i finalment a la secció 6 acabo l'article amb unes conclusions i línies futures.

1.1 Objectius

Aquest TFG consisteix a realitzar una aplicació per a mòbils que et permeti jugar al mastermind contra la màquina en mòbils de gama baixa. Per tant, es trobarà l'algorisme que millor satisfaci les necessitats de recursos ja que l'entorn mòbil és una tecnologia que no disposa de grans recursos de còmput.

Així doncs, d'acord amb aquestes dues grans vessants del projecte, he establert els següents grans objectius:

Obj 1: ***Analitzar els diferents algorismes que resolen el problema.*** S'analitzaran els algorismes en termes de la seva capacitat de resolució en funció dels recursos consumits. Per tal de garantir una alta jugabilitat també es demanarà un compromís entre rapidesa per trobar el codi i complexitat del codi a encertar.

Obj 1.1: Elecció dels algorismes a tenir en compte per fer l'anàlisis. S'ha trobat un conjunt representatiu d'algorismes per a poder fer-ne el seu posterior anàlisis i trobar-ne l'òptim

Obj 1.2 Anàlisi de les propietats teòriques i implementacions d'un conjunt representatiu d'algorismes. S'ha analitzat tant les propietats i complexitat teòriques com la facilitat per migrar els algoritmes a plataforma mòbil.

Obj 1.3: Anàlisi de la jugabilitat que proporciona cada algorisme. S'implementarà un prototipus de cada algoritme en PC per tal de que usuaris potencials donin el seu feed-back de la jugabilitat que permet l'algoritme

Obj 2: ***Implementar la millor solució en Android.*** A partir dels prototipus obtinguts en anteriors fases, s'ha implementat una aplicació Android totalment funcional.

Obj 2.1: Dissenyar interfície gràfica intuïtiva i agradable per a oferir una bona experiència de joc a l'usuari. S'ha utilitzat tècniques de disseny de software per a fer l'aplicació més amigable.

Obj 2.2: Migrar el prototipus de PC a Android. S'ha combinat la part funcional del prototipus de PC amb els dissenys realitzats per a mòbil, incloent en aquesta etapa tests de rendiment de l'aplicació final.

A partir de la definició d'aquests objectius, les diferents seccions d'aquest treball s'utilitzaràn per aprofundir en les tasques i la planificació per a poder assolir-los.

2 Requeriments i Metodología Genèrica del Sistema

El fet que el TFG consisteixi en un joc per a aparells mòbils que es pugui executar en dispositius de gama baixa, genera que apareguin uns requisits que ha de complir l'aplicació per a que pugui utilizar-se correctament. Aquests requisits es mostren a continuació.

A més a més, en aquesta secció també s'exposa la metodología genérica utilitzada juntament amb els motius de la seva elecció.

2.1 Requeriments

Els requeriments definits per al producte final d'aquest treball sorgeixen de l'ambició d' una aplicació que ofereixi una bona jugabilitat i que alhora tingui en compte l'eficiència.

Req 1: L'algorisme escollit ha de permetre la seva implementació en Java per a una futura implementació més senzilla en Android, ja que l'eina Android Studio ofereix més facilitats per a utilitzar funcions Java.

Req 2: L'algorisme triat ha de minimitzar els recursos de memòria i de CPU del dispositiu mòbil el màxim possible.

Req 3: El joc ha de permetre que l'usuari intenti encertar el codi de la màquina i que alhora la màquina intenti encertar el codi de l'usuari mitjançant torns.

Req 4: Les partides han de ser parametritzables en quant al nombre de xifres dels codis i si les xifres es poden repetir.

Req 5: L'aplicació ha de ser entenedora per a poder ser jugada per gent de totes les edats.

Req 6: El joc ha d'oferir una bona jugabilitat.

Req 7: L'aplicació ha de poder-se executar en mòbils de gamma baixa, per tant no ha de consumir gaires recursos.

2.2 Metodología Genèrica

Per a dur a terme el TFG, s'ha seguit la metodologia de la cadena crítica [2].

Aquesta metodologia s'utilitza normalment per a gestionar projectes en els que els seus mòduls són dependents entre si.

El mecanisme de la cadena crítica, consisteix en establir un flux de projecte, en el que es relacionen les diferents tasques d'acord a les seves dependències amb les altres tasques.

Quan una tasca depèn d'una altra, no pot començar fins que la tasca prèvia hagi finalitzat. Per a representar-ho, s'elabora un diagrama en el que s'estableixen les tasques que formen part de la planificació del projecte i es

senyalen les seves dependències mitjançant fletxes. Si una tasca és apuntada per una altra, significa que no pot iniciar-se fins que hagi acabat l'anterior.

En el cas particular d'aquest treball, les diferents tasques que intervenen a la cadena crítica estàn directament relacionades amb el compliment dels diferents objectius plantejats al inici del projecte, i a continuació es mostra l'esquema que s'ha seguit durant el treball. (*Figura 2*)

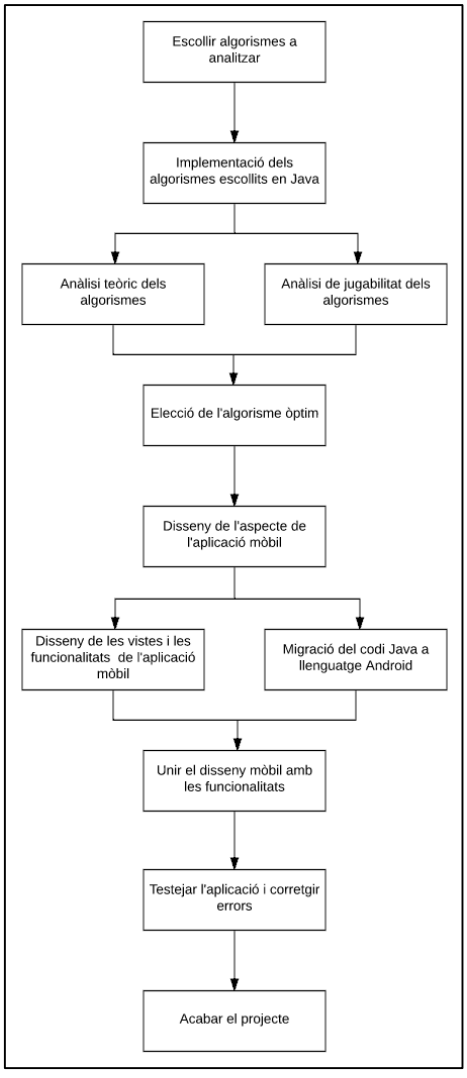


Figura 2. Diagrama de la cadena crítica del projecte

3 ANÀLISI DELS ALGORISMES

Per a escollir l'algorisme òptim per a utilitzar en l'aplicació final del meu TFG, al principi es van escollir tres algorismes a analitzar.

Després d'informar-me dels diferents tipus d'algorismes que podria utilitzar per a realitzar el projecte, vaig escollir que els algorismes de cerca eren els que s'adequaven més per al joc del Mastermind.

Aquesta tipologia d'algorismes, es basen en la cerca d'un element dins un conjunt de molts elements diferents.

Per a fer aquestes búsquedes, s'apliquen restriccions per anar eliminant possibles candidats fins que s'arriba a la situació en que només queda l'element que es vol.

En el cas del meu treball, l'element a localitzar és un codi numèric dins un conjunt de possibles codis. Com s'ha esmentat a la introducció, el joc consisteix en després de provar un codi, rebre una resposta sobre quantes xifres hi ha correctes i quantes regulars. Doncs a partir d'aquesta resposta, s'eliminen els codis que no cumpleixen aquestes restriccions fins arribar a un sol codi que és el que es busca.

Dins dels algorismes de cerca, ens trobem que n'hi ha molts de diferents, però com es va establir als requeriments, l'algorisme havia d'oferir una bona jugabilitat i alhora havia d'utilitzar els mínims recursos possibles.

Aquest fet va ajudar a triar els tres algorismes amb els que s'ha desenvolupat els anàlisis, escollint l'algorisme de Knuth, l'algorisme simple i l'algorisme de la mida esperada.

Els anàlisis realitzats es divideixen en dues etapes. La primera fase consisteix en una exploració teòrica de l'algorisme juntament amb un anàlisis de la seva complexitat i els recursos que consumeix.

La segona part, consta d'avaluacions amb usuaris potencials per a analitzar la jugabilitat que ofereix.

Aquestes dues etapes de l'anàlisis per a escollir l'algorisme òptim estan explicades en els següents apartats.

3.1 Anàlisi teòric i de complexitat

Els tres algorismes escollits per analitzar, tenen la mateixa base de funcionament. En els tres casos, al inici de l'execució s'elabora una llista de possibles codis en base als paràmetres de la partida.

Per exemple, si una partida es juga amb 4 xifres i les xifres es poden repetir, com que cada xifra pot tenir 10 valors (0-9), elabora un llistat de 100.000 possibles codis (10^4) que van des del 0000 fins al 9999.

Una vegada establerta aquesta llista, els algorismes van eliminant possibles codis a partir de les restriccions que cadascun d'ells creu prioritàries.

A continuació, s'explica el funcionament per a cadascun dels tres algorismes.

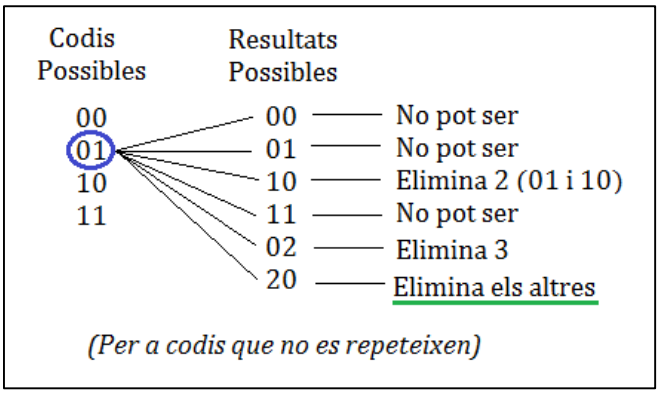
Algorisme de Knuth [3]

El primer dels algorismes, que va ser proposat per Donald Knuth l'any 1977, funciona de la següent manera.

Al principi de tot, es genera un conjunt de possibles codis que està format per totes les possibles combinacions d'acord als paràmetres de la partida.

A partir d'aquest moment, l'algorisme pondera cadascuna de les combinacions d'acord al nombre de combinacions que s'eliminen si s'escull aquell codi, i posteriorment, escull el codi mitjançant el qual redueix al màxim el volum de possibles codis restants.

És a dir, per cada codi del llistat de possibles codis que



queden, analitza quants codis s'eliminarien per cadascun dels resultats possibles restants, i escull aquella predicció que si es dóna el resultat òptim redueix més el nombre de codis que queden disponibles dins la llista restant de codis possibles.

Figura 3. Esquema funcionament algorisme de Knuth

Com es pot veure a la figura 3, per cadascun dels codis possibles, s'analitza quins possibles resultats pot donar. En aquest exemple, els codis només són de dues xifres i els valors només poden ser 0 o 1 per a explicar el funcionament de manera més senzilla. A la columna de resultats possibles, la primera xifra del resultat correspon a les xifres correctes i el segon a les regulars. De tots aquests resultats, s'escull el que elimina el màxim de codis possibles, que en aquest cas és el resultat 2-0 que equival a totes correctes, i a aquest codi, se li assigna una ponderació de 3, que són els codis restants que queden a la llista de codis possibles.

Aquest procés es realitza per cadascun dels codis, i una vegada escollit el que elimina més possibilitats, també s'actualitza la nova llista de codis possibles, per a poder tornar a realitzar una altra iteració del procés seguint els mateixos passos.

Algorisme Simple [4]

El segon algorisme, va ser proposat per Ehud Shapiro.

El funcionament d'aquest algorisme, és bastant similar al primer, també consisteix en realitzar una primera llista de codis possibles que es va reduïnt fins a trobar el codi resultant.

La diferència respecte l'anterior algorisme, es troba a l'hora de ponderar els diferents codis. En aquest cas, simplement es limita a ordenar els codis que queden disponibles de forma ascendent, i escull el primer d'aquesta llista.

Per a fer-ho, es limita a ponderar cadascun dels codis amb el valor 0, i degut a que no hi ha cap que superi a l'anterior, es queda amb el primer de la llista durant totes les iteracions que realitza l'algorisme.

Igualment, quan l'usuari respon al codi que proposa la màquina, s'apliquen restriccions depenent del resultat proposat de la mateixa manera que ho fa l'algorisme de Knuth. És a dir, en el mateix cas exposat en la figura 3, en el supòsit que proposés el codi 01, i que la resposta de l'usuari fós que hi ha una correcta i cap regular, eliminaria els codis 01 i 10, reduïnt els codis possibles.

Al següent torn, tornaría a proposar el primer que hi hagués a aquesta nova llista actualitzada de codis possibles, que en el cas de l'exemple, es correspondria al 00.

Algorisme de la mida esperada [5]

Aquest últim algorisme, també segueix els mateixos mecanismes que els dos anteriors, però la diferència es troba al sistema de ponderacions dels codis.

En aquest cas, l'algorisme de la mida esperada analitza per cadascun dels codis possibles, tots els resultat possibles, i es sumen els valors per a aconseguir la ponderació d'aquell codi.

Per a fer aquestes previsions, el procés consisteix en multiplicar el nombre de codis disponibles que queden disponibles després de realitzar aquell possible moviment per ell mateix, és a dir calcular-ne el quadrat, i una vegada calculat aquest valor, dividir-ho pel nombre actual de codis possibles.

Una vegada obtingut aquests valors, es sumen tots els corresponents a cada codi, i s'obté la ponderació per al codi corresponent.

Tornant al exemple utilitzat per explicar els anteriors algorismes, a la figura 4 s'explica el procés per a aquest algorisme.

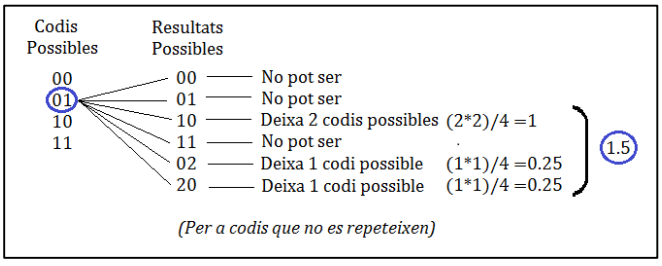


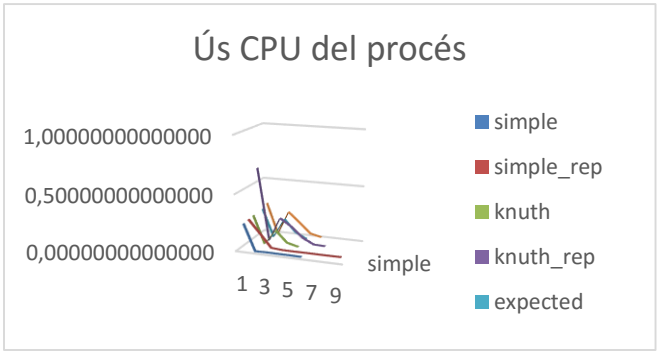
Figura 4. Esquema funcionament algorisme de la mida esperada

Així doncs, seguint l'exemple anterior, per al codi 01, se li assignaria una ponderació de 1.5, i després d'analitzar tots els codis, s'escolliria aquell amb major ponderació.

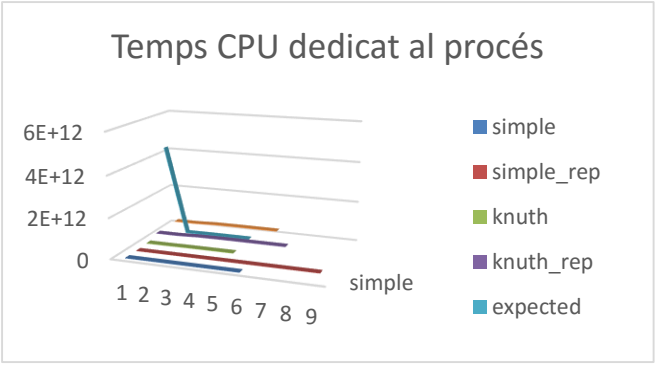
Una vegada exposats els funcionaments dels algorismes, i implementats els primers prototipus en Java, es van realitzar els primers experiments per a analitzar el rendiment de cadascun d'ells i escollir-ne l'òptim.

Per a fer-ho, vaig utilitzar diferents funcions que ofereix Java que analitzen l'estat de la CPU en el moment de la crida. Aquesta funció s'ha cridat a cada torn per analitzar com a mida que es reduïa el nombre de codis possibles, també disminuïa els recursos consumits, alhora que s'analitzava el temps que tardava en realitzar cada una de les prediccions.

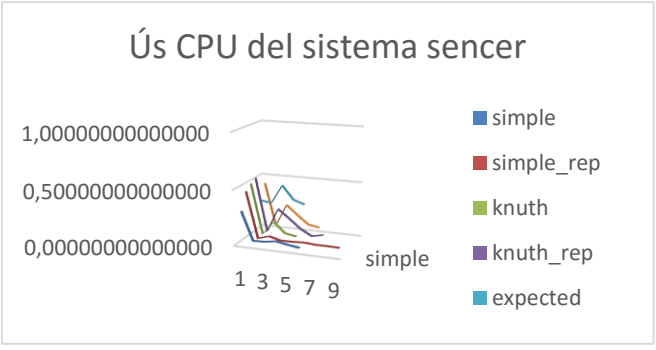
Els resultats obtinguts d'aquest primer experiment van ser els següents.



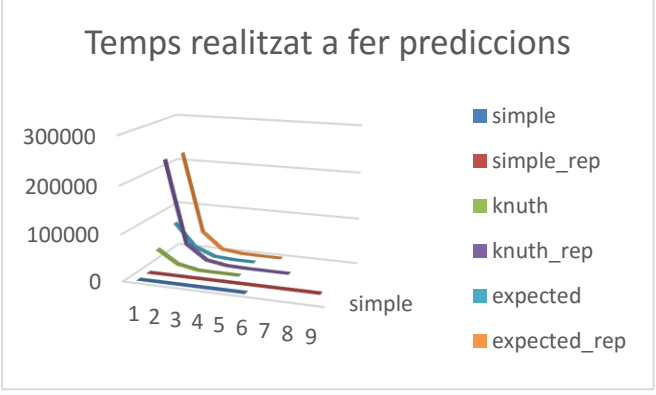
Gràfic 1. Evolució del ús de la CPU per al procés JVM[6] durant el transcurs dels torns.



Gràfic 2. Evolució del temps de la CPU dedicat al procés JVM[6] durant el transcurs dels torns.



Gràfic 3. Evolució del ús de la CPU per al sistema sencer



Gràfic 4. Evolució del temps realitzat a proposar la següent preducció durant el transcurs dels torns.

A partir de les dades obtingudes de les gràfiques extretes, era fàcil d'observar quin d'ells era el més eficient i quin el que menys.

En tots els casos, l'algorisme de la mida esperada, era el que consumia més recursos de còmput per part de la CPU i a l'hora era el que requería més temps d'execució, amb una gran diferència respecte als altres.

Per tant, des d'un inici aquest algorisme va quedar descartat.

Pel que fa als altres dos algorismes, els valors dels càlculs eren una mica més semblants, tot i que també es veia molt clar quin dels dos necessitava menys recursos.

Tant en el temps d'execució com en l'ús de CPU que consumien, l'algorisme simple estava en els dos casos per sota del algorisme de Knuth, oferint una gran millora de recursos respecte els altres.

Això és degut, a que els altres dos algorismes, cada vegada que volen trobar el següent moviment realitzen una cerca intensiva per tots els codis que queden possibles, fent càlculs probabilístics sobre cada un d'aquests codis, mentre que l'algorisme simple, tan sols els ordena i escull el primer sense haver de calcular la ponderació de cada codi ja que les situa a 0 totes.

Per tant, és evident que dels tres algorismes analitzats, el que consumeix menys recursos i que per tant és més eficient parlant en termes de eficiència de còmput, és l'algorisme simple.

3.2 Anàlisi de la jugabilitat

Una vegada implementats els diferents algorismes en Java, i havent-los analitzat respecte la seva complexitat i el seu consum de recursos, vaig observar que els resultats obtinguts amb els anàlisis, en alguns casos no es corresponien amb el grau de jugabilitat que podien oferir.

Per aquest motiu, es va iniciar una segona etapa d'anàlisis en la qual es va analitzar la jugabilitat que oferia cadascun.

Aquest experiment ha consistit en la avaluació de la jugabilitat de cadascun dels algorismes per part d'usuaris potencials de diferents edats.

Es van realitzar les proves amb una població de 10 usuaris amb edats compreses entre 15 i 50 anys.

Per a fer-ho, es va executar cadascun dels 3 algorismes fins a tres vegades amb cada usuari i a més a més de la observació, ja que es feien les proves en persona, al final els usuaris havien d'ordenar els tres algorismes d'acord amb la el grau de jugabilitat que oferien.

A partir d'aquest experiment, les conclusions sobre la jugabilitat que oferia cada algorisme van ser les següents.

Pel que fa al algorisme de Knuth, els usuaris consideraven que durant els primers torns el temps que l'algorisme dedicava a cercar la següent predicció era massa elevada, encara que a mesura que avançaven els torns el temps ja era raonable, i pel que feia al nivell de dificultat, el trobaven correcte, ja que costava guanyar però era possible.

Per altra banda, l'algorisme simple, no es veia penalitzat en el temps dedicat a cercar la següent predicció ja que des dels primers torns el temps no superava els 5 segons. Però pel que feia al nivell de dificultat, en algunes

ocasions resultava massa fàcil.

Finalment, i en referència al algorisme de la mida esperada, els resultats obtinguts no van ser positius ni pel temps d'execució ni pel nivell de dificultat. La totalitat dels usuaris van coincidir que el temps que dedicava a pensar la següent predicció era massa elevat, i que si teníem en compte aquest temps, els hi era indiferent el nivell de dificultat ja que no creien acceptable que tardés tant.

3.3 Elecció de l'algorisme òptim

Una vegada realitzat els diferents experiments, els resultats obtinguts en ambdós eren bastant coherents entre ells.

Els dos coincidien que l'algorisme de la mida esperada tot i trobar bastant ràpid els codis, el seu temps d'execució entre torns era inadmissibles, per tant ràpidament va quedar descartat.

Pel que feia als altres dos algorismes, els dos experiments mostraven el mateix. Els temps d'execució es decantaven al algorisme simple, però la necessitat de torns per encertar el codi afavorien al algorisme de Knuth.

Així doncs, i a partir d'aquestes conclusions, la meva decisió va ser canviar la planificació inicial d'implementar un sol algorisme per a introduir-hi dos algorismes a l'aplicació.

Això es podria aconseguir introduint diferents nivells de dificultat a l'aplicació i depenent del nivell que l'usuari escollís, s'utilitzaria un algorisme o un altre.

Per al nivell normal, s'utilitzaría l'algorisme simple, ja que el seu temps d'execució és òptim respecte als altres, però el nivell de dificultat no és sempre alt, mentre que pel nivell difícil es faria ús del algorisme de Knuth, ja que tot i que als primers torns el seu temps era elevat, si es feia balanç respecte al nivell de dificultat, es podia acceptar la seva tardança.

4 Implementació en Android

Per a implementar l'algorisme escollit en Android, vaig decidir dividir aquesta etapa en dues fases.

En la primera d'elles, s'ha dissenyat com havia de ser l'aplicació visualment, i quines funcionalitats havia de complir, mentre que la segona etapa, ha consistit en migrar el codi en Java a Android i afegir-hi els detalls visuals i funcionals.

4.1 Disseny de l'aplicació

Una vegada establerts els requeriments al inici del projecte, en aquesta etapa s'han elaborat diferents iteracions del disseny de l'aplicació complint els requisits plantejats.

El primer pas a l'hora de dissenyar com havia de ser el joc, va ser enumerar quines eren les funcionalitats que el joc havia de permetre. A partir d'aquest estudi, s'ha elaborat una llista d'aquestes fucions que esmento a continuació.

1. L'aplicació ha d'oferir al usuari, una petita guia en format d'ajuda per a entendre el funcionament del joc.
2. L'usuari ha de poder escollir els paràmetres de la partida que són, el nombre de xifres dels codis i si es poden repetir xifres dins el mateix codi.
3. La partida ha de desenvolupar-se mitjançant torns alternats entre l'usuari i la màquina.
4. En el torn que l'usuari fa una predicció, la màquina ha de mostrar la seva resposta de xifres correctes i regulars.
5. En el torn en que la màquina fa la seva predicció, l'usuari ha de poder introduir la seva resposta.
6. Una vegada introduït el codi o la resposta, no es pot desfer el moviment.
7. Quan un dels dos jugadors encerta el codi de l'altre, la partida es dona per finalitzada mostrant un missatge final.
8. Al acabar la partida, es torna a la pantalla d'inici des d'on es pot tornar a jugar una altra partida.

D'aquestes vuit funcionalitats bàsiques que havia de satisfer l'aplicació se'n va fer un primer esboç a mà en el que es presentaven unes primeres pantalles relacionades entre si per a poder estructurar les activitats en Android i saber quantes pantalles hi haurien i com es relacionaven entre si. (*Figura 5*)

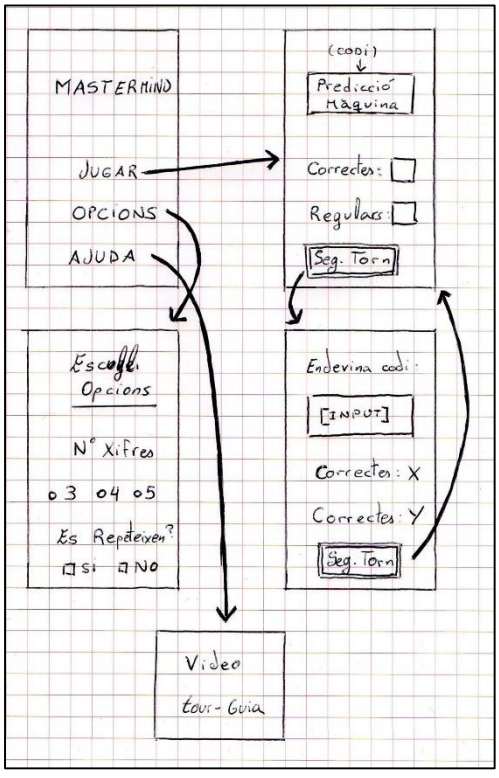


Figura 5. Primers mock-ups[7] de l'aplicació

4.2 Migració de Java a Android

Amb un primer esquema general de l'aplicació ja establert, el següent pas va consistir en passar els algorismes escollits a Android, i unir-ho amb les pantalles que havia dissenyat prèviament.

Per a fer-ho, he fet servir l'eina Android Studio [8], un entorn de desenvolupament integrat per a Android que s'ofereix als usuaris de forma gratuïta.

L'elecció d'aquesta eina es deu al fet que t'ofereix renderitzar l'aplicació que estas treballant a temps real per a veure els avenços, té una interfície d'usuari que permet una bona interacció amb l'eina, i envers als errors, ajuda a trobar-los i proposa possibles solucions. Així doncs és una bona eina per a desenvolupadors que no estan familiaritzats en aquest entorn de programació.

L'estructura del meu projecte, igual que qualsevol projecte Android, consta de 3 elements principals diferenciats. Per una banda hi ha el manifest, que equival al arxiu de configuracions de l'aplicació des d'on s'estableixen paràmetres com el nom de l'aplicació, la icona,... i on es llisten totes les activitats que intervenen a l'aplicació.

Per altra banda, hi ha les activitats, que són les pantalles que es mostren al usuari. Aquestes activitats es defineixen en arxius .xml des d'on es poden realitzar crides a altres funcions.

Finalment també hi ha les classes Java, des de les que es defineixen les funcions i es realitza el còmput de l'aplicació.

A partir dels dissenys elaborats, el projecte s'ha separat el 8 activitats diferents.

La primera d'elles s'utilitza com a pantalla d'inici, des d'on es pot començar una partida o veure les instruccions.

Tres activitats estàn destinades a mostrar una petita guia de com es desenvolupa una partida en el joc, reproduint el cicle d'una partida normal.

També hi ha una altra activitat dedicada a escollir els paràmetres de la partida a jugar.

Dues activitats més, s'utilitzen per a jugar els diferents torns. Una per al torn de l'usuari i l'altra pel torn de la màquina.

I finalment les dues activitats restants es fan servir, una per a mostrar un missatge d'error en els casos que és necessari i una per a mostrar un missatge de victoria, que pot ser per part de la màquina o de l'usuari.

A continuació es mostra un esquema on es veu l'estructura del projecte Android. (*Figura 6*)

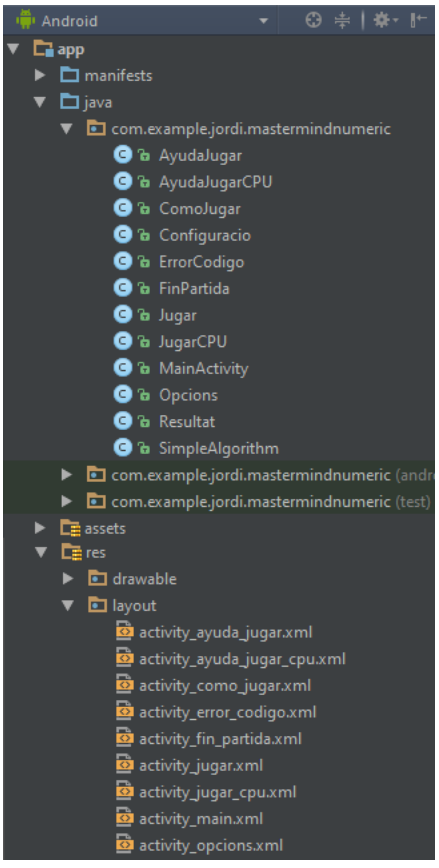


Figura 6. Classes Java i vistes xml del projecte.

Per a introduir les funcionalitats a aquestes pantalles, vaig agafar tots els arxius utilitzats en el prototipus en Java i els vaig importar dins el projecte.

A partir d'aquí, vaig anar traient les funcions i classes que ja no eren necessàries perquè ara ja tenia les vistes, i vaig anar adeqüant les funcions que vaig deixar per a que es poguessin utilitzar des de les activitats.

En aquesta primera implementació de l'aplicació en Android, vaig intentar implementar la versió del joc amb 2 algorismes diferents, decisió sorgida dels experiments d'anàlisis dels algorismes.

Al fer-ho, però, vaig observar que l'algorisme de Knuth, que en els primers torns en l'ordinador ja tardava un temps considerable, al mòbil també ho feia però durant aquest temps la pantalla se'm mostrava de color negre tot i executar-ho des d'un dispositiu capaç d'executar aplicacions que a priori necessiten més recursos.

Tot i intentar solucionar aquest fet, finalment no va ser possible, i per aquest motiu vaig tornar a la planificació inicial en la que només implementava un sol algorisme, en aquest cas el simple.

A causa d'aquest contratemps, es va tornar a revisar les funcions i les classes que es necessitaven i es va tornar a eliminar les que ja no eren necessàries.

Així doncs, finalment el projecte ha quedat de la següent manera.

L'estructura consta de 12 classes programades en Java i 9 layouts que equivalen a les diferents pantalles que disposa el joc.

Dins de les classes, hi ha unes classes principals que s'encarreguen de gestionar tots els càlculs i prediccions sobre els codis proposats, i hi ha un altre tipus de classes que s'encarreguen de comunicar la part funcional amb el disseny realitzat a les vistes.

Entre totes les classes i vistes, s'hi poden diferenciar tres grups clars d'acord a la funció per a que s'utilitza cadascuna d'elles.

El primer d'aquest grup, està format per les classes MainActivity, Opcions, FinPartida i ErrorCodigo, aquestes classes juntament amb les seves respectives activitats, són les encarregades de conduir al usuari durant el transcurs del joc. La primera d'elles s'utilitza com a menú principal, des d'on es pot accedir als diferents apartats de l'aplicació. La segona s'utilitza com a interfície des d'on l'usuari pot escollir els paràmetres de la partida, i les dues restants es fan servir per a mostrar missatges al usuari o bé de finalització de la partida o bé un missatge d'error.

Per altra banda, en el segon grup trobem les tres primeres classes de la figura 4, AyudaJugar, AyudaJugarCPU i ComoJugar, que juntament amb les seves activitats corresponents, s'encarreguen de guiar al jugador per una petita guia mitjançant la qual s'explica els passos a seguir en les diferents pantalles que existeixen.

I ja per acabar, les classes restants s'utilitzen per a realitzar tots els càlculs i els procediments d'inteligència artificial per a poder predir el codi de l'usuari, i permet el sistema de torns al jugador mitjançant l'alternança de l'activitat Jugar amb la de JugarCPU. A més a més de l'estructura explicada, també s'ha utilitzat patrons de disseny[9], per a facilitar la experiència d'usuari.

En concret s'han aplicat dos patrons clarament detectables com són el patró del Tour[10] i la metàfora[11].

El patró del tour, consisteix en l'oferiment d'una guia a l'usuari per a veure el flux de l'aplicació i poder fer-la servir, mentre que la metàfora consisteix en a partir del disseny, simular un aspecte que l'usuari relacioni amb el seu dia a dia.

En aquest cas, s'ha utilitzat un fons de llibreta i la tipologia de la lletra simula una lletra escrita a mà, perquè és un joc que es pot jugar amb una llibreta i intenta fer sentir al usuari com si ho estigués fent.

5 RESULTATS

L'apartat de resultats d'aquest treball té dues vessants. Per una banda, hi ha l'aplicació final amb totes les seves funcionalitats, i per un altre costat hi ha el resultat obtingut de la prova de l'aplicació final amb usuaris potencials.

En relació a l'aplicació final, s'ha aconseguit realitzar un joc funcional al 100%, en el qual es poden parametritzar les partides i es pot jugar contra la màquina.

A continuació es mostren unes captures reals del joc en el que es poden diferenciar algunes de les pantalles de les que disposa l'aplicació, aquestes captures però són tan sols una referència orientativa ja que hi ha altres que no apareixen i que són necessàries per al correcte transcurs de les partides..

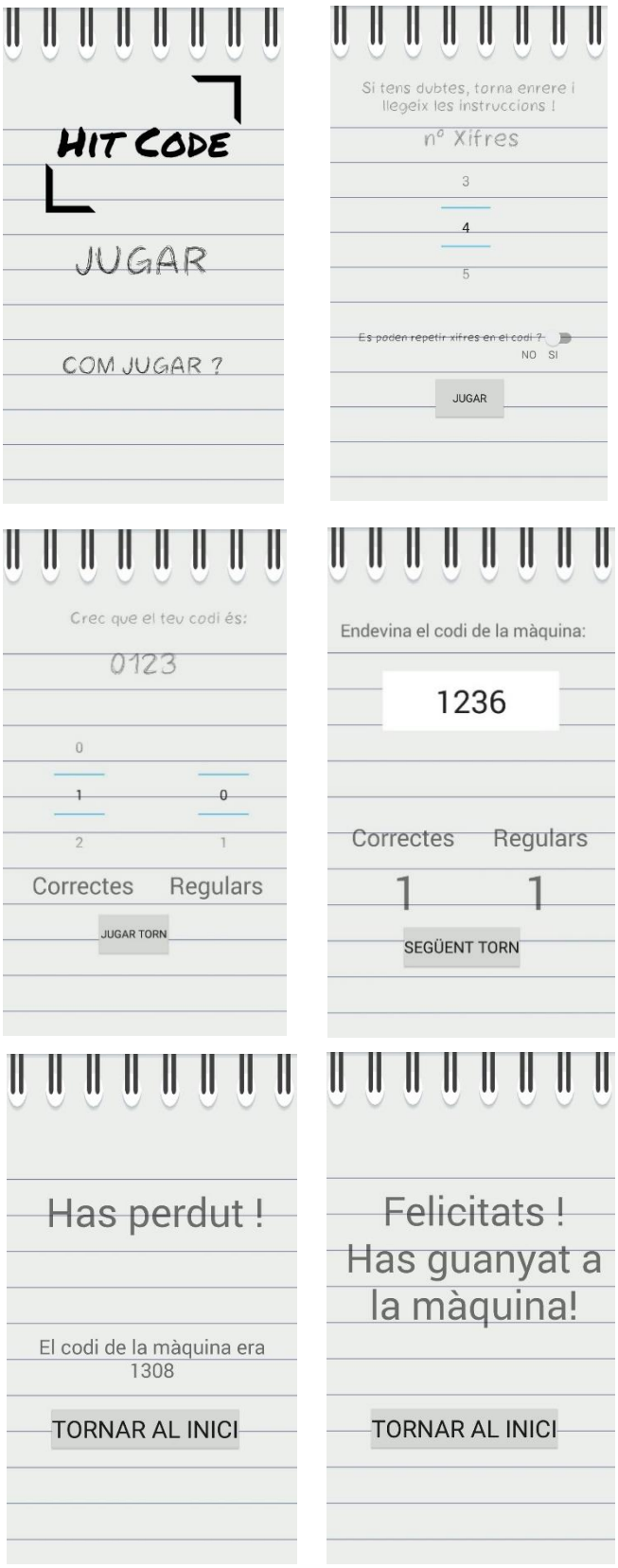


Figura 7. Captures de l'aplicació final

A la figura 7, es pot apreciar en ordre d'inserció, les següents pantalles.

La primera d'elles, és la pantalla d'inici que apareix tan sols obrir el joc, des de la que pots anar a una petita guia

de jugar o pots anar a jugar.

Si esculls la opció de jugar, et porta a la segona imatge, des de la que el jugador escull els paràmetres de la partida a jugar.

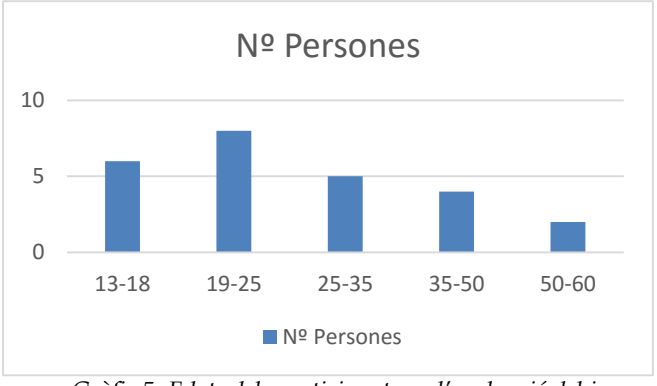
A continuació, es van alternant la tercera i la quarta imatge entre elles a mesura que van succeint els diferents torns entre l'usuari i la màquina.

Finalment, les dues últimes imatges, corresponen al missatge final de partida que pot ser en el cas en que l'usuari encerti el codi de la màquina, o en canvi que la màquina encerti el codi de l'usuari com es pot veure en la última imatge.

La segona part dels resultats, ha estat obtinguda a partir del testing del joc realitzat amb jugadors potencials que han valorat els diferents aspectes del joc així com proposat possibles millores.

Aquesta avaluació, s'ha realitzat en una població de 25 individus, d'edats variades entre els 13 i els 60 anys, amb varietat de sexes i diferents personalitats.

Per a ser més concrets, a continuació es mostra un gràfic amb les edats compreses dels avaluadors.



Gràfic 5. Edats dels participants en l'avaluació del joc

Com es pot observar al gràfic 5, la major part dels usuaris als que he tingut accés són usuaris joves, però tot i això, s'ha examinat a usuaris adults per a veure les seves reaccions.

Pel que fa als sexes, hi ha hagut 17 participants masculins i 8 femenins.

S'ha intentat trobar varietat tant d'edats com de sexes, pel fet que cadascun dels diferents perfils que he analitzat, busca coses diferents en un joc, i es fixa en unes parts més específiques.

Per a fer-ho, s'ha realitzat una demo de l'aplicació sentats a la mateixa taula per a veure quina era cadascuna de les reaccions que es produïen i veure quines eren les tendències de la majoria en quant a fluxe d'activitat.

A més a més, al final de l'experiment, els usuaris van respondre una sèrie de preguntes per a poder treure'n conclusions d'aquest TFG.

Les preguntes que es van realitzar, anaven destinades per una banda a obtenir informació de l'usuari en quant a l'edat, tecnologies que utilitza habitualment i jocs semblants als que hagi jugat, i per un altre costat, també es preguntava sobre si el joc els hi havia sigut amè, si tornarien a jugar a aquest joc en aquell moment, si ho farien més endavant, i finalment si pagaríen per a poder jugar al joc o per a poder gaudir de característiques extres.

A partir dels resultats obtinguts, la major part dels usuaris van considerar que és un joc difícil d'entendre el seu procediment, però que una vegada entesa la dinàmica, es fa interessant i 22 dels 25 usuaris avaluats van concloure que segurament hi tornarien a jugar.

En canvi, van considerar que és un joc pel que en un principi no hi pagarien, ja que no consideraven que pogués haver-hi cap element del joc que fós mereixedor de ser comprat amb diners.

Per tant, com a resultats finals del TFG, a part de l'aplicació final completament funcional, també hi ha les valoracions positives dels usuaris que asseguren que és un joc que entretén i et fa pensar.

6 Conclusions i Línies Futures

Per a concloure aquest treball, cal dir que s'ha assolit els objectius inicials, però hi ha hagut alguns detalls que si s'haguéssin tingut en compte des d'un inici i s'hagués tingut constància d'ells, segurament el temps del projecte s'hagués pogut reduir.

Per una banda, s'ha de tenir en compte que els algorismes a analitzar, no tan sols s'han de fer els estudis sobre el seu rendiment computacional sinó que també hi intervé el factor de la jugabilitat que ofereixen al usuari a l'hora d'escollir-ne l'òptim.

Per tant, si tan sols es té en compte els valors estadístics de fer anàlisis de rendiment, pot ser que la opció escollida no sigui la millor.

Així doncs, hem d'obtenir el compromís entre eficiència i diversió del jugador a l'hora de jugar al nostre joc.

A més a més, a l'hora de fer l'anàlisis de recursos consumits, si s'utilitza un ordinador quan el producte final és un dispositiu mòbil, hi ha la opció que en la posterior implementació els dispositius no suportin aquell procés. Això s'ha demostrat en el meu treball, ja que tot i que al ordinador els dos algorismes triats anaven bé, al moment d'implementar-ho a Android, la majoria de dispositius no ho executaven de forma ràpida. És per això que cal tenir-ho en compte per evitar temps d'implementació en un altre llenguatge si finalment no es farà servir.

Per una altra banda, també cal tenir en compte que per a fer una bona implementació del joc, cal que el disseny s'adapti a qualsevol dispositiu, fent que s'adapti a les diferents mides de pantalles que hi ha actualment al mercat. Per a fer-ho, l'eina Android Studio, t'ofereix la opció de fer les diferents pantalles adaptables. Això s'aconsegueix introduint les distàncies i les mides de forma relativa respecte al espai total de la pantalla en comptes de posant valors absoluts.

Ja per acabar, un altre factor important a treballar correctament és el nivell de dificultat que ofereix la màquina. A partir dels experiments amb els usuaris i del feed-back proporcionat, he pogut observar que si l'algorisme a utilitzar fa que sigui massa difícill guanyar a la màquina en les partides, l'usuari acaba descontent i segurament no voldrà jugar-hi. Per tant, s'ha de trobar la manera per a

que el nivell de dificultat sigui suficient per a no resultar massa fàcil però que a l'hora no sigui impossible.

Per a saber si s'ha trobat aquest equilibri en la dificultat del joc, recomano sobretot realitzar tests de satisfacció dels jugadors, és a dir, fer jugar a usuaris potencials, i demanar-los si creuen que el joc està bé, els possibles canvis a implementar, i sobretot si hi tornarien a jugar. Si s'aconsegueix que a aquesta última pregunta tothom contesti que si, jo prendria l'aplicació com a enllestida.

AGRAÏMENTS

Agraïr a la tutora d'aquest treball Debora Gil, per l'assasorament proporcionat, al professor Enric Martí per als consells donats per a poder analitzar bé el joc, i finalment als meus amics i coneguts que han ofert la seva opinió per a realitzar un millor estudi del joc.

BIBLIOGRAFIA

[1] Mastermind. https://en.wikipedia.org/wiki/Mastermind_(board_game) [Consulta: 10/02/2016]
[2] Metodología de la Cadena Crítica. Cadena Crítica, Eliyahu M. Goldratt, Ediciones Díaz de Santos, 2001.
[3] Algorisme de Knuth. D. E. Knuth, (1976-77). The Computers as a Master Mind, Journal of Recreational Mathematics 9.
[4] Algorisme Simple. Sterling, L. and Shapiro, E. (1994) The Art of Prolog: Advanced Programming techniques
[5] Algorisme de la mida esperada. Justin Dowell, Defeating Mastermind (Paper)
[6] JVM. https://en.wikipedia.org/wiki/Java_virtual_machine [Consulta: 03/04/2016]
[7] Mock-ups. https://es.wikipedia.org/wiki/Mockup [Consulta 23/04/2016]
[8] Android Studio. Eina de desenvolupament Android. https://developer.android.com/studio/intro/ [Consulta 23/04/2016]
[9] Patrons de disseny mòbils. Android Design Patterns: Interaction Design Solutios for Developers, G. Nudelman Wiley, 2013
[10] Tour (Patró) – [11] Metàfora (Patró) www.androidpatterns.com [Consulta: 05/05/2016]