Boat Attack: El desenvolupament d'una aplicació mòbil

Màster Universitari en Aplicacions Multimèdia

Itinerari Professional

Autor: Francesc Casellas López

Consultor: Sergio Schvarstein Liuboschetz

15/06/2015

Crèdits/Copyright

Agraïments

Al Professor Sergio Schvarstein Liuboschetz vull agrair-li la seva ajuda i dedicació, sempre ha resolt tots els dubtes que se m'han plantejat al llarg del treball final de màster i m'ha guiat durant tot el procés.

A la meva família i, sobretot, a la meva estimada Sara, pel seu suport incondicional, i la paciència que han tingut amb mi en les hores de treball i de recerca. Així com, pels seus suggeriments i opinions que, sens dubte, m'han ajudat molt a tirar endavant i completar aquest projecte.

A tots els meus amics i companys que també m'han ajudat sempre que el he necessitat.

Resum

El treball Final de Màster engloba totes les competències i coneixements que he adquirit al llarg de les diferents assignatures del Màster d'Aplicacions Multimèdia, itinerari Professional.

L'objectiu del treball és establir les bases necessàries per al desenvolupament d'un joc natiu pels dispositius *Android*. Així doncs, el TFM consisteix en la planificació, disseny i implementació d'un joc consistent en enfonsar tants vaixells enemics com li sigui possible a l'usuari durant el temps que dura la partida, i vigilant sempre de no enfonsar els vaixells amics que també es mostren per la pantalla.

El resultat d'aquest TFM és una versió inicial amb menys funcionalitats de les que tindrà la versió definitiva, però que servirà per realitzar una prova de concepte, així com per analitzar l'ús que en fan els usuaris. A més dels reptes tecnològics implícits en el desenvolupament d'una aplicació, un dels principals reptes abordats ha estat el disseny complert tant de les pantalles com dels icones que s'utilitzen en l'aplicació, doncs malauradament no tenia experiència en aquesta faceta del desenvolupament.

Darrera el desenvolupament d'aquest treball s'hi amaga un gran interès personal per utilitzar la meva experiència professional i aprofundir-la per adquirir els nous coneixements que de ben segur m'ha aportat el desenvolupament d'un treball centrat en l'àmbit dels jocs multimèdia.

Paraules clau

Treball de Fi de Màster, Memòria, Aplicacions Multimèdia, App game, Android, battle ship, diversió, entreteniment

Abstract

The Master's Thesis encompasses all that skills and knowledge that I have gained through all the subjects of the Multimedia Applications Master, Professional itinerary.

The aim of this Thesis is to lay the foundations for the development of a native game available on *Android* devices. Therefore, the Master Thesis is the planning, design and implementation of a game that consists in sinking as enemy ships as the user can during the time that a game takes, on the other hand, the user should keep an eye on not sinking the friendly ships that are displayed too.

The Thesis outcome is a beta version with the minimum required value and features, that will be used to do a proof of concept, furthermore, this beta version is going to be very useful as it will allow me to analyse the usage that the users do of the application. Besides the technological challenges implicit in the development of an *Android* application, one of the main challenges I have faced has been the full design of both the screens and the icons required for the application, as I didn't had experience in this field of development.

Behind the development of this Thesis there is a great personal interest in using my own professional experience as well as deepen this experience in order to acquire knowledge that the development of a Thesis focused on the field of multimedia games has, for sure, brought me.

Notacions i Convencions

En aquest treball s'han utilitzat dos tipus de font diferents:

- **Arial Narrow** pel text de la memòria.
  - Pels títols dels apartats s'ha utilitzat el format negreta amb un tamany 11 de la lletra i color blau fosc.
  - Pels subapartats de segon nivell s'ha utilitzat un tamany 11 de lletra i color blau fosc.
  - Pels subapartats de tercer nivell s'ha utilitzat un color blau clar, el format de lletra cursiva i un tamany 11.
  - Per les figures s'ha utilitzat un tamany 9 i el format negreta amb color blau clar.
- **Consolas** amb tamany 9 pels fragments de codi de l'aplicació.

Índex

Figures

Introducció: Resum de la proposta

La temàtica del treball final de màster és la planificació, el disseny i la implementació d'un joc natiu pels terminals *Android*. Aquesta aplicació nativa ha de ser suportada pels terminals amb sistema operatiu *Ice Cream Sandwitch* - versió 4 del sistema operatiu *Android* - cap amunt ja que, d'acord amb les estadístiques d'*Android [1]*, només un 0,4% dels usuaris utilitzen *Froyo* - versió 2.2 - i un 6,9% *Gingerbread - versió 2.3 -*. Per tant, serà molt positiu centrar la implementació de l'aplicació en la versió *Lollipop* per tal d'aprofitar novetats tecnològiques com, per exemple, el *material design [2],* però fent ús de la llibreria de suport per garantir el funcionament en les versions anteriors d'*Android* fins a 4.0.
El treball final de màster, per tant, se centrarà en la conceptualització i el disseny de l'aplicació, per tal d'establir uns bons fonaments per al desenvolupament d'aquesta. Així doncs, el treball es dividirà en les següents seccions o fases:

- Descripció de l'aplicació, és a dir, definició de com ha de ser l'aplicació i com ha de funcionar, així com la descripció dels perfils d'usuari a la qual anirà dirigida l'aplicació i el seu context d'ús.
- Llistat de funcionalitats. Establir i detallar totes les funcionalitats que inclourà l'aplicació.
- Disseny funcional i prototip de l'aplicació.
- Implementació i desenvolupament del joc.

El joc consistirà en una pantalla en la qual hi aniran apareixent vaixells a diferents altures de la pantalla i a diferents velocitats. Hi haurà tres categories de vaixells diferents:

- Vaixells enemics: Inicialment el formaran els pirates però a mida que l'usuari jugui i avanci en el joc també podran aparèixer altres tipus de vaixells com, per exemple, els *víkings.*
- Vaixells amics.
- Vaixells amb carregaments de pólvora.

L'objectiu del joc serà enfonsar tants vaixells enemics com sigui possible en 40 segons. Tanmateix, per cadascun dels vaixells enemics que enfonsi l'usuari, se li sumarà un temps addicional, per exemple, per cadascun dels vaixells pirates sumarà 3 segons més. Per altra banda, l'usuari haurà de vigilar de no enfonsar els vaixells amics ja que això el penalitza restant 5 segons. Pel que fa als vaixells amb carregaments de pólvora, el que fan és explotar i enfonsar tots els vaixells que es trobin a un cert radi de distancia, per tant, poden ser utilitzats per enfonsar molts vaixells enemics al mateix temps però, per contra, també podrien enfonsar als vaixells amics i penalitzar.
Per tal d'enfonsar els vaixells l'usuari haurà de fer un *"tap"* amb el dit indicant el lloc exacte de la pantalla on vol que es dispari una bala de canyó. Tanmateix, quan l'usuari fa el "*tap"* a la pantalla la bala de canyó es dispara des de la part inferior de la pantalla, així que triga un temps – en el qual hi haurà una animació del tir – fins que la bala arribi al seu destí, en conseqüència, l'usuari haurà de calcular on vol que es dispari la bala en funció de la situació i la velocitat del vaixell objectiu.
Així doncs, l'objectiu de l'usuari serà enfonsar tants vaixells enemics com li sigui possible per aconseguir accedir al llistat de rànquing dels usuaris amb més bona punteria. Una part molt important del treball es focalitzarà en el disseny de l'aplicació per tal d'establir les bases per a la seva implementació. Per fer-ho es duran a terme les següents tasques:

- Definir les pantalles de l'aplicació.
- Establir el flux de navegació entre les pantalles per tal de garantir una bona usabilitat a l'usuari de l'aplicació.
- Fer un prototip de l'aplicació amb el disseny de cadascuna de les pantalles i on es demostri el flux de navegació establert.

Un cop el joc ja estigui totalment conceptualitzat i definit, i ja es disposi d'un prototip amb els dissenys requerits per cada pantalla, ja es podrà iniciar el desenvolupament de l'aplicació nativa per *Android*. Aquest desenvolupament es farà mitjançant l'eina de treball *Android Studio* així com la *SDK* d'*Android [3].*

Objectiu i abast

L'objectiu d'aquest treball final de màster és el d'englobar i posar en comú tots els coneixements adquirits al llarg del màster així com l'experiència professional prèvia, per tal de conceptualitzar, dissenyar i implementar una aplicació mòbil, que rebrà el mateix nom que el títol del treball – Boat Attack -, i que tindrà el format de joc multimèdia disponible en els dispositius *Android*.

Objectius principals

- **Conceptualització de l'aplicació**. El primer gran objectiu serà el de conceptualitzar una aplicació que encaixi amb les tendències que marquen el mercat, a la vegada que aporti un valor diferencial al que estan aportant les altres aplicacions que hi ha actualment en el mercat.
- **Anàlisis del context d'ús**. Estudiar i concloure els casos d'ús de l'aplicació per tal de concloure el context d'ús així com el perfil dels potencials usuaris.
- **Disseny i implementació**. Dur a terme la part més pràctica del projecte mirant de garantir un flux de navegació correcte així com una òptima experiència d'usuari.
- **Aplicació dels coneixements adquirits**. Un dels principals objectius del projecte és el de posar a la pràctica tots els coneixements que he adquirit amb l'estudi del màster.
- **Realització personal**. Tot i haver tingut l'oportunitat de desenvolupar diverses aplicacions mòbils i pàgines web, mai havia desenvolupat un joc per *Android*. Així doncs, darrera el desenvolupament d'aquest treball s'hi amaga un gran interès personal per utilitzar la meva experiència professional i aprofundir-la per adquirir els nous coneixements que de ben segur m'aportarà el desenvolupament d'un treball centrat en l'àmbit dels jocs multimèdia.

Objectius secundaris

- Publicar l'aplicació desenvolupada en el mercat d'aplicacions d'*Android* i analitzar-ne la utilització que en fan els usuaris. L'objectiu a llarg termini serà afegir noves funcionalitats a l'aplicació així com una estratègia de monetització.

Abast

L'abast del projecte serà la implementació d'una primera versió del joc *Boat Attack*:

- Descripció de l'aplicació, perfil d'usuari i context d'ús.
- Definició del flux de navegació de l'aplicació.
- Disseny dels prototips així com un esborrany de les icones de l'aplicació.
- Implementació de la funcionalitat del menú i la partida individual.

Queden fora de l'abast del projecte:

- La funcionalitat dels rànquings amb les puntuacions.
- Compartició amb les xarxes socials i reptes als amics.
- Integració de les estratègies de monetització.
- Disseny definitiu de les icones de l'aplicació.
- Funcionalitat de joc multi-jugador.

Planificació

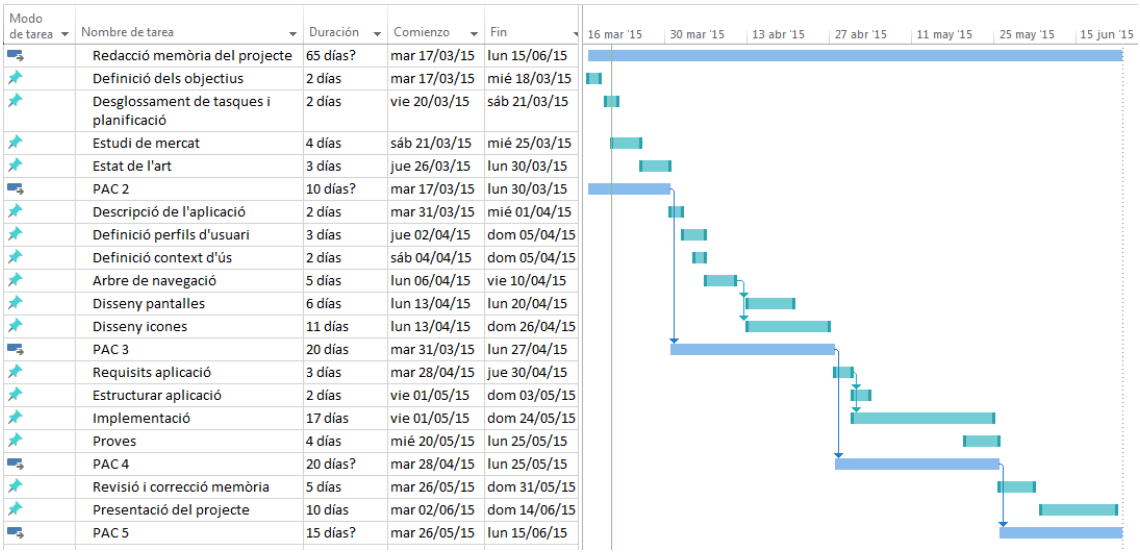
L'objectiu d'aquest apartat és el de desglossar el projecte amb les tasques necessàries per dur-lo a terme així com fer una primera valoració dels temps que requerirà cadascuna d'aquestes tasques.

Fites

Fita	Data
Entrega PAC 2	30 de març de 2015
Definició arbre de navegació	10 d'abril de 2015
Finalització disseny gràfic	26 d'abril de 2015
Entrega PAC 3	27 d'abril de 2015
Implementació aplicació	24 de maig de 2015
Entrega PAC 4	25 de maig de 2015
Elaboració de la presentació	14 de junt de 2015
Entrega PAC 5	15 de juny de 2015

Diagrama de Gantt

La planificació del treball es va veure afectada amb les tasques de disseny de l'aplicació, on em vaig adonar que calia afegir alguna tasca nova, així com tornar a calcular alguns temps planificats. Inicialment el diagrama de Gantt amb la planificació va ser el següent:



Modo de tarea	Nombre de tarea	Duración	Comienzo	Fin
	Redacció memòria del projecte	65 días?	mar 17/03/15	lun 15/06/15
	Definició dels objectius	2 días	mar 17/03/15	mié 18/03/15
	Desglossament de tasques i planificació	2 días	vie 20/03/15	sáb 21/03/15
	Estudi de mercat	4 días	sáb 21/03/15	mié 25/03/15
	Estat de l'art	3 días	jue 26/03/15	lun 30/03/15
	PAC 2	10 días?	mar 17/03/15	lun 30/03/15
	Descripció de l'aplicació	2 días	mar 31/03/15	mié 01/04/15
	Definició perfils d'usuari	3 días	jue 02/04/15	dom 05/04/15
	Definició context d'ús	2 días	sáb 04/04/15	dom 05/04/15
	Arbre de navegació	5 días	lun 06/04/15	vie 10/04/15
	Disseny pantalles	6 días	lun 13/04/15	lun 20/04/15
	Disseny icones	11 días	lun 13/04/15	dom 26/04/15
	PAC 3	20 días	mar 31/03/15	lun 27/04/15
	Requisits aplicació	3 días	mar 28/04/15	jue 30/04/15
	Estructurar aplicació	2 días	vie 01/05/15	dom 03/05/15
	Implementació	17 días	vie 01/05/15	dom 24/05/15
	Proves	4 días	mié 20/05/15	lun 25/05/15
	PAC 4	20 días?	mar 28/04/15	lun 25/05/15
	Revisió i correcció memòria	5 días	mar 26/05/15	dom 31/05/15
	Presentació del projecte	10 días	mar 02/06/15	dom 14/06/15
	PAC 5	15 días?	mar 26/05/15	lun 15/06/15

Figura 1 - Diagrama de Gantt amb la planificació inicial

Tanmateix, els temps de disseny calculats – fins a 11 dies per realitzar la tasca de disseny de les icones, i uns altres 6 dies per dissenyar les pantalles de l'aplicació – eren massa elevats. Després de documentar-me sobre els programes que he utilitzat per fer els dissenys i de dedicar-li forces hores a realitzar tutorials d'aprenentatge, vaig poder reduir considerablement els temps de dedicació d'aquestes dues tasques, rebaixant-los a 5 dies de feina cadascuna. Per altra banda, vaig afegir una nova tasca dins dels temps de realització de la PAC 3 consistent en una ampliació de l'estudi de mercat realitzat a la PAC anterior centrant-me en quatre exemples d'aplicació que, pel seu mode de funcionament, em podien servir d'exemple per emmirallar l'aplicació.
Gràcies a aquests canvis en la planificació de les tasques de disseny, he disposat de més dies per dedicar-li a la tasca d'implementació de l'aplicació. Així doncs, la planificació final del treball realitzat ha estat la següent:

Nombre de tarea	Duración	Comienzo	Fin
Redacció memòria del projecte	65 días?	mar 17/03/15	lun 15/06/15
Definició dels objectius	2 días	mar 17/03/15	mié 18/03/15
Desglossament de tasques i planificació	2 días	vie 20/03/15	sáb 21/03/15
Estudi de mercat	4 días	sáb 21/03/15	mié 25/03/15
Estat de l'art	3 días	jue 26/03/15	lun 30/03/15
PAC 2	10 días	mar 17/03/15	lun 30/03/15
Descripció de l'aplicació	2 días	mar 31/03/15	mié 01/04/15
Definició perfils d'usuari	3 días	jue 02/04/15	dom 05/04/15
Definició context d'ús	2 días	sáb 04/04/15	dom 05/04/15
Arbre de navegació	5 días	lun 06/04/15	vie 10/04/15
Disseny icones	5 días	sáb 11/04/15	jue 16/04/15
Disseny pantalles	5 días	vie 17/04/15	jue 23/04/15
Estudi comparatiu del mercat	2 días	vie 24/04/15	lun 27/04/15
PAC 3	20 días	mar 31/03/15	lun 27/04/15
Requisits aplicació	2 días	mar 28/04/15	mié 29/04/15
Estructurar aplicació	2 días	vie 01/05/15	dom 03/05/15
Implementació	17 días	vie 01/05/15	lun 25/05/15
Proves	2 días	vie 22/05/15	lun 25/05/15
PAC 4	20 días	mar 28/04/15	lun 25/05/15
Restructurar i adaptar la memòria a partir de les PAC's	8 días	mar 26/05/15	jue 04/06/15
Revisió i correcció memòria	2 días	vie 05/06/15	lun 08/06/15
Presentació del projecte	5 días	lun 08/06/15	vie 12/06/15
PAC 5	15 días	mar 26/05/15	lun 15/06/15

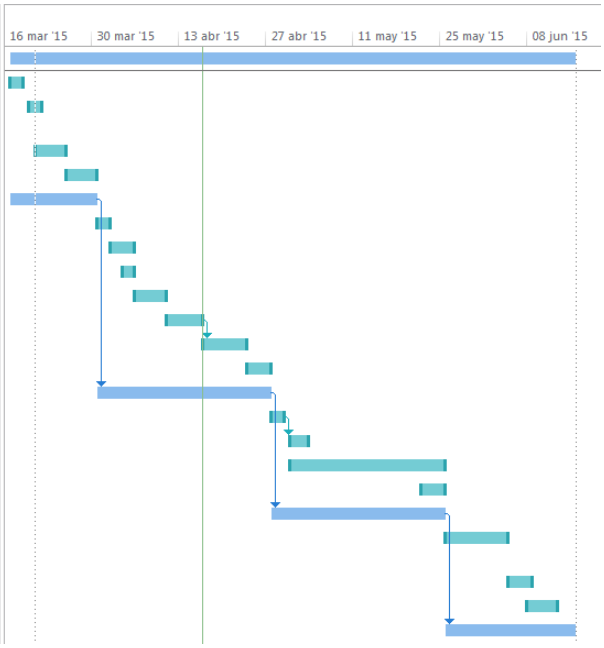


Figura 2 - Diagrama de Gantt amb la planificació final

Anàlisi del mercat

La introducció dels *smartphones* al mercat va suposar un canvi en la manera que tenien els usuaris d'entendre els telèfons mòbils, i va obrir tot un nou ventall de productes i aplicacions que ha suposat una veritable revolució tecnològica. Com a conseqüència d'aquesta revolució la societat ha canviat els seus hàbits i ha integrat aquesta nova tecnologia en la vida quotidiana, esdevenint en poc temps un dispositiu imprescindible per al dia a dia de molta gent.

Estudi de mercat

El nombre de dispositius mòbils, no ha deixat de créixer i enquestes publicades ja mostren que és la tecnologia més utilitzada per accedir a la xarxa *[4].* Així doncs, estudis i enquestes realitzades recentment han conclòs que els usuaris utilitzen cada vegada més els dispositius mòbils en front dels ordinadors d'escriptori tradicionals. Això s'ha traduït en que aquests dispositius ja són molt utilitzats per accedir a pàgines web, xarxes socials, etc.*[5]*
Aquesta tendència no ha passat inadvertida i, per aquest motiu, s'ha estes la filosofia de "*Mobile first*". Aquesta nova forma de pensar implica que en el procés de disseny de qualsevol pàgina web es tingui molt en compte els dispositius mòbils - tant els telèfons mòbils com les tauletes – per tal de garantir que tant el disseny com les funcionalitats siguin correctes independentment del tamany de la pantalla des del que s'accedeix. Per assolir aquest objectiu s'estan utilitzant tecnologies com *CSS3* que ja permeten que els dissenys siguin responsius, és a dir, que s'adaptin al tamany de la pantalla.
Així doncs, la revolució tecnològica impulsada pels telèfons intel·ligents ha implicat, no només les millores tecnològiques en les pàgines web, sinó també en la proliferació d'aplicacions de tot tipus. Concretament, una de les indústries que s'espera que tingui un creixement més important és la dels jocs per mòbils, doncs s'estima que en només dos anys generarà uns ingressos de 23,9 bilions de dòlars. [6] Aquests números es poden explicar, en part, ja que al voltant del 23% de les aplicacions mòbils en *Apple, Android, Blackberry* i *Windows Phone* són jocs. No només això, sinó que entre el 70 i el 80% de les aplicacions que es descarreguen són jocs i un 53% dels usuaris afirmen que juguen diàriament a algun joc per *smartphone [7].*
Una de les claus que expliquen aquest fenomen és que els jocs per mòbil han evolucionat, passant d'una situació en que l'usuari jugava tot sol contra la màquina fins a un nou concepte de jocs socials, on les partides es comparteixen amb els amics i es juga amb grup cadascú des dels seus terminals. De fet, la revista *Forbes* apunta

que l'evolució dels jocs no vindrà determinada per millores en el *hardware*, sinó que seran les millores en les possibilitats de l'usuari i els jocs multi-plataforma les que empenyeran a un creixement del sector *[8].*

Cal destacar que no són només els adolescents els que estan fent un ús més elevat d'aquest tipus d'aplicació, sinó que és una tendència que s'ha estes tant a homes com a dones d'edats molt diverses. Per exemple, als Estats Units es calcula que fins a un 68% dels usuaris que juguen amb el mòbil són majors de 18 anys, i que l'edat mitjana dels jugadors està entre els 30 i 35 anys *[8]*.

Aquests canvis socials no haguessin pogut esdevenir si no hagués estat per la gran evolució que han sofert els telèfons mòbils. L'aparició al mercat de gegants com *Android* o *iOS* ha revolucionat el mercat convertint-lo en un monopoli. Segons estadístiques d'*IDC – International Data Corporation* -, *Android* ha menjat el terreny a altres competidores com *Blackberry, Windows Phone*, o inclòs *iOS*, assolint una quota de mercat del 80,2% el 2014. En la figura següent es pot veure l'evolució de les diferents plataformes des de l'any 2009 fins a principis de 2014. *[9]*

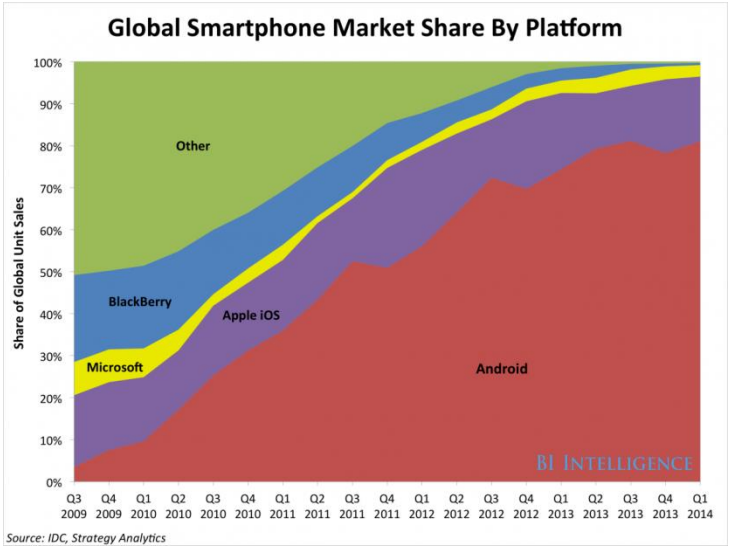


Figura 3 - Evolució de la quota de mercat per plataforma

En la gràfica realitzada per l'*IDC* es pot observar que actualment hi ha dos grans plataformes que es distribueixen gairebé tot el mercat – *Apple iOS* i *Android* -, deixant una quota molt residual a la resta de plataformes. Aquest mateix comportament el trobem a Espanya, on les estadístiques demostren que es venen molts més dispositius *Android* que de qualsevol altre sistema operatiu, per exemple, el mes de setembre de l'any 2014 un 90,4% dels terminals que es van vendre eren *Android*, en front al 6,3% que eren *iOS. [10]*

Una excepció d'aquesta tendència es troba en el mercat d'Estats Units. En aquest país es calcula que el 52% dels prop de 140 milions de *smartphones* que hi ha són dispositius *iOS*, mentre que *Android* té un 42% de quota de mercat. *[11]*

Una de les claus que expliquen aquest clar domini del mercat radica en la nombrosa quantitat de fabricants de gammes, nacionalitats i preus molt diferents que han confiat en aquest sistema operatiu i que l'han convertit en el gegant que és actualment. A més, *Android*, anant de la mà de fabricants com *Samsung,* s'ha anticipat a tota la seva competència en decisions estratègiques com la creació de mòbils de més gran tamany de pantalla – *Samsung Galaxy Note* -, i que *iOS* no ha afrontat fins a la recent comercialització del *iPhone 6 plus*. Per altra banda, un altre motiu d'aquesta elevada quota de mercat que té la companyia és el fet que s'hagin centrat també en captar els usuaris amb menys recursos econòmics i no només la gent amb més ingressos. Com a conseqüència d'això, *Android* és el clar dominador en països amb menys recursos econòmics com, per exemple, la Xina o l'Índia. *[12]*

Justament aquesta estratègia comercial ens pot ajudar a entendre que, degut al perfil dels usuaris, tenir accés a un gran nombre de potencials usuaris no es tradueix necessàriament en aconseguir un major volum d'ingressos. Segons dades de l'*IDC,* l'any 2014 el 66,6% dels ingressos obtinguts en el mercat dels *smartphones* van ser per a

Android, que domina el mercat amb el voltant del 80% de quota de mercat. En canvi, *iOS* va obtenir un 30,4% dels ingressos totals del mercat tenint només un 13,8% de quota de mercat. *[13]*

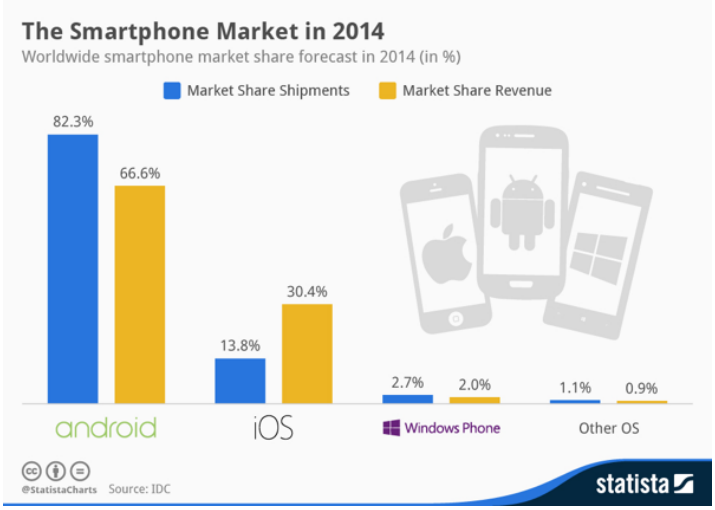


Figura 4 - Comparativa quota de mercat i ingressos [10]

Pel que fa al desgranament per categories, els jocs són la categoria que més descàrregues i més ingressos generen independentment del sistema operatiu *[14]*.

Estratègia de màrqueting

Un error greu que es comet sovint i que caldria evitar és la publicació d'una aplicació – tant *iOS* com *Android* – sense tenir unes estratègies de monetització definides. Si bé es podria publicar una primera versió sense aplicar aquestes estratègies, si que és necessari tenir-les plantejades quan es desenvolupen els dissenys de l'aplicació; de la mateixa manera, cal dissenyar els estils i navegació de l'aplicació en funció de la plataforma on funcionarà així com del context d'ús *[15]*.
Analitzant el mercat actual es pot concloure que hi ha tres estratègies principals de monetització: *[16]*

- **Gratuïta amb publicitat –** Tot i ser la manera més fàcil de començar a monetitzar per una aplicació gratuïta, aquesta és una de les estratègies menys efectives. El gran avantatge d'aquesta estratègia és que no és l'usuari qui paga directament, sinó que els ingressos provenen a partir d'introduir anuncis en l'aplicació. Es monetitza cada vegada que un usuari cliqui en un anunci, tanmateix, les estadístiques mostren que els *CTR – Clicks-Through-Rate* – no solen ser molt elevats, això fa que es necessitin moltes impressions i molts clics per poder obtenir alguns ingressos. A més, cal vigilar on es posa la publicitat per tal de no empitjorar l'experiència de l'usuari dins l'aplicació ni tampoc abromar a l'usuari amb excessiva publicitat. Dins de la publicitat podem trobar tres estratègies diferents: anuncis intercalats que ocupin tota la pantalla i que apareguin en certs moments de la navegació, *banners* publicitaris que estiguin estàtics en un lloc de la pantalla i, una estratègia més recent, consistent en reproduir anuncis – sobretot d'altres aplicacions - en format de vídeo.
- **Freemium –** Aquesta estratègia consisteix en publicar l'aplicació gratuïta en el mercat i obtenir els beneficis a partir de pagaments dins de l'aplicació. S'ha demostrat – i així ho recullen estadístiques de *IDC* i *AppAnnie* – que aquesta estratègia és la més eficient en quant a beneficis, però implica una major complexitat tecnològica i temps de desenvolupament. Així doncs, al ser gratuïta s'eliminen barreres per tal d'obtenir el màxim nombre de descàrregues i, al mateix temps, es busca fidelitzar a l'usuari per a que faci els pagaments per obtenir més beneficis dins de l'aplicació. Un perill que pot comportar aquesta estratègia és que es converteixi en un *"paga per guanyar"*, ja que en aquest cas l'usuari tindrà una mala impressió de l'aplicació.

- ***Premium* –** Consisteix en fer l'aplicació de pagament en el mercat. Aquesta estratègia cada vegada és menys utilitzada degut a que suposa una barrera per obtenir descàrregues, a més, implica que els usuaris han de tenir les targetes de crèdit enllaçades amb la compta del *Google Store* i no tothom està disposat a fer-ho. L'any 2014 només el 20,4% de les aplicacions eren de pagament, en canvi, un 49,7% eren *freemium* amb *in-app purchases*, un 50.3% tenien anuncis intercalats i un 40.8% tenien els anuncis en format de *banner*.

Donant una ullada ràpida al llistat d'aplicacions amb més ingressos – i tal i com indiquen les dades explicades - s'arriba a la conclusió que la millor estratègia per monetitzar és publicar l'aplicació de manera gratuïta i integrar pagaments per obtenir funcionalitats o objectes virtuals que millorin l'experiència en l'aplicació. A més, cal anar amb molta cura quan s'afegeixen pagaments ja que poden generar molta controvèrsia entre els usuaris – com a clars exemples tenim els casos recents de *Whatsapp* o *Tinder* -.
Així doncs, serà necessari fidelitzar a l'usuari i generar-li la necessitat i l'interès per a que faci pagaments dins de l'aplicació i, de manera complementaria, afegir publicitat sense arribar a influir en l'experiència d'usuari ni en la percepció que aquest pugui tenir de l'aplicació. Com sempre, el secret radica en conèixer que busca l'usuari i en no demanar-li més del valor que ofereix l'aplicació – l'usuari no s'ha de sentir enganyat ni utilitat –.
Per altra banda, també és important tenir definides les estratègies de promoció de l'aplicació. Actualment hi ha moltes formes diferents d'arribar als potencials usuaris, unes quantes de les estratègies més utilitzades són:

- **Publicar anuncis en les xarxes socials**. Les xarxes socials són una gran font per arribar als potencials usuaris de l'aplicació ja que permeten segmentar perfectament entre els usuaris i focalitzar-se amb aquells més interessants. Així doncs, una bona estratègia de promoció per les aplicacions mòbils serà la publicitat en xarxes com *Twitter* o *Facebook*, segmentant el públic al que se li publicarà l'anunci per tal de focalitzar-nos en aquells usuaris que més ens interessin. L'experiència demostra que és una gran font per obtenir usuaris amb *CPU* força baixos, però que la qualitat de l'usuari pot ser més baixa, és a dir, es poden obtenir molts usuaris però normalment acostumen a ser menys fidels a l'aplicació que els que provenen de fonts orgàniques.
- **Utilitzar campanyes d'aplicacions del dia**. Una estratègia per aconseguir molts usuaris amb molt poc temps és la de pagar a una de les aplicacions que publiquen aplicacions del dia, per exemple, *App of the day*. Aquestes estratègies disparen la quantitat de descàrregues en un sol dia i ajuda a millorar el posicionament de l'aplicació en el mercat d'aplicacions.
- **Utilitzar "*influencers*".** Tot i que pel tipus d'aplicació d'aquest projecte aquesta no seria una bona solució, si que per molts tipus d'aplicacions diferents és una gran font d'usuaris. L'estratègia consisteix en pagar a una persona coneguda així com a un *blogger* per tal que parli bé de l'aplicació – sense que es noti que ha estat contractada -. Quan més seguidors tingui aquesta persona contractada – en les xarxes socials o visites en el blog -, a més persones arribarà el missatge i, per tant, a més potencials usuaris s'estarà arribant.
- **Utilitzar publicitat dins de les aplicacions.** Una altra estratègia de promoció serà publicitar-se dins d'altres aplicacions mitjançant serveis com, per exemple, *adMob.* Aquest tipus de serveis permeten afegir *banners* o anuncis de l'aplicació a l'interior d'altres aplicacions per, d'aquesta manera, aconseguir augmentar el nombre de descàrregues.
- **Utilitzar els propis usuaris de l'aplicació.** Una estratègia comunament utilitzada és la d'empènyer als propis usuaris a compartir la seva activitat en l'aplicació en les xarxes socials per tal de donar-la a conèixer. Una opció per fer-ho és la d'oferir compensacions dins l'aplicació a aquells usuaris que publiquin informació de l'aplicació en les xarxes socials o la recomanin als seus amics.
- **Publicitat en els mitjans.** Aquesta estratègia de promoció consisteix en fer publicitat en els diferents mitjans de comunicació – revistes, diaris, radio, televisió – i, per tant, no està a l'abast de tothom degut a que implica un gran cost. Una estratègia que fan la gran majoria d'aplicacions que utilitzen aquestes

estratègies de promoció són el que es coneix com *Media for equity*, i que consisteix en intercanviar un percentatge de la companyia a canvi de la publicitat.

- **Presencia en les xarxes socials.** Actualment és necessari estar en les xarxes socials per existir, per aquest motiu una bona estratègia de promoció és la de mantenir actives aquestes xarxes tot fent publicacions periòdiques així com aprofitant esdeveniments coneguts per afegir publicacions.

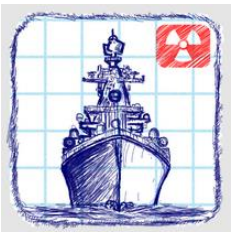
Per tal d'aconseguir usuaris, doncs, serà molt important detallar correctament el perfil d'usuari i executar diverses estratègies en paral·lel per aconseguir descàrregues. A mida que l'aplicació estigui millor situada en el llistat de descàrregues més fàcil serà obtenir-ne de noves i més baix serà el *CPU – Cost per User* -. A més, cal tenir en compte que, en general, el tràfic de més qualitat i més fidel serà el que es generà orgànicament, és a dir, pel boca a boca dels propis usuaris.

Anàlisis DAFO

- **Febleses.**
  - Falta d'experiència en el desenvolupament gràfic.
  - Necessitat d'aconseguir una gran quantitat de descàrregues.
  - Requeriments d'inversió en estratègies de promoció.
  - Requeriment de desenvolupament per adaptar-se a altres sistemes operatius.
- **Fortaleses.**
  - Anys d'experiència desenvolupant aplicacions per *Android*.
  - Experiència en el món de les *start-ups* gestionant projectes.
  - Coneixements sobre les estratègies de promoció.
  - No es necessita un equip de treball ampli per desenvolupar el projecte.
  - No es requereix una gran infraestructura que doni suport al projecte.
- **Amenaces.**
  - Poques barreres d'entrada.
  - Sector molt explotat amb moltes aplicacions al mercat.
  - Les aplicacions competidores provenen a nivell internacional.
  - Falta de finançament.
  - Risc de que et copiïn l'aplicació.
- **Oportunitats.**
  - Gran nombre de potencials usuaris.
  - Facilitat d'arribar als potencials usuaris mitjançant els mercats d'aplicacions.
  - Els usuaris ja estan familiaritzats amb aquest tipus d'aplicacions.
  - Les estadístiques demostren un interès creixent en els jocs per *smartphones*.

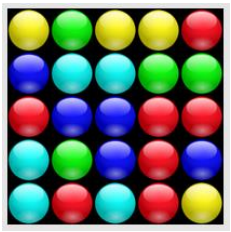
Comparativa de la competència

El mercat d'*Android* està ple d'aplicacions, i hi podem trobar jocs de tots tipus. Tanmateix, és molt beneficiós fixar-se en uns quants jocs que estan oferint a l'usuari la possibilitat de fer partides ràpides per tal d'analitzar quines estratègies de fidelització utilitzen, així com quines són les estratègies de monetització i d'activitat a les xarxes socials.



La batalla naval: Aquesta aplicació consisteix en el joc de *"Hundir la flota"* tal i com el coneixem. Tanmateix, cal destacar que el joc té més de 5 milions de descàrregues en el mercat d'aplicacions. El primer que xoca és veure que els gràfics els han fet com si fossin esbossos de paper, és a dir, quedant lluny del realisme. L'aplicació ofereix poques opcions a l'usuari ja que només hi ha les partides contra la màquina o partides online amb altres

usuaris, tanmateix, és necessari que l'usuari es registri a *Google Plus* per poder jugar en xarxa. La primera estratègia de monetització que han utilitzat és la de la publicitat, ja que l'aplicació mostra *banners* publicitaris en totes les pantalles. A més, per fidelitzar a l'usuari opten per oferir productes virtuals com, per exemple, mines d'aigua o defensa antiaèria, que l'usuari pot comprar mitjançant monedes virtuals per tal d'utilitzar-los en les seves partides. Així doncs, podem concloure que les estratègies de monetització i fidelització són força semblants a les plantejades per *Boat Attack,* ja que es centren en una petita estratègia de publicitats no molesta per l'usuari més oferir productes per fer el joc més entretingut per l'usuari.



Bubble Poke: Tot i ser una temàtica molt diferent, aquest joc planteja moltes similituds amb *Boat Attack.* El joc consisteix en petar les bombolles amb els mateixos colors que estiguin en contacte, quantes més bombolles juntes peti l'usuari a la vegada, més punts sumarà. Així doncs, aquest és un joc que, com *Boat Attack,* no requereix de gaire concentració per jugar-hi sinó que busca entretenir l'usuari i enganxar-lo a seguir jugant una partida darrera de l'altra. Amb aquest concepte tant senzill, aquesta aplicació té més de 10 milions de descàrregues al *Google Play*. Com a mesura de fidelització, l'aplicació planteja que, a més de poder fer una partida d'entrenament, també hi hagi partides per nivells, és a dir, que se li van desbloquejant nous nivells a l'usuari a mida que va superant puntuacions. L'activitat a les xarxes socials la realitzen a partir d'obligar a l'usuari a connectar el seu compte de *Google Plus* per tal de poder accedir als seus assoliments i puntuacions. Per altra banda, l'estratègia de monetització està basada únicament en la publicitat ja que es mostra, per una banda, un *banner* publicitari en totes les pantalles i, per altra banda, quan l'usuari retorna d'una partida se li mostra un anunci que ocupa tota la pantalla – aquest tipus d'anunci està demostrat que reporta més beneficis, però també més molèstia per l'usuari ja que interromp el flux de navegació normal de l'aplicació -.

Fruit ninja: Aquest joc consisteix en partir les fruites que van apareixent en pantalla amb un moviment de dit i evitant que caiguin per la part inferior de la pantalla. Cal destacar que el disseny de l'aplicació està molt elaborat i aconseguit. L'aplicació busca enganxar a l'usuari a fer una partida darrera l'altra a partir del concepte de nivells. Aquest concepte consisteix en que es van sumant tots els punts que va aconseguint l'usuari en cadascuna de les partides per tal que vagi millorant el seu nivell com a usuari. A més, una altra estratègia de fidelització la trobem a partir de la funcionalitat de multi-jugador que permet a dos usuaris jugar a la vegada des del mateix terminal – la pantalla es divideix en dos de forma que els dos usuaris juguen a la vegada l'un contra l'altre -. Per tal d'aconseguir activitat en les xarxes socials, empenyen als usuaris a connectar el seu compte de *Facebook* per tal de continuar les partides des d'un altre dispositiu. A més, després de cadascuna de les partides l'usuari pot compartir la seva puntuació en *Facebook* i *Twitter*. Una particularitat de l'aplicació és que té una versió gratuïta i una altra de pagament – la gratuïta té entre 100 i 500 milions de descàrregues, mentre que la versió de pagament en té entre 1 i 5 milions -. Tanmateix, l'estratègia principal de monetització es basa en els pagaments dins de l'aplicació, ja que ofereix la possibilitat als usuaris de comprar *"fruites estrella"* que posteriorment poden utilitzar per comprar productes virtuals dins de l'aplicació.

Paper Toss 2.0: Un dels punts forts d'aquesta aplicació és la senzillesa de concepte i facilitat d'ús que l'ha portat a tenir més de 10 milions d'usuaris. El joc consisteix en llençar una bola de paper dins d'una paperera tenint en compte que hi ha un ventilador que va canviant de potència. Una particularitat d'aquesta aplicació és que van llençar una primera aplicació – *Paper Toss* – on només es podia llençar la bola de paper a la paperera, havent-hi diferents nivells i una estratègia de monetització totalment enfocada a la publicitat – mitjançant un *banner* publicitari fix així com publicitat a pantalla complerta cada vegada que l'usuari tornava d'una partida -. Tanmateix, en el mercat d'aplicacions també hi trobem *Paper Toss 2.0*, que amplia el joc a partir d'afegir nous objectes a llençar així com la funcionalitat de papereres en moviment. Així doncs, els creadors de l'aplicació van realitzar una primera prova de concepte i, veient l'èxit d'aquesta, han llençat una nova versió de

l'aplicació amb una estratègia de monetització més complerta, ja que han afegit les *"paper points"* que consisteixen en els punts que va acumulant l'usuari de les partides i que després poden intercanviar per altres productes per llençar com, per exemple, tomàquets o grapadores. Aquests punts, a més de guanyar-se a partir de les puntuacions de les partides, també es poden comprar mitjançant pagaments dins de l'aplicació.

Estat de l'art

El mercat de les aplicacions mòbils ha tingut ja un llarg recorregut i es troba en un estat força madur. Per aquest motiu, és fonamental conèixer les etapes que es necessiten per dur a terme el desenvolupament d'una aplicació, així com les principals eines que s'utilitzen actualment per cadascuna de les etapes del projecte. Aquestes eines han anat evolucionant amb el temps i les noves tecnologies, facilitant la feina i ajudant a aconseguir millors resultats.

Actualment ja no es poden deixar detalls a la improvisació, sinó que l'aplicació ha d'estar totalment plantejada i definida. És per això que una de les eines que se solen utilitzar són els programes informàtics per fer esborradors de les pantalles. Fins fa ben poc temps aquests esborradors es feien amb paper i llapis però, des de fa uns anys, ja hi ha a l'abast una gran quantitat de programes - per exemple *Balsamiq* - que ajuden a realitzar aquesta tasca. Amb aquests esborradors ens podem fer una idea més clara de com funcionarà l'aplicació i quin serà el flux de navegació que farà l'usuari. És un cop ja estiguin clars aquests conceptes que caldrà centrar-se en fer un disseny més acurat de les pantalles. Actualment hi ha molts programes disponibles per dur a terme aquests dissenys com, per exemple, *Just in mind*. Aquests programes no solament permeten implementar els dissenys exactes de com es veurà l'aplicació, sinó que també permeten posar a prova el flux de navegació entre les pantalles - ja que es poden enllaçar entre elles -, així com afegir les mateixes animacions que es voldrà que hi hagi en l'aplicació.

Aquesta fase del desenvolupament és molt important ja que, gràcies a aquests dissenys tant exactes, es poden dur a terme *testings* de disseny i de concepte. Aquest tipus de proves s'han posat de moda recentment ja que permeten obtenir *feedback* dels usuaris inclòs abans d'haver començat a desenvolupar l'aplicació. Així doncs, aquests procés de proves consisteix en oferir la maqueta de l'aplicació a un grup d'usuaris per a que aquests duguin a terme un seguit de tasques que se'ls demana. El més important consisteix en observar com reacciona l'usuari i analitzar la facilitat que té per dur a terme les tasques que se li han demanat. D'aquesta manera, serà possible corregir problemes de disseny de l'aplicació centrant-se, no en suposicions, sinó en les reaccions d'usuaris reals. A més, com que aquestes proves permeten descobrir els errors de concepte i de disseny abans de desenvolupar-la, també ajudaran a estalviar el temps que seria necessari dedicar-li si s'hagués de refer alguna de les pantalles.

Un cop ja es tenen clars els dissenys, serà necessari crear una sèrie d'icones, fons de pantalla, etc. Per fer-ho hi ha nombrosos programes informàtics com, per exemple, *FireWorks* o *Illustrator*. També hi ha pàgines webs que ofereixen icones ja fetes, tanmateix, els resultats acostumen a ser més bons quan es desenvolupen els icones especialment per l'aplicació. En el cas concret de *Boat Attack*, serà necessari implementar un fons de pantalla que simulin les onades del mar, una bala de canó, així com els icones de diversos vaixells de diferents mides i estats - vaixell enfonsat o vaixell navegant pel mar -.

Un cop ja esta clar el flux de funcionament de l'aplicació i els seus dissenys, ja es pot començar a treballar amb el codi de l'aplicació. Per desenvolupar una aplicació amb *Android* és necessari instal·lar un programa anomenat *Android Studio*, així com la *SDK* d'*Android*. *Android Studio* és el programa que ha substituït *Eclipse* – que requeria *ADT* - i que et permet dur a terme la programació del codi de l'aplicació per *Android*. *Android* es programa a partir de *Java* i utilitzant la *SDK* pròpia d'*Android*, aquesta *SDK* ofereix moltes classes i funcionalitats que es poden utilitzar o adaptar segons les necessitats. A més, el programa d'*Android Studio* ja té integrats els emuladors per poder provar l'aplicació que s'està desenvolupant sense la necessitat d'un dispositiu físic. Per tal d'instal·lar l'aplicació en un dispositiu es pot utilitzar *Android Studio* per instal·lar-la o, per altra banda, es pot generar un fitxer *APK*. Aquest fitxer és el paquet que conté l'aplicació i que permet distribuir-la. Com que l'*APK* és en definitiva el

codi de l'aplicació comprimit, és una pràctica molt estesa i recomanada la d'ofuscar aquest paquet mitjançant *ProGuard,* de forma que el codi queda protegit a possibles atacs per inspecció del codi.
Una de les pràctiques més utilitzades i necessàries actualment és la d'afegir una llibreria a l'aplicació que s'encarregui d'analitzar l'ús que fa l'usuari d'aquesta. Actualment hi ha moltes opcions al mercat, tant gratuïtes – per exemple *Google Analytics* – com de pagament – per exemple *MixPanel* -, que permeten fer un seguiment de l'aplicació, així com recollir dades i estadístiques anònimes que, en cas de ser ben utilitzades, ajudaran a entendre el que funciona i, sobretot, el que no funciona de l'aplicació per tal d'aconseguir que l'aplicació ofereixi sempre allò que vol l'usuari que l'utilitza.

Definició dels usuaris

L'aplicació resultant d'aquest projecte té com a destinatari objectiu un perfil de públic jove, d'entre 15 i 35 anys, els quals juguen esporàdicament amb l'aplicació per divertir-se i entretenir-se. Aquest tipus d'usuaris necessiten d'al·licients per continuar jugant partides ja que són usuaris acostumats a tenir moltes aplicacions – i molts jocs – en el seu telèfon mòbil i, per tant, se'ls ha d'oferir algun estímul o incentiu per empènyer-los a seguir utilitzant l'aplicació. És per aquest motiu que serà indispensable per l'aplicació tenir una funcionalitat de "assoliment*",* ja que aquesta serà una de les motivacions de l'usuari per seguir enganxat al joc.
Per altra banda, aquest perfil de públic jove estan molt acostumats a utilitzar les xarxes socials i, en conseqüència, a compartir les seves activitats i aficions en aquestes xarxes. Són usuaris, per tant, molt habituats a utilitzar les xarxes socials per comunicar-se i mantenir-se al dia així que veuran amb molt bons ulls – i inclòs necessari – poder compartir els seus assoliments que aconsegueixen dins de l'aplicació amb els seus amics, així com reptar-los a que els superin.
Una altra característica important d'aquest perfil d'usuaris és que són persones molt acostumades a utilitzar el seu telèfon mòbil i les aplicacions d'aquest amb tota naturalitat en gairebé qualsevol moment i lloc on es trobin.
A més, com que el perfil dels usuaris serà principalment gent jove, aquests seran força reticents a fer pagaments, tant per descarregar l'aplicació, com per poder-hi jugar. Així doncs, la millor estratègia de monetització serà a partir de *coins* o monedes virtuals que els usuaris podran utilitzar per aconseguir objectes virtuals dins de l'aplicació. Els usuaris podran guanyar aquestes monedes virtuals sense haver de pagar necessàriament, sinó que les guanyaran a mida que vagin superant assoliments o visualitzant vídeo anuncis. També hi haurà l'opció de que els usuaris paguin per obtenir paquets de monedes virtuals però, donat el perfil jove dels usuaris, les previsions apunten que aquesta opció serà la menys utilitzada i explotada.
Cal destacar que en la versió resultant d'aquest projecte no s'afegiran encara les opcions de monetització, ja que serà necessària una primera validació de la idea de negoci a partir de recollir estadístiques d'ús dels propis usuaris de l'aplicació. Valdrà la pena observar com utilitzen l'aplicació i quines necessitats tenen els usuaris per tal d'explotar-les mitjançant les monedes virtuals i aconseguir un augment dels estímuls que es tradueixi en un major nombre d'usuaris recurrents. Tanmateix, l'estratègia de monetització és una conseqüència directa del perfil d'usuari objectiu definit en aquest projecte.

Context d'ús

El tipus d'ús que faran els usuaris de l'aplicació serà, sobretot, en els seus moments lliures per tal de combatre l'avorriment i entretenir-se, per exemple, quan estan realitzant un trajecte amb metro o autobús. Per aquest motiu, l'aplicació no ha de requerir que l'usuari tingui una bona connexió a la xarxa quan aquest estigui jugant, ja que en molts casos es pot donar la situació que l'usuari estigui jugant en el metro o altres zones amb cobertura deficient. Així doncs, un requeriment important de l'aplicació serà garantir que l'experiència d'usuari sigui correcte independentment de la connexió a internet o el nivell de cobertura del dispositiu.
Serà molt important que el disseny de l'aplicació – així com el flux de navegació – sigui molt intuïtiu i fàcil d'entendre per tal de no requerir la concentració ni l'esforç mental per part de l'usuari, ja que un ús molt majoritari

de l'aplicació podria donar-se en transports públics o en altres situacions en les que l'usuari no pugui centrar-se completament en l'aplicació.
A més, com que un context d'ús molt plausible serà durant els trajectes en transport públic – autobús, ferrocarril o tren -, serà necessari que l'aplicació no requereixi que l'usuari tingui les dues mans en el dispositiu, sinó que aquest pugui jugar perfectament mentre amb l'altra mà s'agafa, per exemple, a la barra o l'agafador de l'autobús. Per tant, a fi de garantir una bona usabilitat de l'aplicació, aquesta funcionarà a partir de "*tocs*" sobre la pantalla del dispositiu que es podran fer inclòs quan l'usuari està agafant el dispositiu mòbil amb una sola mà.
Donat el context d'ús previst de l'aplicació, aquesta haurà d'oferir una modalitat de joc en que les partides siguin molt ràpides i dinàmiques i que el temps de joc sigui curt, serà a partir dels assoliments i els reptes que s'oferiran els estímuls necessaris per que l'usuari enllaci una partida darrera l'altra.

Definició de perfils d'usuari

Un cop definits el perfil d'usuari i el context d'ús que tindrà l'aplicació, val la pena definir diversos escenaris plausibles d'usuaris de l'aplicació:

- En Joan és un noi jove de 25 anys que viu a Sant Feliu amb els seus pares. Recentment ha trobat la seva primera feina com informàtic en una empresa que té les oficines al 22@ de Barcelona. Com a conseqüència, en Joan ha de fer diàriament trajectes en transport públic en els quals s'avorreix. Inicialment va provar de llegir un llibre, però degut a la gran quantitat de gent que hi havia sempre compartint transport, no trobava mai la concentració ni la comoditat per llegir. Per aquest motiu, en Joan juga amb el seu telèfon mòbil per tal d'entretenir-se. La cobertura no és gaire bona i, inclòs en alguns trams del trajecte no hi ha connexió a internet, així que el que vol és entretenir-se jugant amb el mòbil amb aplicacions que no requereixin accés a internet. Per en Joan, doncs, *Boat Attack* pot ser una aplicació ideal. En aquest cas, en Joan no valorarà les funcionalitats de competir amb els seus amics ni de compartir la seva activitat en les xarxes socials, sinó que el que més l'atraurà de l'aplicació serà la possibilitat que aquesta li dona de jugar partides ràpides.
- En Miquel és un noi de 18 anys que està totalment enganxat a les xarxes socials i al *Whatsapp*. Com els seus amics, en Miquel comparteix a *Facebook* tot allò que fa, i el diverteix entrar-hi regularment per veure que fan els seus amics. Tanmateix, a en Miquel també li agradà jugar amb el mòbil quan està assentat al sofà després de sopar mentre de reüll mira la televisió. Per aquest motiu, en Miquel li interessen els jocs que no requereixen un gran nivell de concentració per jugar-hi i divertir-se i que, a la vegada, ofereixen l'al·licient de compartir l'experiència amb els amics. Així doncs, les funcionalitats primordials per aquest jove de 18 anys seran les de compartició de la seva experiència en el joc amb els seus amics, per exemple, compartint les seves puntuacions o reptant als seus amics a superar-les.
- L'Abril és una noia de 34 anys que no acostuma a jugar amb el telèfon mòbil. Tanmateix, de forma esporàdica sí que li agrada jugar quan està a casa seva o té una estona lliure i vol passar l'avorriment. Per l'Abril, el més important serà que l'aplicació li ofereixi un joc dinàmic i atractiu, que li permeti divertir-se sense la necessitat de fer un seguiment del joc. Així doncs, per a ella serà positiu que les partides siguin curtes i independents entre elles.
- En Lluís és un noi de 28 anys que, des de ben petit, li ha agradat competir i superar-se en tots els jocs i activitats que du a terme. No li ha agradat mai quedar segon i sempre intenta superar-se i fer-ho millor. Per aquest motiu, per en Lluís serà molt important el llistat de puntuacions, ja que li permeten veure com ho està fent ell en comparació amb la resta d'usuaris i li serveix de gran motivació i estímul per seguir jugant i superar-se. Una altra de les funcionalitats que més valorarà aquest jove de 28 anys serà la dels *assoliments*, ja que aquesta li oferirà un llistat de reptes que ha d'anar superant i que l'estimularan a seguir jugant. Per a en Lluís, la diversió del joc radicarà en la motivació per superar-se en cadascuna de les partides que juga.

Disseny

Llistat de funcionalitats

L'objectiu del joc serà que l'usuari s'entretingui a partir de jugar partides ràpides i dinàmiques en les quals haurà de mirar de destruir tants vaixells enemics com li sigui possible. Per aquest motiu, tot el funcionament del joc rondarà a partir d'aquest concepte, que es potenciarà amb les següents funcionalitats:

- **Partida –** Funcionalitat consistent en el joc, en la qual van apareixent els vaixells – tant amics com enemics – i l'usuari ha de mirar d'enfonsar-ne tants d'enemics com li sigui possible abans que se li acabi el temps, i sempre vigilant de no enfonsar els vaixells amics.
- **Puntuació –** Llistat amb les puntuacions obtingudes pels usuaris, ordenades de més puntuació a menys puntuació.
- **Assoliments –** Llistat d'assoliments que l'usuari ha de mirar d'aconseguir, per exemple, si l'usuari aconsegueix enfonsar 150 vaixells enemics en una sola partida haurà aconseguit l'assoliment de *mariner*.
- **Configuració d'àudio –** Una funcionalitat serà que l'usuari pugui habilitar o inhabilitar l'àudio i la vibració del joc.

Altres funcionalitats que queden fora de l'abast d'aquest projecte seran:

- **Compartir amb els amics –** L'usuari podrà compartir les seves puntuacions i els seus assoliments amb els seus amics connectant-se amb el seu *Facebook* o mitjançant un missatge de *Whatsapp.*
- **Reptar a un amic –** Aquesta funcionalitat consisteix que un usuari pugui reptar als seus amics de *Facebook* a millorar una puntuació que ha fet ell.
- **Ús de productes virtuals –** Els usuaris podran escollir diversos productes virtuals els quals podran comprar amb monedes virtuals dins de l'aplicació. Aquests productes seran:
  - Bomba: Aquest producte és una arma que els usuaris podran utilitzar durant la partida i que consisteix en amplificar el *toc* que faci l'usuari per tal d'augmentar el radi d'explosió dels vaixells. D'aquesta manera, l'usuari podrà enfonsar molts més vaixells d'un sol *toc*.
  - Escut: Amb aquest producte l'usuari podrà evitar enfonsar els vaixells amics, ja que durant uns quants segons tots els vaixells amics que apareguin en la pantalla tindran un escut i no podran ser enfonsats.
  - Rellotge: L'usuari podrà utilitzar aquest producte per parar el temps durant 5 segons, gràcies a això podrà ampliar el temps que té per enfonsar els vaixells i assolir una millor puntuació.
- **Botiga de la compra –** A partir de la introducció dels productes virtuals, també serà necessari introduir una funcionalitat que permeti als usuaris comprar les monedes virtuals que necessita per aconseguir aquests productes. Bàsicament l'usuari tindrà tres opcions per aconseguir les monedes virtuals:
  - Superant assoliments: Cada vegada que l'usuari aconsegueix un nou assoliment o una bona puntuació, se li donarà un grapat de monedes virtuals.
  - Visualitzant un vídeo anunci: De forma gratuïta pels usuaris podran aconseguir més monedes virtuals quants més vídeo anuncis visualitzin.
  - Cofre de monedes: Els usuaris podran comprar dins de l'aplicació cofres de diferents mides a preus diferents.
- **Nous vaixells –** En etapes posteriors de l'aplicació hi haurà més vaixells. Per exemple, hi haurà vaixells amb carregaments de pólvora que, al ser enfonsats, produiran una explosió que enfonsaran tots els vaixells – tant amics com enemics - que hi hagi a un cert radi seu. A més, també hi haurà més vaixells enemics – no solament els pirates – com, per exemple, vaixells Víkings.

Arbre de navegació

El flux de navegació dins de l'aplicació ha de ser molt simple i intuïtiu per tal de potenciar que l'usuari encadeni una partida darrera l'altra.

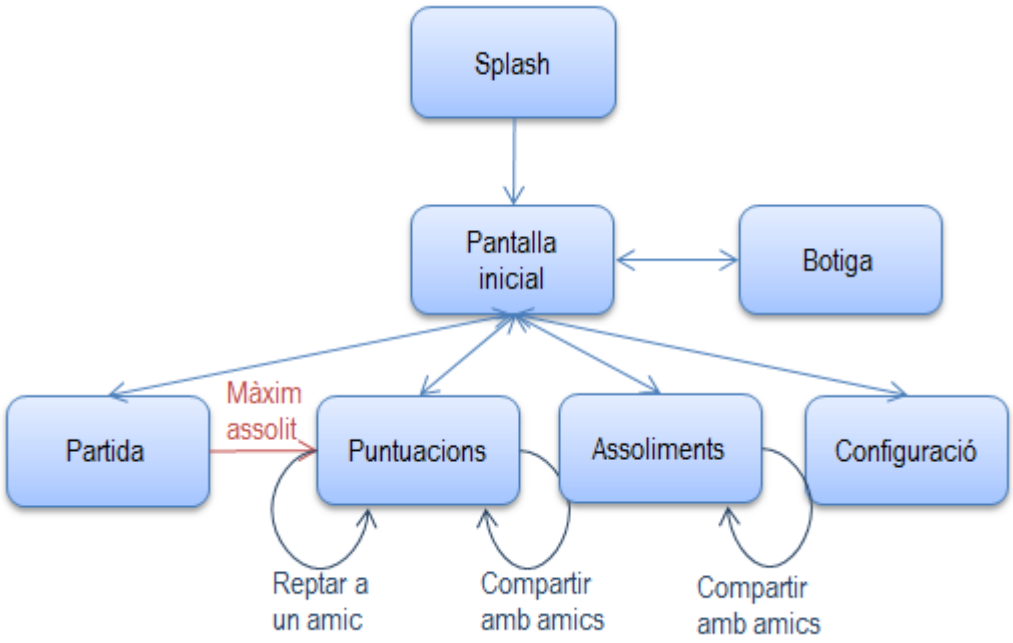


Figura 5 - Arbre de navegació

Quan l'usuari obra l'aplicació se li mostra una pantalla de *Splash* en la qual es mostra la icona de l'aplicació amb una petita animació d'aparició i desaparició. En el moment que la icona desapareix, l'usuari és redireccionat automàticament a la pantalla inicial. En aquesta pantalla l'usuari té totes les opcions que ofereix l'aplicació, però donant-li especial importància a l'opció d'iniciar una partida.

Quan l'usuari inicia la partida desapareixen la resta d'opcions de la pantalla i comença el joc. Quan l'usuari acaba la partida – o l'atura ell abans de que acabi - , aleshores és novament redirigit a la pantalla inicial. Tanmateix, hi ha una petita variant que és quan l'usuari ha assolit un nou màxim de puntuació, ja que en aquest cas si que se l'encamina a l'apartat de puntuacions. Aquest apartat consisteix en el llistat del *ranking* de les puntuacions del joc. Des d'aquesta pantalla l'usuari podrà dur a terme dues accions – marcades en blau fort en el diagrama – i que són les de reptar a un amic o compartir l'activitat amb els amics. Aquestes dues accions es duran a terme mitjançant la xarxa social de *Facebook* – l'usuari s'haurà de connectar amb les seves credencials i podrà compartir textos amb la seva puntuació al mur o bé enviar un missatge privat als seus amics reptant-los a superar la seva puntuació -.

Per altra banda, l'usuari tindrà accés a una pantalla amb el llistat d'assoliments que ja ha aconseguit així com els que li falten per obtenir. Des d'aquesta pantalla, com des de la pantalla de puntuacions, l'usuari podrà compartir la seva activitat a les xarxes socials.

Les altres funcionalitats de l'aplicació – configuració i botiga - són també accessibles des de la pantalla inicial, pantalla a la qual tornen un cop s'han guardat els canvis o s'ha comprat el que es desitjava. Des de la pantalla de configuració l'usuari podrà modificar les opcions de so i de vibració del joc, així com connectar - o desconnectar – el seu compte de *Facebook*. Per altra banda, en la botiga l'usuari podrà intercanviar les monedes virtual per qualsevol dels productes virtuals que s'ofereixen – bombes, escuts o rellotges – que posteriorment podrà utilitzar durant les partides.

Disseny de l'aplicació

Un cop definit el context d'ús de l'aplicació així com el flux de navegació d'aquesta, és moment de realitzar el disseny de l'aplicació per, en una etapa posterior, plasmar-lo en el desenvolupament. L'etapa de disseny de l'aplicació la he dividit en dues fases: una primera fase de definició i disseny de les icones i imatges necessàries en l'aplicació, i una segona de prototipatge del disseny de l'aplicació.

Disseny d'icones

Una de les prioritats marcades després d'analitzar el context d'ús era que l'aplicació fos molt visual i fàcil d'entendre per l'usuari i, per aquest motiu, és molt important realitzar un bon disseny de les icones per tal que aquestes siguin auto-explicatives. Per altra banda, donat que l'aplicació es tracta d'un joc d'enfonsar vaixells pirates, he volgut que les icones tinguessin un aire clàssic i antic i fer-les contrastar amb un fons de pantalla que fos una imatge real.
Per dur a terme aquesta part del treball he utilitzat l'eina *Adobe Illustrator CC [19]*. Aquesta eina té molt potencial i permet elaborar totes les icones tal i com es volen. Tanmateix, mai havia utilitzat cap eina com aquesta per la qual cosa he requerit un temps d'aprenentatge i desenvolupament força elevat.
Un dels objectius personals que em vaig marcar amb aquest treball final de màster va ser aprendre a utilitzar una eina de disseny d'imatges i icones, i estic molt satisfet que, després de dedicar-li moltes hores, hagi aconseguit realitzar totes les icones de l'aplicació tal i com me les havia plantejat des d'un inici. Totes les imatges de les icones explicades a continuació es poden trobar en l'apartat *Annex 1 – Disseny d'icones de l'aplicació* d'aquest mateix treball.
El primer bloc d'icones necessaris en l'aplicació el formen aquelles icones que es mostraran a la pàgina inicial i que li serviran a l'usuari per accedir a les diferents pantalles de l'aplicació. Per cadascuna d'aquestes icones hi ha dues versions, on canvia el color de la forma, ja que la versió de la imatge canviarà mentre l'usuari clica sobre la icona per, un cop l'usuari tregui el dit de la icona, tornar a un estat inicial – això és així per tal de transmetre-li a l'usuari una sensació més real d'haver clicat sobre un botó -. Aquest bloc d'icones el formen:

- Botó de play: Aquest és el botó que utilitzarà l'usuari per iniciar una partida i es mostrarà al mig de la pantalla i d'un tamany més gran que la resta d'icones d'aquesta pantalla principal. Aquesta icona té un tamany de 64x64 píxels. - *Veure figures 7 i 8* -
- Botó de pantalla d'assoliment: Aquest botó redirecciona a l'usuari a la pantalla d'assoliments de l'usuari. El tamany d'aquest botó és de 32x32 píxels. - *Veure figures 9 i 10* -.
- Botó de pantalla de puntuació: Mitjançant aquest botó l'usuari pot accedir a la pàgina amb el llistat de les puntuacions de tots els usuaris. Com en el cas anterior, aquest botó té un tamany de 32x32 píxels. - *Veure figures 11 i 12* -
- Botó de pantalla a configuració: Botó situat a la part superior de la pantalla i que porta a l'usuari a la pantalla amb les opcions de configuració de l'aplicació. Aquesta icona té un tamany de 32x32 píxels. - *Veure figures 13 i 14* -
- Botó de botiga: Amb aquest botó l'usuari accedeix a la botiga on pot visionar vídeos per guanyar monedes virtuals, així com intercanviar les que ja té pels productes virtuals. De la mateixa manera que els altres botons d'opcions, aquest botó té un tamany de 32x32 píxels. - *Veure figures 15 i 16* -

El segon bloc d'icones el formen aquelles icones necessàries per dur a terme una partida:

- Vaixell enemic: Aquest és un dels icones principals doncs representa als vaixells pirates que l'usuari haurà d'enfonsar. Té dues versions que depenen de l'estat del vaixell, per una banda, en el seu estat inicial i, per altra banda, en un segon estat en que el vaixell ha estat enfonsat. El tamany de la icona és de 78x54 píxels. - *Veure figures 17 i 18* -

- Vaixell amic: Aquest vaixell no ha de ser enfonsat i, per aquest motiu, té colors més clars que l'anterior. Tanmateix, igual que en el cas anterior el vaixell té dues versions corresponents als seus dos estats possibles. Aquesta icona té el mateix tamany que el vaixell enemic. - *Veure figures 19 i 20* -
- Bomba: Mitjançant aquesta icona l'usuari podrà disparar una bomba durant la partida. Tot i que aquesta funcionalitat no estarà disponible en la versió obtinguda d'aquest treball, si que es té en compte en els dissenys de l'aplicació. Aquesta icona pot tenir més d'un tamany segons la pantalla on es troba. En la pantalla de la botiga el tamany d'aquesta icona és de 64x64 píxels, mentre que en la pantalla de la partida té un tamany de 32x32 píxels. - *Veure figures 21 i 22* -
- Escut: Igual que en el cas de la bomba, aquesta icona no forma part de les funcionalitats incloses en aquesta primera versió de l'aplicació, però la he desenvolupat per tal de fer un millor disseny inicial de l'aplicació. Aquesta icona es mostrarà en el botó que l'usuari utilitzarà per protegir els vaixells amics dels seus dispars. L'escut té el mateix tamany que té la icona de la bomba. - *Veure figures 23 i 24* -
- Rellotge: Com els casos anteriors, la icona correspon a aclarir els dissenys de l'aplicació i es mostrarà en el botó que l'usuari clicarà quan desitgi que es pari el temps de la partida durant 5 segons. Com en els casos anteriors, aquesta icona té un tamany de 64x64 píxels en la pantalla de la botiga i 32x32 píxels en la partida. - *Veure figures 25 i 26* -
- Bala de canó: Bala que es mostrada durant l'animació quan un usuari clica a la pantalla per tal d'enfonsar un vaixell. El tamany d'aquest icona anirà canviant mitjançant una animació de llançament, passant d'un tamany inicial de 16x16 píxels fins al tamany final a l'arribar al vaixell de 5x5 píxels. - *Veure figures 27* -
- Botó de stop: Botó utilitzat per l'usuari en cas de voler aturar la partida abans de que s'esgoti el temps. Aquest botó té un tamany de 32x32 píxels. - *Veure figures 28 i 29* -

Finalment, en l'aplicació hi ha dues altres icones de gran importància:

- Botó per tornar endarrere: Aquest botó s'utilitzarà per tal que l'usuari retorni a la pàgina principal des de qualsevol de les pantalles – excepte des d'una partida, ja que en aquest cas utilitzaria el botó d'aturar -. El tamany de la icona és de 32x32 píxels. - *Veure figures 30 i 31* -
- Icona de moneda virtual: Aquest icona representa a les monedes virtuals que es poden obtenir i utilitzar dins de l'aplicació. Aquesta icona s'utilitza en la pantalla de la botiga i tindrà un tamany de 64x64 píxels. - *Veure figures 32 i 33* -

Per tal d'obtenir un efecte més xocant per l'usuari i fer destacar l'estil més clàssic de les icones dels vaixells, el fons de pantalla de l'aplicació serà una imatge real de l'oceà – *veure figura 34 -.*

Disseny de l'aplicació

Per realitzar el prototip de l'aplicació he utilitzat el programa *Just in mind [20].* Actualment en el mercat hi ha una gran varietat de programes i serveis que permeten dissenyar tant les aplicacions mòbils com les pàgines web, tanmateix, he escollit *Just in mind* ja que disposa d'un gran nombre de funcionalitats. Per una banda, el programa permet fer el disseny de les pantalles tenint en compte tots els detalls necessaris per tal que sigui el més realista possible, per exemple, utilitzant els icones que prèviament ja havia creat per l'aplicació amb *Adobe Illustrator.* Per altra banda, aquest programa permet enllaçar les diferents pantalles dissenyades per tal de recrear el flux de navegació que tindrà l'aplicació real. Una altra funcionalitat important que m'ha fet decantar per aquesta solució ha estat que aquesta eina permet incorporar animacions en el prototip, de forma que s'aconsegueix recrear amb una gran fidelitat el resultat que es voldrà obtenir amb l'aplicació *[21].*

Un cop finalitzat el prototip de l'aplicació, l'he pujat a la xarxa i l'he publicat per tal que es pugues accedir remotament. Per fer-ho, només cal accedir a la URL https://www.justinmind.com/usernote/tests/14687015/14816657/14830079/index.html. Cal destacar que, com que el prototip de l'aplicació fa ús d'un gran nombre d'imatges i d'animacions, quan es testeja mitjançant la direcció adjuntada es poden tenir problemes de lentitud, per exemple, al carregar la imatge de fons de pantalla o les animacions.

A més d'accedir mitjançant la URL, adjuntat a aquest treball s'hi pot trobar el fitxer *vp* amb el prototip de l'aplicació així com un vídeo demostratiu del funcionament del prototip.

Un dels problemes més importants que m'he trobat al realitzar el disseny de l'aplicació ha estat decidir l'abast que havia de tenir el prototip, ja que la versió beta resultant d'aquest treball final de màster no disposarà de totes les funcionalitats que es preveuen de l'aplicació, sinó que serà una primera versió del joc que permetrà fer una prova de concepte. Finalment, he optat per que el prototip final tingui incorporades totes les funcionalitats de l'aplicació, ja que considero que és molt valuós tenir els dissenys complerts de l'aplicació abans de començar amb la fase de desenvolupament del codi. A més, el desenvolupament del prototip ha sofert molts canvis a mida que anava avançant ja que he utilitzat la funcionalitats de *Just in mind* per realitzar les primeres proves amb els potencials usuaris. Aquestes proves han consistit en ensenyar el prototip realitzat als usuaris i veure com reaccionaven, si trobaven fàcilment el que els demanava i si els semblava un funcionament intuïtiu. D'aquesta manera he pogut anar depurant el disseny i el flux de navegació de l'aplicació. En l'apartat *Annex 2 – Disseny de les pantalles de l'aplicació* s'hi troben totes les captures de les pantalles del prototip.

La primera pantalla de l'aplicació serà el *Splash – veure figura 35* -, aquesta serà la pantalla de benvinguda al joc en la qual l'usuari no haurà de fer cap acció, sinó que només s'hi mostrarà una animació. Inicialment la pantalla tindrà el fons de pantalla amb l'oceà i un text gran amb el nom de l'aplicació – *Boat Attack -,* al cap d'unes mil·lèsimes de segon una bala de canó apareixerà des de la part inferior esquerra de l'aplicació i xocarà contra les lletres fent-les desaparèixer per, a continuació, redirigir a l'usuari cap a la pantalla amb el menú principal.

Aquesta segona pantalla consisteix amb les opcions que tindrà l'usuari mostrades de forma molt gràfica mitjançant icones. A la part superior hi haurà les opcions addicionals – puntuació, assoliments, configuració i, en la versió no beta, botiga -, on cadascuna de les opcions serà una icona de 32x32 píxels de tamany. En canvi, en el centre de la pantalla es mostrarà un gran icona de *play* per tal que l'usuari inici una nova partida. Aquesta icona principal tindrà un tamany de 64x64 píxels ja que és l'opció més important de l'aplicació. *– veure figura 36 per veure la versió completa i la figura 37 per la versió beta -*

Pel que fa a la pantalla de la partida, també he desenvolupat dues versions. Una primera versió en que no hi ha les funcionalitats amb els productes virtuals – bomba, escut i rellotge- que correspon a la versió beta resultant d'aquest treball – *veure figura 38* -. Per altra banda, la segona versió – la qual està visible en la versió definitiva del prototip – si que disposa d'aquestes funcionalitats, per aquest motiu, s'afegeixen a la part inferior de la pantalla els botons per utilitzar cadascun dels productes virtuals, així com el número de productes que disposa l'usuari de cada tipus – *veure figura 39* -. A la part superior de la pantalla s'hi mostren un botó de *stop* de 32x32 píxels a la banda esquerra i que serveix per que l'usuari pugui aturar la partida abans d'hora, un comptador del temps que queda fins que s'acabi la partida a l'esquerra i, al centre, el comptador amb els punts que està acumulant l'usuari en la partida. La resta de part visible de la pantalla és on els vaixells aniran apareixent d'una banda a una altra per tal que l'usuari, fent un toc a la pantalla, els dispari per enfonsar tants com pugui.

L'usuari podrà accedir a la pantalla de puntuacions des del menú principal o serà redirigit després d'una partida en cas que superi la seva puntuació màxima. Aquesta pantalla té un llistat amb el *rànking* del joc, així com un espai on l'usuari pot escriure el seu nom per tal que s'afegeixin les seves puntuacions en aquest llistat – *veure figura 40* -. A més, al costat dret de les puntuacions de l'usuari hi ha un botó de F*acebook* des del qual l'usuari pot compartir la seva puntuació en aquesta xarxa social o reptar a un amic que la superi. Quan l'usuari cliqui sobre el botó s'executarà la *SDK* de *Facebook* que prèviament s'haurà integrat a l'aplicació, un cop s'hagi compartit l'activitat a *Facebook* se li mostrarà un missatge d'èxit a l'usuari per tal d'informar-lo que tot ha funcionat correctament – *veure*

figura 41 -. Així doncs, en els prototips realitzats amb *Just in mind* s'han saltat el pas intermedi de *Facebook* per tal de dissenyar el diàleg resultant dins de l'aplicació.
Pel que fa a la pantalla d'assoliments, consisteix en un llistat amb tots els possibles assoliments de l'aplicació. En cas que l'usuari ja hagi acomplert un assoliment la icona que es mostra canvia, passant de ser un trofeu en gris – *figura 9* - a un trofeu de color negre – *figura 10* -. A més s'afegeix un text indicant-li a l'usuari que ja ha superat un nou assoliment – *veure figura 42* -. A més, en la part superior dreta de la pantalla hi ha un botó de *Facebook* per tal que l'usuari pugui compartir els seus assoliments en aquesta xarxa social. Com en el cas de la puntuació, s'utilitzarà la *SDK* de *Facebook* per realitzar aquesta tasca i, un cop realitzada, se li mostrarà un diàleg d'èxit a l'usuari. – *veure figura 43* -
L'usuari podrà modificar les opcions del joc des de la pantalla de configuració – *veure figura 44* -. Des d'aquesta pantalla, doncs, podrà activar o desactivar el so del joc, així com els efectes de vibració que s'executen quan l'usuari enfonsa un vaixell. Addicionalment, l'usuari podrà registrar o cancel·lar el seu compte de F*acebook.*
Finalment, en la versió no beta del joc hi haurà disponible la pantalla de la botiga. En aquesta pantalla l'usuari pot veure de quantes monedes virtuals disposa així com intercanviar-les pels productes virtuals. A més, en aquesta pantalla també podrà aconseguir més monedes virtuals mitjançant el visionat de vídeos o comprant-los amb *in-app purchase* – *veure figures 45 i 46* -. Cada vegada que l'usuari compra un nou producte o un cofre de monedes virtuals, se li mostra un diàleg informatiu de l'èxit de l'acció. – *veure figures 47 i 48* -

Implementació

Requisits de l'aplicació

El primer que cal decidir per tal de començar a desenvolupar l'aplicació per *Android* és la versió mínima dels dispositius que suportaran l'aplicació. Aquest és un procés delicat ja que, si s'escull una versió molt antiga, tot i donar suport al 100% dels dispositius, es perdrien una gran quantitat de funcionalitats noves. En canvi, si s'escull una versió molt recent, tot i tenir totes les últimes funcionalitats disponibles, es deixaria d'arribar a una gran quantitat de dispositius.
Finalment, m'he decantat per donar suport a partir de la versió 15 de la *SDK* d'*Android*, és a dir, tots aquells dispositius que tinguin una versió *Ice Cream Sandwitch* 4.0.3 o superior. *[22]*
Els motius principals que m'han portat a aquesta decisió han estat:

- Gairebé un 95% dels dispositius *Android* que hi ha al mercat ja estan utilitzant la versió *Ice Cream Sandwitch* o superior, així que, a l'escollir aquest nivell mínim de l'*API*, estem garantint un gran volum de dispositius i de potencials usuaris. *[23]*
- Una de les novetats que va portar l'aparició de la versió *Lollipop* va ser un nou *UI widget* anomenat *RecyclerView*. Aquest *widget*, tot i ser més complex d'utilitzar, aporta alguns avantatges importants com, per exemple, una gran flexibilitat d'arquitectura que permet modificar el seu comportament i adaptar-lo a les seves necessitats. A més, té una gran eficiència gràcies a la reutilització de les vistes que impliquen un augment en la velocitat de pintat i una disminució de la memòria consumida. *[24]* Aquesta nova funcionalitat, tot i estar introduïda per *Android Lollipop – versió 5.0* -, també pot ser utilitzada a partir del nivell 15 sempre i quan s'afegeixi un paquet de suport *[25]*. Per fer-ho, només caldrà instal·lar les dependències i afegir "compile 'com.android.support:recyclerview-v7:+" en el fitxer de *gradle* corresponent.
- A partir del nivell 11 del *SDK* d'*Android* es poden utilitzar les funcionalitats *setX* i *setY* per tal de situar una vista en la pantalla directament per les seves coordenades. Aquesta funcionalitat serà molt important per l'aplicació ja que permetrà simplificar el posicionament dels objectes com els vaixells o les bales de canyó i ajudarà a implementar les seves animacions corresponents.

- Amb la introducció de la versió *Honeycomb* – versió 3.0 d'*Android* – van aparèixer els *Fragments.* Aquest element, que sempre ha d'estar integrat a una *Activity*, representa una part de la interfície d'usuari de forma que es poden combinar múltiples fragments dins d'una mateixa pantalla. Així doncs, aquest element ofereix molta flexibilitat d'implementació i serà necessari per compartir més d'una interfície d'usuari en una mateixa pantalla. *[26]*

Estructura de l'aplicació

L'aplicació estarà dividida en mòduls clarament independents que, utilitzats conjuntament, donaran el funcionament de l'aplicació. Aquests diferents mòduls estaran ordenats i agrupats per blocs segons les seves funcionalitats, de forma que hi haurà quatre blocs de mòduls:

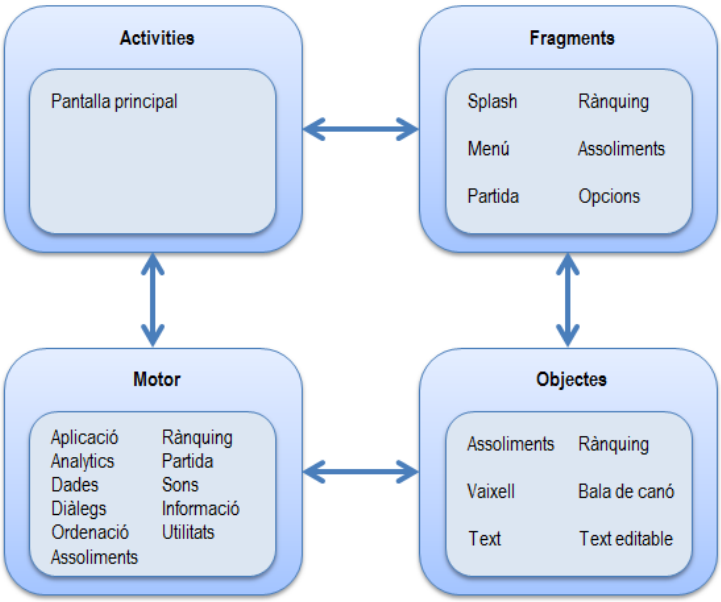


Figura 6 - Estructura de l'aplicació

- **Activities –** En *Android*, es pot resumir una *Activity* com una pantalla de l'aplicació. Al seu temps, aquesta *Activity* pot contenir diversos *Fragments* així com les vistes dels elements. Així doncs, el primer bloc de l'aplicació serà dedicat a totes les *Activities* de l'aplicació que, en aquest cas, només serà una – *MainActivity* -. Això serà així ja que la resta de pantalles de l'aplicació funcionaran en fragments que s'aniran mostrant o amagant mitjançant animacions segons el cas corresponent. Aquesta única activitat tindrà els receptors dels esdeveniments dels botons i serà l'encarregada de gestionar aquests esdeveniments per tal d'alliberar els *Fragments* de fer aquesta tasca.
- **Fragments –** Aquest segon bloc contindrà tots els *Fragments* necessaris per l'aplicació. Els *Fragments* construiran totes les vistes però no faran cap funcionalitat afegida, ja que aquestes es faran des dels respectius motors de l'aplicació. L'objectiu, doncs, és estructurar bé el codi per tal que cada mòdul sigui independent i s'encarregui duna funcionalitat específica.
  A partir del flux de navegació previst, es pot concloure que hi haurà 6 fragments:
  - FragmentSplash per la pantalla del *Splash* inicial.
  - FragmentMenu per la pantalla del menú.
  - FragmentGame per la pantalla de la partida.
  - FragmentRanking pel llistat de rànquings.
  - FragmentAchievements pel llistat d'assoliments.
  - FragmentSettings per la pantalla de configuració.

- **Engine –** Aquest bloc agruparà tots aquells mòduls que s'encarreguen d'alguna funcionalitat i que conformen el motor de l'aplicació. També agruparà els mòduls amb les funcionalitats i les constants necessàries per altres mòduls de l'aplicació. Cal destacar que els motors de l'aplicació s'implementaran com a *singletons* per tal de garantir que siguin mòduls independents i reutilitzables dins de l'aplicació.
    - Analytics amb el motor de funcionament de *Google Analytics*.
    - *BoatAttack* amb la classe d'inicialització de l'aplicació.
    - ComparatorRanking per ordenar el llistat de rànquing.
    - DataManagement amb totes les funcionalitats per guardar i recuperar les dades guardades.
    - Dialogs amb les funcionalitats dels diàlegs de pantalla.
    - EngineAchievements que serà el motor de funcionament dels assoliments.
    - EngineRanking que serà el motor de funcionament del rànquing.
    - EngineGame que serà el motor de funcionament d'una partida.
    - EngineSound que garantirà el funcionament dels efectes de so.
    - Information amb totes les constants globals de l'aplicació.
    - Utils amb un grapat de funcions estàtiques necessàries des de diversos mòduls de l'aplicació.
- **Objects –** Aquest bloc serà l'encarregat d'agrupar els mòduls que corresponen a un objecte d'informació o de vista. Aquests objectes seran utilitzats pels mòduls de motor de l'aplicació per tal de realitzar les seves funcionalitats.

  L'aplicació tindrà 6 objectes:
    - ObjectAchievement que correspon a l'objecte o bloc de dades d'un assoliment.
    - ObjectRanking que correspondrà a la informació d'una posició de rànquing.
    - ObjectShip que serà l'objecte amb la vista i animació d'un vaixell.
    - ObjectShoot que serà l'objecte amb la vista i animació d'una bala de canó.
    - CustomTextView que serà la vista utilitzada pels texts de l'aplicació, utilitzant la font *Stencil*.
    - CustomEditText que serà la vista necessària per tal que els elements de tipus *EditText* – camp editable, per exemple, perquè l'usuari introdueixi el seu nom d'usuari - utilitzin la font *Stencil*.

A més de l'estructura en blocs i mòduls, és important garantir una bona estructura de nombrat dins de l'aplicació. Així doncs, tots els fragments de l'aplicació començaran el seu nom amb *Fragment*, per exemple, FragmentSplash. De la mateixa manera, els mòduls de motor començaran el seu nom per un *Engine* i els d'objectes per *Object.* Els mòduls que no són específicament cap d'aquests casos tindran un nom que sigui auto-explicatiu com, per exemple, *Analytics* o *Dialogs.* A més, les variables i classes també tindran un mateix patró per tal de garantir una bona comprensió del codi. Així doncs, caldrà complir el següent:

- Totes les classes de l'aplicació començaran per majúscula. Per exemple: public class ObjectAchievement.
- Totes les variables de l'aplicació començaran per minúscula i, si tenen més d'una paraula, aquestes estaran juntes acabant una paraula amb minúscula i iniciant la segona amb majúscula. Per exemple: private boolean hasAchieved.
- En cas de les constants globals que no canviaran mai de valor, aleshores estaran en majúscules separant les paraules amb un guió baix. Per exemple: public static final String SHARED_PREFERENCE_PREFS_ACHIEVEMENT.

- A més d'aquesta estructura, també s'implementaran un fitxer amb els estils de l'aplicació – fitxer styles.xml que garantiran la consistència entre totes les pantalles de l'aplicació en quant a tipus de lletra, tamany del text, etc. -, així com els fitxers *strings.xml* que contindran tots els textos de l'aplicació amb les seves corresponents traduccions als diferents idiomes als quals es vulgui donar suport.

Activitats

L'aplicació resultant d'aquest treball final de màster té una única activitat en la qual s'hi mostren diferents fragments segons la pantalla que se li vol mostrar a l'usuari. Així doncs, és important que aquesta classe estengui *FragmentActivity* per tal de donar suport als fragments. Aquesta és una de les classes més senzilles de l'aplicació, però també una de les més importants, ja que és la base sobre la qual funcionen totes les vistes del joc. Com a funcions de la classe s'hi troben totes aquelles que modifiquen el fragment visible de l'activitat així com els escoltadors dels botons. Aquests escoltadors són importants ja que són els que identifiquen quin botó ha estat clicat i reaccionen en conseqüència.
En *l'Annex 3 – MainActivity.java* d'aquest mateix treball s'hi troba el codi implementat d'aquesta classe. Per altra banda, en *l'Annex 4- main_activity.xml* s'hi pot trobar el *xml d'aquesta pantalla*. Aquest fitxer *xml* és el que s'utilitza en *Android* per implementar els dissenys de les vistes. En aquest cas concret, com que la classe *MainAcitivty* és una activitat contenidora de fragments i no de vistes, el seu corresponent *xml* és molt senzill i simplificat ja que només requereix d'un *FrameLayout* – element en el qual s'hi enganxen i mostren els fragments -.

Fragments

El bloc de fragments conté els sis fragments, cadascun correspon a una pantalla diferent de l'aplicació.

Splash

Aquest fragment correspon a la pantalla inicial de l'aplicació, en la qual únicament s'hi mostra una animació. Quan es crea el fragment es mostra un text centrat en la pantalla amb el nom de l'aplicació, en aquest mateix moment s'inicia una animació amb un retard de 750 mili segons. Després d'aquest temps s'afegeix la imatge de la bala de canó amb una altra animació que la trasllada des de la part inferior esquerra fins al centre de la pantalla. Quan la bala de canó arriba al seu punt destí es fa desaparèixer tant la bala de canyó com el text amb el nom de l'aplicació, i es reemplaça aquest fragment pel fragment de la pantalla de menú. Per dur a terme aquestes animacions el que es fa és afegir inicialment tant el text amb el nom com la bala de canó des del *xml* del fragment – anomenat *fr_splash.xml* -, el qual és carrega en el mètode de creació d'aquest. Pel que fa a les animacions, s'utilitza la classe *TranslateAnimation* de la *SDK* d'*Android*, la qual permet moure un element per la pantalla. A més, aquesta mateixa classe ja permet establir uns escoltadors per saber quan acaba l'animació i executar el codi corresponent. En els *Annexes 5 i 6* s'hi troben el codi de la classe en *java* així com el codi *xml* amb les vistes de la pantalla.

Menú

El fragment de menú correspon a la pantalla amb el menú d'opcions del joc. En aquesta primera versió *beta* l'usuari té quatre opcions: jugar, veure el rànquing, veure els assoliments i accedir a les opcions de configuració. Així doncs, aquest fragment carrega la vista amb els botons a partir del fitxer *xml* que es troba en la carpeta de recursos de l'aplicació. Des d'aquest fitxer es creen els botons a partir de les icones creades anteriorment i se'ls dona uns certs identificadors únics. És també en aquest fitxer *xml* que es posicionen els elements com es vol i es dona el tamany idoni a cadascun dels botons. Cal destacar que aquesta feina es fa tenint en compte que hi ha nombrosos dispositius amb pantalles de tamany molt variables. El codi *java* d'aquesta classe només ha de generar la vista a partir del fitxer *xml* ja que, com he explicat anteriorment, els escoltadors dels botons es fan des de la pantalla *MainActivity* a partir dels identificadors únics establerts en el *xml* del fragment. El codi elaborat es pot veure en els *Annexes 7 i 8* d'aquest treball.

Partida

Una bona estructura del codi és clau per realitzar una bona aplicació, que tingui un codi fàcil de mantenir i actualitzar. Per això és molt important dividir correctament el codi en mòduls independents que facin cadascun una feina concreta. Per exemple, el fragment de la partida correspon a la vista d'una partida del joc i, com que és

el codi del fragment, únicament inicialitza la vista inicial amb el botó de parar, el comptador de puntuació i el cronòmetre. Aquests elements es creen en el fitxer *fr_game.xml* i es carreguen en el mètode de creació del fragment. En aquest mateix mètode també s'inicialitza el motor del joc que, com explicaré posteriorment en el treball, és el que s'encarrega de crear el motor dels vaixells – encarregat d'anar mostrant vaixells cada certs intervals de temps – així com el motor de les puntuacions. És en aquest mateix fragment que s'estableix un escoltador d'esdeveniments de contacte, que identifiquen quan l'usuari clica sobre la pantalla i inicien el motor de dispar de la bala de canó. Per tal de comunicar-se entre els diferents motors del joc i la pantalla de la partida s'utilitza una classe d'*Android* anomenada *BroadcastReceiver [27],* aquesta classe permet que el fragment de la partida es subscrigui a uns certs esdeveniments – puntuació, temps i final de partida – i reaccionar en conseqüència. Per exemple, des del motor de la partida s'envien esdeveniments de puntuació amb la nova puntuació per tal que, des del fragment de la partida s'actualitzi la vista amb la nova puntuació. El mateix es fa amb els esdeveniments de temps, que són els utilitzats per anar actualitzant el temps del cronòmetre. Finalment, quan el motor de la partida – classe *EngineGame* – detecta que s'ha acabat el temps de joc, envia un esdeveniment de *gameOver* de forma que, quan el rep el fragment de la partida mostri un diàleg de final de partida. Aquest diàleg té tres opcions segons el resultat de la partida: final de partida estàndard, final de partida amb un nou assoliment assolit o final de partida amb una nova puntuació en el rànquing. En el primer cas, quan l'usuari clica el botó *d'acceptar*, es redirigit cap al menú. En canvi, en els altres dos casos l'usuari és redirigit cap a la pantalla corresponent segons el diàleg. En els *Annexes 9 i 10* s'hi troben el codi de la classe *FragmentGame.java* i de la vista *fr_game.xml.*

Rànquing

Aquest fragment correspon a la pantalla amb el llistat de posicions i, com en els casos anteriors, es pot llegir el codi en els *Annexes 11, 12 i 13*. Per realitzar la llista de les posicions he utilitzat una nova classe introduïda en *Android 5* anomenada *RecylcerView*. Aquesta classe el que fa és afegir una vista per cadascun dels elements de la llista, però de manera eficient tant de velocitat com de consum de memòria, ja que reutilitza aquests elements a mida que l'usuari va fent *scroll* pel llistat. Cadascuna de les vistes dels elements de la llista es carreguen des del fitxer *v_ranking.xml* que conté el disseny d'aquesta vista. Pel cas dels rànquings la vista dels elements de la llista només contenen dos elements de text, un amb el nom de l'usuari i un segon amb el número de la puntuació – aquest fitxer es pot veure en *l'Annex 13* -. Per mostrar la llista s'agafa aquesta del motor de rànquing que és el que té tota la intel·ligència d'aquesta part de l'aplicació, ja que aquest fragment únicament mostra per pantalla els resultats. A més de la llista també hi ha un camp editable on l'usuari ha d'introduir el seu nom, doncs, fins que no ho fa les seves puntuacions no són tingudes en compte en el llistat de posicions.

Assoliments

De la mateixa manera que en el cas del rànquing, el fragment dels assoliments càrrega el motor d'assoliments per tal d'obtenir el llistat d'assoliments possibles així com aquells que l'usuari ja ha assolit. Aquest llistat és mostrat en pantalla mitjançant una classe *RecyclerView.* Cadascun dels elements de la llista carreguen la vista des del fitxer *v_achievement.xml* amb les vistes de l'element – en aquest cas, el nom de l'assoliment, la descripció d'aquest, una icona (que canvia de color segons si l'usuari ja l'ha assolit o no), així com un petit text en cas que l'usuari efectivament hagi assolit l'assoliment -. Així doncs, aquest fragment té una classe *java* que inicialitza un primer fitxer *fr_achievement.xml* amb la vista de la pantalla i que utilitza un altre fitxer *xml* amb les vistes de cadascun dels elements de la llista. Els codis d'aquests tres fitxers es troben en els *Annexes* del treball – *14,15 i 16* -.

Configuració

El fragment de configuració implementa la vista de la configuració del joc. Aquesta classe té una gran simplicitat ja que únicament hi ha dues opcions de configuració. Així doncs, el disseny de la vista és carrega des del fitxer *fr_settings.xml* amb els dos botons, un per activar les opcions de so i un altre per activar les opcions de vibració.

Per guardar les preferències de l'usuari s'emmagatzema en un *SharedPreference*, és a dir, en memòria. Per fer-ho, el fragment de configuració utilitza el motor de dades per tal de, en primer lloc, obtenir les preferències guardades i mostrar d'aquesta manera el botó en l'estat correcte – és a dir, si l'usuari té activades les opcions de so, mostrar aquest en estat *ON*, mentre que si les tingués desactivades s'haurien de mostrar en estat *OFF* -. En segon lloc, en el codi *java* d'aquesta classe s'hi estableix un escoltador per capturar cada vegada que l'usuari canvia l'estat dels botons per, d'aqueta manera, actualitzar la informació guardada en memòria.
El codi tant de la classe *FragmentSettings.java* com de la vista *fr_settings.xml* es troben en els *Annexes 17 i 18* respectivament.

Engines

Aquest bloc conté tots els diferents motors que s'utilitzen des d'altres punts de l'aplicació i que, utilitzats en conjunt, fan possible el funcionament de l'aplicació. Dins dels motors de l'aplicació s'hi poden trobar de dues classes diferents. Per una banda hi ha uns motors que el que fan és agrupar un conjunt de funcionalitats segons la temàtica per tal que el codi sigui més entenedor com, per exemple, els motors d'*Analytics* o *Diàlegs*. En canvi, altres parts més sensibles de l'aplicació tenen motors que són *singletons* de forma que és un motor únic per tota l'aplicació i que s'encarrega d'una part funcional d'aquesta com, per exemple, els motors de la partida, del rànquing o dels assoliments.

Motors per funcionalitats

Aquests tipus de motor agrupen un conjunt de mètodes o variables segons les seves funcionalitats per tal de fer el codi molt més entenedor i, a la vegada, garantir que no s'està duplicant codi en cap moment. Duplicar codi és un greu error de programació ja que pot portar a errors i a dificultats futures per tal de mantenir aquest codi. És per això que és una bona pràctica agrupar les funcionalitats per classes. Una particularitat d'aquests motors és que utilitzen classes i variables estàtiques ja que han de poder ser accedides de forma comuna des de qualsevol punt de l'aplicació. En el codi podem trobar els següents motors per funcionalitat:

- **Dialogs –** Aquesta classe *Dialogs.java* – disponible en *l'Annex 19* del treball – agrupa totes les funcionalitats que fan referència a crear i mostrar un diàleg en l'aplicació. Així doncs, aquesta classe conté quatre mètodes públics i estàtics, és a dir, quatre mètodes que poden cridar-se des de qualsevol punt de l'aplicació. Aquests mètodes serveixen per crear un diàleg i mostrar-lo en pantalla. Un exemple de la seva utilitat és en el fragment de la partida per mostrar el diàleg corresponent al final de cadascuna de les partides de l'usuari.
- **Utils –** La classe *Utils.java* conté tots aquells mètodes que són d'utilitat pels altres motors i classes de l'aplicació. Per una banda, té els mètodes de creació dels *BroadcastReceivers* així com els mètodes d'enviament per aquest – com he explicat, aquests receptors són imprescindibles per garantir la comunicació entre els diferents motors i classes de l'aplicació -. Per altra banda, també conté el mètode encarregat de fer vibrar el mòbil – aquest mètode es crida des del motor de partida quan l'usuari enfonsa un vaixell i té la vibració activada -. En *l'Annex 20* es pot veure com s'ha implementat aquesta classe.
- **Information –** Aquesta classe conté totes les constants globals de l'aplicació. S'entén per constants globals totes aquelles variables que tenen un valor fix i que no canviarà en cap moment de l'aplicació. Per exemple, és una constant global emmagatzemada en aquesta classe les variables amb els noms de les dades que es guarden en les *SharedPreferences*. La utilitat de fer-ho d'aquesta manera és que sempre que es vulgui utilitzar aquest nom, com que s'utilitza la mateixa variable global, assegurem que no ens equivoquem al escriure-la i, per tant, sempre funcionarà. També facilita mantenir el codi en el futur. El codi d'aquesta classe es troba en *l'Annex 21* d'aquest treball.
- **ComparatorRanking –** Aquesta classe que es pot llegir en *l'Annex 22* serveix per ordenar el llistat del rànquing. Així doncs, la classe agafa dos objectes de tipus rànquing i retorna un valor segons si el primer

objecte ha d'anar abans o després del segon objecte. Per tant, aquesta classe és utilitzada des del motor de rànquing per tal d'ordenar correctament la llista del rànquing en ordre de puntuació.

- **BoatAttack –** Aquesta classe es crida automàticament quan s'inicialitza l'aplicació ja que estén la classe *Application* d'*Android.* La principal utilitat d'aqueta classe és que inicialitza tots els motors de tipus *singleton* de l'aplicació. Això és important ja que si un *singleton* s'inicialitza des d'una activitat, *l'Android* podria aturar-lo en algun moment que tingués poca memòria encara que no aturés l'aplicació sencera i això podria provocar errors en el joc. En canvi, si s'inicialitzen en aquesta classe, *l'Android* no podrà aturar els *singletons* fins que no es tanqui l'aplicació. En *l'Annex 23* s'hi troba el codi d'aquesta classe *BoatAttack.java.*
- **DataManagement –** Aquesta classe – veure *Annex 24* – agrupa totes les funcionalitats d'escriptura i lectura de memòria. Com he explicat, tota la informació necessària per l'aplicació es guarda en memòria mitjançant les *SharedPreferences* d'*Android*. Doncs bé, aquesta classe és la que conté tots els mètodes d'escriptura i lectura a memòria. Per exemple, té els mètodes d'escriptura i lectura de les opcions de configuració, de la màxima puntuació d'un usuari, del nom de l'usuari, del total de punts que porta acumulats l'usuari, etc.

Analytics

Aquesta classe agrupa tots els mètodes que fan referència a la funcionalitat de *Google Analytics.* En aquest cas s'utilitza un *singleton* ja que el que es busca és no haver d'inicialitzar el *tracker* de Google Analytics cada vegada que s'ha d'utilitzar, sinó inicialitzar-lo una sola vegada per tota l'aplicació i, d'aquesta manera, tenir un *tracker* que és únic per totes les pantalles i esdeveniments. A més, hi ha dues classes públiques – les quals es poden accedir des de qualsevol punt de l'aplicació -, una per enviar un esdeveniment de pantalla – esdeveniment que s'envia cada vegada que l'usuari navega a una nova pantalla (menú, partida, rànquing,...) – i una altra per enviar els esdeveniments – per exemple, l'inici d'una partida, final de partida, canvi en les preferències de so o vibració, ... -. El codi d'aquesta classe *Analytics.java* es pot llegir en *l'Annex 25.*

Motor de partida

La classe *EngineGame.java*, la qual es pot llegir en *l'Annex 26* d'aquest treball, és l'encarregada de controlar el funcionament d'una partida. Per tant, és la classe clau per fer que el joc funcioni ja que és el motor que controla els vaixells que apareixen per pantalla i el temps que queda per jugar, així com l'encarregada de notificar a la resta de motors i fragments quan hi ha novetats en una partida. Així doncs, quan s'inicialitza aquest motor el que es fa és inicialitzar dos h*andlers [28]* amb dos temps diferents. Per una banda, un *handler* periòdic cada segon que servirà per controlar el temps que falta per jugar i que anirà enviant l'esdeveniment de BROADCAST_*ACTION*_TIMER amb el temps que falta perquè acabi la partida, per tal que el fragment de la partida capturi aquest esdeveniment i actualitzi la vista. Per altra banda, el segon *handler* s'executarà cada cert interval de temps aleatori i el que farà és inicialitzar un nou objecte vaixell. A més de tenir el motor del temps i de vaixell, aquesta classe també conté les variables amb la puntuació que porta l'usuari ja que està subscrit al esdeveniment de *SHIP_SUNKEN* que és enviat per l'objecte vaixell quan l'usuari l'enfonsa. En rebre aquest esdeveniment aquest motor actualitza la puntuació i el temps i envia un nou esdeveniment per tal que el fragment actualitzi les vistes. Com que les puntuacions i els temps de partida es controlen des d'aquesta classe, és aquesta la que té les constants amb els valors dels punts que guanya o perd un usuari per enfonsar un vaixell amic o enemic, així com els segons que se li sumen o resten a l'enfonsar els vaixells.

Motor d'assoliments

Aquest *singleton* és el motor de funcionament dels assoliments. Està creat com un *singleton* ja que interessa que el motor sigui únic per tota l'aplicació ja que la llista d'assoliments, la qual s'inicialitza quan s'obre l'aplicació carregant-la des de memòria, ha de ser única durant tota la vida de l'aplicació. Aquesta classe, a més de tenir i

retornar el llistat d'assoliments, també implementa el mètode que analitza si donada una nova puntuació l'usuari haurà assolit algun nou assoliment o no. El codi d'aquesta classe es troba en l'*Annex 27 – EngineAchievements.java.*

Motor de rànquing

De la mateixa manera que en el cas anterior, aquesta classe és el motor de la funcionalitat de rànquing. Així doncs, aquest *singleton* inicialitza el llistat amb el rànquing i és l'encarregat de controlar si, cada vegada que l'usuari finalitza una partida amb una nova puntuació, aquesta nova puntuació ha de ser afegida al rànquing de puntuacions. Com que en aquesta versió *beta* de l'aplicació no hi ha un servidor que controli els rànquings, aquesta tasca és realitza des d'aquesta classe interna de l'aplicació. Per altra banda, des d'aquesta classe també s'inicialitza un primer llistat de rànquing el primer cop que l'usuari accedeix a l'aplicació per tal de millorar la seva experiència d'usuari amb un llistat que no estigui buit el primer cop que accedeixi. En *l'Annex 28* s'hi troba el codi implementat per aquesta funcionalitat.

Motor de so

Aquest últim motor de l'aplicació, però no menys important, controla els efectes de so de les partides. Quan s'inicialitza el motor s'inicialitzen els dos sons possibles – el so del dispar i el d'un vaixell enfonsat – i es preparen per poder ser executats en qualsevol instant. Aquesta classe, doncs, és un *singleton* justament per garantir que els sons s'inicialitzin una sola vegada a l'inicialitzar l'aplicació i, d'aquesta manera, garantir que sempre estaran preparats per sonar en qualsevol instant de l'aplicació. Per altra banda, aquest motor també conté els mètodes per executar i fer que sonin els dos sons cada vegada que és necessari. *L'Annex 29 – EngineSound.java* conté el codi d'aquesta classe.

Objectes

El darrer dels blocs, però no menys important, és el bloc dels objectes. Hi ha dos tipus d'objectes en l'aplicació, els objectes que agrupen una informació per una certa funcionalitat, i els objectes que inclouen un element de vista així com la seva pròpia animació.

Rànquing

Aquest objecte conté la informació d'un element de rànquing, per tant, el llistat de rànquing està conformat per deu objectes rànquing. Cadascun d'aquests objectes rànquing conté la informació necessària per aquesta funcionalitat, és a dir, la posició, el nom de l'usuari i els punts. A més dels mètodes per actualitzar aquesta informació i retornar-la per poder-la mostrar per pantalla, aquest objecte també conté el mètode que crida al motor de dades per escriure aquesta posició en memòria. *L'Annex 30* d'aquest treball té la implementació d'aquesta classe.

Assoliment

Igual que en el cas anterior, aquest objecte conté la informació d'un assoliment en particular. Per tant, la llista d'assoliments està formada per tants objectes assoliment com assoliments hi hagi. La informació que té aquest objecte és el nom de l'assoliment, la descripció, la puntuació necessària per superar aquest assoliment, un booleà sobre si l'usuari l'ha assolit o no, i un booleà sobre si l'assoliment és de tipus partida o en total – tipus partida vol dir que la puntuació s'ha de revisar per una partida en concret i tipus total vol dir que l'usuari ha d'haver superat una puntuació acumulada total -. Com en el cas anterior, aquest objecte utilitza les funcions d'escriptura i lectura a memòria per mantenir actualitzat l'estat de l'assoliment en totes les partides que jugui l'usuari. El codi d'aquesta classe es troba en l'*Annex 31*.

Vaixell

Aquest objecte és diferent als dos anterior ja que aquest és l'encarregat de mostrar un vaixell en particular i executar la seva corresponent animació. El motor del joc – *EngineGame* – controla els vaixell i va inicialitzant objectes vaixell cada un cert interval de temps. Quan s'inicialitza un nou objecte vaixell aquest aleatòriament decideix si és de tipus amic o enemic, així com la seva posició inicial en l'eix vertical. Aquesta posició vertical es calcula a partir del tamany de la pantalla tenint en compte que hi haurà quatre carrils de vaixells a diferents altures. Un cop ja s'ha inicialitzat el tipus i la posició, s'inicialitza una animació que portarà el vaixell d'esquerra a dreta de la pantalla a una velocitat aleatòria dins d'un llistat de velocitats possibles. Per altra banda, aquest objecte implementa el mètode que comprova si la posició de la bala de canó disparada per l'usuari ha tocat aquest vaixell en concret o no. En cas que sí que l'hagi tocat, el vaixell actualitza el seu estat a *sunken* i canvia la vista de la imatge – passant a la corresponent al seu tipus però enfonsat -, a més, envia un esdeveniment pel *BroadcastReceiver* per tal que el capturi el motor de joc i actuï en conseqüència. En l'*Annex 32 – ObjectShip.java* s'hi troba el codi d'aquesta classe.

Bala de canó

Aquest objecte s'encarrega de mostrar la bala de canó que dispara un usuari. Així doncs, una nova instància d'aquest objecte és inicialitzada cada vegada que l'usuari clica en pantalla per fer un dispar. La inicialització de l'objecte és molt senzilla ja que únicament afegeix la vista de l'element bala de canó en pantalla i executa la seva animació des del punt inicial – part inferior esquerra de l'aplicació – fins al seu destí corresponent – punt de la pantalla que ha clicat l'usuari -. Cal destacar que en aquest mètode d'inicialització es calcula el temps que durarà l'animació en funció de la distancia que hi ha entre el punt inicial i el punt destí, d'aquesta manera, quan l'usuari clica en un punt proper a l'origen la bala de canó aquesta arribarà més ràpid que si l'usuari clica en un punt més llunyà. Aquesta funcionalitat s'ha fet així per donar un efecte més real a l'aplicació. Quan finalitza l'animació, és a dir, quan la bala de canó arriba al seu destí, aquest objecte executa la funció que revisa si s'ha enfonsat algun vaixell amb aquest dispar. Per fer-ho recórrer tots els objectes vaixell que es troben actualment en pantalla i els crida el seu mètode corresponent per que siguin aquests qui, sabent la posició de la bala de canó, comparin la seva posició actual en pantalla – tenint en compte el seu tamany – i decideixin si han estat enfonsats o no. El codi d'aquest objecte es troba adjuntat en l'*Annex 33- ObjectShoot.java*.

Text

En *Android*, per tal d'utilitzar una font pels texts diferent a les que venen per defecte amb el sistema operatiu cal realitzar uns quants passos. En primer lloc, cal emmagatzemar en la carpeta *src/main/assets/fonts* els fitxers *TTF* amb les noves fonts que es volen afegir en l'aplicació. Un cop fet si, com en aquest cas, es vol que els texts utilitzin diferents fonts segons uns certs paràmetres, caldrà crear també un nou recurs en el fitxer *src/main/res/vàlues/attrs.xml*. En el cas de *BoatAttack*, aquest nou recurs s'utilitzarà per establir que els texts que volem que siguin de tipus *bold* utilitzin la font *StencilExpanded*, mentre que si volem que tinguin un estil més fí, aleshores utilitzaran la font *Stencil*. Un cop preparades les fonts cal crear el fitxer d'objecte *CustomTextView* que s'inicialitzarà amb els valors que se'ls passi en els *xml* de les vistes i mostrarà per pantalla el text amb la font que li pertoqui segons el cas. Tant el fitxer *attrs.xml* com l'objecte *CustomTextView.java* es poden llegir en els *Annexes 34 i 35* d'aquest treball.

Text editable

Amb els passos explicats en l'objecte *CustomTextView* s'ha aconseguit crear una nova vista pels texts que utilitzin fonts externes a les que venen per defecte en el sistema operatiu. Tanmateix, per tal que el camp editable on l'usuari haurà d'introduir el seu nom d'usuari també utilitzi aquestes noves fonts, caldrà crear un altre objecte

CustomEditText que, com en l'objecte anterior, s'inicialitzi des del *xml* de la vista i utilitzi la font *Stencil* pel text del camp editable. El codi implementat per aquest objecte es troba en l'*Annex 36 – CustomEditText.java.*

Altre codi d'interès

A més de totes les classes i vistes implementades, per tal que una aplicació *Android* funcioni correctament i estigui ben organitzada cal també desenvolupar un seguit de fitxers que s'implementen en la carpeta de recursos de l'aplicació. Per una banda, per agrupar tots els dissenys de les vistes i evitar haver de repetir-los en cadascuna d'aquestes, he implementat el fitxer *styles.xml*. Allà s'hi troba el disseny, per exemple, dels diferents tipus de text – gran, normal i petit -, els estils de les imatges, els estils dels botons de l'aplicació, etc. Fent-ho d'aquesta manera podem simplificar enormement el codi de les vistes a la vegada que garantim que el codi no estigui duplicat, sinó centralitzat en un mateix lloc, de forma que si en un futur volem canviar algun detall del disseny – per exemple el tamany de la lletra – ho podrem fer canviant només una línia d'un fitxer de l'aplicació. Dins de la carpeta *values* del directori de recursos de l'aplicació, també he implementat un fitxer *arrays.xml*. Aquest fitxer serveix per implementar *arrays* de texts que, en aquest cas, seran utilitzats per enviar els esdeveniments de *Google Analytics.* A més, en aquest directori s'hi ha implementat els texts traduïts de l'aplicació – actualment l'aplicació està en tres idiomes: anglès, català i castellà -.
Finalment, per tal que l'aplicació es pugui instal·lar correctament en els dispositius *Android,* és necessari desenvolupar un fitxer *AndroidManifest.xml,* en aquest fitxer s'hi detallen el nom de l'aplicació, les *activities* d'aquesta, els permisos que se li demanaran a l'usuari per que la pugui instal·lar, el tema de l'aplicació, etc.
Tots aquests fitxers aquí definits es poden trobar a l'apartat d'*Annexes* d'aquest mateix treball.

Instruccions d'instal·lació del fitxer *APK*

Les aplicacions d'*Android* utilitzen una extensió anomenada *APK – Application PacKage file* – que és la que s'instal·la en els dispositius. Bàsicament un fitxer *APK* és un fitxer comprimit com seria un *zip* i que conté tots els fitxes que conformen l'aplicació. Així doncs, aquest fitxer empaqueta els fitxers de l'aplicació i s'utilitza per instal·lar-la en els dispositius *Android,* tant mòbils com tauletes. *[29],[30]*
Adjunt a aquest treball s'hi troba el fitxer *BoatAttack.apk* que és l'instal·lable de l'aplicació desenvolupada en aquest treball. Per instal·lar-la en un dispositiu *Android* només cal seguir els següents passos:

- Per defecte, i per motius de seguretat, els *Androids* no confien en les aplicacions externes al seu mercat d'aplicacions, ja que podria tractar-se de virus o aplicacions que no compleixin amb els requisits establerts per *Android*. Tanmateix, per instal·lar directament d'un *APK* caldrà fer una excepció i fer que l'*Android* permeti aquest tipus d'instal·lacions. Per fer-ho només cal anar a la pantalla de Configuració del dispositiu i entrar en l'apartat de *Seguretat*. Allà s'hi troba l'opció d'*Orígens desconeguts* que per defecte no està marcada, doncs bé, tot el que cal fer és activar aquesta opció.
- Un cop hem garantit que l'*Android* permeti les aplicacions externes al mercat, cal emmagatzemar el fitxer *APK* a la memòria interna o a la targeta *SD* del dispositiu. Per fer-ho podem connectar el dispositiu a un ordinador i des d'allà copiar i enganxar el fitxer, o podem enviar-lo per e-mail i descarregar el fitxer directament des de l'aplicació d'email del mòbil.
- Un cop el fitxer ja es troba en el dispositiu, hem de navegar pels fitxers fins trobar-lo. En les darreres versions d'*Android* ja es disposa d'una aplicació que permet navegar pels fitxers emmagatzemats en memòria i executar-los, en canvi, si la versió és anterior caldrà descarregar una aplicació del mercat d'*Android* per poder-ho fer.
- Un cop hem navegat fins al fitxer *BoatAttack.apk* li fem un clic perquè s'executi. Si no s'hagués seguit el primer pas, aquí ens apareixeria un missatge conforme s'ha bloquejat la instal·lació per motius de seguretat, en cas contrari, es mostrarà una pantalla que ens preguntarà si volem instal·lar l'aplicació. A més, en aquesta pantalla apareixen els permisos que l'aplicació requereix, en aquest cas, accés a

internet – necessari per enviar els esdeveniments de *Google Analytics* – així com de vibració – necessari per fer els efectes de vibració del dispositiu -. En aquesta pantalla cal clicar en el botó de la part inferior dreta que posa *Instal·lar.*

- Al clicar el botó d'instal·lar s'iniciarà el procés d'instal·lació que durarà molt pocs segons. Al finalitzar, es mostrarà una nova pantalla que ens informa que el procés s'ha realitzat correctament i que dona l'opció d'obrir el joc o de continuar des de la pantalla de menú. En aquest punt l'aplicació ja està instal·lada i s'hi podrà accedir sempre que es vulgui des del llistat d'aplicacions del dispositiu.

Conclusions i línies de futur

Aquest treball ha establert les bases i ha servit per implementar una primera versió de l'aplicació *Boat Attack* que servirà per fer una prova de concepte. A més, tal i com m'havia plantejat en els objectius del treball, m'ha servit per desenvolupar el meu primer joc en *Android* així com per aprendre a utilitzar eines de disseny per tal de crear els meu propis icones.

Conclusions

Aquest Treball Final de Màster engloba tots els punts necessaris per realitzar una aplicació per *Android*, des del seu plantejament inicial, l'estudi de mercat i del context d'ús, així com el disseny del flux de navegació i de totes les icones necessàries, fins a la seva final implementació.

Amb aquest treball, doncs, he posat a la pràctica molts dels coneixements que he assolit al llarg de totes les assignatures que he cursat en aquest màster d'Aplicacions Multimèdia i que m'han permès el desenvolupament d'un joc per dispositius *Android* realitzant totes les etapes necessàries.

En l'inici d'aquest Treball Final de Màster m'havia marcat l'objectiu de conceptualitzar una aplicació que s'adeqüés a les tendències del mercat d'aplicacions però que tingués un caràcter diferencial, a la vegada de realitzar-ne un disseny que complís amb els requeriments pre-establerts pel seu context d'ús.

Doncs bé, en el treball s'ha fet un exhaust estudi de mercat, per tal de concloure una clara tendència que marca un augment de les aplicacions mòbils que són gratuïtes i que monetitzen a partir d'oferir productes virtuals. També s'ha estudiat i s'ha conclòs que cada vegada hi ha més usuaris d'aquest tipus d'aplicacions, ja que *Android* ocupa una gran part del mercat dels dispositius mòbils i que la tendència sembla indicar que *Android* seguirà menjant terreny sobre els seus competidors. A més, les estadístiques estudiades indiquen un clar augment de l'interès dels usuaris per aquest tipus d'aplicacions.

Per altra banda, s'ha posat molt èmfasi en l'estudi dels perfils dels potencials usuaris així com del context d'ús d'aquesta aplicació, per tal d'establir les bases per fer un disseny que s'adapti a les necessitats d'ús dels usuaris. D'aquesta manera, s'ha conclòs que l'aplicació havia de ser molt fàcil d'entendre, amb poc text i botons grans, ja que els usuaris haurien de ser capaços d'utilitzar-la amb una sola mà i, fins i tot, en contexts en els que no poden estar totalment concentrats en l'aplicació, per exemple, quan viatgen en transport públic.

A partir de les conclusions extretes s'ha pogut fer un esquema del flux de navegació, així com els dissenys de les pantalles de l'aplicació i dels icones d'aquesta. Valoro de forma especialment positiva aquesta part del treball ja que malauradament mai havia tingut l'oportunitat de treballar amb programes de disseny com l'*Adobe Illustrator CC.* A nivell personal m'havia marcat l'objectiu d'aprendre ha realitzar icones amb aquestes tecnologia, pel que estic molt satisfet de, després de dedicar-li nombroses hores a realitzar tutorials, haver estat capaç de desenvolupar tots els icones de l'aplicació.

Amb totes les bases establertes he pogut dur a terme el desenvolupament del codi de l'aplicació. En aquesta fase he hagut d'afrontar forces complicacions com, per exemple, els nombrosos tamanys diferents que hi ha dels dispositius d'*Android* i que han dificultat la implementació d'una solució pel joc. Tot i que tinc experiència desenvolupant aplicacions per aquest sistema operatiu, mai havia desenvolupat un joc, així que estic molt satisfet d'aquesta primera experiència.

Així doncs, faig una valoració molt positiva d'aquest treball ja que considero que he complert tots els objectius que m'havia plantejat inicialment, tant a nivell personal com a nivell de coneixements.

Línies futures

La versió final d'aquest treball és una primera versió del joc *Boat Attack* el qual m'agradaria publicar en el mercat d'aplicacions en un curt període de temps. Per fer-ho caldrà seguir treballant en les següents funcionalitats de l'aplicació:

- Tal i com es va definir en els objectius del Treball Final de Màster, les icones desenvolupades en aquest treball serviran per una primera fase de prova de concepte, però caldrà redissenyar-les i adaptar-les per pròximes versions de l'aplicació.
- Informar dels resultats parcials de la partida a l'usuari, és a dir, el nombre de vaixells totals enfonsats –juntament amb la puntuació total -, detallant quants vaixells amics i quants enemics s'han enfonsat, així com el nombre total de vaixells que han navegat.
- Mostrar un canó des del qual s'inicialitzi l'animació del dispar al centre de la part inferior de la pantalla. D'aquesta manera, el canó s'anirà movent en funció d'on dispari l'usuari.
- Acabar d'ajustar la velocitat i la trajectòria tant de les bales de canó com dels vaixells, per tal de que l'experiència de l'usuari sigui la millor possible.
- Caldrà treballar en noves funcionalitats de l'aplicació, per exemple, integrar la *SDK* de *Facebook* per tal que els usuaris puguin compartir la seva activitat dins l'aplicació en aquestes xarxes socials, així com puguin reptar als seus amics a jugar. Amb aquesta funcionalitat el que es busca és fer el joc una mica més viralitzador i mirar de portar més usuaris a partir del "*boca a boca*".
- En aquesta primera versió *beta* del joc, com que no s'ha implementat cap servidor, el llistat de rànquing és administrat directament per l'aplicació, la qual cosa no es sincronitzat entre els diferents usuaris. Per tal de publicar aquesta aplicació al mercat, doncs, serà imprescindible implementar un servidor que dugui a terme aquestes tasques d'administració del llistat de rànquing per tal que aquest sigui sincronitzat i igual entre tots els usuaris de l'aplicació.
- Tal i com s'ha definit en apartats anteriors d'aquest treball, altres tasques futures impliquen desenvolupar noves funcionalitats per tal d'afegir productes virtuals i monetització al joc. Així doncs, en el futur caldrà afegir els productes que s'han definit en les bases de l'aplicació realitzades en aquest treball final de màster, així com afegir noves funcionalitats, per exemple, nous tipus de vaixells –tant vaixells amb carregaments de pólvora, com vaixells amics i enemics-.

Bibliografia

[1] https://developer.android.com/about/dashboards/index.html, 28 de febrer de 2015
[2] https://developer.android.com/about/versions/lollipop.html, 28 de febrer de 2015
[3] http://developer.android.com/sdk/index.html, 01 de març de 2015
[4] http://www.elmundo.es/television/2014/02/28/530f9d5f22601df05e8b458c.html, 21 de març de 2015
[5] http://www.movilwe.com/estadisticas-del-consumidor-movil/, 21 de març de 2015
[6] http://www.bigfishgames.com/blog/2014-android-iphone-ipad-tablet-mobile-video-game-stats/, 21 de març de 2015
[7] http://mobilefomo.com/2014/04/mobile-gaming-statistics/, 21 de març de 2015
[8] http://www.bigfishgames.com/blog/2014-global-gaming-stats-whos-playing-what-and-why/, 21 de març de 2015
[9] http://www.businessinsider.com/iphone-v-android-market-share-2014-5, 21 de març de 2015
[10] http://www.xatakamovil.com/mercado/android-vuelve-a-superar-el-90-de-cuota-de-ventas-en-espana-mientras-blackberry-toca-suelo, 21 de març de 2015
[11] http://spin.atomicobject.com/2014/07/18/mobile-phone-statistics/, 21 de març de 2015
[12] http://www.businessinsider.com/android-v-apple-ios-market-share-revenue-income-2014-6, 22 de març de 2015
[13] http://www.mobileindustryreview.com/2014/12/android-revenue-less-than-you-think.html, 22 de març de 2015
[14] http://www.androidauthority.com/q2-2014-app-stats-show-google-play-ios-improve-407106/, 22 de març de 2015
[15] http://partners.gamehouse.com/7-mistakes-app-developers-make-monetizing-apps-part-1, 25 de març de 2015
[16] http://www.androidauthority.com/how-to-monetize-android-app-379638/, 25 de març de 2015
[17] http://www.statista.com/statistics/297024/most-popular-mobile-app-monetization-models/, 25 de març de 2015
[18] http://www.scribd.com/doc/141905110/4-Analisis-DAFO-grupo-2-presentacion#scribd, 26 de març de 2015
[19] http://www.adobe.com/es/products/illustrator.html, 4 d'abril de 2015
[20] http://www.justinmind.com/, 5 d'abril de 2015
[21] http://www.justinmind.com/features, 5 d'abril de 2015
[22] http://developer.android.com/about/versions/android-4.0.3.html, 28 d'abril de 2015
[23] https://developer.android.com/about/dashboards/index.html, 28 d'abril de 2015
[24] https://developer.android.com/reference/android/support/v7/widget/RecyclerView.html, 29 d'abril de 2015
[25] https://developer.android.com/reference/android/support/v7/app/package-summary.html, 29 d'abril de 2015
[26] http://developer.android.com/guide/components/fragments.html, 29 d'abril de 2015
[27] http://developer.android.com/reference/android/content/BroadcastReceiver.html, 3 de maig de 2015
[28] http://developer.android.com/reference/android/os/Handler.html, 3 de maig de 2015
[29] http://es.wikipedia.org/wiki/APK_%28formato%29, 22 de maig de 2015
[30] http://www.androidsis.com/que-es-un-apk-y-como-se-utiliza/, 22 de maig de 2015

Annex 1 – Disseny d'icones de l'aplicació

Figura 7 - Botó d'inici de partida

Figura 8 - Botó d'inici de partida amb estat clicat

Figura 9 - Botó a pantalla d'assoliments

Figura 10 - Botó a pantalla d'assoliments amb estat clicat

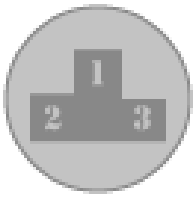


Figura 11 - Botó a pantalla de puntuacions

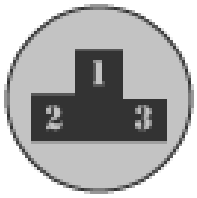


Figura 12 - Botó a pantalla de puntuacions amb estat clicat

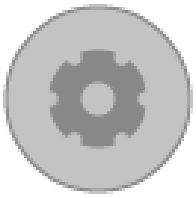


Figura 13 - Botó a pantalla de configuració

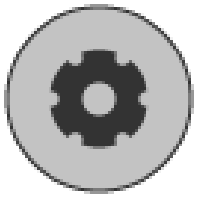


Figura 14 - Botó a pantalla de configuració amb estat clicat

Figura 15 - Botó a pantalla de botiga

Figura 16 - Botó a pantalla de botiga amb estat clicat

Figura 17 - Vaixell pirata

Figura 18 - Vaixell pirata enfonsat

Figura 19 - Vaixell amic

Figura 20 - Vaixell amic enfonsat

Figura 21 - Bomba

Figura 22 - Bomba clicada

Figura 23 - Escut

Figura 24 - Escut clicat

Figura 25 - Rellotge

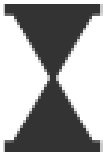


Figura 26 - Rellotge clicat

Figura 27 - Bala de canó

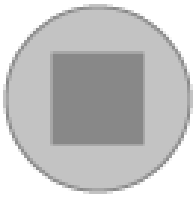


Figura 28 - Botó d'aturar

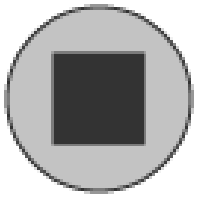


Figura 29 - Botó d'aturar amb estat clicat

Figura 30 - Botó de tornar

Figura 31 - Botó de tornar amb estat clicat

Figura 32 - Moneda virtual

Figura 33 - Moneda virtual amb estat clicat

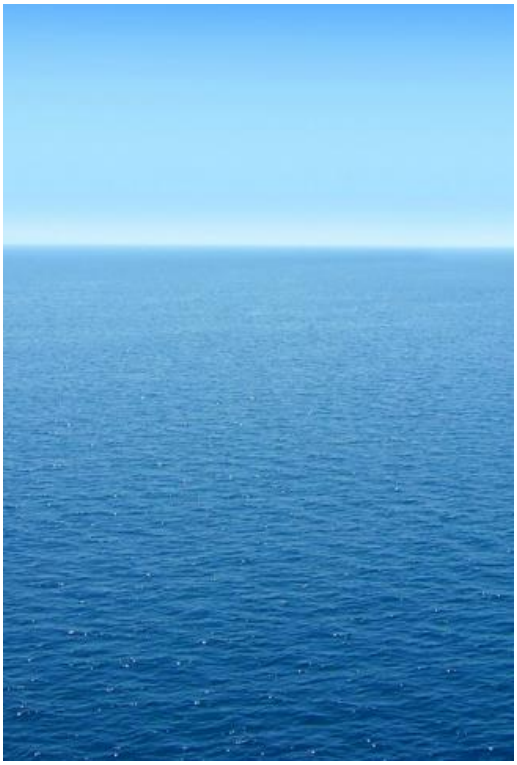


Figura 34 - Fons de pantalla

Annex 2 – Disseny de les pantalles de l'aplicació

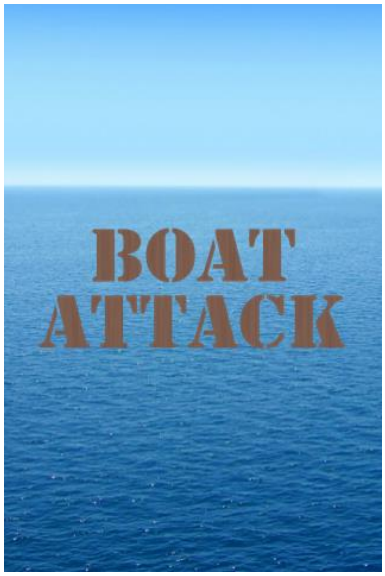


Figura 35 - *Splash*

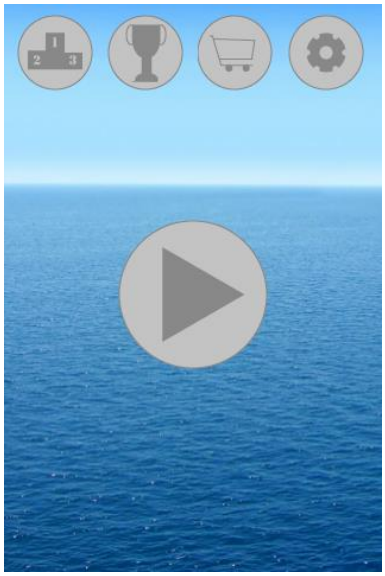


Figura 36 - Menú principal

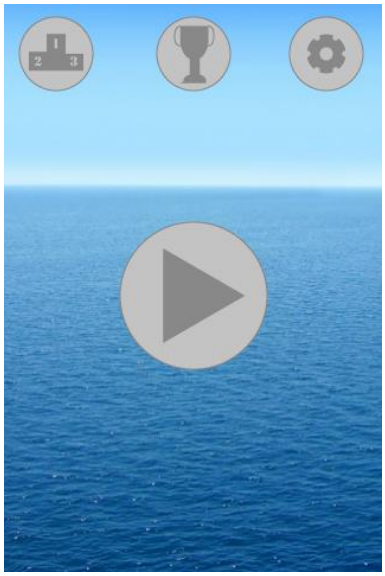


Figura 37 - Menú principal versió beta

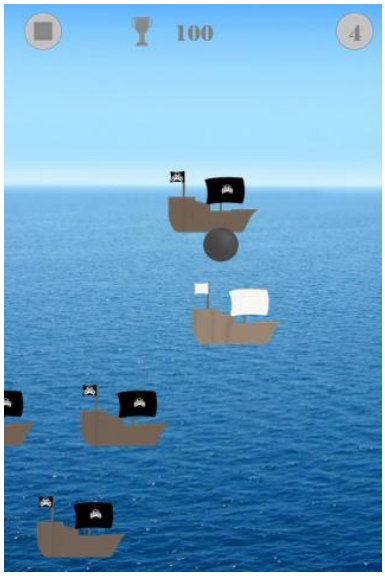


Figura 38 - Pantalla partida versió beta

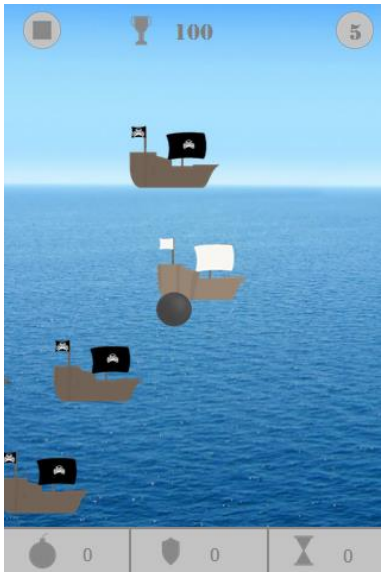


Figura 39 - Pantalla partida

Figura 40 - Puntuació

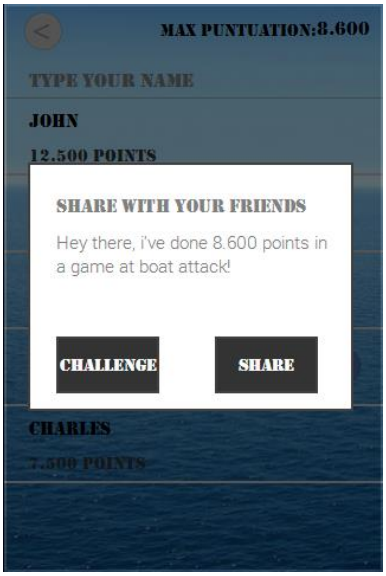


Figura 41 - Diàleg compartir puntuació

Figura 42 - Assoliments

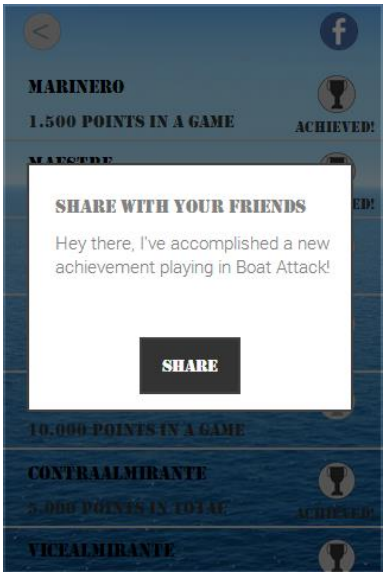


Figura 43 - Diàleg compartir assoliments

Figura 44 - Configuració

Figura 45 - Botiga de productes

Figura 46 - Botiga de monedes virtuals

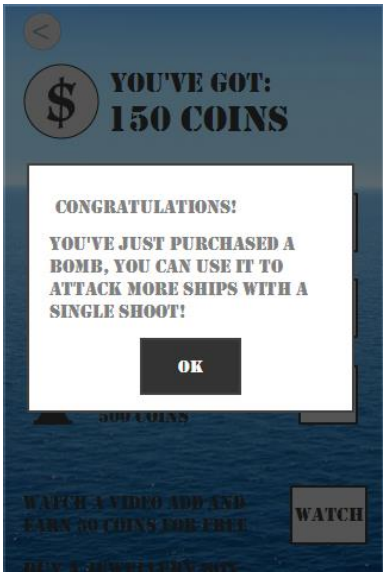


Figura 47 - Diàleg producte virtual comprat

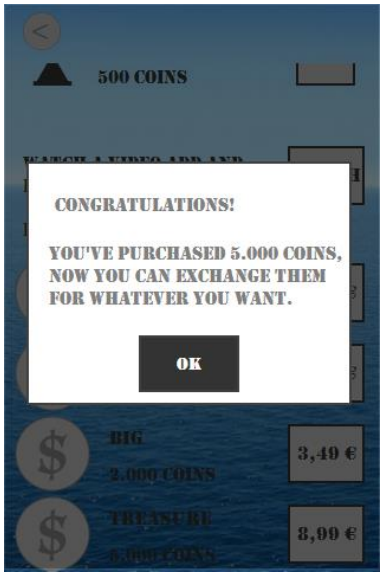


Figura 48 - Diàleg monedes virtual comprades

Annex 3 – MainActivity.java

```
/**
 * Main Activity that contains and manage all the fragments of the application
 */
public class MainActivity extends FragmentActivity {

    // Constants
    private final String ACHIEVEMENTS_FRAGMENT_NAME = "achievements";
    private final String GAME_FRAGMENT_NAME = "game";
    private final String MENU_FRAGMENT_NAME = "menu";
    private final String SETTINGS_FRAGMENT_NAME = "settings";
    private final String SPLASH_FRAGMENT_NAME = "splash";
    private final String RANKING_FRAGMENT_NAME = "ranking";

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);
        if (savedInstanceState == null) {
            changeToSplash();
        }
    }

    /**
     * Replace the current's fragment
     * @param fg The new fragment that should be displayed
     * @param name The new fragment's name
     * @param isAnimated Whether the change of fragment should be done with an animation or not
     */
    private void changeFragment(Fragment fg, String name, boolean isAnimated) {
        Analytics.getInstance().sendScreenEvent(getApplicationContext(), name);

        FragmentTransaction ft = getSupportFragmentManager().beginTransaction();

        if(isAnimated) {
            ft.setCustomAnimations(R.anim.slide_in_right, R.anim.slide_out_left);
        }
        ft.replace(R.id.container, fg, name).commitAllowingStateLoss();

    }

    /**
     * Change the fragment to the Achievements one
     */
    public void changeToAchievements() {
        changeFragment(new FragmentAchievements(), ACHIEVEMENTS_FRAGMENT_NAME, true);
    }

    /**
     * Change the fragment to the Game one
     */
    public void changeToGame() {
        changeFragment(new FragmentGame(), GAME_FRAGMENT_NAME, true);
    }

    /**
     * Change the fragment to the Menu one
     */
    public void changeToMenu() {
        changeFragment(new FragmentMenu(), MENU_FRAGMENT_NAME, true);
    }

    /**
     * Change the fragment to the Settings one
```

```
     */
    public void changeToSettings() {
        changeFragment(new FragmentSettings(), SETTINGS_FRAGMENT_NAME, true);
    }

    /**
     * Change the fragment to the Splash one
     */
    public void changeToSplash() {
        changeFragment(new FragmentSplash(), SPLASH_FRAGMENT_NAME, false);
    }

    /**
     * Change the fragment to the Ranking one
     */
    public void changeToRanking() {
        changeFragment(new FragmentRanking(), RANKING_FRAGMENT_NAME, true);
    }

    /**
     * Listener called each time than an option's button is clicked.
     * @param v The view that has been clicked
     */
    public void optionsButtonClicked(View v) {
        switch (v.getId()) {
            case R.id.bt_achievements:
                changeToAchievements();
                break;
            case R.id.bt_ranking:
                changeToRanking();
                break;
            case R.id.bt_settings:
                changeToSettings();
                break;
            case R.id.bt_play:
                changeToGame();
                break;
            default:
                break;
        }

    }

    /**
     * Each time that the back button is clicked, the fragment should be changed to the menu's
one.
     * @param v The view that has been clicked
     * */
    public void backButtonClicked(View v) {
        if(v.getId() == R.id.bt_stop) {
            Analytics.getInstance().sendEvent(this, R.array.game_stop);
        }
        changeToMenu();
    }
}
```

Annex 4 – main_activity.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:id="@+id/container"
    android:layout_width="match_parent" android:layout_height="match_parent"
    tools:context=".MainActivityActivity" tools:ignore="MergeRootFrame"
    android:background="@drawable/bg_medium"/>
```

Annex 5 – FragmentSplash.java

```
/**
 * FragmentSplash fragment
 */
public class FragmentSplash extends Fragment {
    private ImageView bullet;
    private RelativeLayout rl_splash;
    private ImageView title;

    public FragmentSplash() {
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View rootView = inflater.inflate(R.layout.splash, container, false);

        // Bullet element
        rl_splash = (RelativeLayout) rootView.findViewById(R.id.rl_splash);
        title = (ImageView) rootView.findViewById(R.id.title);
        bullet = (ImageView) rootView.findViewById(R.id.bullet);
        shootBullet();
        return rootView;
    }

    /** Start the bullet's animation after 750 milliseconds */
    private void shootBullet() {
        Handler handler = new Handler();
        handler.postDelayed(new Runnable(){
            @Override
            public void run(){
                moveBulletToScreenCenter();
            }
        }, 750);

    }

    /**
     * Move the bullet view to the center of the screen
     */
    private void moveBulletToScreenCenter()
    {
        DisplayMetrics dm = new DisplayMetrics();
        getActivity().getWindowManager().getDefaultDisplay().getMetrics( dm );
        int statusBarOffset = dm.heightPixels - rl_splash.getMeasuredHeight();

        int originalPos[] = new int[2];
        bullet.getLocationOnScreen( originalPos );

        int xDest = dm.widthPixels/2;
        xDest -= (bullet.getMeasuredWidth()/2);
        int yDest = rl_splash.getMeasuredHeight()/2 - (bullet.getMeasuredHeight()/2) +
statusBarOffset;

        TranslateAnimation anim = new TranslateAnimation( 0, xDest - originalPos[0] , 0, yDest
- originalPos[1] );
        anim.setDuration(500);
        anim.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {
                bullet.setVisibility(ImageView.VISIBLE);
            }
            @Override
            public void onAnimationRepeat(Animation animation) {}
```

```
            @Override
            public void onAnimationEnd(Animation animation) {
                bullet.setVisibility(ImageView.GONE);

                Animation fadeOut = new AlphaAnimation(1, 0);
                fadeOut.setInterpolator(new AccelerateInterpolator()); // and this
                fadeOut.setStartOffset(100);
                fadeOut.setDuration(250);
                fadeOut.setFillAfter(true);
                title.startAnimation(fadeOut);

                // Change to menu
                Handler handler = new Handler();
                handler.postDelayed(new Runnable(){
                    @Override
                    public void run(){
                        MainActivity mainActivity = (MainActivity) getActivity();
                        if(mainActivity != null) {
                            mainActivity.changeToMenu();
                        }
                    }
                }, 750);
            }
        });
        bullet.startAnimation(anim);
    }
}
```

Annex 6 – fr_splash.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivityActivity$PlaceholderFragment"
    android:background="@color/transparent"
    android:id="@+id/rl_splash">

    <ImageView
        android:layout_width="239dp"
        android:layout_height="100dp"
        android:id="@+id/title"
        android:cropToPadding="false"
        android:adjustViewBounds="true"
        android:layout_centerVertical="true"
        android:layout_centerHorizontal="true"
        android:src="@drawable/boat_attack_title" />

    <ImageView
        android:visibility="invisible"
        android:layout_width="32dp"
        android:layout_height="32dp"
        android:id="@+id/bullet"
        android:src="@drawable/bullet"
        android:padding="0dp"
        android:cropToPadding="false"
        android:adjustViewBounds="true"
        android:layout_alignParentBottom="true"
        android:layout_alignParentLeft="true"/>

</RelativeLayout>
```

Annex 7 – FragmentMenu.java

```
/**
 * Menu's fragment
 */
public class FragmentMenu extends Fragment {


    public FragmentMenu() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fr_menu, container, false);
    }


}
```

Annex 8 – fr_menu.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    style="@style/FrameLayoutStyle"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.boatattack.francesc.boatattack.Fragments.FragmentMenu">


    <ImageButton
        style="@style/OptionsButton"
        android:id="@+id/bt_ranking"
        android:layout_gravity="left|top"
        android:src="@drawable/bt_ranking"/>

    <ImageButton
        style="@style/OptionsButton"
        android:id="@+id/bt_achievements"
        android:layout_gravity="center_horizontal|top"
        android:src="@drawable/bt_achievements"/>

    <ImageButton
        style="@style/OptionsButton"
        android:id="@+id/bt_settings"
        android:layout_gravity="right|top"
        android:src="@drawable/bt_settings"/>

    <ImageButton
        style="@style/OptionsButton"
        android:layout_width="128dp"
        android:layout_height="128dp"
        android:id="@+id/bt_play"
        android:layout_gravity="center"
        android:src="@drawable/bt_game"/>
</FrameLayout>
```

Annex 9 – FragmentGame.java

```
/**
 * Game's Fragment
 */
public class FragmentGame extends Fragment {

    private RelativeLayout rl_game;
    private TextView tv_timer;
    private TextView tv_points;
    private EngineGame engineGame;

    public FragmentGame() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View rootView = inflater.inflate(R.layout.fr_game, container, false);
        rl_game = (RelativeLayout) rootView.findViewById(R.id.rl_game);
        rl_game.setOnTouchListener(new View.OnTouchListener() {
            public boolean onTouch(View v, MotionEvent event) {
                if(event.getAction() == MotionEvent.ACTION_DOWN){
                    //hoot on each tap event
                    int x = (int)event.getX();
                    int y = (int)event.getY();

                    ObjectShoot objectShoot = new ObjectShoot(getActivity(), rl_game, x, y);
                    objectShoot.shoot();
                }
                return true;
            }
        });
        tv_timer = (TextView) rootView.findViewById(R.id.tv_timer);
        tv_points = (TextView) rootView.findViewById(R.id.tv_points);


        return rootView;
    }

    @Override
    public void onResume() {
        super.onResume();

        //Register broadcast that will be used to detect when the give me one has been done
        String[] broadcast_actions = new String[]{Information.BROADCAST_ACTION_TIMER,
Information.BROADCAST_ACTION_GAME_OVER, Information.BROADCAST_ACTION_POINTS};
        Utils.registerBroadcastReceiver(getActivity().getApplicationContext(), receiver,
broadcast_actions);
        // Start game's engine
        engineGame = EngineGame.getInstance();
        engineGame.resume(getActivity(), rl_game);
        tv_timer.setText(engineGame.getTime() + "");
        tv_points.setText(engineGame.getPoints()+"");

    }

    @Override
    public void onPause() {
        super.onPause();
        //Unregister broadcast receiver
        Utils.unregisterBroadcastReceiver(getActivity().getApplicationContext(), receiver);
```

```
        if(engineGame != null) {
            engineGame.pause();
        }
    }

    private void gameOver(Bundle extras) {
        tv_timer.setText("0");

        if(extras.getBoolean(Information.BROADCAST_ACTION_KEY_ACHIEVEMENTS_UPDATED)) {
            Dialogs.showAlert(getActivity(), R.string.dialog_achievements_title,
R.string.dialog_achievements_msg, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    MainActivity mainActivity = (MainActivity) getActivity();
                    mainActivity.changeToAchievements();
                }
            });
        } else if (extras.getBoolean(Information.BROADCAST_ACTION_KEY_RANKING_UPDATED)) {
            Dialogs.showAlert(getActivity(), R.string.dialog_ranking_title,
R.string.dialog_ranking_msg, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    MainActivity mainActivity = (MainActivity) getActivity();
                    mainActivity.changeToRanking();
                }
            });
        } else {
            Dialogs.showAlert(getActivity(), R.string.dialog_game_over_title,
R.string.dialog_game_over_msg, new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which) {
                    MainActivity mainActivity = (MainActivity) getActivity();
                    mainActivity.changeToMenu();
                }
            });
        }
    }


    /*******************     BROADCAST RECEIVER     ************************/

    /** Broadcast receiver. This will be used to detect when the give me one has been done */
    private BroadcastReceiver receiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            Bundle extras = intent.getExtras();
            switch(action) {
                case Information.BROADCAST_ACTION_TIMER:
                    int pendingTime =
extras.getInt(Information.BROADCAST_ACTION_KEY_TIMER_TIME, 30);
                    if(pendingTime < 6) {

tv_timer.setTextColor(getResources().getColor(android.R.color.holo_red_light));
                    } else {

tv_timer.setTextColor(getResources().getColor(android.R.color.black));
                    }
                    tv_timer.setText(""+pendingTime);
                    break;
                case Information.BROADCAST_ACTION_POINTS:
                    int points = extras.getInt(Information.BROADCAST_ACTION_KEY_POINTS, 0);
                    tv_points.setText(NumberFormat.getNumberInstance().format(points));
                    break;
                case Information.BROADCAST_ACTION_GAME_OVER:
                    gameOver(extras);
```

```
                break;
            default:
                break;
        }
    }
};

}
```

Annex 10 – fr_game.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    style="@style/FrameLayoutStyle"
    android:layout_height="match_parent"
    tools:context="com.boatattack.francesc.boatattack.Fragments.FragmentGame">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="top|left"
            android:layout_weight="0">

            <ImageButton
                style="@style/StopButton"
                android:layout_alignParentLeft="true"
                android:layout_centerVertical="true"
                android:id="@+id/bt_stop"/>
            <com.boatattack.francesc.boatattack.Objects.CustomTextView
                style="@style/BigText"
                android:id="@+id/tv_points"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_centerInParent="true"
                android:textSize="32sp" />

            <com.boatattack.francesc.boatattack.Objects.CustomTextView
                style="@style/BigText"
                android:id="@+id/tv_timer"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:layout_centerVertical="true"
                android:gravity="center"
                android:background="@drawable/timer"/>

        </RelativeLayout>
        <RelativeLayout
            android:id="@+id/rl_game"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_weight="1">

        </RelativeLayout>
    </LinearLayout>
</FrameLayout>
```

Annex 11 – FragmentRanking.java

```
/**
 * Positions' fragment
 */
public class FragmentRanking extends Fragment {

    private boolean waitingToUpdateUsername = false;
    private ArrayList<ObjectRanking> myDataset = new ArrayList<>();
    private EditText et_username;
    private Handler mHandler = new Handler();
    private Runnable mRunnable = new Runnable() {

        @Override
        public void run() {
            updateUserName();
        }
    };


    public FragmentRanking() {
        // Required empty public constructor
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        Context currentContext = getActivity().getApplicationContext();

        myDataset = EngineRanking.getInstance(getActivity()).getRankingList();
        // Inflate the layout for this fragment
        View rootView = inflater.inflate(R.layout.fr_ranking, container, false);


        TextView tv_ranking_max = (TextView) rootView.findViewById(R.id.tv_ranking_max);
        TextView tv_ranking_total = (TextView) rootView.findViewById(R.id.tv_ranking_total);

        tv_ranking_max.setText(getString(R.string.ranking_max,
DataManagement.getMaxRanking(currentContext)));
        tv_ranking_total.setText(getString(R.string.ranking_total,
DataManagement.getTotalPunctuation(currentContext)));

        et_username = (EditText) rootView.findViewById(R.id.et_username);
        et_username.setText(DataManagement.getUserName(currentContext));
        et_username.addTextChangedListener(new TextWatcher(){
            public void afterTextChanged(Editable s) {
            }
            public void beforeTextChanged(CharSequence s, int start, int count, int after){}
            public void onTextChanged(CharSequence s, int start, int before, int count){
                waitingToUpdateUsername = true;
                mHandler.removeCallbacks(mRunnable);
                mHandler.postDelayed(mRunnable, 1000);
            }
        });


        RecyclerView mRecyclerView = (RecyclerView) rootView.findViewById(R.id.rv_ranking);
        // use this setting to improve performance if you know that changes
        // in content do not change the layout size of the RecyclerView
        mRecyclerView.setHasFixedSize(true);

        // use a linear layout manager
        RecyclerView.LayoutManager mLayoutManager = new LinearLayoutManager(getActivity());
        mRecyclerView.setLayoutManager(mLayoutManager);
```

```
        // specify an adapter (see also next example)
        RecyclerView.Adapter mAdapter = new MyAdapter(myDataset);
        mRecyclerView.setAdapter(mAdapter);

        return rootView;
    }

    @Override
    public void onResume() {
        super.onResume();

        // Show alert dialog if the user hadn't set his username
        String username = DataManagement.getUserName(getActivity());

        if(username == null || username.length() == 0) {
            Dialogs.showAlert(getActivity(), R.string.dialog_username_title,
R.string.dialog_username_msg);
        }

    }

    @Override
    public void onPause() {
        super.onPause();
        // If the username must be updated and it hasn't been updated yet
        if(waitingToUpdateUsername) {
            mHandler.removeCallbacks(mRunnable);
            updateUserName();
        }
    }

    private void updateUserName() {
        Context context = getActivity().getApplicationContext();
        boolean hadInitialized = DataManagement.getUserNameInitialized(context);
        waitingToUpdateUsername = false;
        String username = et_username.getText().toString();
        DataManagement.setUserName(context, username);

        // If the user hadn't stored his name yet, then check whether its max punctuation must
be added in the ranking list.
        if(!hadInitialized) {
            // Check whether the max punctuation must be added in rankings
            int maxRanking = DataManagement.getMaxRanking(context);
            if(maxRanking != 0) {
                EngineRanking.getInstance(context).updateRanking(context, maxRanking);
            }
        }

    }

    public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {
        private ArrayList<ObjectRanking> mDataset;

        // Provide a reference to the views for each data item
        // Complex data items may need more than one view per item, and
        // you provide access to all the views for a data item in a view holder
        public class ViewHolder extends RecyclerView.ViewHolder {
            // each data item is just a string in this case
            public View mView;
            public ViewHolder(View v) {
                super(v);
                mView = v;
            }
        }

        // Provide a suitable constructor (depends on the kind of dataset)
```

```
        public MyAdapter(ArrayList<ObjectRanking> myDataset) {
            mDataset = myDataset;
        }

        // Create new views (invoked by the layout manager)
        @Override
        public MyAdapter.ViewHolder onCreateViewHolder(ViewGroup parent,
                                                       int viewType) {
            // create a new view
            View v = LayoutInflater.from(parent.getContext())
                    .inflate(R.layout.v_ranking, parent, false);

            return new ViewHolder(v);
        }

        // Replace the contents of a view (invoked by the layout manager)
        @Override
        public void onBindViewHolder(ViewHolder holder, int position) {
            // - get element from your dataset at this position
            // - replace the contents of the view with that element
            Context currentContext = getActivity().getApplicationContext();
            ObjectRanking ranking = mDataset.get(position);

            TextView mTitle = (TextView) holder.mView.findViewById(R.id.tv_ranking_name);
            mTitle.setText(ranking.getName());
            TextView mDescription = (TextView)
holder.mView.findViewById(R.id.tv_ranking_description);
            mDescription.setText(ranking.getDescription(currentContext));
        }

        // Return the size of your dataset (invoked by the layout manager)
        @Override
        public int getItemCount() {
            return mDataset.size();
        }
    }


}
```

Annex 12 – fr_ranking.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    style="@style/FrameLayoutStyle"
    android:layout_height="match_parent"
    tools:context="com.boatattack.francesc.boatattack.Fragments.FragmentRanking">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="0"
            android:paddingLeft="5dp"
            android:paddingRight="5dp"
            android:layout_gravity="top">
            <ImageButton
                style="@style/BackButton"
                android:layout_centerVertical="true"
                android:layout_alignParentLeft="true"
                android:id="@+id/bt_back"/>

            <LinearLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:orientation="vertical"
                android:layout_centerVertical="true"
                android:layout_alignParentRight="true">
                <com.boatattack.francesc.boatattack.Objects.CustomTextView
                    style="@style/MediumText"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="right"
                    android:id="@+id/tv_ranking_max"/>
                <com.boatattack.francesc.boatattack.Objects.CustomTextView
                    style="@style/MediumText"
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="right"
                    android:id="@+id/tv_ranking_total"/>
            </LinearLayout>

        </RelativeLayout>

        <com.boatattack.francesc.boatattack.Objects.CustomEditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:inputType="textPersonName"
            android:text="Name"
            android:ems="10"
            android:hint="@string/username_palceholder"
            android:id="@+id/et_username"
            android:layout_gravity="center_horizontal"
            android:layout_weight="0" />

        <android.support.v7.widget.RecyclerView
            android:id="@+id/rv_ranking"
            android:scrollbars="vertical"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1" />
```

```
    </LinearLayout>

</FrameLayout>
```

Annex 13 – v_ranking.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="match_parent"
    android:layout_height="match_parent" android:layout_margin="5dp"
android:layout_marginTop="10dp" android:layout_marginBottom="10dp">

    <com.boatattack.francesc.boatattack.Objects.CustomTextView
        style="@style/BigText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/tv_ranking_name" />

    <com.boatattack.francesc.boatattack.Objects.CustomTextView
        style="@style/SmallText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dp"
        android:id="@+id/tv_ranking_description"/>

</LinearLayout>
```

Annex 14 – FragmentAchievement.java

```
/**
 * Achievements fragment
 */
public class FragmentAchievements extends Fragment {
    private RecyclerView mRecyclerView;
    private RecyclerView.Adapter mAdapter;
    private RecyclerView.LayoutManager mLayoutManager;

    private ArrayList<ObjectAchievement> myDataset = new ArrayList<>();


    public FragmentAchievements() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {


        myDataset = EngineAchievements.getInstance(getActivity()).getAchievementsList();

        // Inflate the layout for this fragment
        View rootView = inflater.inflate(R.layout.fr_achievements, container, false);

        mRecyclerView = (RecyclerView) rootView.findViewById(R.id.rv_achievements);
        // use this setting to improve performance if you know that changes
        // in content do not change the layout size of the RecyclerView
        mRecyclerView.setHasFixedSize(true);

        // use a linear layout manager
        mLayoutManager = new LinearLayoutManager(getActivity());
        mRecyclerView.setLayoutManager(mLayoutManager);

        // specify an adapter (see also next example)
        mAdapter = new MyAdapter(myDataset);
        mRecyclerView.setAdapter(mAdapter);

        return rootView;
    }



    public class MyAdapter extends RecyclerView.Adapter<MyAdapter.ViewHolder> {
        private ArrayList<ObjectAchievement> mDataset;

        // Provide a reference to the views for each data item
        // Complex data items may need more than one view per item, and
        // you provide access to all the views for a data item in a view holder
        public class ViewHolder extends RecyclerView.ViewHolder {
            // each data item is just a string in this case
            public View mView;
            public ViewHolder(View v) {
                super(v);
                mView = v;
            }
        }

        // Provide a suitable constructor (depends on the kind of dataset)
        public MyAdapter(ArrayList<ObjectAchievement> myDataset) {
            mDataset = myDataset;
        }
```

```
        // Create new views (invoked by the layout manager)
        @Override
        public MyAdapter.ViewHolder onCreateViewHolder(ViewGroup parent,
                                                       int viewType) {
            // create a new view
            View v = LayoutInflater.from(parent.getContext())
                    .inflate(R.layout.v_achievement, parent, false);

            return new ViewHolder(v);
        }

        // Replace the contents of a view (invoked by the layout manager)
        @Override
        public void onBindViewHolder(ViewHolder holder, int position) {
            // - get element from your dataset at this position
            // - replace the contents of the view with that element
            Context currentContext = getActivity().getApplicationContext();
            ObjectAchievement achievement = mDataset.get(position);

            TextView mTitle = (TextView) holder.mView.findViewById(R.id.tv_achievement_title);
            mTitle.setText(achievement.getTitle(currentContext));
            TextView mDescription = (TextView)
holder.mView.findViewById(R.id.tv_achievement_description);
            mDescription.setText(achievement.getDescription(currentContext));
            TextView tv_achievement_label = (TextView)
holder.mView.findViewById(R.id.tv_achievement_label);
            ImageView mLogo = (ImageView) holder.mView.findViewById(R.id.iv_achievement_logo);
            if(achievement.hasAchieved()) {
                mLogo.setImageResource(R.drawable.trophy_black);
                tv_achievement_label.setVisibility(TextView.VISIBLE);
            } else {
                mLogo.setImageResource(R.drawable.trophy);
                tv_achievement_label.setVisibility(TextView.INVISIBLE);
            }

        }

        // Return the size of your dataset (invoked by the layout manager)
        @Override
        public int getItemCount() {
            return mDataset.size();
        }
    }


}
```

Annex 15 – fr_achievements.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    style="@style/FrameLayoutStyle"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.boatattack.francesc.boatattack.Fragments.Achievements">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <ImageButton
            style="@style/BackButton"
            android:layout_weight="0"
            android:id="@+id/bt_back"/>

        <android.support.v7.widget.RecyclerView
            android:id="@+id/rv_achievements"
            android:scrollbars="vertical"
            android:layout_width="match_parent"
            android:layout_height="0dp"
            android:layout_weight="1" />
    </LinearLayout>

</FrameLayout>
```

Annex 16 – v_achievement.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:layout_width="wrap_content"
    android:layout_height="wrap_content" android:layout_margin="5dp"
android:layout_marginTop="10dp" android:layout_marginBottom="10dp">

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_alignParentLeft="true"
        android:layout_centerVertical="true">
        <com.boatattack.francesc.boatattack.Objects.CustomTextView
            style="@style/BigText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:id="@+id/tv_achievement_title" />

        <com.boatattack.francesc.boatattack.Objects.CustomTextView
            style="@style/SmallText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="15dp"
            android:id="@+id/tv_achievement_description"/>
    </LinearLayout>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:gravity="center">

        <ImageView
            style="@style/ImageViewStyle"
            android:id="@+id/iv_achievement_logo"
            android:layout_width="32dp"
            android:layout_height="32dp"
            android:src="@drawable/trophy"/>
        <com.boatattack.francesc.boatattack.Objects.CustomTextView
            style="@style/SmallText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/achieved_label"
            android:layout_marginTop="5dp"
            android:id="@+id/tv_achievement_label" />
    </LinearLayout>


</RelativeLayout>
```

Annex 17 – FragmentSettings.java

```
/**
 * Settings' fragment
 */
public class FragmentSettings extends Fragment {


    public FragmentSettings() {
        // Required empty public constructor
    }


    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        View rootView = inflater.inflate(R.layout.fr_settings, container, false);

        CheckBox cb_sounds = (CheckBox) rootView.findViewById(R.id.cb_sounds);
        cb_sounds.setChecked(DataManagement.getIsSoundEnabled(getActivity()));
        cb_sounds.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

Analytics.getInstance().sendEvent(getActivity().getApplicationContext(),R.array.sounds,
isChecked+"");
                DataManagement.setIsSoundsEnabled(getActivity().getApplicationContext(),
isChecked);
            }
        });
        CheckBox cb_vibration = (CheckBox) rootView.findViewById(R.id.cb_vibration);
        cb_vibration.setChecked(DataManagement.getIsVibrationEnabled(getActivity()));
        cb_vibration.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
            @Override
            public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
                Analytics.getInstance().sendEvent(getActivity().getApplicationContext(),
R.array.vibration, isChecked+"");
                DataManagement.setIsVibrationEnabled(getActivity().getApplicationContext(),
isChecked);
            }
        });

        return rootView;
    }


}
```

Annex 18 – fr_settings.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    style="@style/FrameLayoutStyle"
    tools:context="com.boatattack.francesc.boatattack.Fragments.FragmentSettings">

    <ImageButton
        style="@style/BackButton"
        android:id="@+id/bt_back"/>

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:orientation="vertical">

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:layout_marginBottom="10dp">

            <com.boatattack.francesc.boatattack.Objects.CustomTextView
                style="@style/MediumText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentLeft="true"
                android:layout_centerVertical="true"
                android:text="@string/settings_enable_sound"/>

            <CheckBox
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:layout_centerVertical="true"
                android:id="@+id/cb_sounds"
                android:layout_gravity="right|center_vertical" />

        </RelativeLayout>

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="horizontal">

            <com.boatattack.francesc.boatattack.Objects.CustomTextView
                style="@style/MediumText"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentLeft="true"
                android:layout_centerVertical="true"
                android:text="@string/settings_enable_vibration"/>

            <CheckBox
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentRight="true"
                android:layout_centerVertical="true"
                android:id="@+id/cb_vibration"
                android:layout_gravity="right|center_vertical" />

        </RelativeLayout>
```

```
    </LinearLayout>


</FrameLayout>
```

Annex 19 – Dialogs.java

```
/**
 * Dialog' method
 */
public class Dialogs {

    /** Creates a generic alert dialog with a single positive button.
     * @param activity The current activity
     * @param titleResource The resource of the title text
     * @param msgResource The resource of the message text
     * @param listener The ok button's listener
     * @return AlertDialog The generic alert dialog */
    private static AlertDialog createAlert(Activity activity, int titleResource, int
msgResource, DialogInterface.OnClickListener listener) {
        AlertDialog.Builder builder;
        if(android.os.Build.VERSION.SDK_INT >= Build.VERSION_CODES.HONEYCOMB){
            builder = new AlertDialog.Builder(activity, AlertDialog.THEME_HOLO_LIGHT);
        }
        else{
            builder = new AlertDialog.Builder(activity);
        }
        builder.setTitle(titleResource)
                .setMessage(msgResource)
                .setCancelable(false)
                .setPositiveButton(R.string.dialog_error_positive_btn, listener);
        // Create the AlertDialog object
        return builder.create();
    }

    /** Creates a generic alert dialog with a single positive button.
     * @param activity The current activity
     * @param titleResource The resource of the title text
     * @param msgResource The resource of the message text
     * @return AlertDialog The generic alert dialog */
    private static AlertDialog createAlert(Activity activity, int titleResource, int
msgResource) {
        return createAlert(activity, titleResource, msgResource, new
DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int id) {
                dialog.dismiss();
            }
        });
    }

    /** Shows a generic alert dialog with a single positive button.
     * @param activity The current activity
     * @param titleResource The resource of the title text
     * @param msgResource The resource of the message text
     * @param listener The ok button's listener */
    public static void showAlert(Activity activity, int titleResource, int msgResource,
DialogInterface.OnClickListener listener) {
        AlertDialog alert = createAlert(activity, titleResource, msgResource, listener);
        alert.show();
    }

    /** Shows a generic alert dialog with a single positive button.
     * @param activity The current activity
     * @param titleResource The resource of the title text
     * @param msgResource The resource of the message text */
    public static void showAlert(Activity activity, int titleResource, int msgResource) {
        AlertDialog alert = createAlert(activity, titleResource, msgResource);
        alert.show();
    }
}
```

Annex 20 – Utils.java

```
/**
 * Utils features
 */
public class Utils {

    /** Register the broadcast receiver for a given activity
     * @param context The context of the activity
     * @param receiver The broadcast receiver from that activity
     * @param broadcast_actions A String[] of the broadcast actions that the receiver must take
int account*/
    public static void registerBroadcastReceiver(Context context,BroadcastReceiver receiver,
String[] broadcast_actions){
        if(context != null) {
            IntentFilter filter = new IntentFilter();
            for(int i=0; i<broadcast_actions.length; i++){
                filter.addAction(broadcast_actions[i]);
            }
            context.registerReceiver(receiver, filter);
        }
    }
    /** Send a broacast message
     * @param broadcast_action The action that will be sent on the broadcast message*/
    public static void sendBroadcastMessage(Context context, String broadcast_action) {
        sendBroadcastMessage(context, broadcast_action, null, null);
    }

    /** Send a broacast message
     * @param broadcast_action The action that will be sent on the broadcast message
     * @param extra_title The extra title
     * @param extra_value The extra value*/
    public static void sendBroadcastMessage(Context context, String broadcast_action, String
extra_title, Object extra_value){
        if(context != null){
            Intent broadcast = new Intent();
            if(extra_title != null && extra_value != null){
                String exra_value_class = extra_value.getClass().toString().toLowerCase();
                if(exra_value_class.contains("string")){
                    broadcast.putExtra(extra_title, (String) extra_value);
                }
                else if(exra_value_class.contains("boolean")){
                    broadcast.putExtra(extra_title, (Boolean) extra_value);
                }
                else if(exra_value_class.contains("int")){
                    broadcast.putExtra(extra_title, (Integer) extra_value);
                } else if(exra_value_class.contains("bundle")) {
                    broadcast.putExtras((Bundle) extra_value);
                }
            }
            broadcast.setAction(broadcast_action);
            context.sendBroadcast(broadcast);
        }
    }


    /** Unregister the broadcast receiver for a given activity
     * @param context The context of the activity
     * @param receiver The broadcast receiver we want to unregister from that activity */
    public static void unregisterBroadcastReceiver(Context context,BroadcastReceiver
receiver){
        if(context != null && receiver != null){
            try {
                context.unregisterReceiver(receiver);
            }catch (IllegalArgumentException e) {}
```

```
        }
    }

    /**
     * If the user has enabled vibration, then vibrate for 300 milliseconds
     * @param context The context of the activity
     */
    public static void vibrate(Context context) {
        if(DataManagement.getIsVibrationEnabled(context)) {
            // Get instance of Vibrator from current Context
            Vibrator v = (Vibrator) context.getSystemService(Context.VIBRATOR_SERVICE);

            // Vibrate for 300 milliseconds
            v.vibrate(300);

        }
    }
}
```

Annex 21 – Information.java

```
/**
 * Class with global constants.
 */
public class Information {

    // Shared preferences
    public static final String SHARED_PREFERENCE_PREFS_ACHIEVEMENT = "boatAttackAchievements";
    public static final String SHARED_PREFERENCE_PREFS_APP = "boatAttackApp";
    public static final String SHARED_PREFERENCE_PREFS_RANKING = "boatAttackRanking";
    public static final String SHARED_PREFERENCE_ACHIEVEMENT_BASE = "ba_achievement_";
    public static final String SHARED_PREFERENCE_RANKING_BASE = "ba_ranking_";
    // Achievements in a game
    public static final int ACHIEVEMENT_SAILOR_ID = 1;
    public static final int ACHIEVEMENT_GRAND_MASTER_ID = 2;
    public static final int ACHIEVEMENT_BOATSWAIN_ID = 3;
    public static final int ACHIEVEMENT_OFFICIAL_ID = 4;
    public static final int ACHIEVEMENT_SKIPPER_ID = 5;
    // Achievements in app
    public static final int ACHIEVEMENT_CABIN_BOY_ID = 6;
    public static final int ACHIEVEMENT_REAR_ADMIRAL_ID = 7;
    public static final int ACHIEVEMENT_VICE_ADMIRAL_ID = 8;
    public static final int ACHIEVEMENT_ADMIRAL_ID = 9;
    public static final int ACHIEVEMENT_CAPTAIN_ID = 10;
    //User
    public static final String SHARED_PREFERENCE_USER_RANKING_MAX = "ba_user_ranking_max";
    public static final String SHARED_PREFERENCE_USER_RANKING_TOTAL = "ba_user_ranking_total";
    public static final String SHARED_PREFERENCE_USERNAME = "ba_username";
    public static final String SHARED_PREFERENCE_USERNAME_INITIALIZED =
"ba_username_initialized";
    public static final String SHARED_PREFERENCE_SETTINGS_SOUNDS = "ba_settings_sounds";
    public static final String SHARED_PREFERENCE_SETTINGS_VIBRATION = "ba_settings_vibration";
    // Images types
    public static final String TAG_BULLET = "bullet";
    public static final String TAG_SHIP = "ship";

    // Broadcast receivers
    public static final String BROADCAST_ACTION_GAME_OVER = "ba_broadcast_action_game_over";
    public static final String BROADCAST_ACTION_SHIP_SUNKEN=
"ba_broadcast_action_ship_sunken";
    public static final String BROADCAST_ACTION_POINTS = "ba_broadcast_action_points";
    public static final String BROADCAST_ACTION_TIMER = "ba_broadcast_action_timer";
    public static final String BROADCAST_ACTION_KEY_ACHIEVEMENTS_UPDATED =
"achievements_updated";
    public static final String BROADCAST_ACTION_KEY_GAME_OVER_BUNDLE = "game_over";
    public static final String BROADCAST_ACTION_KEY_POINTS = "points";
    public static final String BROADCAST_ACTION_KEY_RANKING_UPDATED = "ranking_updated";
    public static final String BROADCAST_ACTION_KEY_SHIP_TYPE= "ship_type";
    public static final String BROADCAST_ACTION_KEY_TIMER_TIME = "time";
}
```

Annex 22 – ComparatorRanking.java

```
/**
 * Compare ranking in order to order the list
 */
public class ComparatorRanking implements Comparator<ObjectRanking> {
    @Override
    public int compare(ObjectRanking o1, ObjectRanking o2) {
        return ((Integer)o1.getPosition()).compareTo(o2.getPosition());
    }
}
```

Annex 23 – BoatAttack.java

```
/**
 * Boat Attack application's engine
 */
public class BoatAttack extends Application {

    @Override
    public void onCreate() {
        super.onCreate();

        // Init singletons
        EngineAchievements.getInstance(getApplicationContext());
        EngineRanking.getInstance(getApplicationContext());
        Analytics.getInstance().enableAdvertisingIdCollection(getApplicationContext());
    }

    /**
     * Get a string identifier by text
     * @param context Current application's context
     * @param name String resource's name
     * @return  Return's the resource's id
     */
    public int getStringIdentifier(Context context, String name) {
        return context.getResources().getIdentifier(name, "string",
context.getPackageName());
    }
}
```

Annex 24 – DataManagement.java

```
/**
 * Class with static data methods.
 */
public class DataManagement {


    /**
     * Get the user's sounds preferences
     * @param context The current application context
     * @return boolean True whether the user has sounds enabled
     */
    public static boolean getIsSoundEnabled(Context context) {
        SharedPreferences mPrefs =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE);
        return mPrefs.getBoolean(Information.SHARED_PREFERENCE_SETTINGS_SOUNDS, true);
    }
    /**
     * Get the user's vibration preferences
     * @param context The current application context
     * @return boolean True whether the user has vibration enabled
     */
    public static boolean getIsVibrationEnabled(Context context) {
        SharedPreferences mPrefs =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE);
        return mPrefs.getBoolean(Information.SHARED_PREFERENCE_SETTINGS_VIBRATION, true);
    }
    /**
     * Get the user's max ranking.
     * @param context The current application context
     * @return int User's max ranking.
     */
    public static int getMaxRanking(Context context) {
        SharedPreferences mPrefs =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE);
        return mPrefs.getInt(Information.SHARED_PREFERENCE_USER_RANKING_MAX, 0);
    }

    /**
     * Get the given position object
     * @param context The current application context
     * @param position Position
     */
    public static String getPosition(Context context, int position) {
        SharedPreferences mPrefs =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_RANKING,
Context.MODE_PRIVATE);
        return mPrefs.getString(Information.SHARED_PREFERENCE_RANKING_BASE + position, null);
    }

    /**
     * Get the user's total punctuation
     * @return int User's total punctuation
     */
    public static int getTotalPunctuation(Context context) {
        SharedPreferences mPrefs =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE);
        return mPrefs.getInt(Information.SHARED_PREFERENCE_USER_RANKING_TOTAL, 0);
    }
```

```
    /**
     * Get the user's name
     * @param context Context The current application's context
     * @return String User's name
     */
    public static String getUserName(Context context) {
        SharedPreferences mPrefs =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE);
        return mPrefs.getString(Information.SHARED_PREFERENCE_USERNAME, "");
    }

    /**
     * Check whether the user name has already been initialized.
     * @param context Context The current application's context
     * @return boolean True whether the username has already been initialized some time.
     */
    public static boolean getUserNameInitialized(Context context) {
        SharedPreferences mPrefs =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE);
        return mPrefs.getBoolean(Information.SHARED_PREFERENCE_USERNAME_INITIALIZED, false);
    }

    /**
     * Set the user's sounds preferences.
     * @param context Context The current application's context
     * @param enabled boolean Whether the sound is enabled
     */
    public static void setIsSoundsEnabled(Context context, boolean enabled) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE).edit();
        editor.putBoolean(Information.SHARED_PREFERENCE_SETTINGS_SOUNDS, enabled);
        editor.commit();
    }

    /**
     * Set the user's vibration preferences.
     * @param context Context The current application's context
     * @param enabled boolean Whether the vibration is enabled
     */
    public static void setIsVibrationEnabled(Context context, boolean enabled) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE).edit();
        editor.putBoolean(Information.SHARED_PREFERENCE_SETTINGS_VIBRATION, enabled);
        editor.commit();
    }

    /**
     * Set the user's max ranking.
     * @param max int User's max rating
     */
    public static void setMaxRanking(Context context, int max) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE).edit();
        editor.putInt(Information.SHARED_PREFERENCE_USER_RANKING_MAX, max);
        editor.commit();
    }

    /**
     * Set the user's total punctuation.
     * @param punctuation int User's total punctuation
     */
```

```
    public static void setTotalPunctuation(Context context, int punctuation) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE).edit();
        editor.putInt(Information.SHARED_PREFERENCE_USER_RANKING_TOTAL,
(DataManagement.getTotalPunctuation(context) + punctuation));
        editor.commit();
    }

    /**
     * Set the user's total punctuation.
     * @param context Context The current application's context
     * @param name String User's name
     */
    public static void setUserName(Context context, String name) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE).edit();
        editor.putString(Information.SHARED_PREFERENCE_USERNAME, name);
        editor.commit();
        DataManagement.setUserNameInitialized(context);
    }

    /**
     * Set that the user's name has been initialized.
     * @param context Context The current application's context
     */
    public static void setUserNameInitialized(Context context) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_APP,
Context.MODE_PRIVATE).edit();
        editor.putBoolean(Information.SHARED_PREFERENCE_USERNAME_INITIALIZED, true);
        editor.commit();

    }

    /**
     * Store the object into shared preference
     * @param context The current application context
     * @param position Position
     * @param json Object's json as string
     */
    public static void setPosition(Context context, int position, String json) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_RANKING,
Context.MODE_PRIVATE).edit();
        editor.putString(Information.SHARED_PREFERENCE_RANKING_BASE + position, json);
        editor.commit();
    }

    /**
     * Clear the positions
     * @param context The current application context
     */
    public static void clearPositions(Context context) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_RANKING,
Context.MODE_PRIVATE).edit();
        editor.clear();
        editor.commit();
    }
}
```

Annex 25 – Analytics.java

```
/**
 * Google Analytics engine
 */
public class Analytics {

    private static Analytics mInstance;
    private Tracker mTracker = null;

    public Analytics() {
    }

    public static Analytics getInstance() {
        if(mInstance == null) {
            mInstance = new Analytics();
        }
        return mInstance;
    }

    /**
     * Get the analytics tracker
     * @param context The current application's context
     * @return Analytic's trigger
     */
    private Tracker getTracker(Context context) {
        if (mTracker == null) {
            GoogleAnalytics analytics = GoogleAnalytics.getInstance(context);
            mTracker = analytics.newTracker(R.xml.global_tracker);

        }
        return mTracker;
    }

    /**
     * Set enable advertising id collection to true
     * @param context The current application's context
     */
    public void enableAdvertisingIdCollection(Context context) {
        // Tracker
        Tracker t = getTracker(context);
        // Enable Advertising Features.
        t.enableAdvertisingIdCollection(true);
    }

    /**
     * Send screen event
     * @param context The current application's context
     * @param screenName Screen's name
     */
    public void sendScreenEvent(Context context, String screenName) {
        // Get tracker.
        Tracker t = getTracker(context);

        // Set screen name.
        t.setScreenName(screenName);

        // Send a screen view.
        if(screenName == "splash") {
            t.send(new HitBuilders.ScreenViewBuilder()
                    .setNewSession()
                    .build());
        } else {
            t.send(new HitBuilders.ScreenViewBuilder().build());
        }
```

```
    }

    public void sendEvent(Context context, int keyId) {
        sendEvent(context, keyId, null);
    }

    public void sendEvent(Context context, int keyId, String label) {
        String[] keys = context.getResources().getStringArray(keyId);
        // Get tracker.
        Tracker t = getTracker(context);

        // Build and send an Event.
        t.send(new HitBuilders.EventBuilder()
                .setCategory(keys[0])
                .setAction(keys[1])
                .setLabel(label != null? label:"")
                .build());
    }

}
```

Annex 26 – EngineGame.java

```
/**
 * Game's engine as a singleton. Start showing ships, it has the game counter and timer.
 */
public class EngineGame {

    private final int PIRATE_POINTS = 75; // Points that the user wins when the user sunks a
pirate
    private final int PIRATE_SECONDS = 5; // Seconds that the user wins when the user sunks a
pirate
    private final int SHIP_POINTS = 50; // Points that the user loses when the user sunks a ship
    private final int SHIP_SECONDS = 10; // Seconds that the user loses when the user sunks a
ship


    private static EngineGame mInstance;
    private Context context = null;
    private RelativeLayout rl_game;
    // Shipping' engine
    private Handler mShippingHandler = new Handler();
    private Runnable mShippingRunnable = null;
    private final int MINIMUM_SHIPS_INTERVAL = 300;
    private final int SHIPS_INTERVAL = 500;
    // Timer
    private int timerTime = 40; // It starts with 30 seconds
    private Handler mTimerHandler = new Handler();
    private Runnable mTimerRunnable = new Runnable() {
        @Override
        public void run() {
            timerTime--;
            mTimerHandler.postDelayed(mTimerRunnable, 995);
            updateTimer();
        }
    };
    // Points
    private int points = 0;

    private EngineGame() {
    }

    /**
     * Create the singleton's instance that will be used on the application.
     * @return EngineGame Game's instance.
     */
    public static EngineGame getInstance() {
        if (mInstance == null)  {
            // Create the instance
            mInstance = new EngineGame();
        }

        return mInstance;
    }

    /**
     * Iniatilize game's variables
     * @param context The current application's context
     * @param rl_game Relative layout where the game is played
     */
    private void initGame(Context context, RelativeLayout rl_game) {
        // Set up game's context
        this.context = context;
        this.rl_game = rl_game;
        this.timerTime = 40;
        this.points = 0;
```

```
    }

    public void resume(Context cntx, RelativeLayout rl_gme) {
        initGame(cntx, rl_gme);
        //Register broadcast that will be used to detect when the give me one has been done
        String[] broadcast_actions = new String[]{Information.BROADCAST_ACTION_SHIP_SUNKEN};
        Utils.registerBroadcastReceiver(context, receiver, broadcast_actions);
        // Set up shipping handler in order to show ships
        mShippingRunnable = new Runnable() {
            @Override
            public void run() {
                ObjectShip objectShip = new ObjectShip(context, rl_game);
                objectShip.sail();
                mShippingHandler.postDelayed(mShippingRunnable, getShipsInterval());
            }
        };
        // Resume the timer
        mTimerHandler.postDelayed(mTimerRunnable, 995);
        // Resume the ships
        mShippingHandler.postDelayed(mShippingRunnable, getShipsInterval());
        // Send start event
        Analytics.getInstance().sendEvent(cntx, R.array.game_start);
    }

    public void pause() {
        //Unregister broadcast receiver
        Utils.unregisterBroadcastReceiver(context, receiver);
        // Stop the timer
        mTimerHandler.removeCallbacks(mTimerRunnable);
        // Stop the ships
        if(mShippingRunnable != null) {
            mShippingHandler.removeCallbacks(mShippingRunnable);
            mShippingRunnable = null;
        }
    }

    private void gameOver() {
        pause();
        boolean rankingUpdated = EngineRanking.getInstance(context).updateRanking(context,
points);
        boolean achievementsUpdated =
EngineAchievements.getInstance(context).updateAchievements(context, points);

        Bundle bundle = new Bundle();
        bundle.putBoolean(Information.BROADCAST_ACTION_KEY_RANKING_UPDATED,
rankingUpdated);
        bundle.putBoolean(Information.BROADCAST_ACTION_KEY_ACHIEVEMENTS_UPDATED,
achievementsUpdated);

        Utils.sendBroadcastMessage(context, Information.BROADCAST_ACTION_GAME_OVER,
Information.BROADCAST_ACTION_KEY_GAME_OVER_BUNDLE, bundle);
    }

    /**
     * Get the time to the next ship
     * @return int Milliseconds until next ship is shown
     */
    private int getShipsInterval() {
        Random random = new Random();
        return MINIMUM_SHIPS_INTERVAL + random.nextInt(SHIPS_INTERVAL);
    }

    /**
     * Get the current punctuation
     * @return int Points
     */
```

```
    public int getPoints() {
        return points;
    }

    /**
     * Get the current timer's time
     * @return int Time 'til the game is over
     */
    public int getTime() {
        return timerTime;
    }

    /**
     * Triggers the broadcast action required in order to update the timer's view.
     */
    private void updatePoints() {
        Utils.sendBroadcastMessage(context, Information.BROADCAST_ACTION_POINTS,
Information.BROADCAST_ACTION_KEY_POINTS, points);
    }

    /**
     * Triggers the broadcast action required in order to update the timer's view.
     */
    private void updateTimer() {
        if(timerTime >= 0) {
            Utils.sendBroadcastMessage(context, Information.BROADCAST_ACTION_TIMER,
Information.BROADCAST_ACTION_KEY_TIMER_TIME, timerTime);
        } else {
            gameOver();
        }
    }


    /*******************     BROADCAST RECEIVER     *************************/

    /** Broadcast receiver. This will be used to detect when the give me one has been done */
    private BroadcastReceiver receiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {
            String action = intent.getAction();
            Bundle extras = intent.getExtras();
            if(action.equals(Information.BROADCAST_ACTION_SHIP_SUNKEN)){
                int shipType = extras.getInt(Information.BROADCAST_ACTION_KEY_SHIP_TYPE, -1);
                switch(shipType) {
                    case ObjectShip.PIRATE:
                        points += PIRATE_POINTS;
                        timerTime += PIRATE_SECONDS;
                        break;
                    case ObjectShip.SHIP:
                        points -= SHIP_POINTS;
                        if(points < 0) {
                            points = 0;
                        }
                        timerTime -= SHIP_SECONDS;
                        break;
                    default:
                        break;
                }
                updatePoints();
                updateTimer();
            }
        }
    };
}
```

Annex 27 – EngineAchievements.java

```
/**
 * Achievement's engine as a singleton.
 */
public class EngineAchievements {

    private static EngineAchievements mInstance;

    private ArrayList<ObjectAchievement> mAchievements = new ArrayList<>();

    private EngineAchievements() {
    }

    /**
     * Create the singleton's instance that will be used on the application.
     * @return EngineAchievements Achievement's instance.
     */
    public static EngineAchievements getInstance(Context context) {
        if (mInstance == null)  {
            // Create the instance
            mInstance = new EngineAchievements();
            // Create the achievements list
            mInstance.createAchievements(context);
        }

        return mInstance;
    }

    /**
     * Create the achievements list
     */
    private void createAchievements(Context context) {
        this.mAchievements.clear();
        // Achievements in a game
        mAchievements.add(new ObjectAchievement(context, Information.ACHIEVEMENT_SAILOR_ID,
1500, true));
        mAchievements.add(new ObjectAchievement(context,
Information.ACHIEVEMENT_GRAND_MASTER_ID, 2500, true));
        mAchievements.add(new ObjectAchievement(context,
Information.ACHIEVEMENT_BOATSWAIN_ID, 5000, true));
        mAchievements.add(new ObjectAchievement(context,
Information.ACHIEVEMENT_OFFICIAL_ID, 7500, true));
        mAchievements.add(new ObjectAchievement(context, Information.ACHIEVEMENT_SKIPPER_ID,
10000, true));
        // Achievements in app
        mAchievements.add(new ObjectAchievement(context,
Information.ACHIEVEMENT_CABIN_BOY_ID, 5000, false));
        mAchievements.add(new ObjectAchievement(context,
Information.ACHIEVEMENT_REAR_ADMIRAL_ID, 10000, false));
        mAchievements.add(new ObjectAchievement(context,
Information.ACHIEVEMENT_VICE_ADMIRAL_ID, 20000, false));
        mAchievements.add(new ObjectAchievement(context, Information.ACHIEVEMENT_ADMIRAL_ID,
50000, false));
        mAchievements.add(new ObjectAchievement(context, Information.ACHIEVEMENT_CAPTAIN_ID,
100000, false));

    }

    /**
     * Check whether a new achievement has been achieved
     * @param context The current app context
     * @param newPoints Last user's punctuation
     * @return boolean True whether the user has achieved a new achievement
     */
```

```
    public boolean updateAchievements(Context context, int newPoints) {
        boolean newAchievements = false;
        int currentTotalPunctuation = DataManagement.getTotalPunctuation(context);
        for(ObjectAchievement achievement : mAchievements) {
            if(!achievement.hasAchieved()) {
                int achievementLimit = achievement.getLimit();
                int comparationLimit = achievement.isLimitInGame()? newPoints :
currentTotalPunctuation;
                if(comparationLimit >= achievementLimit) {
                    newAchievements = true;
                    achievement.setHasAchieved(context, true);
                    Analytics.getInstance().sendEvent(context,
R.array.achievements_updated, achievement.getTitle(context));
                }
            }
        }
        return newAchievements;
    }

    /**
     * Get the achievements list of the application.
     * @return ArrayList<ObjectAchievement> Achievement's list.
     */
    public ArrayList<ObjectAchievement> getAchievementsList() {
        return mAchievements;
    }


}
```

Annex 28 – EngineRanking.java

```
/**
 * Ranking's engine as a singleton.
 */
public class EngineRanking {

    private static EngineRanking mInstance;

    private ArrayList<ObjectRanking> mRanking = new ArrayList<>();

    private EngineRanking() {
    }

    /**
     * Create the singleton's instance that will be used on the application.
     * @param context The current application's context.
     * @return EngineRanking Ranking's instance.
     */
    public static EngineRanking getInstance(Context context) {
        if (mInstance == null)  {
            // Create the instance
            mInstance = new EngineRanking();
            // Create the achievements list
            mInstance.createRanking(context);
        }

        return mInstance;
    }

    /**
     * Create the first fake ranking in order to avoid that the list was empty.
     * @param context The current application's context.
     */
    private void initRanking(Context context) {
        ObjectRanking first = new ObjectRanking(1, "Sara", 12500);
        first.store(context);
        mRanking.add(first);
        ObjectRanking second = new ObjectRanking(2, "Abril", 11800);
        second.store(context);
        mRanking.add(second);
        ObjectRanking third = new ObjectRanking(3, "Clara", 9700);
        third.store(context);
        mRanking.add(third);
        ObjectRanking fourth = new ObjectRanking(4, "Pepi", 8600);
        fourth.store(context);
        mRanking.add(fourth);
        ObjectRanking fifth = new ObjectRanking(5, "Francesc", 7500);
        fifth.store(context);
        mRanking.add(fifth);
        ObjectRanking six = new ObjectRanking(6, "Ignasi", 6900);
        six.store(context);
        mRanking.add(six);
        ObjectRanking seven = new ObjectRanking(7, "George", 5800);
        seven.store(context);
        mRanking.add(seven);
        ObjectRanking eight = new ObjectRanking(8, "Joe", 4900);
        eight.store(context);
        mRanking.add(eight);
        ObjectRanking nine = new ObjectRanking(9, "Charles", 4100);
        nine.store(context);
        mRanking.add(nine);
        ObjectRanking ten = new ObjectRanking(10, "Pere", 3900);
        ten.store(context);
        mRanking.add(ten);
```

```
    }
    /**
     * Create the ranking list.
     * @param context The current application's context.
     */
    private void createRanking(Context context) {
        this.mRanking.clear();

        boolean end = false;
        int counter = 1;
        while(!end) {
            Gson gson = new Gson();
            String json = DataManagement.getPosition(context, counter);
            if(json != null) {
                ObjectRanking obj = gson.fromJson(json, ObjectRanking.class);
                this.mRanking.add(obj);
            } else {
                end = true;
            }
            counter++;
        }


        if(this.mRanking.size() == 0) {
            initRanking(context);
        }
    }

    /**
     * Get the ranking list.
     * @return ArrayList<ObjectRanking> Ranking list.
     */
    public ArrayList<ObjectRanking> getRankingList() {
        return this.mRanking;
    }

    /**
     * Update the user's total punctuation
     * @param context The current application's context.
     * @param newPoints New user's punctuation
     */
    private void updateTotalPoints(Context context, int newPoints) {
        DataManagement.setTotalPunctuation(context, newPoints);
    }

    /**
     * Check whether the user has done a new max punctuation and, in this case, update the user's
max punctuation
     * @param context The current application's context.
     * @param newPoints New user's punctuation
     * @return boolean True whether a new max punctuation has been achieved by the user
     */
    private boolean updateMaxPunctuation(Context context, int newPoints) {
        boolean maxPunctuationUpdated = false;
        int maxPunctuation = DataManagement.getMaxRanking(context);
        if(newPoints > maxPunctuation) {
            DataManagement.setMaxRanking(context, newPoints);
            maxPunctuationUpdated = true;
        }
        return maxPunctuationUpdated;
    }
    /**
     * Update the ranking list with a new possible punctuation.
     * @param context The current application's context.
     * @param newPoints New user's punctuation
     * @return True whether the ranking has been updated
```

```
     */
    public boolean updateRanking(Context context, int newPoints) {
        // Update user's max punctuation
        updateMaxPunctuation(context, newPoints);
        // Update user's total punctuation
        updateTotalPoints(context, newPoints);
        // Update ranking list
        String username = DataManagement.getUserName(context);
        ObjectRanking newRanking = null;
        Iterator<ObjectRanking> iterator = mRanking.iterator();
        while(iterator.hasNext()) {
            ObjectRanking obj = iterator.next();
            if(newRanking != null) {
               obj.increasePosition();
            }
            else if(newPoints > obj.getPoints()) {
                newRanking = new ObjectRanking(obj.getPosition(), username, newPoints);
                obj.increasePosition();
            }
        }
        if(newRanking != null) {
            Analytics.getInstance().sendEvent(context, R.array.ranking_updated,
""+newRanking.getPoints());
            mRanking.add(newRanking);
            Collections.sort(mRanking, new ComparatorRanking());
            while(mRanking.size() > 10) {
                mRanking.remove(mRanking.size()-1);
            }
            storeRankingList(context);
        }
        return (newRanking != null);
    }

    private void storeRankingList(Context context) {
        DataManagement.clearPositions(context);
        Iterator<ObjectRanking> iterator = mRanking.iterator();
        while(iterator.hasNext()) {
            iterator.next().store(context);
        }
    }

}
```

Annex 29 – EngineSound.java

```
/**
 * Sound's engine as a singleton
 */
public class EngineSound {
    private SoundPool soundPool;
    boolean plays = false, loaded = false;
    float actVolume, maxVolume, volume;
    private int explosionID, shootID;
    AudioManager audioManager;
    private Context context;

    private static EngineSound mInstance;

    private EngineSound(Context context) {
        this.context = context;

        audioManager = (AudioManager) context.getSystemService(Context.AUDIO_SERVICE);
        actVolume = (float) audioManager.getStreamVolume(AudioManager.STREAM_MUSIC);
        maxVolume = (float) audioManager.getStreamMaxVolume(AudioManager.STREAM_MUSIC);
        volume = actVolume / maxVolume;

        //Hardware buttons setting to adjust the media sound
        ((Activity) context).setVolumeControlStream(AudioManager.STREAM_MUSIC);

        soundPool = new SoundPool(10, AudioManager.STREAM_MUSIC, 0);

        explosionID = soundPool.load(context, R.raw.explosion, 1);
        shootID = soundPool.load(context, R.raw.shoot, 1);


    }

    /**
     * Create the singleton's instance that will be used on the application.
     * @return EngineSound Sound's instance.
     */
    public static EngineSound getInstance(Context context) {
        if (mInstance == null)  {
            // Create the instance
            mInstance = new EngineSound(context);
        }

        return mInstance;
    }

    private void play(int soundID) {
        if(DataManagement.getIsSoundEnabled(context)) {
            soundPool.play(soundID, volume, volume, 1, 0, 1f);
        }
    }

    public void explosion() {
        play(explosionID);
    }

    public void shoot() {
        play(shootID);
    }

}
```

Annex 30 – ObjectRanking.java

```
/**
 * Object with the position information
 */
public class ObjectRanking {
    private int position;
    private String name;
    private int points;

    public ObjectRanking(int position, String name, int points) {
        this.position = position;
        this.name = name;
        this.points = points;
    }

    /**
     * Get the rankings' name
     * @return String Rankings' name
     */
    public String getName() {
        return name;
    }

    /**
     * Get the rankings' description
     * @param context The current application context
     * @return String Rankings' description
     */
    public String getDescription(Context context) {
        return context.getString(R.string.ranking_description,
NumberFormat.getNumberInstance().format(points));

    }

    /**
     * Get the rankings' punctuation
     * @return int Rankings' punctuation
     */
    public int getPoints() {
        return points;
    }

    /**
     * Get the rankings' position
     * @return int Rankings' position
     */
    public int getPosition() {
        return position;
    }

    /**
     * Get the object's as String
     * @return String JSON as string
     */
    private String getJSON() {
        Gson gson = new Gson();
        return gson.toJson(this);
    }

    /**
     * Increase the ranking position in one position.
     */
    public void increasePosition() {
        position++;
```

```
    }
    /**
     * Store the object into shared preference
     * @param context The current application context
     */
    public void store(Context context) {
        DataManagement.setPosition(context, position, getJSON());
    }
}
```

Annex 31 – ObjectAchievement.java

```
/**
 * Achievement object.
 */
public class ObjectAchievement {

    /** Achievement's id. */
    private int id;
    /** Achievement's title. */
    private String title;
    /** Achievement's description. */
    private String description;
    /** Limit of points required in order to achieve that achievement*/
    private int limit;
    /** True whether the limit is related to the punctuation in a single game */
    private boolean limitInGame;
    /** True whether the user has already achieved that achievement. */
    private boolean hasAchieved;

    public ObjectAchievement(Context context, int id, int limit, boolean limitInGame) {
        this.id = id;
        this.limit = limit;
        this.limitInGame = limitInGame;

        updateHasAchieved(context);
    }

    /**
     * Set the hasAchieved variable from shared preference
     * @param context The current context
     */
    private void updateHasAchieved(Context context) {
        String achivementPreferenceId = Information.SHARED_PREFERENCE_ACHIEVEMENT_BASE + id;
        SharedPreferences sp =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_ACHIEVEMENT,
Context.MODE_PRIVATE);

        hasAchieved = (sp.contains(achivementPreferenceId) &&
sp.getBoolean(achivementPreferenceId, false));
    }

    /**
     * Set the hasAchieved shared preference and update's the variable's value
     * @param context The current application's context
     * @param hasAchieved Has achieved value
     */
    public void setHasAchieved(Context context, boolean hasAchieved) {
        SharedPreferences.Editor editor =
context.getSharedPreferences(Information.SHARED_PREFERENCE_PREFS_ACHIEVEMENT,
Context.MODE_PRIVATE).edit();
        editor.putBoolean(Information.SHARED_PREFERENCE_ACHIEVEMENT_BASE + id, hasAchieved);
        editor.commit();

        this.hasAchieved = hasAchieved;

    }
    /**
     * Check whether the user has already achieved that achievement.
     * @return Boolean True whether the user has already achieved that achievement.
     */
    public boolean hasAchieved() {
        return hasAchieved;
    }
```

```
    /**
     * Check whether the achievement's limit is in game or in total
     * @return True whether the achievement's limit is in game
     */
    public boolean isLimitInGame() {
        return limitInGame;
    }

    /**
     * Get the achievement's id
     * @return int Achievement id
     */
    public int getId() {
        return id;
    }

    /**
     * Get the achievement's limit.
     * @return int Achievement's limit
     */
    public int getLimit() {
        return limit;
    }
    /**
     * Get the achievement's title.
     * @param context The current application's context
     * @return String Achievement's title.
     */
    public String getTitle(Context context) {
        int resourceId = ((BoatAttack)
context.getApplicationContext()).getStringIdentifier(context, "achievement_title_"+id);
        return context.getString(resourceId);
    }

    /**
     * Get the achievement's description.
     * @param context The current application's context
     * @return String Achievement's description.
     */
    public String getDescription(Context context) {
       int resourceId = limitInGame? R.string.achievement_description_game :
R.string.achievement_description_app;
       return context.getString(resourceId, NumberFormat.getNumberInstance().format(limit));
    }
}
```

Annex 32 – ObjectShip.java

```
/**
 *  ObjectShip animation. It does translate the ship from side to side.
 */
public class ObjectShip {

    // SHOOT MARGINS
    private final int MARGIN = 15;

    // TYPES
    public static final int PIRATE = 0;
    public static final int SHIP = 1;
    // STATES
    private final int OK = 0;
    private final int SUNKEN = 1;
    // RATES IN MILLISECONDS
    private final int SLOW = 3000;
    private final int MEDIUM = 2000;
    private final int FAST = 1500;
    private final int SUPER_FAST = 1000;
    /** ObjectShip's type */
    private int type = PIRATE;
    /** ObjectShip's state */
    private int state = OK;
    /** ObjectShip's rate */
    private int rate = MEDIUM;

    private long startTime = -1;
    private int distance = -1;
    private int yRoad = -1;

    /** Translate animation engine */
    private TranslateAnimation translateAnimation = null;
    /** ObjectShip image view */
    private ImageView ship = null;
    /** Relative layout element */
    private RelativeLayout rl_game = null;
    /** Context where the ship has been created */
    private Context context = null;

    public ObjectShip(Context context, RelativeLayout rl_gme) {
        this.rl_game = rl_gme;
        this.context = context;

        // Set the ship's type
        Random random = new Random();
        type = random.nextInt(2);

        // Set the ship's rate
        int rate_type = random.nextInt(4);
        switch(rate_type) {
            case 0:
                rate = SLOW;
                break;
            case 1:
                rate = MEDIUM;
                break;
            case 2:
                rate = FAST;
                break;
            default:
                rate = SUPER_FAST;
                break;
        }
```

```
        // Set the y position (4 possible rows)
        yRoad = ((rl_game.getMeasuredHeight()/4) * (random.nextInt(4)));

        distance = rl_game.getMeasuredWidth()+200;

        ship = new ImageView(context);
        ship.setTag(this);
        setImage();
        rl_game.addView(ship);
        // Set the ship initial position
        ship.setX(-100);
        ship.setY(yRoad);

        // Set the translate animation
        translateAnimation = new TranslateAnimation( 0, distance, 0, 0);
        translateAnimation.setDuration(rate);
        translateAnimation.setFillAfter(true);
        translateAnimation.setFillEnabled(true);
        translateAnimation.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {
                startTime = System.currentTimeMillis();
            }

            @Override
            public void onAnimationRepeat(Animation animation) {
            }

            @Override
            public void onAnimationEnd(Animation animation) {
                // Destroy the ship as it has finished
                rl_game.removeView(ship);
                ship.setImageDrawable(null);
                ship = null;
            }
        });
    }

    /**
     * Start to sail, so translate the ship form side to side
     */
    public void sail() {
        if(ship != null && translateAnimation != null) {
            ship.startAnimation(translateAnimation);
        }
    }

    /**
     * Set the ship's image depending on its type and state
     */
    private void setImage() {
        int resourceId = -1;
        switch(type) {
            case SHIP:
                resourceId = (state == SUNKEN)? R.drawable.ship_sunken : R.drawable.ship;
                break;
            default:
                resourceId = (state == SUNKEN)? R.drawable.pirate_sunken : R.drawable.pirate;
                break;
        }
        if(ship != null && resourceId != -1) {
            ship.setImageResource(resourceId);
        }
    }
```

```
    /**
     * Set this ship as sunken and updates its image
     */
    private void sunk() {
        state = SUNKEN;
        setImage();
        Utils.vibrate(context);
        EngineSound.getInstance(context).explosion();
        Utils.sendBroadcastMessage(context, Information.BROADCAST_ACTION_SHIP_SUNKEN,
Information.BROADCAST_ACTION_KEY_SHIP_TYPE, type);
    }

    /**
     * Check whether the ship is located where the bullet has been shot
     * @param x Bullet's x position
     * @param y Bullet's y position
     */
    public void bulletAt(int x, int y) {
        int[] location = new int[2];
        if(ship != null && state == OK) {
            ship.getLocationOnScreen(location);

            long now = System.currentTimeMillis();
            long timeFromStart = now - startTime;
            float percentage = (float) timeFromStart/rate;
            float currentXLeftPosition = (distance*percentage)+location[0];

            int xDistance =  x - (int) currentXLeftPosition;
            int yDistance = y - yRoad;

            if(xDistance >= 0 && xDistance <= (ship.getMeasuredWidth()) && yDistance > 0 &&
yDistance <= (ship.getMeasuredHeight())) {
                sunk();
            }
        }
    }
}
```

Annex 33 – ObjectShoot.java

```
/**
 * ObjectShoot animation. It does translate the bullet view and resize it at the same time.
 */
public class ObjectShoot {

    /** @return Time in milliseconds that the translate animation takes (once the distance has
been taken into account) */
    private final int TRANSLATE_DURATION = 500;
    /** @return Minimum time that the translation must take in milliseconds */
    private final int TRANSLATE_MINIMUM_DURATION = 100;
    /** Translate animation engine */
    private TranslateAnimation translateAnimation = null;
    /** Animation's target element */
    private ImageView bullet = null;
    /** Relative layout element */
    private RelativeLayout rl_game = null;
    private int xDest = 0;
    private int yDest = 0;
    private Context context;

    public ObjectShoot(Context cntxt, RelativeLayout rl_gme, int xDest, int yDest) {
        this.context = cntxt;
        this.rl_game = rl_gme;
        this.xDest = xDest;
        this.yDest = yDest;

        bullet = new ImageView(context);
        bullet.setImageResource(R.drawable.bullet);
        rl_game.addView(bullet);

        bullet.getLayoutParams().height = (int)
context.getResources().getDimension(R.dimen.bullet_offset);
        bullet.getLayoutParams().width = (int)
context.getResources().getDimension(R.dimen.bullet_offset);
        bullet.setX(-100);
        bullet.setY(rl_game.getMeasuredHeight()-100);


        int xTranslation = xDest + 100;
        int yTranslation = yDest - rl_game.getMeasuredHeight() + 100;

        double maxDistance = Math.sqrt(Math.pow(rl_game.getMeasuredWidth(),2) +
Math.pow(rl_game.getMeasuredHeight(), 2));
        double distance = Math.sqrt(Math.pow(xTranslation, 2) + Math.pow(yTranslation, 2));

        int translateTime = (int) (distance/maxDistance * TRANSLATE_DURATION) +
TRANSLATE_MINIMUM_DURATION;

        translateAnimation = new TranslateAnimation( 0, xTranslation, 0, yTranslation);
        translateAnimation.setDuration(translateTime);
        translateAnimation.setFillAfter(true);
        translateAnimation.setAnimationListener(new Animation.AnimationListener() {
            @Override
            public void onAnimationStart(Animation animation) {
                EngineSound.getInstance(context).shoot();
            }

            @Override
            public void onAnimationRepeat(Animation animation) {
            }

            @Override
            public void onAnimationEnd(Animation animation) {
```

```
                // Destroy the shoot as it has finished
                rl_game.removeView(bullet);
                bullet.setImageDrawable(null);
                bullet = null;
                checkIfSunken();
            }
        });


    }

    /***
     * Start the shoot animation
     */
    public void shoot() {
        if(bullet != null && translateAnimation != null) {
            bullet.startAnimation(translateAnimation);
        }
    }

    private void checkIfSunken() {
        int childcount = rl_game.getChildCount();
        for(int i=0; i<childcount; i++) {
            View v = rl_game.getChildAt(i);

            if(v.getTag() != null) {
                ObjectShip objectShip = (ObjectShip) v.getTag();

                objectShip.bulletAt(xDest, yDest);

            }
        }
    }

}
```

Annex 34 – attrs.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="CustomTextView">
        <attr name="fontType" format="enum">
            <enum name="light" value="0"/>
            <enum name="regular" value="1"/>
        </attr>
    </declare-styleable>
</resources>
```

Annex 35 – CustomTextView.java

```
/**
 * Custom application's text view
 */
public class CustomTextView extends TextView {
    private int fontType = 0;

    public CustomTextView(Context context) {
        super(context);
        fontType(context, null);
    }

    public CustomTextView(Context context, AttributeSet attrs) {
        super(context, attrs);
        fontType(context, attrs);
    }

    public CustomTextView(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        fontType(context, attrs);
    }

    private void fontType(Context context, AttributeSet attrs) {
        if(attrs != null) {

            TypedArray a = context.getTheme().obtainStyledAttributes(
                    attrs,
                    R.styleable.CustomTextView,
                    0, 0);

            try {
                fontType = a.getInteger(R.styleable.CustomTextView_fontType, 0);
            } finally {
                a.recycle();
            }
        }

        Typeface lightTypeface = Typeface.createFromAsset(context.getAssets(),
"fonts/Stencil.ttf");
        Typeface regularTypeface = Typeface.createFromAsset(context.getAssets(),
"fonts/StencilExpanded.ttf");
        switch(fontType) {
            case 0:
                super.setTypeface(lightTypeface, Typeface.NORMAL);
                break;
            default:
                super.setTypeface(regularTypeface, Typeface.NORMAL);
                break;
        }
    }
}
```

Annex 36 – CustomEditText.java

```
/**
 * Custom application's edit text
 */
public class CustomEditText extends EditText {
    private int fontType = 0;

    public CustomEditText(Context context) {
        super(context);
        fontType(context, null);
    }

    public CustomEditText(Context context, AttributeSet attrs) {
        super(context, attrs);
        fontType(context, attrs);
    }

    public CustomEditText(Context context, AttributeSet attrs, int defStyle) {
        super(context, attrs, defStyle);
        fontType(context, attrs);
    }

    private void fontType(Context context, AttributeSet attrs) {
        if(attrs != null) {

            TypedArray a = context.getTheme().obtainStyledAttributes(
                    attrs,
                    R.styleable.CustomTextView,
                    0, 0);

            try {
                fontType = a.getInteger(R.styleable.CustomTextView_fontType, 0);
            } finally {
                a.recycle();
            }
        }

        Typeface lightTypeface = Typeface.createFromAsset(context.getAssets(),
"fonts/Stencil.ttf");
        Typeface regularTypeface = Typeface.createFromAsset(context.getAssets(),
"fonts/StencilExpanded.ttf");
        switch(fontType) {
            case 0:
                super.setTypeface(lightTypeface, Typeface.NORMAL);
                break;
            default:
                super.setTypeface(regularTypeface, Typeface.NORMAL);
                break;
        }
    }
}
```

Annex 37 – styles.xml

```
<resources>

    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light.NoActionBar">
        <!-- Customize your theme here. -->
    </style>

    <!-- Main Frame style -->
    <style name="FrameLayoutStyle">
        <item name="android:padding">10dp</item>
    </style>

    <!-- Global image view style -->
    <style name="ImageViewStyle">
        <item name="android:background">@color/transparent</item>
        <item name="android:focusable">true</item>
        <item name="android:layout_height">64dp</item>
        <item name="android:layout_width">64dp</item>
        <item name="android:cropToPadding">false</item>
        <item name="android:adjustViewBounds">true</item>
    </style>

    <!-- Options button style -->
    <style name="OptionsButton" parent="ImageViewStyle">
        <item name="android:clickable">true</item>
        <item name="android:onClick">optionsButtonClicked</item>
    </style>

    <!-- Back button -->
    <style name="BackButton" parent="ImageViewStyle">
        <item name="android:clickable">true</item>
        <item name="android:layout_gravity">left|top</item>
        <item name="android:src">@drawable/bt_back</item>
        <item name="android:onClick">backButtonClicked</item>
        <item name="android:layout_marginBottom">10dp</item>
    </style>

    <style name="StopButton" parent="BackButton">
        <item name="android:src">@drawable/bt_stop</item>
    </style>

    <!-- Text styles -->
    <style name="TextStyle">
    </style>

    <!-- Title -->
    <style name="BigText" parent="TextStyle">
        <item name="android:textSize">18sp</item>
        <item name="android:textStyle">bold</item>
        <item name="fontType">regular</item>
    </style>
    <!-- Medium -->
    <style name="MediumText" parent="BigText">
        <item name="android:textSize">16sp</item>
    </style>
    <!-- Small -->
    <style name="SmallText" parent="TextStyle">
        <item name="android:textSize">16sp</item>
        <item name="fontType">light</item>
    </style>

    <color name="transparent">#00000000</color>
</resources>
```

Annex 38 – arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="sounds">
        <item>Settings</item>
        <item>Sounds</item>
    </string-array>
    <string-array name="vibration">
        <item>Settings</item>
        <item>Vibration</item>
    </string-array>
    <string-array name="game_start">
        <item>Game</item>
        <item>Start</item>
    </string-array>
    <string-array name="game_stop">
        <item>Game</item>
        <item>Stop</item>
    </string-array>
    <string-array name="ranking_updated">
        <item>Ranking</item>
        <item>Updated</item>
    </string-array>
    <string-array name="achievements_updated">
        <item>Achievements</item>
        <item>Achieved</item>
    </string-array>
</resources>
```

Annex 39 – strings.xml

```
<resources>
    <string name="app_name">Boat Attack</string>

    <!-- Dialogs -->
    <string name="dialog_error_positive_btn">OK</string>
    <string name="dialog_username_title">You should introduce your username</string>
    <string name="dialog_username_msg">You must introduce your username if you want your
punctuations to be in the rankings.</string>
    <string name="dialog_achievements_title">Congratulations!</string>
    <string name="dialog_achievements_msg">You have achieved a new achievements! Go on this way
to become a real captain!</string>
    <string name="dialog_ranking_title">Congratulations!</string>
    <string name="dialog_ranking_msg">You have had a great punctuation! Now you are in the TOP-10
ranking!</string>
    <string name="dialog_game_over_title">You run out of time!</string>
    <string name="dialog_game_over_msg">The game is over! Keep playing to achieve more
achievements and become one of the bests users in the world!</string>


    <!-- Achievement label -->
    <string name="achieved_label">Achieved!</string>
    <!-- Achievements titles -->
    <string name="achievement_title_1">Sailor</string>
    <string name="achievement_title_2">Grand master</string>
    <string name="achievement_title_3">Boatswain</string>
    <string name="achievement_title_4">Official</string>
    <string name="achievement_title_5">Skipper</string>
    <string name="achievement_title_6">Cabin boy</string>
    <string name="achievement_title_7">Rear admiral</string>
    <string name="achievement_title_8">Vice-admiral</string>
    <string name="achievement_title_9">Admiral</string>
    <string name="achievement_title_10">Captain</string>
    <!-- Achievements description -->
    <string name="achievement_description_game">%1$s in a game</string>
    <string name="achievement_description_app">%1$s in total</string>


    <!-- Ranking strings -->
    <string name="username_palceholder">Type your name...</string>
    <string name="ranking_description">%1$s points</string>
    <string name="ranking_max"><b>Max:</b> %1$s</string>
    <string name="ranking_total"><b>Total:</b> %1$s</string>

    <!-- Settings -->
    <string name="settings_enable_sound">Enable sound</string>
    <string name="settings_enable_vibration">Enable vibration</string>
</resources>
```

Annex 40 – strings-ca.xml

```
<resources>
    <string name="app_name">Boat Attack</string>

    <!-- Dialogs -->
    <string name="dialog_error_positive_btn">D\'acord</string>
    <string name="dialog_username_title">Has d\'introduir el teu nom</string>
    <string name="dialog_username_msg">Has d\'introduir el teu nom d\'usuari si vols que les
teves puntuacions siguin tingudes en compte en el llistat de puntuacions.</string>
    <string name="dialog_achievements_title">Enhorabona!</string>
    <string name="dialog_achievements_msg">Has assolit un nou assoliment. Segueix així per
convertir-te en un autèntic capità!</string>
    <string name="dialog_ranking_title">Enhorabona!</string>
    <string name="dialog_ranking_msg">Has obtingut una gran puntuació, ara ets entre els 10
millors del joc!</string>
    <string name="dialog_game_over_title">T\'has quedat sense temps!</string>
    <string name="dialog_game_over_msg">Final del joc! Segueix jugant per assolir més
assoliments i convertir-te en un dels millors jugadors del món.</string>


    <!-- Achievement label -->
    <string name="achieved_label">Assolit!</string>
    <!-- Achievements titles -->
    <string name="achievement_title_1">Mariner</string>
    <string name="achievement_title_2">Gran mestre</string>
    <string name="achievement_title_3">Contramestre</string>
    <string name="achievement_title_4">Oficial</string>
    <string name="achievement_title_5">Capità</string>
    <string name="achievement_title_6">Mosso de cabina</string>
    <string name="achievement_title_7">Contralmirall</string>
    <string name="achievement_title_8">Vicealmill</string>
    <string name="achievement_title_9">Almirall</string>
    <string name="achievement_title_10">Comandant</string>
    <!-- Achievements description -->
    <string name="achievement_description_game">%1$s en una partida</string>
    <string name="achievement_description_app">%1$s en total</string>


    <!-- Ranking strings -->
    <string name="username_palceholder">Escriu el teu nom...</string>
    <string name="ranking_description">%1$s punts</string>
    <string name="ranking_max"><b>Màx:</b> %1$s</string>
    <string name="ranking_total"><b>Total:</b> %1$s</string>

    <!-- Settings -->
    <string name="settings_enable_sound">Habilitar so</string>
    <string name="settings_enable_vibration">Habilitar vibració</string>
</resources>
```

Annex 41 – strings-es.xml

```
<resources>
    <string name="app_name">Boat Attack</string>

    <!-- Dialogs -->
    <string name="dialog_error_positive_btn">De acuerdo</string>
    <string name="dialog_username_title">Debes introducir tu nombre</string>
    <string name="dialog_username_msg">Debes introducir un nombre de usuario si quieres que tus
puntuaciones sean tenidas en cuenta en el listado de puntuaciones.</string>
    <string name="dialog_achievements_title">¡Enhorabuena!</string>
    <string name="dialog_achievements_msg">Has aconseguido un nuevo logro. ¡Sigue así para
convertire en un auténtico capitán!</string>
    <string name="dialog_ranking_title">¡Enhorabuena!</string>
    <string name="dialog_ranking_msg">Has logrado una gran puntuación, ¡Ahora estás entre los
10 mejores del juego!</string>
    <string name="dialog_game_over_title">¡Te has quedado sin tiempo!</string>
    <string name="dialog_game_over_msg">¡Final de la partida! Sigue jugando para obtener más
logros y convertirte en uno de los mejores jugadores del mundo.</string>


    <!-- Achievement label -->
    <string name="achieved_label">Logrado</string>
    <!-- Achievements titles -->
    <string name="achievement_title_1">Marinero</string>
    <string name="achievement_title_2">Gran maestro</string>
    <string name="achievement_title_3">Contramaestre</string>
    <string name="achievement_title_4">Oficial</string>
    <string name="achievement_title_5">Capitán</string>
    <string name="achievement_title_6">Mozo de cabina</string>
    <string name="achievement_title_7">Contralmirante</string>
    <string name="achievement_title_8">Vicealmirante</string>
    <string name="achievement_title_9">Almirante</string>
    <string name="achievement_title_10">Comandante</string>
    <!-- Achievements description -->
    <string name="achievement_description_game">%1$s en una partida</string>
    <string name="achievement_description_app">%1$s en total</string>


    <!-- Ranking strings -->
    <string name="username_palceholder">Escribe tu nombre...</string>
    <string name="ranking_description">%1$s puntos</string>
    <string name="ranking_max"><b>Máx:</b> %1$s</string>
    <string name="ranking_total"><b>Total:</b> %1$s</string>

    <!-- Settings -->
    <string name="settings_enable_sound">Habilitar sonidos</string>
    <string name="settings_enable_vibration">Habilitar vibración</string>
</resources>
```

Annex 42 – AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.boatattack.francesc.boatattack" >

    <uses-permission android:name="android.permission.VIBRATE"/>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />


    <application
        android:name=".Engine.BoatAttack"
        android:allowBackup="true"
        android:icon="@drawable/ship_sunken"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >

        <meta-data android:name="com.google.android.gms.version"
            android:value="@integer/google_play_services_version" />

        <activity
            android:name=".Activities.MainActivity"
            android:screenOrientation="portrait"
            android:configChanges="keyboardHidden|orientation|screensize"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```