



Col·legi Episcopal
Lleida

Treball de Recerca

"Programació d'aplicacions pel Sistema Operatiu Android en el llenguatge de programació Java"

Pablo Fraile Alonso

Abstract

Programming is one of the most important and powerful branches of the technological environment. This is the reason why this research project presents the development of two different apps programmed in Java for the Android Operating System. The project includes an exhaustive study on the chronology of the process during the creation of both applications, from the first line of code to the process of uploading them to the Google Play Store. The conclusions drawn in this project are satisfactory since the different purposes planned for the apps have been fulfilled. Having as a result a simple app with some type of utility.

Agraïments:

*Al meu tutor Salvador Sol, per la seva ajuda i comprensió en el desenvolupament
d'aquest Treball de Recerca.*

*A la meva família per el suport incondicional des de l'inici del Treball, i per motivar-me
en aquells moments en que no ho veia tot clar.*

*Als meus amics i tots els que han decidit provar les diferents aplicacions i invertir una
part del seu temps en el meu treball.*

Índex:

Introducció:

Part Teòrica:

2. Introducció al llenguatge de programació:

2.1 Programar:.....	6
2.2 Programa:	8
2.2.1 Explicació del codi font:	8
2.2.1.1 Paraules reservades en el codi font:	9
2.2.1.2 Anotacions:.....	10
2.2.2 Executable	10
2.2.3 Aplicació	11
2.3 Diferents tipus de llenguatge de programació i la seva utilitat:.....	11
2.4 Tipus de llicències en els llenguatges de programació i la seva importància:.....	12
2.4.1 Les Llicències de Software Lliure:.....	12
2.4.1 Les Llicències de Software Propietari:.....	13

3. Compiladors:

3.1 Definició de compilador i la seva funció.....	14
3.2 Fases del Compilador	14

4. Introducció a Java:

4.1 Definició i característiques del llenguatge de programació Java:	16
4.1.2 Programació orientada a objectes:	16
4.2 Introducció a l'ecosistema de java	17
4.2.1 Java JDK:	17
4.3 Creació de l'aplicació a partir del codi font.....	20
4.3.1 Procés de compilació.....	20
4.4 Java API.....	22

5. Android Studio:

5.1 Introducció a Android Studio	24
5.2 Android SDK	27
5.3 Gradle	27
5.4 El procés de Compilació:	28

6. Java en Android:

6.1 Funcionament d'Android	29
6.2 Funcionament del Android Runtime (ART)	32
6.3 Fonaments de la creació d'una aplicació Android:	33
6.4 Recursos de l'aplicació	34
6.5 Recurs Layout:	35

Part Pràctica

7. Calculadora:

7.1 Creació del projecte:	38
7.2 Explicació de la interfície gràfica (layout).....	42
7.3 Explicació de les classes	50
7.4 Explicació del codi font.....	52
7.4 Penjar l'aplicació a la Google Play.....	66

8. Dreceres:

8.1 El perquè del projecte i la creació d'aquest.....	69
8.2 Explicació de la interfície gràfica	69
8.3 Explicació de les classes	80
8.4 Explicació del codi font.....	84
8.4.1 Activitat_Principal.java.....	84
8.4.2 Explicació de “configuració.java”	104
8.4.3 Explicació de “info.java”	115
8.5 Penjar l'aplicació a la Google Play.....	116

Introducció:

En el present treball de recerca, es pretén programar dues aplicacions útils i funcionals pel sistema operatiu Android mitjançant el llenguatge de programació Java i l'entorn de desenvolupament “Android Studio”.

Per a poder realitzar aquest projecte, s'ha estructurat en tres parts diferents: una part teòrica i dues parts pràctiques. En la part teòrica, es farà una introducció al món de la programació, sobretot al món dels compiladors i del codi font, ja que és essencial per a entendre la part pràctica del projecte. Tot seguit es presentarà el món de la programació adaptat al sistema operatiu Android, i finalment s'explicaran la creació de les dues aplicacions, tant la part visual com la part de codi. Finalment es penjaran aquestes a la Google Play Store.

PART TEÒRICA

2. Introducció al llenguatge de programació:

2.1 Programar:

El verb programar es refereix a l'art de fer que un dispositiu electrònic (com per exemple un telèfon mòbil) dugui a terme unes determinades accions que l'usuari anomenat programador li ha encomanat anteriorment. Hi ha diferents tipus de llenguatges de programació que poden oferir més accions que altres, però finalment tots tenen com a funció realitzar accions encomanades anteriorment pel programador.

2.2 Programa:

Segons la Universitat Internacional d' Atlanta :

"Un programa és un conjunt d'instruccions o ordres basades en un llenguatge de programació que un ordinador interpreta per a resoldre un problema o una funció específica."

El conjunt de programes per a fer diferents funcions d'un dispositiu electrònic s'anomena Software.

2.2.1 Explicació del codi font:

El codi font és una de les parts més importants d'un programa. El codi font d'un programa és el conjunt de línies que un programador escriu amb un llenguatge de programació amb els passos que ha de seguir un dispositiu per executar dit programa.

Un exemple de codi font és el de la figura 1, que com es pot apreciar és un conjunt de línies escrites en un llenguatge de programació (en aquest cas el llenguatge utilitzat és Java).

Aquestes línies de codi són essencials per a que el programa, quan sigui executat, tingui les ordres del que ha de fer.

```
public class codifont{  
    public static void main(String []args){  
        System.out.println("Codi Font");  
    }  
}
```

Figura 1, codi font.

2.2.1.1 Paraules reservades en el codi font:

Dins del codi font, encara que aquest depengui del llenguatge de programació, es poden trobar ítems que es reconeixen en tots els llenguatges de programació. Un d'aquests ítems són les paraules reservades.

Les paraules reservades, són identificadors dins d'un codi font, predefinitos pel propi llenguatge de programació que tenen un significat "especial" assignat i que no es poden utilitzar com identificadors de qualsevol element en el seus programes.

Per a entendre millor aquest fet, s'introduirà una paraula reservada en el codi font d'un programa de Java (podria ser aplicat a qualsevol llenguatge de programació, amb qualsevol paraula reservada d'aquest.)

En Java, una paraula reservada és per exemple la paraula "int". Per tant, si se li assigna un identificador, el programa donarà error a l'hora de executar-se.

```
public class noerror{  
    public static void main(String []args){  
        System.out.println("No hi ha error");  
    }  
}  
  
public class int{  
    public static void main(String []args){  
        System.out.println("Hi ha error");  
    }  
}
```

Figura 2, paraules reservades.

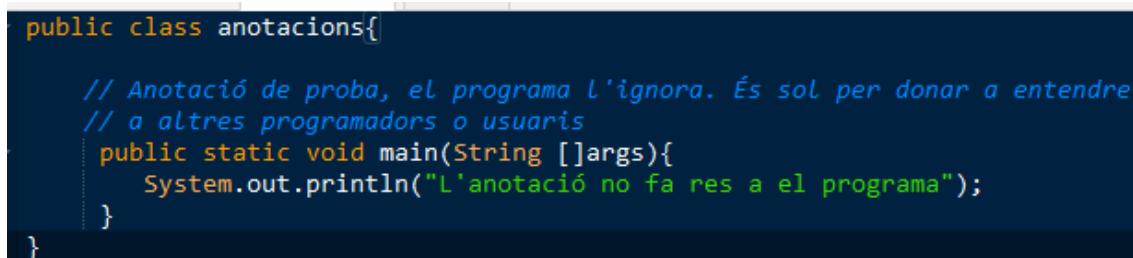
Com es pot apreciar en la figura 2, si s'escriu la paraula int en qualsevol lloc del programa (exceptuant algunes funcions, que seran explicades a la part pràctica) el llenguatge la detectarà com a una paraula reservada.

En canvi com la paraula "noerror" no és una paraula reservada, el codi s'executarà sense cap error.

2.2.1.2 Anotacions:

En gairebé tots els llenguatges de programació, podem trobar una eina que s'utilitza per a organitzar el codi font i explicar a altres programadors per a què serveix i que fa aquell codi font. Aquesta eina són els anomenats símbols d'anotacions. En cada llenguatge s'associen dos o tres símbols que son ignorats pel programa i que l'únic que fan és que el programador que escrigui el codi pugui explicar per a què serveix qualsevol línia del codi font, i per tant, donar informació a altres programadors o usuaris que entenguin el llenguatge de programació.

En el cas de Java, c o c++ les anotacions simples són donades amb els símbols "/* */" com es pot apreciar en la Figura 3.



```
public class anotaciones{  
    // Anotació de prova, el programa l'ignora. És sol per donar a entendre  
    // a altres programadors o usuaris  
    public static void main(String []args){  
        System.out.println("L'anotació no fa res a el programa");  
    }  
}
```

A screenshot of a Java code editor showing a class named 'anotaciones'. Inside the class, there is a single-line comment starting with '//' followed by a multi-line explanatory text. The explanatory text discusses the purpose of annotations: to provide documentation for the code, specifically for other programmers or users. It states that the program ignores the annotations but they are useful for understanding the code. The code then defines a main method that prints a message to the console.

Figura 3,anotacions.

2.2.2 Executable

Un executable és el conjunt d'arxius que formen un programa. Això vol dir que un executable inclou el codi font i, a més a més, altres ítems que poden ajudar a fer el programa més extens o comprensible. Un exemple d'arxius que es poden trobar dins d'un executable poden ser imatges o arxius en llenguatge de programació XML (llenguatge de programació orientat a interfícies gràfiques) que no executen cap acció sinó que donen una interfície gràfica que pot ajudar a l'usuari que executi el programa (un exemple pot ser el arxiu icon.png de la figura 4)

Aquests arxius addicionals es solen ficar en carpetes per a diferenciar-ho d'altres arxius i per a disposar d'un executable organitzat (com es pot apreciar en la figura 4, en l'el·ipse de color vermell).

Simple-Calculator / fastlane / metadata / android / en-US / images /		
		Create new file Find file History
 tibbi changing the launcher icons to make the foreground smaller		Latest commit 0600de9 11 days ago
"		
 phoneScreenshots	updating screenshots	2 months ago
 sevenInchScreenshots	updating screenshots	2 months ago
 teninchScreenshots	updating screenshots	2 months ago
 featureGraphic.png	updating some fastlane info	4 months ago
 icon.png	changing the launcher icons to make the foreground smaller	11 days ago

Figura 4, executable

2.2.3 Aplicació

Per a poder explicar el terme aplicació, s'ha de tornar a explicar la definició de programa per poder entendre certes diferències.

Tal i com s'ha mencionat anteriorment, un programa és un conjunt d'instruccions escrites amb un llenguatge de programació que un aparell interpreta per a resoldre un problema o una funció específica.

Una aplicació no és més que un estil de programa dissenyat únicament per facilitar a l'usuari la realització d'un determinat treball. Això marca una gran diferència amb altres programes que estan pensats per realitzar treballs més avançats i que no afecten de manera comú al usuari (com podrien ser per exemple programes que fan funcionar un sistema operatiu).

Un exemple d'aplicació pot ser un programa que fa el treball de calcular les notes finals dels treballs de recerca a partir de certs percentatges donats.

2.3 Diferents tipus de llenguatge de programació i la seva utilitat:

Tipus de llenguatges de programació segons la seva dificultat:

- Llenguatges de programació de nivell baix: Es programen tenint en compte les característiques del processador al qual se li donen les ordres. Solen ser ordres lògiques i no tenen molta complexitat.

Exemples de llenguatges de programació de nivell baix: Basic, python o bash.

.- Llenguatges de programació de nivell mitjà: Són llenguatges de programació que ajuden als llenguatges de programació de nivell alt. Són molt útils implementant algoritmes, per fer taules mitjançant codi, etc.

Exemples de llenguatges de programació de nivell mitjà: C o Java-script.

.-Llenguatges de programació de nivell alt: Són els llenguatges de programació que més s'aproximen al llenguatge dels éssers humans. Això vol dir que a vegades ignoren el funcionament de la màquina. Són molt utilitzats en Intel·ligència Artificial.

Exemples de llenguatge de programació de nivell alt: Java, Ruby o Kotlin.

2.4 Tipus de llicències en els llenguatges de programació i la seva importància:

Segons l' IEC, la paraula llicència té el següent significat: "Llibertat de fer o de dir alguna cosa en virtut d'una permisió donada." Per tant, si apliquem la paraula llicència al món de la programació, trobem que el significat variarà, ja que una llicència en el món de programació i la informàtica és un contracte entre el creador d'un programa, software o un llenguatge de programació i un programador o un usuari comú. Dins d'aquesta llicència, que ha d'existir, en podem trobar de diferents tipus, les llicències de software lliure (on el contracte explícit anteriorment seria el idoni per a l'usuari o programador) i les llicències de software propietari (on el " contracte explícit anteriorment, seria idoni pels creadors del llenguatge de programació).

2.4.1 Les Llicències de Software Lliure:

Les llicències de software lliure, són aquelles llicències en que, el "contracte entre el usuari/programador i els creadors del llenguatge de programació o software beneficien sobretot als usuaris. Aquest benefici sorgeix dels següents motius:

- .- El codi del llenguatge/software és accessible per tothom.
- .- No té cost per utilitzar ell llenguatge o software
- .- Dona total llibertat a l'usuari.

- .- No requereix actualitzacions.
- .- Es pot trobar orientacions a fòrums, blogs i wikis (que solen ser més properes als usuaris).

Un exemple de llicència de Software Lliure és la llicència anomenada GNU GPL , què és la que utilitzen la majoria de llenguatges de programació lliures, com ara Java (Figura 5).

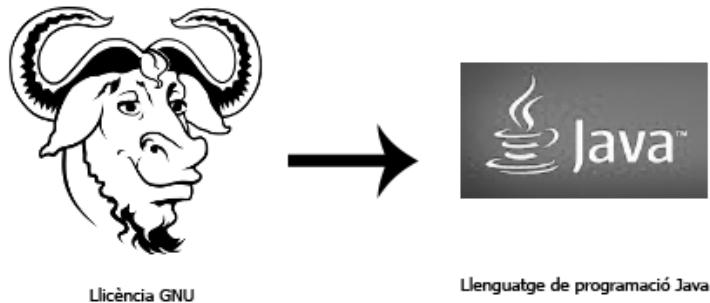


Figura 5, llicència gnu

2.4.1 Les Llicències de Software Propietari:

Les llicències de software propietari, són les llicències que beneficien sobretot a els creadors d'un llenguatge de programació o software. Els motius d'aquest benefici són els següents:

- .- El usuari no té accés al codi.
- .- Requereix actualitzacions
- .- Té un determinat cost
- .- Limita l'ús de l'usuari
- .- Orientació únicament mitjançant pàgines oficials y manuals autoritzats.

Un exemple de software propietari pot ser Windows, que compleix tots els punts anomenats anteriorment

3. Compiladors :

3.1 Definició de compilador i la seva funció

Segons Advanced Compiler Design and Implementation de Steven S. Muchnick, "Els compiladors són sistemes de software que tradueixen programes escrits en llenguatges de programació avançats en el seu equivalent en llenguatge màquina o codi objecte, per a la seva execució en determinada màquina" – definició agafada del llibre Advanced Compiler Design and Implementation, Steven S. Muchnick.

3.2 Fases del Compilador

Un compilador té diferents fases des de que s'executa fins que obté el producte finalitzat, ja sigui un arxiu en codi màquina o un executable. Segons el llibre Advanced Compiler Design and Implementation de Steven S. Muchnick, els compiladors segueixen sempre 4 fases:

- 1) Anàlisis lèxic: El compilador rep el codi font, analitza el codi font i separa les línies de codi per a transformar-les en tokens *. El compilador té aquesta habilitat de fer un anàlisis lèxic del codi font ja que s'ha programat per identificar les paraules reservades del codi font, els identificadors d'aquests, etc.
*Un token, consisteix en agafar les línies de codi independents i separar-les com si fossin frases, pera després analitzar-les.
- 2) Anàlisis Sintàctic o anàlisi: Es processen la seqüència de tokens i es produeix un arbre d'anàlisis sintàctic (com es faria a l'hora d'analitzar una frase) per a comprovar si l'expressió dels tokens es sintàcticament correcta. Després d'aquest anàlisis es genera un codi entremig, que és un codi entremig del llenguatge de programació i el codi màquina
- 3) Procés de comprovació de la semàntica: Es comprova si l'anàlisi d'arbre construït segueix les regles del llenguatge de programació, com per exemple si els identificadors del codi font s'identifiquen abans d'utilitzar-se o no

- 4) La generació del codi: Transforma el codi entremig en el seu equivalent en codi màquina.

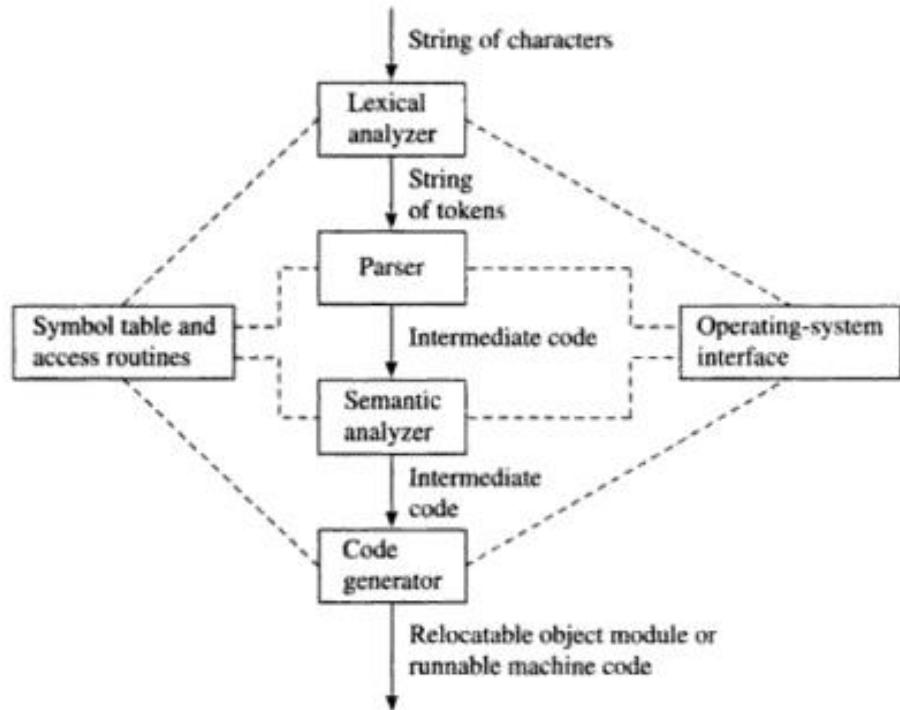


Figura 6, fases del compilador.

4. Introducció a Java:

4.1 Definició i característiques del llenguatge de programació Java:

Java és un llenguatge de programació multi plataforma, amb un propòsit general orientat a objectes.

En una única frase, s'han pogut introduir tres nous conceptes: el llenguatge de programació multi plataforma , un propòsit general i la programació orientada a objectes. Ara es definiran breument aquests conceptes que després s'explicaran més detalladament.

.- El llenguatge de programació multi plataforma: És aquell llenguatge de programació on el seu codi pot ser executat en qualsevol màquina mitjançant un compilador.

.-Llenguatge de propòsit general: Són aquells llenguatges de programació que poden ser utilitzats per a diferents propòsits, com podrien ser des de la creació d'applicacions fins a bases de dades complexes.

.-Programació orientada a objectes: Es una nova filosofia de programar, que canvia el tipus de programació lògica de les màquines a una altra més adaptada al pensament humà.

4.1.2 Programació orientada a objectes:

Segons Elivar Largo (desenvolupador de Software durant 4 anys) explica la programació orientada a objectes és la forma d'agrupar característiques similars de diferents objectes anomenades classes, i a partir d'aquestes classes fer que aquests objectes es comuniquin entre ells mitjançant missatges de codi, per tal de que la unió d'aquests missatges i dels objectes de les determinades classes formin un programa.

A partir d'aquesta definició, han sorgit dos conceptes nous molt importants que defineixen la programació dedicada a objectes:

- 1) Classe: Una classe és una plantilla, un molle o un model per aconseguir crear objectes.
Una classe està composta per unes propietats i atributs (anomenats variables) i per

uns comportaments (anomenats mètodes) que creen un tipus de model a seguir i que després seguirà un objecte. Un exemple de classe podria ser una bicicleta. On les seves variables serien que té dues rodes, un manillar i uns frens i els mètodes serien que pot accelerar i frenar.

- 2) Objecte: Un objecte es una instància d'una classe, un cas específic d'aquesta. Per a entendre-ho millor, Elivar ens cita un exemple de com seria una classe de mòbils amb diferents objectes dins d'aquesta. El exemple descriu que encara que la classe es defineixi per a mòbils no tots els mòbils son iguals , tenen diferent grandària , pes, color, etc. Però segueixen compartint unes característiques bàsiques que fan que aquells objectes siguin mòbils, com poden ser la funcionalitat de trucar o de fer fotografies.

4.2 Introducció a l'ecosistema de java.

Java és un llenguatge de programació multi plataforma, el que vol dir que funciona en tots els dispositius, però per a que el codi font pugui funcionar en tots els sistemes operatius i tots els processadors disponibles en el mercat, les aplicacions que són escrites per aquest llenguatge de programació necessiten un ecosistema on puguin convertir aquest codi en un executable compatible per a un sistema operatiu (en el cas de Windows un arxiu .exe, en cas de mac os .app)

Aquest ecosistema d'aplicacions que convertiran el codi font en un executable pel nostre sistema operatiu venen en una aplicació anomenada java JDK.

4.2.1 Java JDK:

Java JDK (Java Development Kit) és un instal·lador de tots els diferents programes necessaris per a poder programar i executar un codi font Java en qualsevol sistema operatiu.

Aquests programes indispensables per a crear un executable programat en Java i que el JDK inclou són els següents:

.- JavaFX (SDK): Proporciona diferents ordres i tecnologies per a poder desenvolupar aplicacions relacionades amb navegadors, ordinadors o dispositius mòbils.

.- Java Mission Control: Eina que proporciona tot tipus d'informació (collecta d'errors i funcionament de l'executable o l'aplicació) al desenvolupador.

.- Private Java JRE : És utilitzat per a la compilació d'una aplicació Java feta per un usuari.

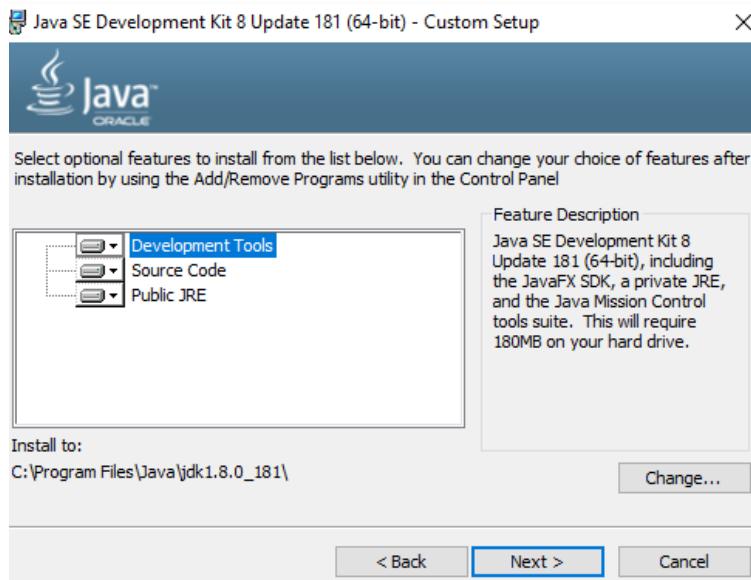


Figura 7, instal·lació de java JDK

.- Source Code : Instal·la l'API (Application Programming Interface, o en Català, Interfície de programació d'aplicacions) necessària per a poder treballar amb totes les opcions que els desenvolupadors del llenguatge de programació Java han preparat per a l'escriptura del codi font.

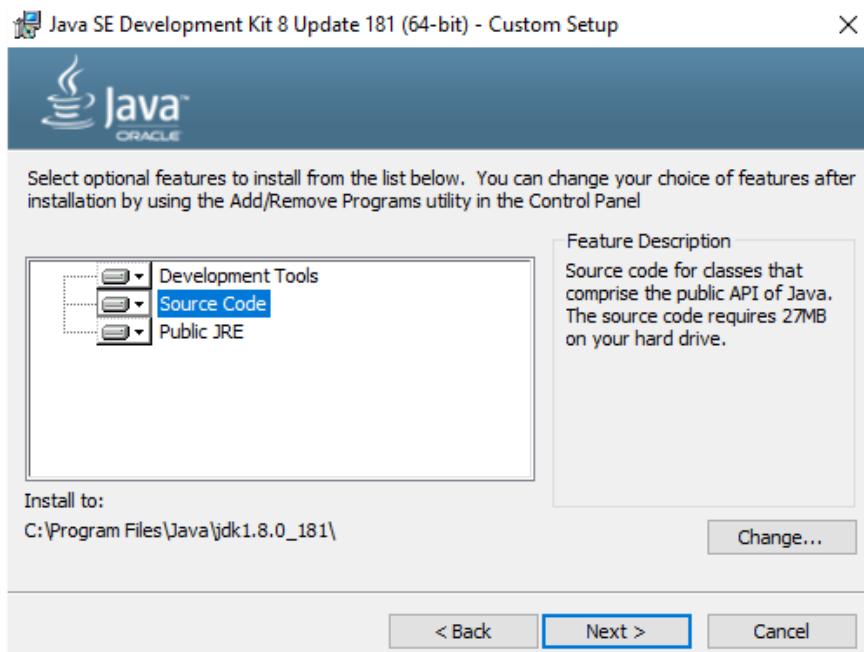


Figura 8, instal·lació de Java JDK (Windows)

.- Public Java JRE: És utilitzat per totes les aplicacions Java (no únicament per compilar les del usuari) a més a més d'incloure un plug-in per al navegador d'internet en cas de que una pàgina tingui que fer anar un codi Java o Java-script (variant de Java per a pàgines html, normalment pàgines web).

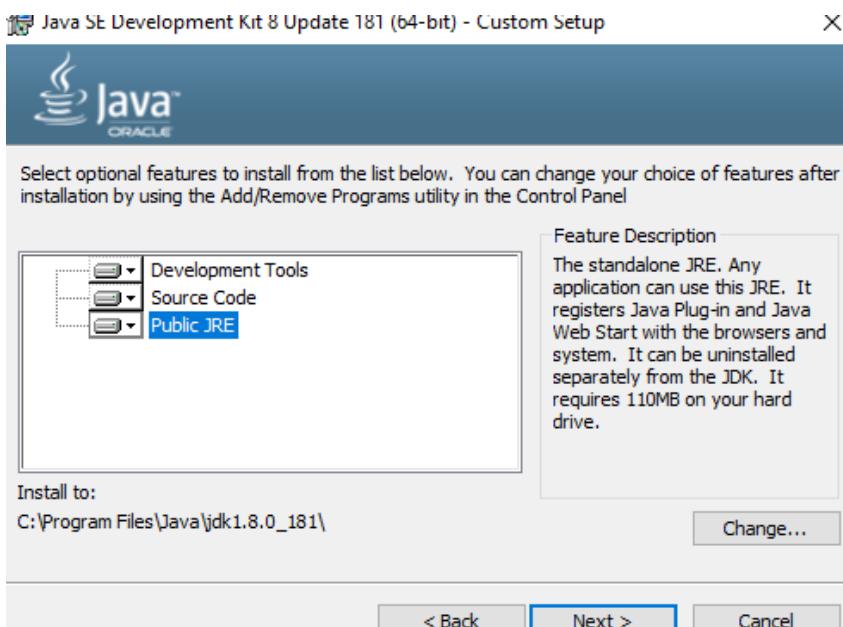


Figura 9, instal·lació de Java JDK a Windows

4.3 Creació de l'aplicació a partir del codi font

Una vegada explicades totes les aplicacions necessàries per a executar una aplicació Java i després d'entendre què és el codi font, cal explicar com es passa de les lletres de codi, a un executable totalment funcional en qualsevol sistema operatiu.

En el cas d'aquest llenguatge de programació el que fa que sigui compatible amb tots els dispositius no es el codi en si, si no l'aplicació JRE que inclou el paquet d'aplicacions de Java JDK.

Aquesta aplicació és única per cada sistema operatiu, i de moment està disponible per a Windows, Mac OS i Linux. Això vol dir que si Java JRE (o una variant d'aquesta, com per exemple Dalvik o ART, en el sistema operatiu Android) no és compatible amb un sistema operatiu, aquest no podrà executar cap aplicació programada amb Java, al menys fins que s'adapti una versió de la Java Virtual Machine al sistema.

4.3.1 Procés de compilació

La compilació és el procés de convertir les diferents classes definides en el codi font escrites en el llenguatge de programació java en un arxiu en format ".class". Els arxius class contenen dins un codi binari especialitzat per a la Java Virtual Machine anomenat ByteCode. Com el codi binari ByteCode es lleigble per la Java Virtual Machine, aquest, continua sent multiplataforma en tots aquells sistemes que suportin la JVM.

El compilador que s'utilitza per a passar d'un arxiu .java a un arxiu .class s'anomena javac (de les sigles java compiler). Aquest programa ha d'executar una finestra de comandaments, tant a Windows, Mac o Linux.

En aquest cas, per a comprovar com funciona el procés de compilació java, s'ha utilitzat un sistema operatiu Linux (ja que sol ser més fàcil en quant a comandaments i compilació en diferents llenguatges de programació). El comandament seria el mateix en Mac OS i en Windows.

A la figura 2.3.1.1, es pot observar un arxiu anomenat java-example.java, i una finestra de comandaments on hi ha el nom del compilador (javac) i l'arxiu que volem compilar.

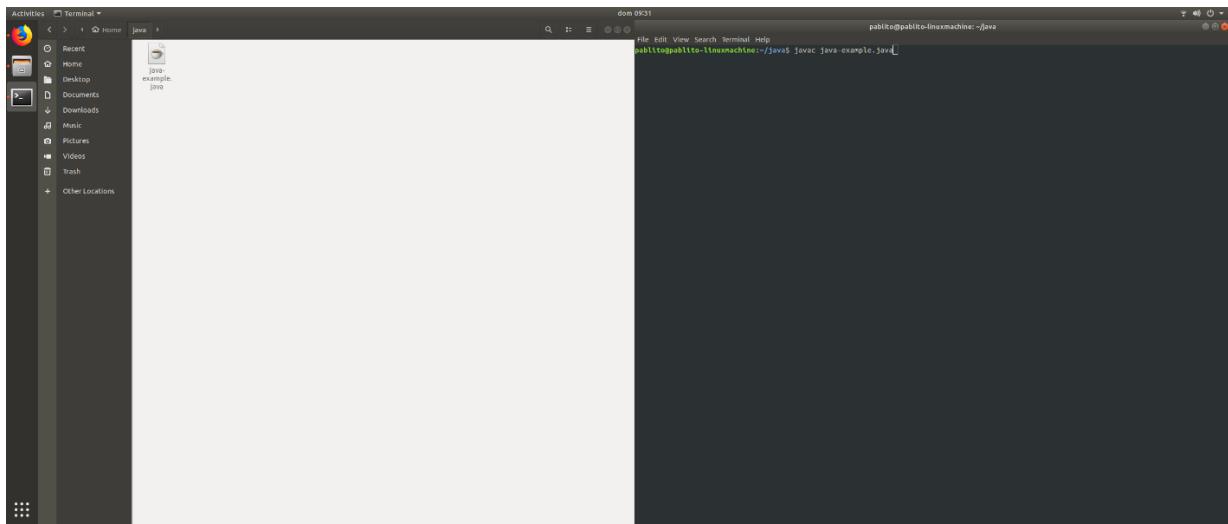


Figura 10, comandament Javac

Una vegada executem el comandament, es pot apreciar a la figura 2.3.1.2 que automàticament es crea un arxiu amb el nom de la classe programada en el arxiu `java-example.java` . Això és perquè el compilador crearà tants arxius en bytecode com classes hi hagin a l'arxiu `.java` (ja que les classes són les encarregades de donar ordres als programes) .

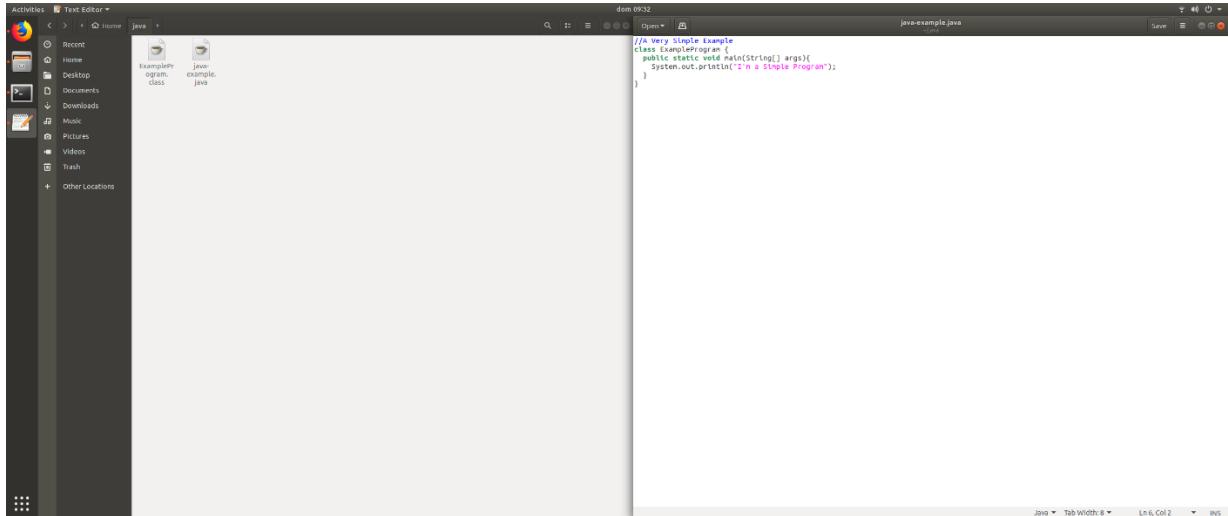


Figura 11, creació d'un arxiu `.class`

En el cas de que es vulgui obrir un arxiu `.class` (figura 2.3.1.3) , el Sistema no serà capaç de llegir-lo, ja que és un idioma màquina que només la Java Virtual Machine interpreta.

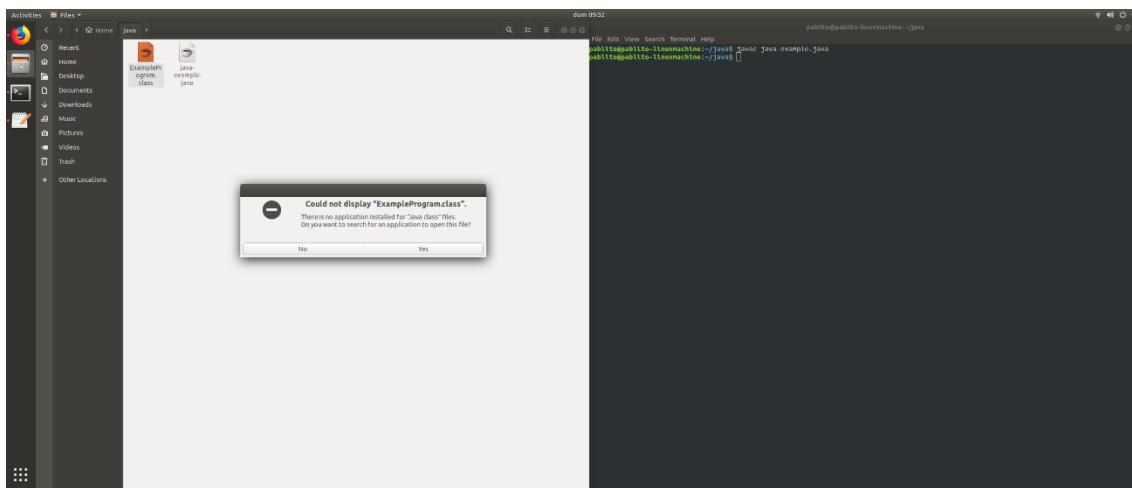


Figura 12, intent d'obrir un arxiu class

Quan es té múltiples classes, i per tant, múltiples arxius .class, es comprimeixen tots en un arxiu .jar, que prové de les sigles Java ARchive (que en català vol dir arxiu de java) que és una variació d'un arxiu .zip. El arxiu .jar, és igualment llegit per la Java Virtual Machine.

4.4 Java API

La paraula API prové de les sigles Application Programming Interface, que en català significa Interfície de Programació d'Aplicacions. En el cas de Java, una API es una llista de totes les classes predefinides que són part del Java Development Kit. L'API inclou tots els paquets de Java, classes, interfícies junt amb els seus mètodes i constructors.

Totes aquestes classes pre-escrites que són guardades en l'API de Java són de gran ajuda per a els programadors, i aquests tenen que estar atents a les diferents actualitzacions o modificacions d'aquestes. Totes les diferents classes que venen instal·lades amb el Java JDK es poden trobar en el següent document de Oracle: <https://docs.oracle.com/javase/7/docs/api/>

En cas que algun programador vulgui utilitzar una classe de la Java API, necessita incloure una declaració que està important aquesta classe al seu codi al principi del seu codi font. Per exemple, si es vol utilitzar la classe “ Scanner” que dona permís al programa per acceptar text escrit pel usuari amb el teclat, es tindria que incloure la següent línia de codi al codi font:

```
import java.util.Scanner;
```

Aquesta declaració d'importació de la classe permet que el programador pugui utilitzar qualsevol mètode inclòs en la classe Scanner.

En cas del java JDK, sol incloure en tots els arxius del codi font, diferents declaracions d'importació que s'utilitzen molt sovint, com per exemple java.util o java.lang (que incorporen les classes de les variables, classes de suma,resta,etc).

5. Android Studio :

5.1 Introducció a Android Studio.

Android Studio:

Android Studio és el IDE oficial per a la creació d'aplicacions Android. Aquest IDE (Integrated Development Environment, també anomenat ambient integrat per a desenvolupar) està basat en el projecte IntelliJ IDEA, un dels projectes IDE més grans per a diferents llenguatges de programació i sobretot Kotlin (un llenguatge de programació basat en Java).

Android Studio, incorpora diferents característiques i utilitats per a la creació d'aplicacions com per exemple:

- .- Un sistema de construcció amb la eina Gradle
- .- Un emulador Android per a poder provar la app sense un dispositiu
- .- Un ambient unificat on es pot desenvolupar per a tots els dispositius android
- .- Opció d'aplicar canvis a la app sense haver de compilar tota la apk de nou
- .- Plantilles de codi i la integració de la pàgina web github per a poder importar diferents parts de codi d'altres usuaris
- .- Una varietat d'eines per a testejar l'app i els frameworks
- .- Eines per a poder testejar la usabilitat, la compatibilitat i la potència de l'aplicació.
- .- Compatibilitat amb altres llenguatges de programació com C++ o NDK
- .- Integració amb la plataforma de google cloud, fent més senzill d'integrar serveis de google com per exemple el Google Cloud Messaging

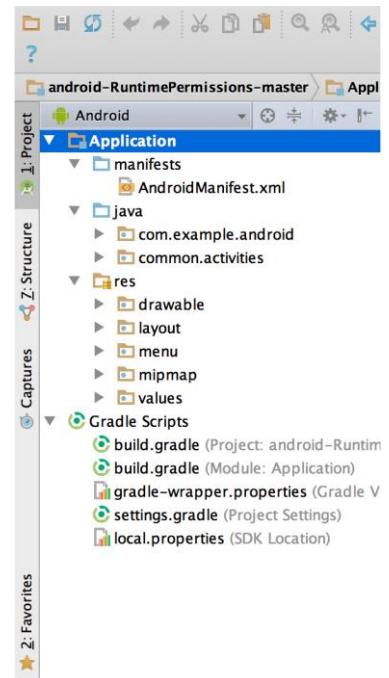


Figura 13, Estructura d'Android Studio

Estructura d'un projecte en Android Studio:

Cada projecte d'Android Studio conté un o més mòduls de codi font i arxius de Recursos. Els arxius de Recursos són tots aquells arxius addicionals i contingut estàtic que el codi font utilitza, com per exemple tota la part gràfica (animacions, interfície, etc.)

Els diferents tipus de mòduls que podem trobar inclouen:

- Mòduls de l'Android App
- Mòduls de Llibreries.
- Mòduls del Google App Engine

De base, Android Studio mostra els arxius del projecte en l'apartat d'Android Studio anomenat la vista del projecte Android (figura 4.1.1). Aquesta vista organitza els arxius per mòduls per a proporcionar una vista més ràpida i ordenada.

Tots els arxius de compilació són visibles davall de Gradle Scripts, mentre que a la part superior es pot trobar els diferents mòduls de la app organitzats en les següents carpetes:

- manifests: Conté l'arxiu AndroidManifest.xml
- java: Conté el o els codis fonts programats en Java
- res: Inclou tots els recursos que no varien, com per exemple la interfície d'usuari, imatges, etc.

La vista de projecte Android pot variar depenent de la opció que se li encomana, per exemple, com es pot apreciar a la figura 4.1.1, per a veure l'estruccura del projecte, estem utilitzant la vista Android però també es poden utilitzar diferents vistes depenent de la situació. Per exemple, si es selecciona la vista “Problems” (figura 4.1.2) es mostren tots els arxius que contenen algun error sintàctic o elements no declarats.

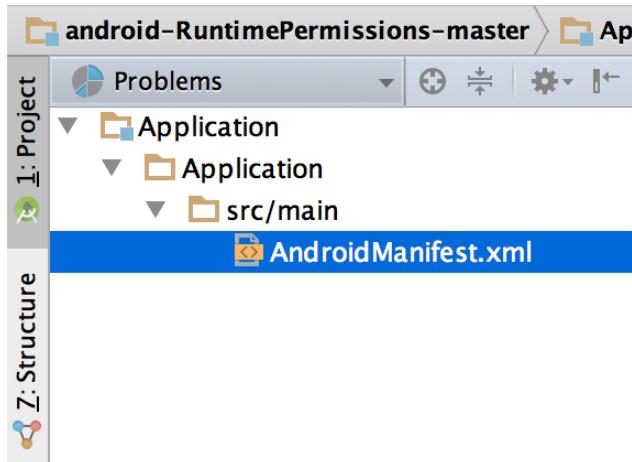


Figura 14, arxius Android Studio

Interfície d'usuari en Android Studio:

La finestra principal d'Android Studio està composta de diferents àrees, com es pot apreciar en la figura 4.1.3.

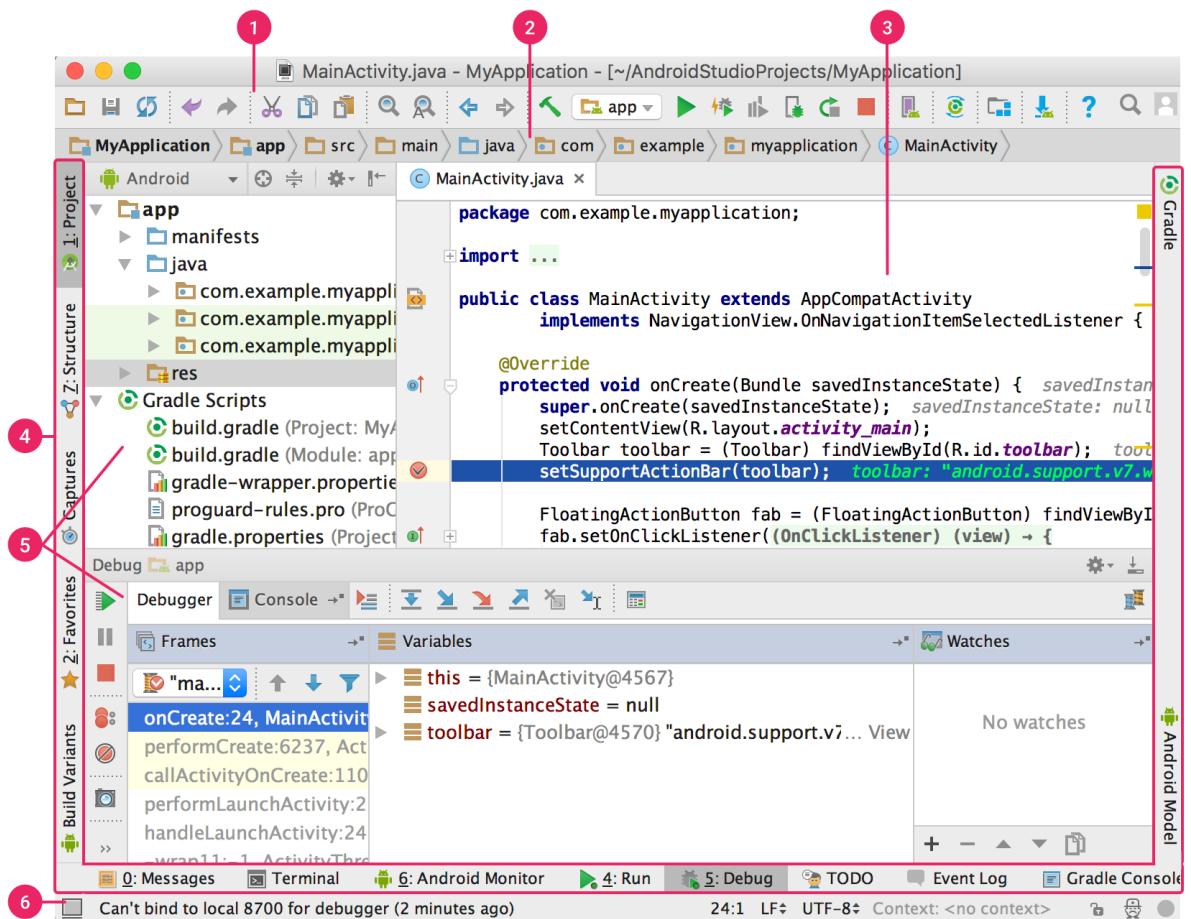


Figura 15, interfície d'usuari d'Android Studioç

- 1) La barra d'eines proporciona diferents accions, com per exemple executar l'aplicació des de l'emulador inclòs, o simplement compilat l'aplicació.
- 2) La barra de navegació, ajuda a l'hora de navegar pels diferents arxius dins del propi projecte. Incorpora una vista reduïda de la finestra de la vista del projecte.
- 3) La finestra d'edició: És on es pot modificar i crear el diferent codi. Depenent del tipus de codi modificat, la finestra pot canviar. Per exemple si s'està modificant una part gràfica de l'aplicació, la finestra serà diferent que si s'està programant el codi font.
- 4) La barra de finestres d'eines, es troba al voltant d'Android Studio i conté els botons que permeten expandir o minimitzar diferents finestres amb eines.
- 5) L'eina de finestres, dona accés específic a tasques com la gestió de projectes, recerca, versió de l'aplicació, etc.
- 6) La barra d'estat treu per pantalla els estat del projecte i de el IDE, així com també diferents avisos o missatges.

5.2 Android SDK

El Android SDK (Software Development Kit, en català kit de desenvolupament de programació) és una composició de múltiples aplicacions que són requerides pel desenvolupament de la plataforma Android.

Dins del Android SDK, es pot trobar que depenen de la versió d'Android per a la que es vulgui programar una aplicació, s'haurà de baixar una versió del SDK o un altra, ja que cada versió del SDK inclou les diferents API per a poder programar l'app, i si es programa una aplicació amb una API més antiga de la que pertoca al dispositiu, no es podran afegir a l'aplicació totes les funcions que la versió d'Android incorpora.

5.3 Gradle

Gradle és una eïna que automatitza al compilació del codi font creada per a millorar la rapidesa en la que es compila el codi i personalitzar el procés de compilació.

Android Studio utilitza Gradle per automatitzar i administrar el procés de compilació, al mateix temps que permet definir configuracions de compilació personalitzades i flexibles.

Gradle se executa independentment de Android Studio, per tant, el resultat de compilació serà el mateix si es compila el projecte des de una línia de comandament, una màquina remota o utilitzant Android Studio

5.4 El procés de Compilació:

El procés de compilació per a una aplicació per Android segueix els següents passos:

- 1) El compilador converteix el codi font en arxius DEX (executables per la Màquina Virtual Dalvik) els quals inclouen el codi de bytes que s'executa en dispositius Android, i en tots els altres recursos compilats.
- 2) L'empaquetador de els arxius APK combina els arxius DEX i els recursos compilats en un únic APK. No obstant, abans de que una aplicació es pugui instal·lar e implementar en un dispositiu Android, es té que firmar l'APK.
- 3) L'empaquetador d'arxius APK, firma de forma diferent el APK dependent de la versió compilada.
 - .- Si es compila una versió depuració de l'aplicació (és a dir, una aplicació que únicament es vulgui utilitzar per a proves) l'empaquetador firmarà l'aplicació amb una clau de depuració. Android Studio configura automàticament nous projectes amb una clau de depuració.
 - .- Si es compila una versió de llançament (una aplicació que es vulgui instal·lar de manera externa, és a dir, com a versió final) l'empaquetador firmarà l'aplicació amb una clau de llançament, creada pel desenvolupador de l'aplicació.

Al final del procés de compilació, s'aconseguirà un arxiu APK que es podrà executar en un dispositiu Android

6. Java adaptat a Android:

6.1 Funcionament d'Android

Segons Android Developers (pàgina oficial de desenvolupadors d'Android), Android és un conjunt de software de codi lliure basat en Linux per a una varietat amplia de dispositius i factors de forma. En la figura 16 es mostren tots els components principals de la plataforma Android.



Figura 16, funcionament d'Android

Kernel de Linux:

La base de la plataforma Android es el kernel de Linux. El kernel és la part del telèfon on es mantenen tots els controladors del hardware que un dispositiu té incorporat, fent així que siguin llegits pel HAL . A més a més, ART la màquina virtual d'Android, també es basa en el kernel de Linux per a funcionalitats subjacentes com la generació de subprocessos i l'administració de memòria de baix nivell.

A més a més, l'ús del kernel de Linux permet que Android aprofiti funcions de seguretat com per exemple IPC (comunicació segura entre processos) que facilita la comunicació segura entre aplicacions que funcionen en diferents processos.

Capa d'abstracció de Hardware (HAL) :

La capa d'abstracció de hardware (HAL) brinda interfícies estàndards que exposen les capacitats de hardware del dispositiu al framework de la Java API de nivell més alt. La HAL consisteix en diferents mòduls de biblioteca on cada un d'aquests incorpora una interfície per a un component de hardware específic, com per exemple el mòdul de càmera o de Bluetooth. Quant el framework d'una API realitza una trucada per accedir al hardware del dispositiu, el sistema Android carga el mòdul de biblioteca pel component de hardware en qüestió.

Temps d'execució d'Android :

Pels dispositius amb Android 5.0 (Nivell d'API 21) o versions posteriors, cada aplicació executa els seus propis processos amb ART. El ART està programat per executar diferents màquines virtuals en dispositius de memòria baixa executant arxius en format DEX, un format de codi de bytes (com seria bytecode) però dissenyat especialment per a Android i optimitzat per a ocupar un espai de memòria mínim. Els arxius DEX, són els que es poden executar en la plataforma Android.

Algunes de les funcions principals del ART són les següents:

- Tipus de compilació ahead-of-time (AOT) i just-in-time (JIT)
- Compatibilitat amb la depuració d'errors, informes d'errors i la capacitat d'establir punts de controls per a controlar determinades parts del sistema.

Abans d'Android 5.0 (API 21), Dalvik era el temps d'execució del sistema operatiu. Si una aplicació s'executa bé en ART, també tindria que funcionar bé en Dalvik, encara que és possible que no succeeixi el contrari

Biblioteques C/C++ natives

Molts dels components i serveis centrals del sistema Android, tal com ART i HAL, es basen en codi natiu que requereix les biblioteques natives escriptes en C i C++. La plataforma Android proporciona la API del framework Java per a exposar la funcionalitat d'algunes d'aquestes biblioteques natives a les aplicacions. Per exemple, es pot accedir a OpenGL ES (llibreria per a gràfics 2D i 3D) a través de la Java OpenGL API que incorpora el framework d'Android per a que les aplicacions puguin agregar compatibilitat amb dibuixos i gràfics.

Framework de la Java API:

Tot el conjunt de funcions del Sistema Operatiu (S.O) està disponible mitjançant APIS escriptes en el llenguatge Java. Aquestes APIS son els ciments que es necessiten per a crear aplicacions Android simplificant la reutilització de components del sistema i serveis centrals i modulars, com les següents:

- Un sistema de vista: Enrichidor i extensible que es pot utilitzar per a compilar la interfície d'usuari (IU) d'una aplicació. Aquesta inclou llistes, quadrícules, quadrats de text, botons e inclòs un navegador web integrable. Aquest sistema de vista es pot incorporar a la aplicació amb els objectes View i ViewGroup.
- Un administrador de recursos que brinda accés als recursos (res) de l'aplicació, com per arxius de disseny.
- Un administrador de notificacions que permet que totes les aplicacions mostrin alertes personalitzades en la barra de notificacions del Sistema Operatiu.
- Un administrador d'activitat que administra el cicle de vida de les aplicacions i proporciona una pila d'activitats comuna.
- Proveïdors de contingut que permeten que les aplicacions accedeixin a dades des d' altres aplicacions com per exemple que una aplicació agafi dades de l'aplicació de contactes o comparteixin les seves pròpies dades.

Els desenvolupadors tenen accés total a les mateixes APIS del framework que utilitzen les aplicacions del sistema Android. Tots els paquets d'APIS proporcionats pels desenvolupadors

d'Android es poden trobar aquí: <https://developer.android.com/reference/packages?hl=es-419>

Aplicacions del sistema :

Android inclou un conjunt d'aplicacions centrals per a correu electrònic, missatgeria SMS, calendari, navegació en internet, contactes, entre altres elements. Les aplicacions incloses en la plataforma no tenen un estat especial entre les aplicacions que el usuari decideix instal·lar. Per això, una aplicació externa es pot convertir en el navegador web, el sistema de missatgeria o inclòs en el teclat predeterminat per l'usuari (encara que existeixen algunes excepcions com per exemple l'aplicació configuració del sistema)

Les aplicacions del sistema funcionen com aplicacions per a usuaris i brinden capacitats claus a les quals els desenvolupadors poden accedir des de les seves propis aplicacions. Per exemple, si una aplicació intenta integrar un missatge SMS, no es necessari compilar aquesta funcionalitat un altre cop; com alternativa es pot invocar l'aplicació de SMS que ja està instal·lada per entregar un missatge al receptor especificat.

6.2 Funcionament del Android Runtime (ART)

Android Runtime (ART) és la màquina virtual utilitzada per les aplicacions i alguns serveis del sistema en el sistema operatiu Android. ART i el seu predecessor Dalvik van ser originalment creats pel projecte Android. ART pot executar els formats Dalvik Executable i DEX bytecode.

ART i Dalvik tenen compatibilitat executant arxius en format DEX Bytecode, així que aplicacions desenvolupades per a l'arquitectura de màquina virtual Dalvik, tindrien que funcionar amb ART. Malgrat això, algunes tècniques que funcionen amb Dalvik, no tenen perquè funcionar amb ART

Millores i afegits de l'arquitectura de la Màquina Virtual ART respecte Dalvik:

- Mètode de Compilació:**

ART introduceix el tipus de compilació anomenat ahead-of-time (AOT)

Aquest nou tipus de compilació, incrementa el rendiment de les aplicacions respecte el mètode Just In Time (JIT) utilitzat per Dalvik.

6.3 Fonaments de la creació d'una aplicació Android:

*** Els components de una aplicació Android:**

Els components d'una aplicació són blocs essencials per a crear una aplicació Android. Cada component és un punt d'entrada a través del qual el sistema o el usuari pot ingressar a una aplicació. Alguns components depenen d'uns altres.

Hi han quatre tipus de components:

- Activitats (“Activities”): Són el punt d'entrada per interactuar amb el usuari. És representat per una sola pantalla amb una interfície d'usuari. Per exemple, una aplicació de correus, segurament té una activitat que mostra una llista amb nous correus, una altra activitat per a escriure un correu, una altra activitat per llegir els correus,etc.
- Serveis (“Services”): És un punt d'entrada amb el propòsit general de mantenir una aplicació executant-se en segon pla. Aquest fet pot passar per múltiples motius, com per exemple per sincronitzar dades (de correu per exemple) mentre el telèfon no s'està utilitzant per el usuari.
- Receptors de difusió (“BroadCast Receivers”): És un component que permet que el sistema lliuri esdeveniments a l'aplicació sense un ús de l'usuari, com per exemple la notificació de bateria baixa o la notificació de que s'ha realitzat una captura de pantalla.
- Proveïdors de continguts (“Content Providers”): Gestiona un conjunt compartit de dades de l'aplicació que es pot emmagatzemar en el sistema de fitxers, a la web o a qualsevol emmagatzematge que pugui accedir l'aplicació.

El Android Manifest:

Abans de que el sistema operatiu Android pugui executar un component, el sistema ha de saber que el component existeix llegint el Manifest de l'aplicació (AndroidManifest.xml). L'aplicació ha de declarar tots els components en aquest arxiu.

A més, el Android Manifest també declara:

- Els permisos que necessita l'aplicació per funcionar (llegir el emmagatzematge , internet, etc)
- El nivell d'API mínim per a poder executar la aplicació
- Declarar característiques de software o hardware utilitzades per la aplicació com per exemple la càmera o el Bluetooth
- Declara llibreries API que no tenen relació amb les APIs incorporades amb Android.

6.4 Recursos de l'aplicació

Els recursos són arxius addicionals i de contingut estàtic que el codi font utilitza, com per exemple Strings per a la interfície d'Usuari, animacions, etc.

La recomanació d' Android Developers és que els arxius de recurs es guardin en un subdirectorí dintre del projecte anomenat “res” (que és on es guarden de forma predeterminada en Android Studio)

Els diferents directoris suportats dintre de el directori res són els següents:

- animator/: Arxius XML que defineixen animacions amb propietat
- anim/: Arxius XML que defineixen animacions bàsiques
- color/: Arxius XML que defineixen una llista de colors
- drawable/: Arxius (.png , .9.png, .jpg, .gif)
- mipmap/: Arxius “drawable” però amb diferents densitats
- layout/: Arxius XML que defineixen la interfície d'usuari.
- menu/: Arxius XML que defineixen menús.
- raw/:
- values/: Arxius XML com per exemple arrays.xml, colors.xml, strings.xml i styles.xml
- xml/: Arxius XML que poden ser llegits al mateix cop que el programa s'executa
- font/: Arxius que contenen diferents tipus de lletra

Depenent del projecte que es vulgui crear es poden crear diferents arxius Strings i per tant, es poden traduir l'aplicació a diferents idiomes.

Per accedir a un recurs des de el codi font s'utilitzarà la referència amb el ID. Tots els recursos tenen un ID que es poden cridar de la següent forma:

R. "tipus de directori" . "id del recurs".

Un exemple seria: R.string.hola

6.5 Recurs Layout:

Un dels recursos més importants és el Recurs Layout, que és aquell recurs que defineix l'estructura de la interfície de usuari d'una activitat o un component de la interfície d'usuari. Depenent de la interfície que es vulgui presentar a l'usuari es pot utilitzar un tipus de layout o un altre. Els més coneguts són:

- Relative Layout: Sol ser el més utilitzat donat a totes les funcions que incorpora. Consisteix en un grup de visualització que mostra visualitzacions secundàries en posicions relatives. La posició de cada element es pot especificar en relació amb els elements del costat o amb àrees primàries com per exemple la part inferior, esquerra o centre de la pantalla.



Figura 17, exemple de Relative Layout

- Grid View: Mostra elements en un tipus de reixa; va perfecte per a separar elements en parts iguals.

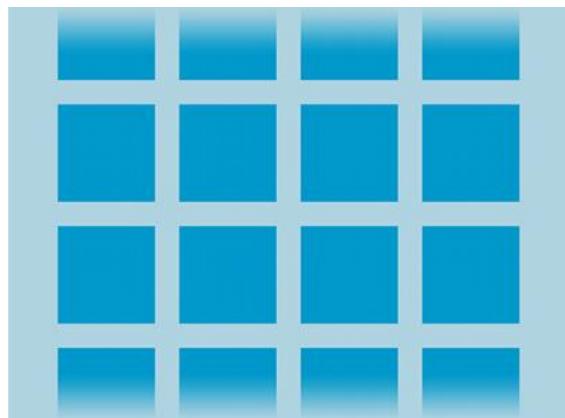


Figura 18, exemple de Grid View

- Linear Layout: Alinea tots els elements en una única direcció. Verticalment o horitzontalment.

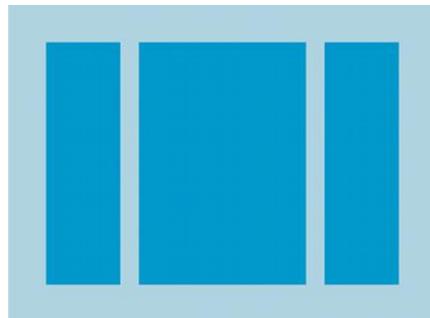


Figura 19, exemple de Linear Layout

PART PRÀCTICA

7. Calculadora.

7.1 Creació del projecte:

Per a crear el projecte es va obrir l'aplicació Android Studio, i seleccionar l'opció de crear un nou projecte (figura 20)

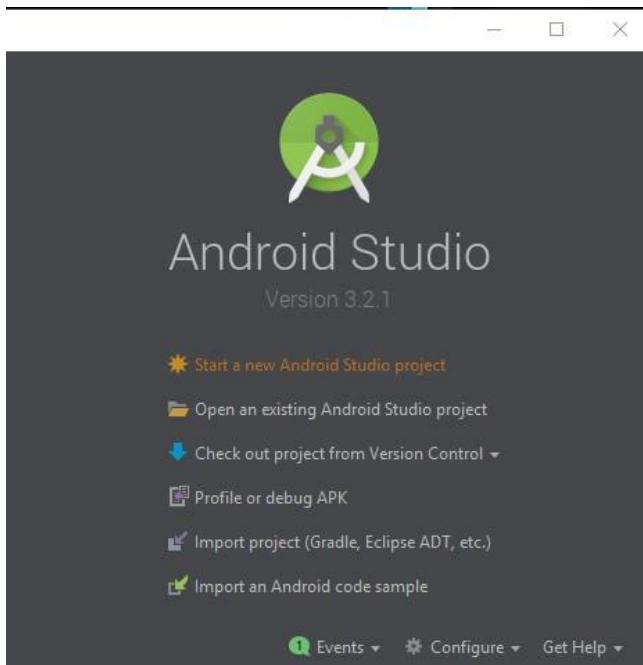


Figura 20, pantalla de benvinguda d'Android Studio

Després d'aquesta acció, va aparèixer una nova pantalla on es va escriure el domini de l'empresa (que com en aquest cas no existeix, aquest serà denominat com pablo.ordinador.com) i el nom de l'aplicació que es volia crear. (figura 21)

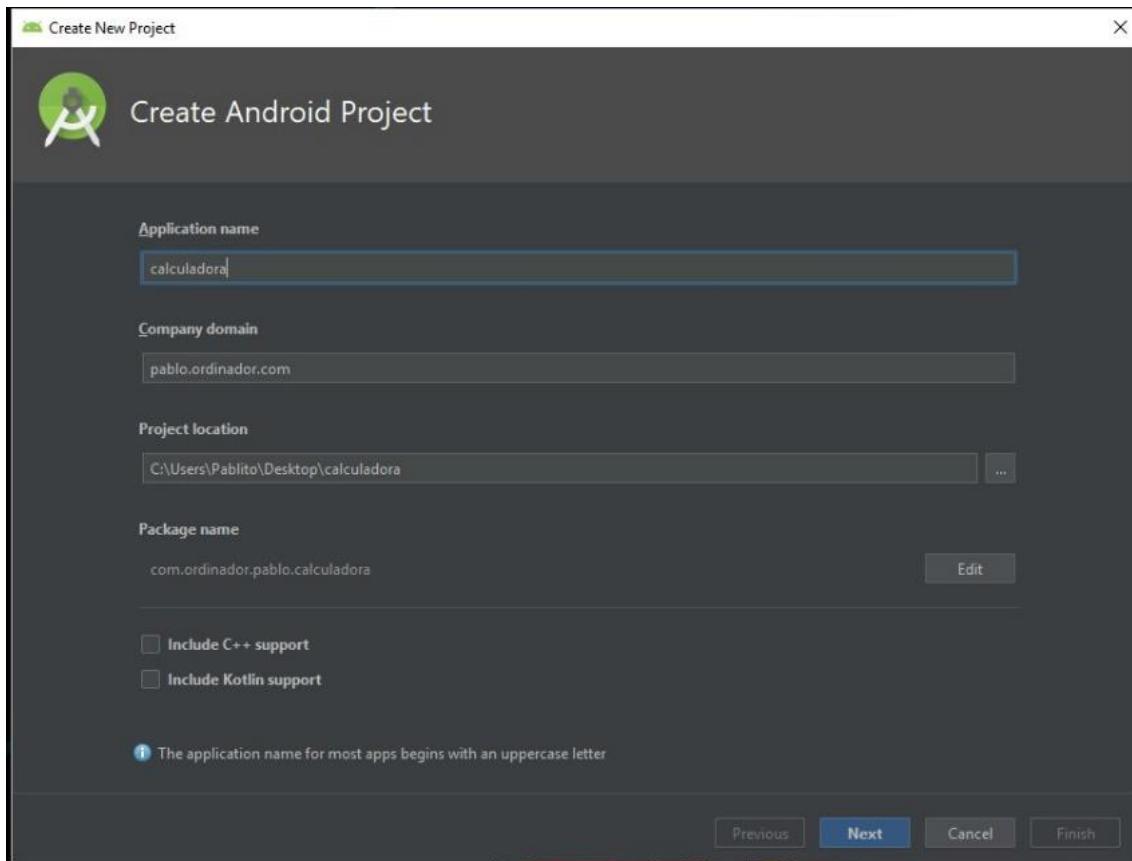


Figura 21, creació d'un projecte a Android Studio

Després de seleccionar el domini i el nom de l'aplicació, va aparèixer l'opció de triar la versió d'Android mínima per al funcionament de la app (figura 22) . En aquest cas, es va triar l'opció predeterminada per Android Studio, que és la API 22 la qual pertany a la versió d'Android 5.1. Ficar aquesta API com a la mínima preferida, fa que l'aplicació sigui compatible amb el 80% del dispositius Android.

Com l'aplicació es únicament per a telèfons, s'ignoraran les altres opcions què afegeixen la funcionalitat de suportar l'aplicació en rellotges intel·ligents, televisions o automòbils.

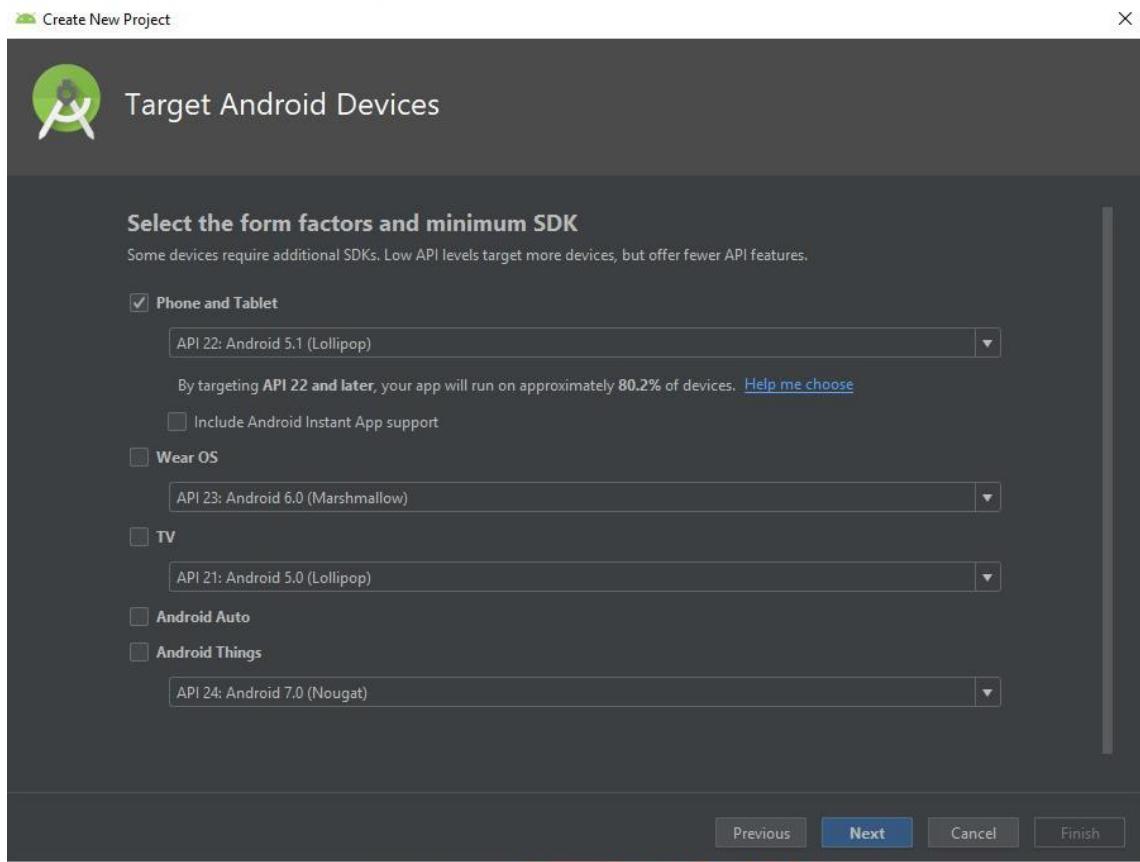


Figura 22, configuració del projecte d'Android Studio

Una vegada seleccionada l'API mínima per al funcionament de la aplicació, Android Studio va mostrar l'opció de canviar el nom de l'activitat principal de l'aplicació (figura 23). En aquest cas, es va decidir nombrar l'activitat principal amb el nom de `activitat_principal`, i el layout es va nombrar `activity_activitat_principal`.

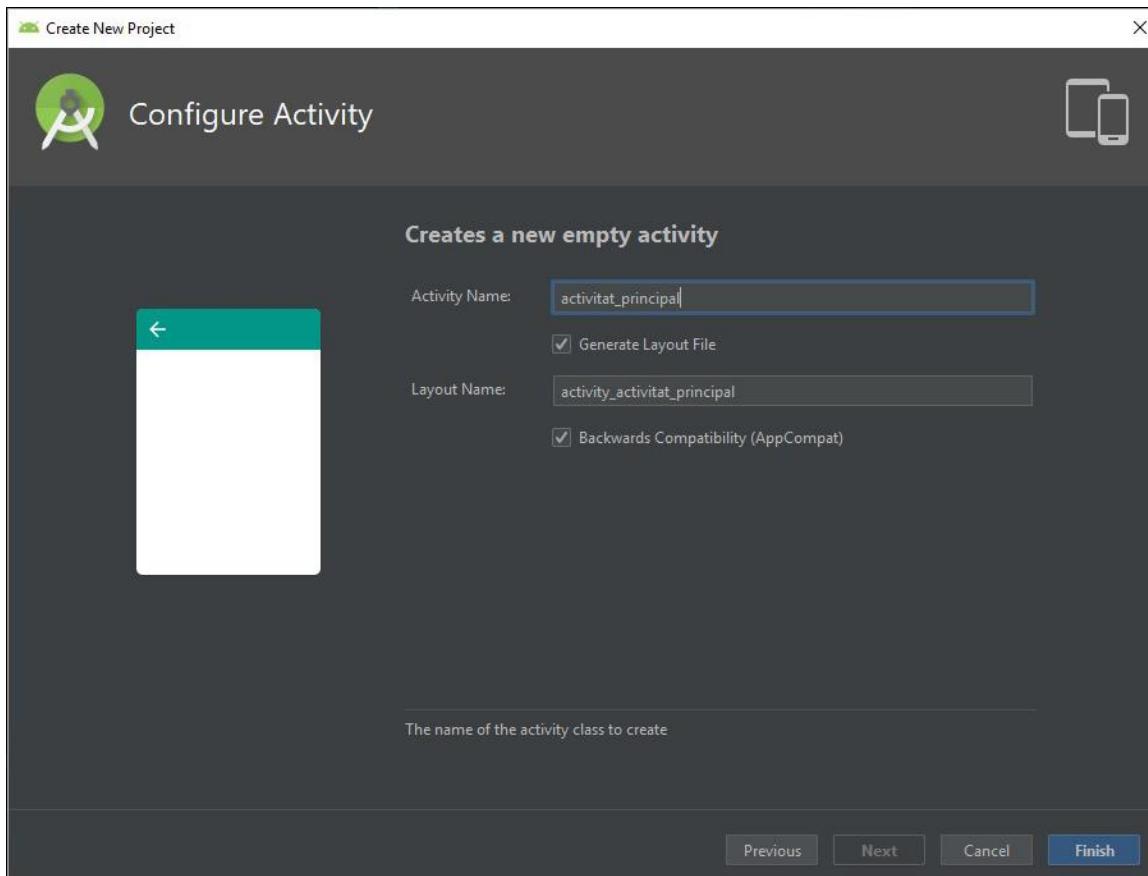


Figura 23, creació de la Activitat i el Layout

Per finalitzar amb la configuració, Android Studio va donar l'opció d'elegir diferents plantilles per a la activitat_principal (figura 24). En aquest cas, es va decidir triar una plantilla buida per tal d'anar afegint a posteriori els diferents ítems de la calculadora.

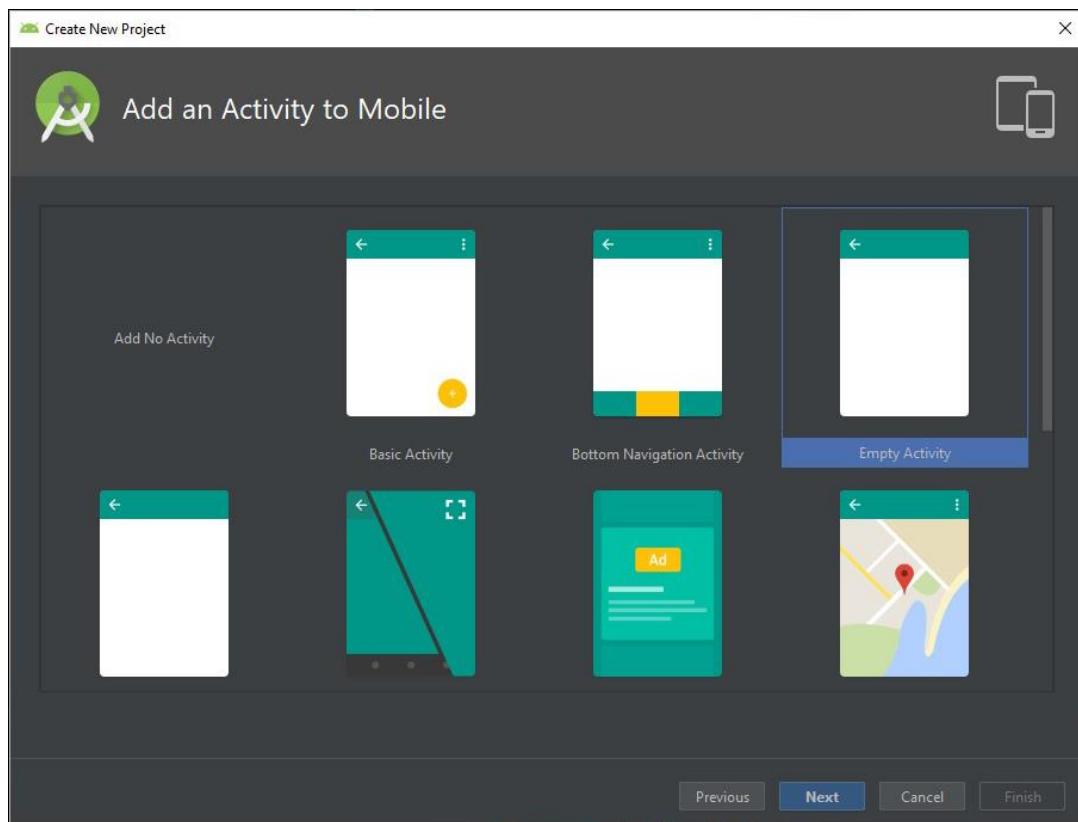


Figura 24, diferents plantilles oferides per Android Studio

7.2 Explicació de la interfície gràfica (layout)

Per a la calculadora, es van utilitzar dues diferents layouts i un menú. Cal recalcar que els dos layouts estaran utilitzant el model Relative Layout, ja que és el més personalitzable i el que inclou més opcions en quant a canvis de colors, funció que serà necessària per al mode obscur.

En quant a l'idioma del text, s'utilitzaran recursos Strings, amb l'idioma principal Anglès i després amb les seves respectives traduccions al Català i al Castellà.

Layout Principal:

El layout principal, serà el que doni la interfície de calculadora a l'aplicació. Estarà format per 19 botons (els quals indicaran els diferents nombres i signes), un textbox (el qual serà utilitzat per la entrada i sortida de els nombres) i un switch que serà utilitzat pel mode obscur de l'aplicació (figura 25)

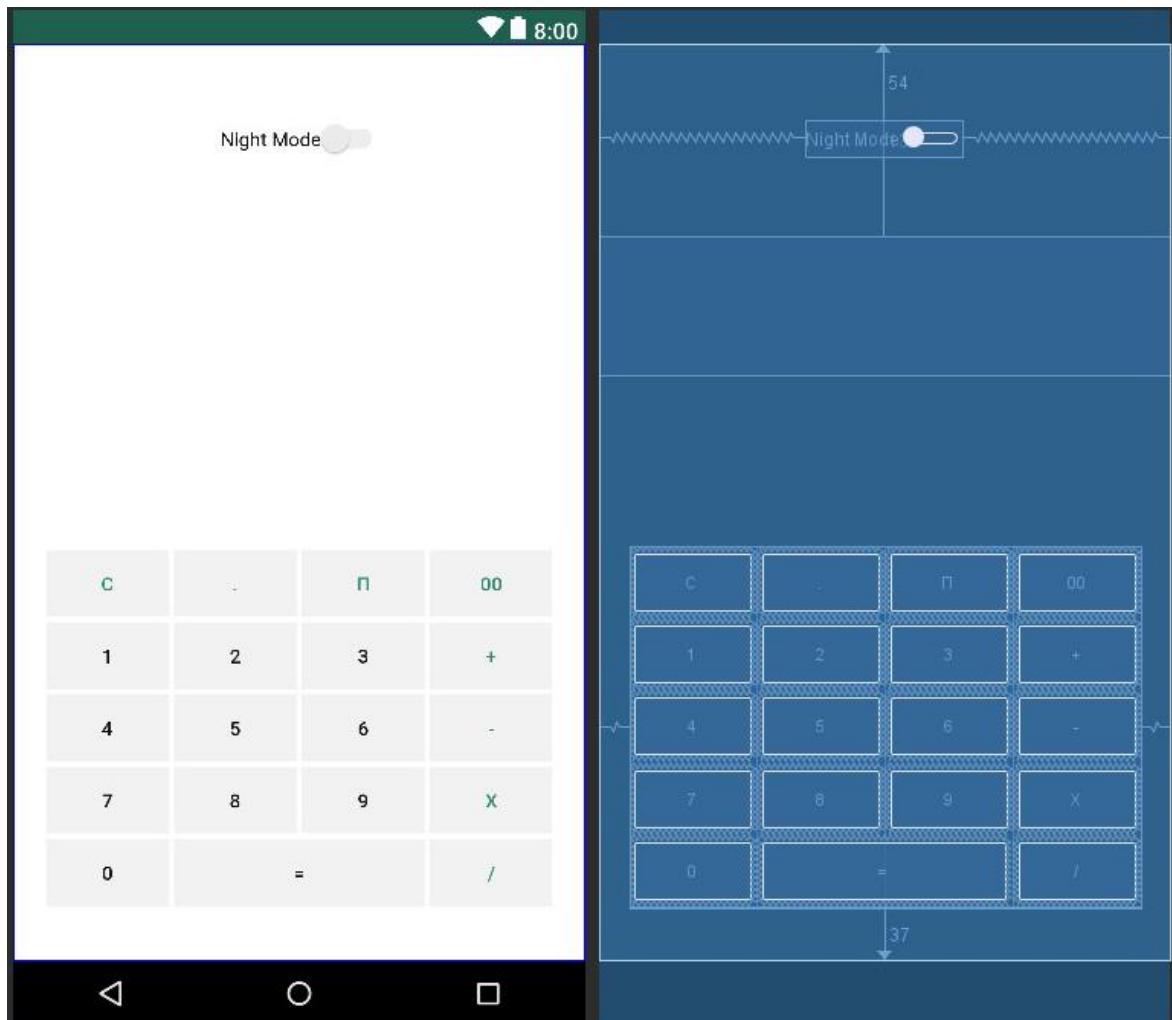


Figura 25, Layout_Principal

Encara que els objectes switch i textbox estiguin incorporats en el layout gràcies a l'editor de layouts d'Android Studio, els nombres es van haver d'introduïr en un GridLayout, el qual va estar creat dins del propi layout RelativeLayout (Figura 26).

GridLayout va permetre la incorporació de les fileres i columnes de botons d'una forma fàcil i eficaç. A més, a partir de l'editor text del XML d'Android Studio, es va poder afegir diferents paràmetres que no es permetien fer amb precisió l'editor de layouts d'Android Studio. Aquests paràmetres varen ser per exemple la separació de píxels de cada botó amb el més proper.

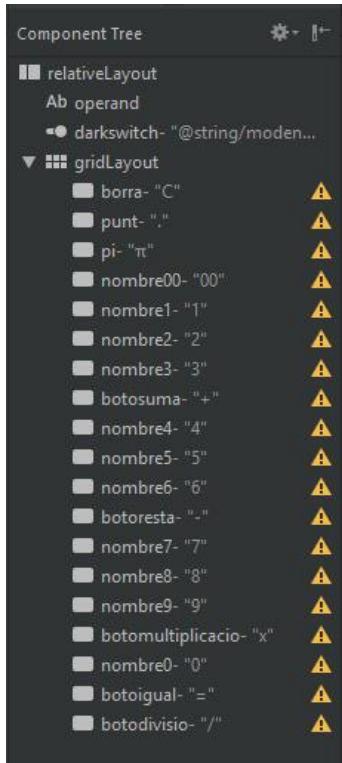


Figura 26, components del layout

En quant als diferents colors de l'aplicació i del text, es va haver d'acudir a diferents conceptes bàsics de xml, que es van trobar sense problema a la web d'Android Developers.

Primer, es va pensar que si es volia tenir un mode obscur, quan es premés el switch, caldria canviar el color de fons, dels botons, del text i del switch. Per tant, com hi havien tants valors a canviar durant l'activació del mode obscur i això amb el llenguatge Java ocuparia bastantes línies de codi, es va decidir incorporar un tema obscur al layout xml d'Android Studio. Per tant, quan es premés el switch, el layout canviaria al tema obscur i quan es desactivés el switch, es canviaria al tema clar (que és el predeterminat per l'aplicació).

Per tant, el primer va ser crear diferents atributs en un arxiu attrs.xml (figura 27). Aquests atributs , tots de format color, seran els encarregats de canviar depenent del tema seleccionat.

```

<?xml version="1.0" encoding="utf-8"?>

<resources>
    <declare-styleable name="ds">
        <attr name="backgroundcolor" format="color" />
        <attr name="buttoncolor" format="color" />
        <attr name="textcolor" format="color" />
        <attr name="colortextespecial" format="color" />
    </declare-styleable>
</resources>

```

Figura 27, atributs declarats en attrs.xml

Una vegada definits els atributs, es va haver d'assignar un arxiu .xml anomenat colors, el qual inclou el valor hexadecimal dels colors utilitzats tant en el tema obscur com en el tema clar (Figura 28). Com es pot apreciar, en l'aplicació hi haurà sempre colors constants, els quals són el color del switch, de la barra de notificacions, dels noms “especials” com són pi o els signes i el color de la barra superior de l'aplicació.

```

<?xml version="1.0" encoding="utf-8"?>
<resources>
    <!-- Colors constants -->
    <color name="colorbarra">#2d8e76</color>
    <color name="colorbaranot">#1f6050</color>
    <color name="colorswitch">#2d846e</color>
    <color name="colortextespecial">#2d8e76</color>

    <!-- Colors tema blanc -->
    <color name="colorfonsblanc">#ffffff</color>
    <color name="colorbotoblanc">#f2f2f2</color>
    <color name="colortextnegre">#111111</color>

    <!-- Colors tema negre -->
    <color name="colorfonsnegre">#333333</color>
    <color name="colorbotonegre">#444444</color>
    <color name="colortextblanc">#ffffff</color>
</resources>

```

Figura 28, colors.xml

Per finalitzar l'apartat dels colors, es va definir en l'arxiu “styles.xml” un nou tema anomenat “DarkTheme”. Aquest, canvia els atributs del fons, color dels botons, text i el color del text especial als colors del tema obscur definits en colors.xml (Figura 29)

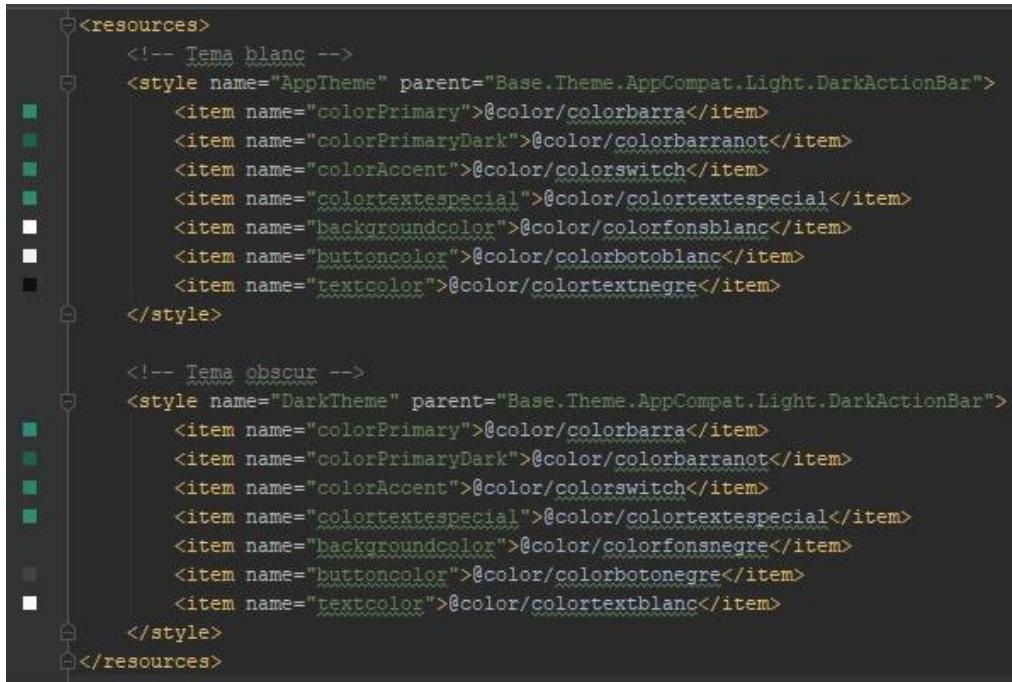


Figura 29, styles.xml

Per finalitzar amb els colors de l'aplicació en el layout, va ser necessari declarar el fons i la lletra dels botons i el text de l' "activitat_principal" als atributs nombrats en attrs.xml. Per a realitzar aquesta acció, es va clicar a cada botó i textView on es volia que canvies el color durant el mode obscur i es van canviar els atributs de fons (background) i de color del text (textColor) pels atributs definits a attrs.xml (figura 30).

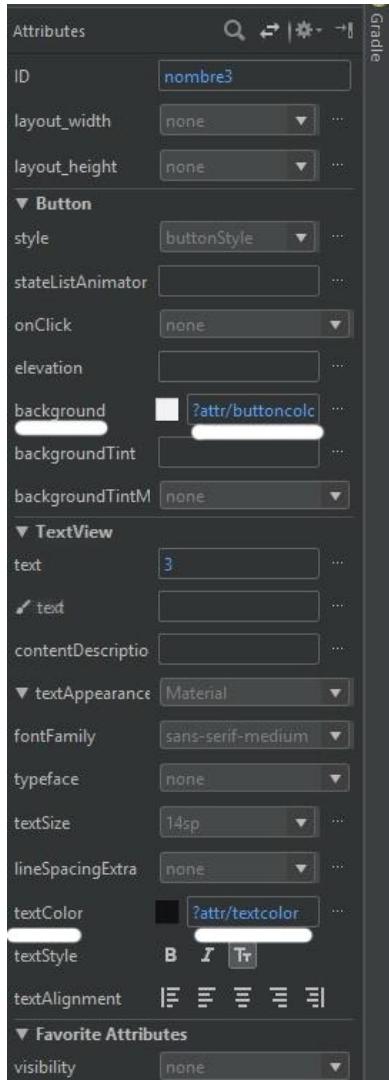


Figura 30, valors dels atributs per a el text i el fons

Menú:

Per a poder mostrar una nova activitat anomenada “sobre l’aplicació” a l’usuari, es va decidir implementar un apartat de menú a l’aplicació. Aquesta implementació inclou creacions d’arxiu xml i canvis en el codi font. Aquests últims seran explicats en l’explicació del codi font. Cal recalcar que els arxius de menú xml, estan basats en els exemples donats en la guia de menús de la web Android Developers.

Per a crear l’arxiu menu.xml, es va fer clic dret a la carpeta res, i es va seleccionar crear un nou arxiu de recurs (Figura 31) . Es va nombrar l’arxiu com menu i es va seleccionar el tipus de recurs com “Menu”. Un cop fet aquest procediment, va aparèixer a la carpeta res, una nova carpeta anomenada menu, la qual incloïa l’arxiu menu.xml.

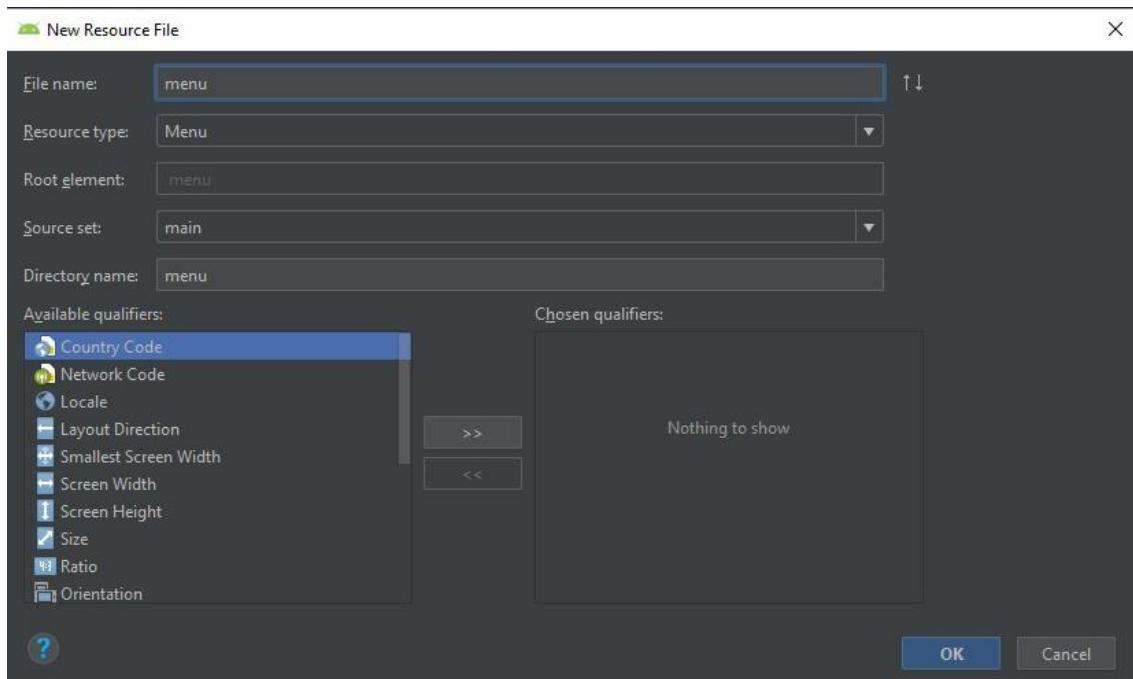


Figura 31, creació del menú

En l'arxiu menu.xml , es va definir un ítem anomenat “about”, al qual se li va assignar el string “about” (Figura 32). Aquest ítem serà l'encarregat d'obrir la nova activitat.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:id="@+id/about" android:title="@string/about" />
</menu>
```

Figura 32, menu.xml

Layout Informació:

El layout informació és l'encarregat de mostrar la informació sobre l'aplicació, com per exemple el perquè de l'aplicació i tenir accés directe a tot el projecte i a la llicència sobre la qual es troba aquest.

El layout es troba format per dos textviews (les quals mostraran la informació sobre l'aplicació) i per dos botons (els quals faran que s'obri la pàgina web on està penjat o la llicència, respectivament) (Figura 33).

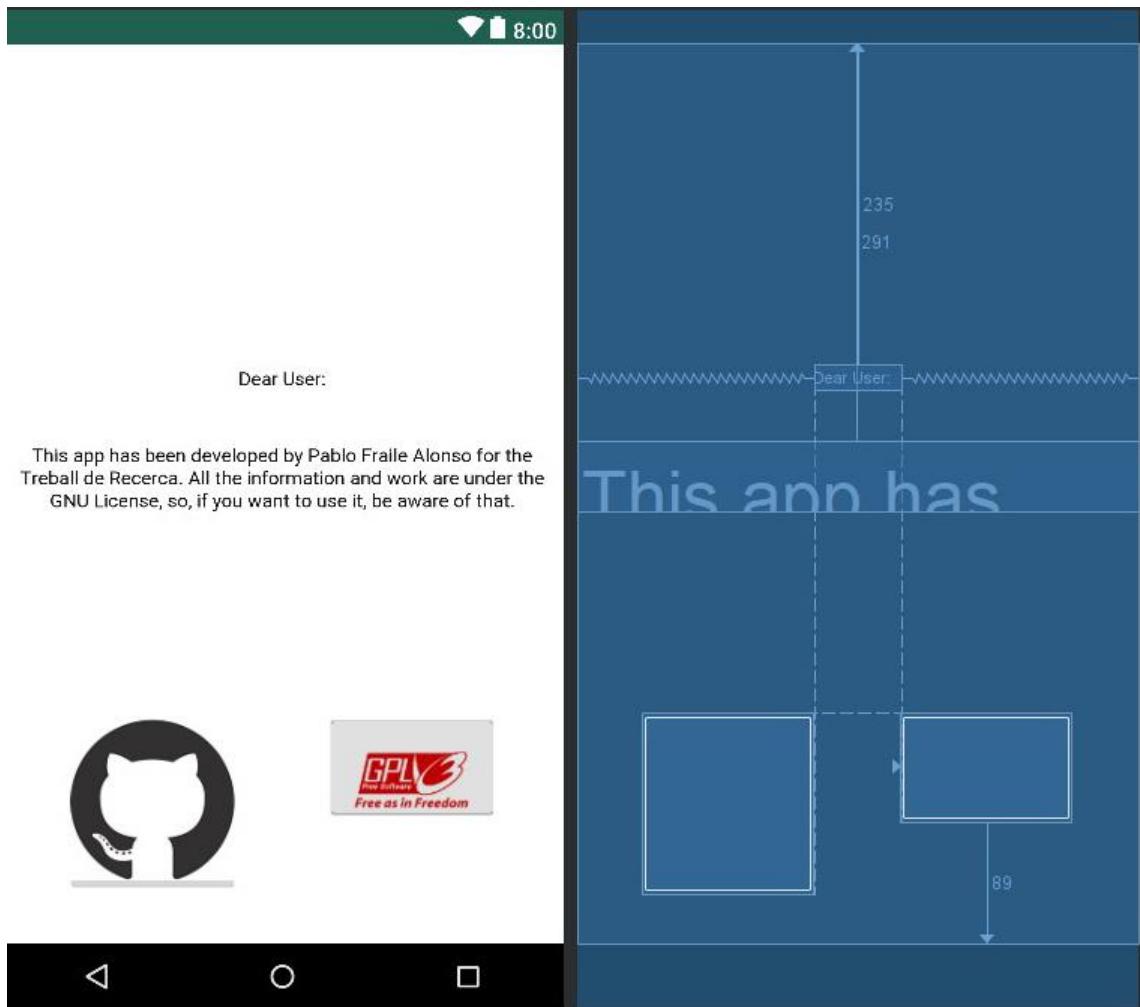


Figura 33, menu_layout.xml

La creació d'aquest layout, ha estat senzill en comparació al layout principal, donat que tots els canvis han estat fets amb l'editor gràfic d'Android Studio, per tant únicament feia falta moure els diferents objectes per la plantilla.

Per a afegir imatges als dos botons, (els quals indiquen la pàgina web on està penjada el codi i la llicència) es va haver d'afegir les dues imatges a la carpeta res/drawable i afegir el següent atribut als dos botons:

“android:drawableBottom=@drawable/nomdelaimatge”

Aquest atribut afegeix una imatge després del nom del botó, que en aquest cas no existeix (Figura 34)

```

<Button
    android:id="@+id/botogithub"
    android:layout_width="126dp"
    android:layout_height="133dp"
    android:layout_alignTop="@+id/botognu"
    android:layout_toStartOf="@+id/importantinfo"
    ● android:drawableBottom="@drawable/githublogo" />

<Button
    android:id="@+id/botognu"
    android:layout_width="126dp"
    android:layout_height="wrap_content"
    android:layout_alignParentBottom="true"
    android:layout_marginBottom="89dp"
    android:layout_toEndOf="@+id/importantinfo"
    android:backgroundTint="?attr/colorControlHighlight"
    ● android:drawableBottom="@drawable/gnugpl" />

```

Figura 34, atributs per als dos botons

7.3 Explicació de les classes

Les classes utilitzades seran separades depenent de l'activitat

Activitat Principal:

L'activitat principal inclou les següents classes (Figura 35):

```

package com.ordinador.pablo.calculadora;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import android.support.v7.app.AppCompatDelegate; // Classe Mode Nocturn
import android.content.Intent; // Classe necessària per a canviar de Activity
import android.widget.Button; // Classe necessària per als botons
import android.view.View; // Classe necessària per a mostrar el xml de l'activity
import android.widget.CompoundButton; // Classe necessària per al switch obscur
import android.widget.Switch; // Classe necessària per a el switch
import android.widget.TextView; // Classe necessària per la textView nombres
import android.content.Context; // Classe necessària per a agafar el contexte per als missatges Toast
import android.widget.Toast; // Classe necessària per als missatges Toast

import android.view.Menu; // Classe necessària per mostrar el Menu
import android.view.MenuItem; // Classe necessària per a mostrar els items del Menu.

import android.content.DialogInterface; // Classe necessària per a el Dialog abans de sortir de la app
import android.support.v7.app.AlertDialog; //Classe necessària per a el Dialog abans de sortir de la app

```

Figura 35, classes importades (Activitat_Principal)

Les classes “android.support.v7.app.AppCompatActivity” i “android.os.Bundle” ja han estat incorporades per Android Studio, donat que es va triar l’opció d’elegir un layout simple.

La classe “ android.support.v7.app.AppCompatDelegate” serà utilitzada per a guardar informació sobre el mode nocturn.

La classe “ android.content.Intent” serà utilitzada per a iniciar una nova activitat o reiniciar l’actual.

La classe “android.widget.Button” serà utilitzada per a programar les diferents accions quan es premi un botó.

La classe “android.view.View” serà utilitzada per a representar les diferents interfícies de usuari.

La classe “android.widget.CompoundButton” serà utilitzada per a el Switch que habilita el mode obscur, donat que es una classe que serveix per a mostrar l’estat de botons amb dos estats (habilitat i desactivat).

La classe “android.widget.Switch” serà utilitzada per a saber l’estat en el què es troba el switch i per a nombrar i programar la utilitat d’aquest.

La classe “android.widget.TextView” serà utilitzada per a llegir i mostrar text a l’usuari.

La classe “android.content.Content” serà utilitzada per a saber informació global sobre l’aplicació i sobre el medi en el que es troba.

La classe “ android.widget.Toast” serà utilitzada per a mostrar missatges flotants per pantalla a l’usuari.

La classe “android.view.Menu” serà utilitzada per a mostrar i programar accions amb el menú.

La classe “android.view.MenuItem” serà utilitzada per a interaccionar amb els ítem del menú.

La classe “android.content.DialogInterface” serà utilitzada per a mostrar un missatge abans de sortir de l’aplicació.

La classe “android.support.v7.app.AlertDialog” és una subclasse de la classe DialogInterface i consisteix en afegir un, dos o tres botons que es poden mostrar en una “caixa” de diàleg.

Activitat menuopcions:

La classe “android.view.View” serà utilitzada per a representar les diferents interfícies d’usuari.

La classe “android.view.Menu” serà utilitzada per a NO mostrar el menú.

La classe “`android.widget.Button`” serà utilitzada per a programar les diferents accions quan es premi un botó.

La classe “`android.content.Intent`” serà utilitzada per a iniciar una nova activitat o reiniciar l’actual.

La classe “`android.net.Uri`” serà utilitzada per a guardar dades de la pàgina web

7.4 Explicació del codi font

1) Activitat_principal.java

1.1) Funcionament de la calculadora

Variables:

Primerament es van crear tres variables de format double, anomenades anterior, actual i numfinal que són igual a 0. A més es va crear una variable String anomenada signe que no és igual a cap valor (Figura 36).

```
// Variables necessàries per a la calculadora
double anterior = 0;
double actual = 0;
double numfinal = 0;
String signe = "";
```

Figura 36, variables necessàries

Funcionament dels botons nombres i punt:

Mitjançant la id assignada en el layout activitat_principal, es van declarar tots els objectes botons i l’objecte textview amb els seus respectius noms i les seves id (Figura 37)

Com l’objecte operand (TextView) haurà de ser utilitzat per altres classes (els botons) ha d’estar declarat com a final. Aquest es un error que va marcar Android Studio durant el procés i pel qual va donar la solució.

```

// Dona nom als botons i al switch mitjançant la ID
Button nombre0 = findViewById(R.id.nombre0); // Botó 0
Button nombre1 = findViewById(R.id.nombre1); // Botó 1
Button nombre2 = findViewById(R.id.nombre2); // Botó 2
Button nombre3 = findViewById(R.id.nombre3); // Botó 3
Button nombre4 = findViewById(R.id.nombre4); // Botó 4
Button nombre5 = findViewById(R.id.nombre5); // Botó 5
Button nombre6 = findViewById(R.id.nombre6); // Botó 6
Button nombre7 = findViewById(R.id.nombre7); // Botó 7
Button nombre8 = findViewById(R.id.nombre8); // Botó 8
Button nombre9 = findViewById(R.id.nombre9); // Botó 9
Button botoigual = findViewById(R.id.botoigual); // Botó igual
Button botodivisio = findViewById(R.id.botodivisio); // Botó divisió
Button botosuma = findViewById(R.id.botosuma); // Botó suma
Button botoresta = findViewById(R.id.botoresta); // Botó resta
Button botomultiplicacio = findViewById(R.id.botomultiplicacio); // Botó multiplicació
Button borra = findViewById(R.id.borra); // Botó de Borrar;
Button punt = findViewById(R.id.punt); // Botó de punt/coma
Button pi = findViewById(R.id.pi); // Botó de Pi
Button doblezero = findViewById(R.id.nombre00); // Botó doble zero
final TextView operand = findViewById(R.id.operand); // Caixa de Text on apareixerà els nombres

```

Figura 37, configuració dels elements

Per programar el funcionament dels botons, es va haver d'utilitzar l'acció “setOnClickListener” que es una de les opcions que brinda la classe “android.widget.Button” i serveix per a que quan es premi el botó nombrat, es faci una acció (Figura 38 i Figura 39). Primerament, es van programar els botons bàsics dels nombres. Aquests botons funcionen de la següent manera:

- 1) Crea una variable String de nom input la qual es igual al valor en String del TextView operand.
- 2) Afegeix el valor del nombre del botó a la variable string.
- 3) Fa que el text mostrat pel TextView operand sigui igual a la variable input.

```

// Programació de totes les accions dels botons
nombre0.setOnClickListener(new View.OnClickListener() { // Administració del nombre 0
    @Override
    public void onClick(View v) {
        String input = operand.getText().toString();
        input = input + "0";
        operand.setText(input);
    }
});
nombre1.setOnClickListener(new View.OnClickListener(){ // Administració del nombre 1
    @Override
    public void onClick(View v) {
        String input = operand.getText().toString();
        input = input + "1";
        operand.setText(input);
    }
});
nombre2.setOnClickListener(new View.OnClickListener(){ // Administració del nombre 2
    @Override
    public void onClick(View v) {
        String input = operand.getText().toString();
        input = input + "2";
        operand.setText(input);
    }
});
nombre3.setOnClickListener(new View.OnClickListener(){ // Administració del nombre 3
    @Override
    public void onClick(View v) {
        String input = operand.getText().toString();
        input = input + "3";
        operand.setText(input);
    }
});
nombre4.setOnClickListener(new View.OnClickListener(){ // Administració del nombre 4
    @Override
    public void onClick(View v) {
        String input = operand.getText().toString();
        input = input + "4";
        operand.setText(input);
    }
});

```

Figura 38, funcionament de els nombres I

```

nombre5.setOnClickListener(new View.OnClickListener() { // Administració del nombre 5
    @Override
    public void onClick(View v) {
        String input = operand.getText().toString();
        input = input + "5";
        operand.setText(input);
    }
});
nombre6.setOnClickListener(new View.OnClickListener() { // Administració del nombre 6
    @Override
    public void onClick(View v) {
        String input = operand.getText().toString();
        input = input + "6";
        operand.setText(input);
    }
});
nombre7.setOnClickListener(new View.OnClickListener() { // Administració del nombre 7
    @Override
    public void onClick(View v) {
        String input = operand.getText().toString();
        input = input + "7";
        operand.setText(input);
    }
});
nombre8.setOnClickListener(new View.OnClickListener() { // Administració del nombre 8
    @Override
    public void onClick(View v) {
        String input = operand.getText().toString();
        input = input + "8";
        operand.setText(input);
    }
});
nombre9.setOnClickListener(new View.OnClickListener() { // Administració del nombre 9
    @Override
    public void onClick(View v) {
        String input = operand.getText().toString();
        input = input + "9";
        operand.setText(input);
    }
});

```

Figura 39, funcionament dels noms II

Pel funcionament dels botons “00” i “π”, el funcionament és el següent:

- Funcionament del “00”:
 - 1) Es crea una String input que serà igual a la String del TextView operand.
 - 2) S’afegeix “00” a la String input
 - 3) El text del Texview Operand és igual a la String Input.

- Funcionament del botó punt:
 - 1) Es crea una String Input que serà igual a la String del TextView operand
 - 2) S'afegeix “.” a la String input
 - 3) El text del TextView operand és igual a la String Input
- Funcionament de “π”:
 - 1) Es crea una variable double nomenada “numpi” que és igual a el nombre Pi
(Math.PI és una de les opcions que dona la classe Math incorporada per Java)
 - 2) Es crea una string anomenada input que és igual al valor en String del TextView operand.
 - 3) Es crea una declaració if/else. Si a la String “input” hi ha algun valor, es crea una variable double anomenada multiplica que serà igual al valor en Double del Text multiplicat per la variable pi. Després d'aquesta operació, es crearà una String anomenada multiplicació que serà igual al valor en String del Double multiplica. Finalment s'assignarà el valor “multiplicacio” al TextView operand.
En cas que input no tingui cal valor, s'assignarà el valor pi a el TextView operand.

En aquest cas, el nombre pi té un funcionament inusual donat que en el cas de que s'utilitzés el mateix mètode que amb els altres nombres, si el TextView no estigués igual a “” (per tant, si hi hagués algun nombre), el nombre pi afegiria el nombre pi al nombre de la TextView. Per exemple:

Si a la TextView es trobés el nombre 25 i s'hagués programat el nombre pi amb el mateix mètode que els altres nombres, el nombre resultant del TextView seria 253.1415..

En canvi, si es fa la mateixa operació amb el mètode en el que s'ha programat finalment, el nombre 25 es multiplicaria pel nombre pi, evitant així l'error anterior.

```

doblezero.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { // Administració del botó doble zero
        String input = operand.getText().toString();
        input = input + "00";
        operand.setText(input);
    }
});
pi.setOnClickListener(new View.OnClickListener() { // Administració del botó pi
    @Override
    public void onClick(View v) {
        double numpi = Math.PI;
        String input = operand.getText().toString();
        if (!input.equals("")){
            Double multiplica = Double.parseDouble(input) * numpi;
            String multiplicacio = String.valueOf(multiplica);
            operand.setText(multiplicacio);
        }
        else {
            input = input + numpi;
            operand.setText(input);
        }
    }
});
punt.setOnClickListener(new View.OnClickListener() { // Administració del botó punt
    @Override
    public void onClick(View v) {
        String input = operand.getText().toString();
        input = input + ".";
        operand.setText(input);
    }
});

```

Figura 40, funcionament de pi, 00 i punt.

Funcionament dels botons de signes:

Per un funcionament òptim de la calculadora, es va descobrir que els càlculs havien de fer-se cada vegada que es prenia el botó igual i qualsevol botó de signes, ja que, si el botó igual acumulava totes les operacions, el programa no guardava els diferents signes que s'anaven prenent i per tant no es podien fer operacions de més de dos nombres diferents.

Per això, es va decidir crear el mètode calcula (figura 41), que és aquell que s'encarregarà del funcionament de les operacions després de pressionar qualsevol botó de signe o el botó igual.

El funcionament del mètode calcula és el següent:

El mètode calcula és un mètode que té que retornar un valor double, per tant, sempre retornarà un nombre decimal. A més, el mètode demana la presència de les variables anterior, actual i signe definides anteriorment. Un cop es crida el mètode, aquest mira la variable signe i depenent de el signe ("+","-","*","/") farà una operació o una altra entre el nombre anterior i actual , i , després d'aquesta operació retornarà la variable numfinal, que serà el resultat de les operacions anteriors.

```
// Mètode double Calcula
public double calcula(Double anterior, Double actual, String signe){
    switch (signe){
        case "+":
            numfinal = anterior + actual;
            break;
        case "-":
            numfinal = anterior - actual;
            break;
        case "*":
            numfinal = anterior * actual;
            break;
        case "/":
            numfinal = anterior / actual;
            break;
    }
    return(numfinal);
}
```

Figura 41, mètode calcula

En quant als botons que contenen signes, el seu funcionament es el següent:

- **Funcionament de “+, /, *”:**
 - 1) Es crea una declaració if/else, la qual si el nombre anterior i el nombre actual es igual a 0, anomenarà anterior a el valor Double del text que es troba a operand, i una vegada guardat aquest valor, ficarà el operand a zero.

- 2) En canvi, si no es compleix que les variables anterior i actual són igual a 0, el valor actual serà igual a el valor Double del text que es troba a operand i es dirà que el valor anterior sigui igual al nombre que tregui el mètode calcula. Un cop assignat el nou valor a la variable anterior, ficarà l'operand a zero.
- 3) Després de la declaració if/else, s'assigna el valor +, / o * (depenent del botó pres) a la variable signe.

- **Funcionament de “-”:**

- 1) Es crea una variable String anomenada text que serà igual al valor en String de el Text operand.
- 2) Es crea una declaració if/else if/else. La qual indica que si al variable text és igual a zero, afegeixi a el TextView operand el signe “-“ (ja que si no hi ha cap mena de nombre a el TextView operand, és un indicí de què l'usuari vol ficar un nombre negatiu).

En canvi, si la variable text no és igual a zero, però la variable anterior i actual sí que son igual a zero, és un indicí de que l'usuari vol fer una operació matemàtica amb el signe menys, per tant, el valor de la variable anterior serà igual al valor Double del TextView operand, el text serà restaurat a 0.

Si no es compleix cap de les altres declaracions, és perquè és una operació amb més de dos nombres, per tant, primer es guarda el valor String del TextView en la variable actual, es calcula el nombre anterior, que serà igual a el resultat de el mètode calcula i es ficarà el text a zero. Finalment, es canviarà la variable de signe a “-“

- **Funcionament de “=”:**

- 1) Es canvia la variable actual pel valor Double de el nombre que es troba a operand.
- 2) Es calcula el numfinal amb el mètode calcula
- 3) Es mostra el valor String del numfinal a el TextView operand.
- 4) Es canvien les variables anterior, numfinal i actual a 0, per tal de què el usuari pugui operar amb el resultat donat.

Funcionament del botó esborrar:

- Funcionament per a esborrar únicament un nombre:
 - 1) Es crea una variable String anomenada input, la qual es igual a el valor en String del text d'operand.
 - 2) Input serà igual a la longitud de operand menys 1.
 - 3) El text d'operand serà igual a input
- Funcionament per a esborrar tots els nombres (pressionant durant un segon el botó)
Per a poder esborrar tots els nombres de la calculadora el botó esborrar ha estat programat amb l'acció “setOnLongClickListener” la qual permet saber quan l'usuari haprés el botó durant un determinat temps.
El botó esborrar per a eliminar tots els nombres utilitza aquesta acció per tal de borrar tots els paràmetres quan l'usuari premi durant un determinat temps el botó.

1.2) Funcionament del Switch

Primerament, es va declarar el switch mitjançant ,la id donada en el layout (Figura 42)

```
// Declaració del SwitchObscur i la seva funció.  
Switch switchobscur;  
switchobscur = findViewById(R.id.darkswitch);
```

Figura 42, declaració del switch.

Mitjançant les classes “android.widget.Switch” i “android.widget.CompoundButton” es va programar el funcionament de el Switch, el qual contendrà un estament if/else.

En el cas de que el switch estigui activat, es ficarà la variable Integer setDefaultNightMode (inclosa en la classe “android.support.v7.app.AppCompatDelegate”) a el valor “2” que indica activat.

En canvi, si el switch no està activat, al variable setDefaultNightMode serà igual a “1” que indica desactivat.

Tant l'estament if com en el else, es cridarà al mètode activityprincipal, que reiniciarà l'activitat_principal per a poder tornar a carregar el layout amb el nou tema.

```

switchobscur.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked){
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
            activityprincipal();
        }
        else{
            AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
            activityprincipal();
        }
    }
});
```

Figura 43, declaració de l'acció del switch

Una vegada programat el switch, es va afegir un estament if/else per a que canviés el tema abans de que la activitat mostrés el layout principal (ja que si es mostrava el layout principal i després es canvia el tema, l'aplicació es tancava). L'estament consistia en que si la variable NightMode era igual a “2”, el tema principal passaria a ser el obscur, i si la variable era igual a “1” el tema principal passaria a ser el tema clar. Cal recalcar que un cop que es tanqués l'aplicació i s'eliminés el procés, la variable NightMode es tornaria a posar a “1”.

```

if (AppCompatDelegate.getDefaultNightMode()==AppCompatDelegate.MODE_NIGHT_YES){
    setTheme(R.style.DarkTheme);
}else{
    setTheme(R.style.AppTheme);
}
```

Figura 44, configuració de el Style

A més, després de provar el codi, va ser visible que després d'aplicar els canvis, el switch tornava a estar desactivat encara que el mode obscur estigués activat. Es per aquesta raó per la qual abans de la programació del switch, es va afegir un estament if que deia que si la variable NightMode era igual a “2”, el switch pasaría a estar activat.

```

if (AppCompatDelegate.getDefaultNightMode()==AppCompatDelegate.MODE_NIGHT_YES){
    switchobscur.setChecked(true);
}
```

Figura 45, configuració del switch

Per tant, el codi sencer quedaría de la següent forma:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    if (AppCompatDelegate.getDefaultNightMode() == AppCompatDelegate.MODE_NIGHT_YES){
        setTheme(R.style.DarkTheme);
    }else{
        setTheme(R.style.AppTheme);
    }

    // Mostra el layout principal, (tema ja escollit)
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_activitat_principal);

    // Declaració del SwitchObscur i la seva funció.
    Switch switchobscur;
    switchobscur = findViewById(R.id.darkswitch);

    if (AppCompatDelegate.getDefaultNightMode() == AppCompatDelegate.MODE_NIGHT_YES){
        switchobscur.setChecked(true);
    }
    switchobscur.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (isChecked){
                AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_YES);
                activityprincipal();
            }
            else{
                AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);
                activityprincipal();
            }
        }
    });
}
```

Figura 46, codi de funcionament del Switch

1.3) Funcionament del menú

Per a mostrar i programar el menú, el codi s'ha basat en els exemples de Android Developers.

Primerament, es crea un mètode anomenat “onCreateOptionsMenu” que retorna un valor Boolean (true o false). Aquest mètode transforma els arxius xml en objectes. Per tant, troba el menú gràcies a la id donada anteriorment i el nombra amb el nom menuconf. Finalment, retorna true, ja que així s'indica que existeix un menú i el mostra.

```

public boolean onCreateOptionsMenu(Menu menuconf) {
    getMenuInflater().inflate(R.menu.menu, menuconf);
    return true;
}

```

Figura 47, mètode per a mostrar el menú

Per a mostrar les opcions, es crea el mètode anomenat “onOptionsItemSelected” el qual troba la id dels diferents ítems del menú, els assigna un nombre int anomenat id i si el usuari el tria, utilitza el mètode activityopcions (figura 48) que executa l’activitat del layout sobre l’aplicació. Finalment retorna true o false dependent si s’ha triat alguna opció del menú o no.

```

// Mètode per a dir que es fa si es pren determinat apartat del menú
public boolean onOptionsItemSelected (MenuItem opciomenu) {
    int id=opciomenu.getItemId();
    if (id==R.id.about){
        activityopcions(null); // No es pot mostrar cap objecte view ( cap widget,etc.) per tant, li donem un parametre null (no tenim res)
    }
    return super.onOptionsItemSelected(opciomenu);
}

```

Figura 48, mètode per a els ítems seleccionats del menú

El mètode activityopcions va estar programat amb la finalitat d’obrir una nova activity que mostrés la informació sobre la aplicació. Per a realitzar aquesta acció, es necessita editar el Android Manifest i afegir un nou arxiu .java, a més de programar el mètode.

- 1) A l’Android Manifest, es va afegir la següent línia de codi (Figura 49) la qual indicava la presència d’una nova activitat (a més de la principal) en l’aplicació.

```

1      <!-- Activitat del Menú -->
2      <activity android:name=".menuopcions" android:label="@string/about" />

```

Figura 49, declaració d’una Activitat a l’arxiu AndroidManifest.xml

- 2) La creació d'un nou arxiu .java el qual serà el que tindrà el codi de funcionament del segon layout (Figura 50). Aquest arxiu es dirà menuopcions.java i serà una extensió de l'activitat_principal, per tal de poder seguir treballant amb el mode nocturn.

```
package com.ordinador.pablo.calculadora;
import android.os.Bundle;

public class menuopcions extends ActivitatPrincipal{
    super.onCreate(savedInstanceState);
    setContentView(R.layout.menulayout);

}
```

Figura 50, creació del arxiu menuopcions.java

- 3) Mètode activityopcions (Figura 51) crea un Intent (acció) sota el nom “i” el qual obre l'arxiu menuopcions.class (compilació de l'arxiu .java) necessari per al segon layout.

```
public void activityopcions(View view){
    Intent i = new Intent(this, menuopcions.class);
    startActivity(i);
}
```

Figura 51 creació del mètode activityopcions

1.4) Funcionament del Diàleg de Sortida.

Pel funcionament del diàleg de sortida, s'haurà d'utilitzar el mètode onBackPressed (Figura 52), que es cridat sempre que es detecta que l'usuari pressiona el botó enrere. La implementació per defecte simplement finalitza l'activitat actual, però en aquest cas es programarà per a mostrar el missatge d'alerta. Es crea un mètode anomenat mostraalerta, el qual crearà un missatge d'Alerta. Primerament, es crea un AlertDialog.builder que s'anomenarà “crea”. A “crea” se li assignaran diferents valors com per exemple el títol o el missatge (que seran

Strings) i la propietat de que el missatge NO es cancelable, per tant, s'ha de contestar a la pregunta d'alerta per a poder continuar amb l'aplicació.

Després, s'assigna un botó positiu i un negatiu. El botó positiu, finalitza l'activitat, i per tant, tanca l'aplicació. En canvi, el botó negatiu, tanca el diàleg.

Finalment es programa que es construeixi el diàleg amb els paràmetres anteriors i que es mostri a l'usuari quan es cridi el mètode.

```
public void onBackPressed(){ // Quan pren el botó enrere, executa el mètode mostraalerta
    mostraalerta();
}

private void mostraalerta(){ // Utilitza la classe AlertDialog.Builder i DialogInterface
    final AlertDialog.Builder crea = new AlertDialog.Builder(this);
    crea.setCancelable(false);
    crea.setTitle(R.string.titolalerta);
    crea.setMessage(R.string.missatgealerta);

    crea.setPositiveButton(R.string.respontasi, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            finish();
        }
    });

    crea.setNegativeButton(R.string.respontano, new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    });
    crea.create().show(); // Crea el missatge i mostra'l
}
```

Figura 52, mètodes per a el missatge d'alerta

2) menuopcions.java

L'arxiu menuopcions.java es aquell que serà executat en el format .class quan l'usuari hagi clicat l'opció en el menú. Aquest arxiu, és una extensió de l'arxiu activitat_principal.java, per tant, cal recalcar que totes les línies de codi escrites anteriorment en l'activitat principal, seran també executats. Tenint en compte aquest fenomen, caldrà doncs, desactivar tant el mètode per a mostrar el menú (Figura 53) com el mètode per a mostrar el missatge d'alerta (Figura 54)

) ja que sinó ens mostrarà el menú i el missatge d'alerta, cosa que no té sentit donada la situació.

Per a eliminar el mètode de menú, es retorna un valor false al mètode, per a que l'aplicació entengui que no hi ha cap menú.

```
public boolean onCreateOptionsMenu(Menu menuconf) {  
    return false;  
}
```

Figura 53, eliminació del menú

Per a eliminar el missatge d'alerta, es programa el mètode “onBackPressed” per tal de que quan es cliqui el botó enrere, no mostri el missatge sinó que finalitzi l'activitat.

```
// Com La Activity MenuOpcions es una ampliació de La Activity Principal, tot el programat a L'Activity Principal s'aplica a menuopcions.  
// Per tant, s'ha de tornar a programar el botó enrere per a que no mostri el mètode mostraalerta, sinò que finalitzi la activitat menuopcions.  
public void onBackPressed(){  
    finish();  
}
```

Figura 54, mètode onBackPressed

Per a la configuració dels botons, primerament es van declarar com a botons mitjançant la id nombrada anteriorment (Figura 55), i posteriorment es van configurar amb l'acció “setOnClickListener”. L'acció consistia en crear un uri.parse (que consisteix en un proveïdor de continguts, simplement guarda dades) amb les pàgines web que es volien executar. Finalment, s'iniciava un Intent amb una ACTION_VIEW, que vol dir que es vol mostrar dades a l'usuari (Figura 56).)

```
Button botogithub;  
botogithub = findViewById(R.id.botogithub);  
Button botognu;  
botognu = findViewById(R.id.botognu);
```

Figura 55, declaració dels botons

```

botogithub.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Uri pagweb = Uri.parse("https://github.com/Pablit02020/Treball-de-Reerca");
        Intent githubweb = new Intent(Intent.ACTION_VIEW, pagweb);
        startActivity(githubweb);
    }
});

botognu.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Uri pagweb = Uri.parse("https://raw.githubusercontent.com/Pablit02020/Treball-de-Reerca/master/LICENSE");
        Intent gnu = new Intent(Intent.ACTION_VIEW, pagweb);
        startActivity(gnu);
    }
});

```

Figura 56, funcionament dels botons

7.5 Penjar l'aplicació a la Google Play

Finalment, per a finalitzar amb el procediment pràctic de l'aplicació Calculadora, es va decidir penjar l'aplicació a la Google Play Store. Per a duu a terme aquest procediment, primerament es va haver de crear una conta de desenvolupador de google i pagar vint-i-cinc euros simbòlics per tal de poder començar a penjar aplicacions.

Un cop amb accés per a penjar aplicacions, es va haver de compilar una versió de l'aplicació signada, per tal de tenir un certificat digital propi per a tenir permís de penjar-se a la Google Play. Un cop fet el procediment es va ser seguir els diferents passos marcats en la pàgina web, els quals es troben separats per categories (Figura 57). Els passos obligatoris a seguir són els que tenen una icona de insígnia al costat (Versió de l'aplicació, Fitxa de la Google Play, etc.)

-  Panel de control
-  Versiones de la aplicación 
-  Aplicaciones Instantáneas Android
-  Biblioteca de artefactos
-  Catálogo de dispositivos
-  Firma de aplicaciones
-  Ficha de Play Store 
-  Clasificación de contenido 
-  Precio y distribución 
-  Productos de compra en la aplicación
-  Servicio de traducción
-  Servicios y APIs
-  Consejos de optimización

Figure 57, opciones de el play console

8. Dreceres.

8.1 El perquè del projecte i la creació d'aquest

Després de la creació de la calculadora com a primera aplicació de la part pràctica, es va decidir que es volia crear una aplicació que sigués útil i que treballés e interactués amb diferent hardware del dispositiu. Es d'aquesta base d'on va néixer "Dreceres". Una aplicació que consisteix en substituir la pàgina d'inici del dispositiu mòbil per una pantalla minimalistà amb dotze accessos directes a aplicacions bàsiques i fonamentals per a un SmartPhone.

La creació del projecte "Dreceres" , va ser igual que l'aplicació Calculadora, exceptuant el nom de l'aplicació. La versió mínima d'Android, segueix sent 5.1 i es va començar amb un layout buit.

8.2 Explicació de la interfície gràfica

El projecte Dreceres consta de tres diferents layouts. Un layout principal ("Activitat_Principal"), un layout per a la configuració ("configuracio") i un layout per a l'informació sobre l'aplicació (info).

Layout Activitat_Principal:

El layout principal, està format per 10 botons, 2 toggles, 2 TextViews (els quals indicaran la funcionalitat dels toggles) i un AdView (el qual inclourà publicitat).

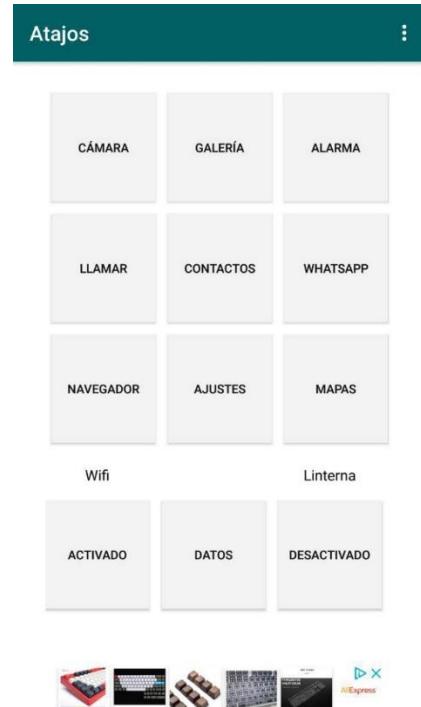


Figura 58, layout principal.

Tots els objectes menys el adview es trobaran en un GridLayout (al igual que a l'aplicació calculadora) donat que l'important en l'aplicació es que els botons estiguin ordenats i tinguin la mateixa mida, i en aquest cas el GridLayout és el millor tipus de layout.

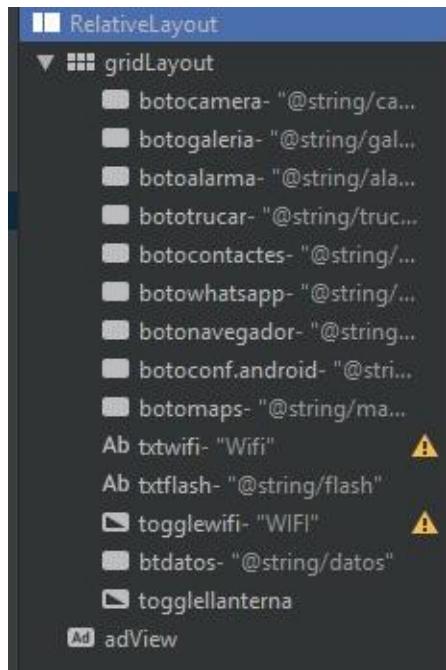


Figura 59, components del layout

Com a novetat, també es va voler incorporar el suport de diversos tipus de pantalles a l'aplicació. Per això es va haver de crear tres versions diferents de layout, a més de incorporar aquesta funció a l'Android Manifest.

- 1) Creació dels tres diferents layouts: Per a la creació dels diferents layouts, es va haver d'accedir a la carpeta res, fer clic dret i seleccionar la opció de crear un nou recurs d'Android. Fet aquest pas es va ficar de nom Activitat_Principal, es va seleccionar el recurs de layout i es va seleccionar el qualificador de pantalla en normal, després en llarg i finalment en extra llarg. La creació de diferents layouts per a diferents estils de pantalla no implica que hi hagi diferències entre els layouts, la única variació que s'aprecia són els píxels que ocupa cada ítem en la pantalla.

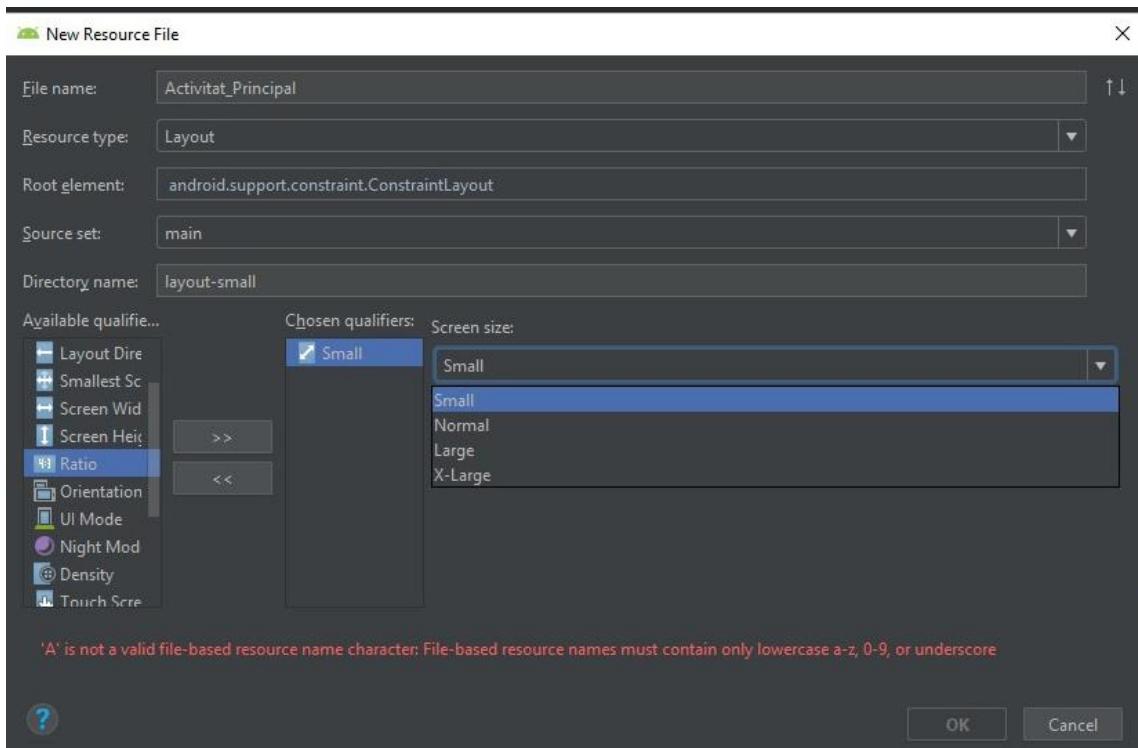


Figura 60, creació de diferents Activitat_Principal

- 2) Afegir el suport per a diferents tipus de pantalles al Android Manifest: Es va haver d'afegir les següents línies de codi al Android Manifest (figura 61) per tal de que el sistema detectés el suport de diferents mides de pantalla

```
<supports-screens
    android:largeScreens="false"
    android:normalScreens="true"
    android:smallScreens="false"
    android:xlargeScreens="true"/>
```

Figura 61, suport per a diferentes pantalles a l'AndroidManifest

A més, l'aplicació també inclou un canviador de tema (el qual vol dir que deixa triar a l'usuari el color principal de l'aplicació.) Aquest fet implica la creació de múltiples temes (no únicament un tema obscur com a l'aplicació calculadora) encara que el procediment és el mateix.

Primerament es crear un arxiu attrs.xml on s'indicaran els diferents atributs que s'utilitzaran per a indicar el color de fons dependent del tema (Figura 62)

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <declare-styleable name="ds">
        <attr name="colorfons" format="color" />
        <attr name="colorboto" format="color" />
        <attr name="colortext" format="color" />
    </declare-styleable>
</resources>
```

Figura 62, declaració d'atributs

Una vegada definits els atributs, es va haver d'assignar un arxiu anomenat colors.xml, que inclou el valor hexadecimal dels colors utilitzats en tots els diferents temes (Figura 63)

```
1 | <?xml version="1.0" encoding="utf-8"?>
2 | <resources>
3 |
4 |     <!-- Colors elegits gràcies a la pàgina web material
5 |          link de Material.io: https://material.io/develop/
6 |
7 |     <!-- Color de la App de Sèrie ( Rosa )--&gt;
8 |     &lt;color name="colorPrimary"&gt;#d81b60&lt;/color&gt;
9 |     &lt;color name="colorPrimaryDark"&gt;#a00037&lt;/color&gt;
10 |    &lt;color name="colorAccent"&gt;#ff5c8d&lt;/color&gt;
11 |
12 |    <!-- Color de la App Blau --&gt;
13 |    &lt;color name="colorPrimaryBlau"&gt;#3F51B5&lt;/color&gt;
14 |    &lt;color name="colorPrimaryDarkBlau"&gt;#303F9F&lt;/color&gt;
15 |    &lt;color name="colorAccentBlau"&gt;#8C9EFF&lt;/color&gt;
16 |
17 |    <!-- Color de la App fosc (negre-gris) --&gt;
18 |    &lt;color name="colorPrimaryGris"&gt;#393939&lt;/color&gt;
19 |    &lt;color name="colorPrimaryDarkGris"&gt;#2f2f2f&lt;/color&gt;
20 |    &lt;color name="colorAccentGris"&gt;#52b7f8&lt;/color&gt;
21 |
22 |    <!-- Color de la App cyan --&gt;
23 |    &lt;color name="colorPrimaryCyan"&gt;#006064&lt;/color&gt;
24 |    &lt;color name="colorPrimaryDarkCyan"&gt;#00363a&lt;/color&gt;
25 |    &lt;color name="colorAccentCyan"&gt;#428e92&lt;/color&gt;
26 |
27 |    <!-- Color de la App verd --&gt;
28 |    &lt;color name="colorPrimaryVerd"&gt;#64dd17&lt;/color&gt;
29 |    &lt;color name="colorPrimaryDarkVerd"&gt;#1faa00&lt;/color&gt;
30 |    &lt;color name="colorAccentVerd"&gt;#9cff57&lt;/color&gt;
31 |
32 |    <!-- Color de la App Taronja --&gt;
33 |    &lt;color name="colorPrimaryTaronja"&gt;#f44336&lt;/color&gt;
34 |    &lt;color name="colorPrimaryDarkTaronja"&gt;#ba000d&lt;/color&gt;
35 |    &lt;color name="colorAccentTaronja"&gt;#ff7961&lt;/color&gt;
36 |
37 |    <!-- Tema Obscur i blanc --&gt;
38 |    &lt;color name="colorfonsblanc"&gt;#ffffff&lt;/color&gt;
39 |    &lt;color name="colorbotoblanco"&gt;#f2f2f2&lt;/color&gt;
40 |    &lt;color name="colortextnegre"&gt;#111111&lt;/color&gt;
41 |
42 |    &lt;color name="colorfonsnegre"&gt;#333333&lt;/color&gt;
43 |    &lt;color name="colorbotonegre"&gt;#444444&lt;/color&gt;
44 |    &lt;color name="colortextblanc"&gt;#ffffff&lt;/color&gt;
45 |
46 |
47 | &lt;/resources&gt;</pre>
```

Figura 63, declaració dels colors

Per finalitzar amb els arxius .xml originaris del canvi de color de l'aplicació, es van definir 5 temes nous (Figura 64) i 2 temes necessaris per al mode obscur automàtic (Figura 65)

```
<resources>

    <!-- Color de La App de Sèrie ( Rosa )-->
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </style>

    <!-- Color de La App Blau -->
    <style name="TemaBlau" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimaryBlau</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDarkBlau</item>
        <item name="colorAccent">@color/colorAccentBlau</item>
    </style>

    <!-- Color de La App fosc (negre/gris) -->
    <style name="TemaGris" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimaryGris</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDarkGris</item>
        <item name="colorAccent">@color/colorAccentGris</item>
    </style>

    <!-- Color de La App Cyan -->
    <style name="TemaCyan" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimaryCyan</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDarkCyan</item>
        <item name="colorAccent">@color/colorAccentCyan</item>
    </style>

    <!-- Color de la App Verd -->
    <style name="TemaVerd" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimaryVerd</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDarkVerd</item>
        <item name="colorAccent">@color/colorAccentVerd</item>
    </style>

    <!-- Color de La App Taronja -->
    <style name="TemaTaronja" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="colorPrimary">@color/colorPrimaryTaronja</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDarkTaronja</item>
        <item name="colorAccent">@color/colorAccentTaronja</item>
    </style>
```

Figura 64, diferents estils

```
<style name="FonsBlanc">
    <item name="colorfons">@color/colorfonsblanc</item>
    <item name="colorboto">@color/colorbotoblan</item>
    <item name="colortext">@color/colortextnegre</item>
</style>

<!-- Fons negre -->
<style name="FonsNegre" parent="Theme.AppCompat.Light.DarkActionBar">
    <item name="colorfons">@color/colorfonsnegre</item>
    <item name="colorboto">@color/colorbotonegre</item>
    <item name="colortext">@color/colortextblanc</item>
</style>

</resources>
```

Figura 65, diferents estils per al canvi de tema automàtic

Layout configuració:

El layout configuració està format per 2 botons, 3 TextViews i un “spinner” (Figura 66 i Figura 67). El “spinner” és un component gràfic que permetrà a l’usuari tenir una manera ràpida de seleccionar un valor de un conjunt (que en aquest cas serà el color predeterminat de l’aplicació).

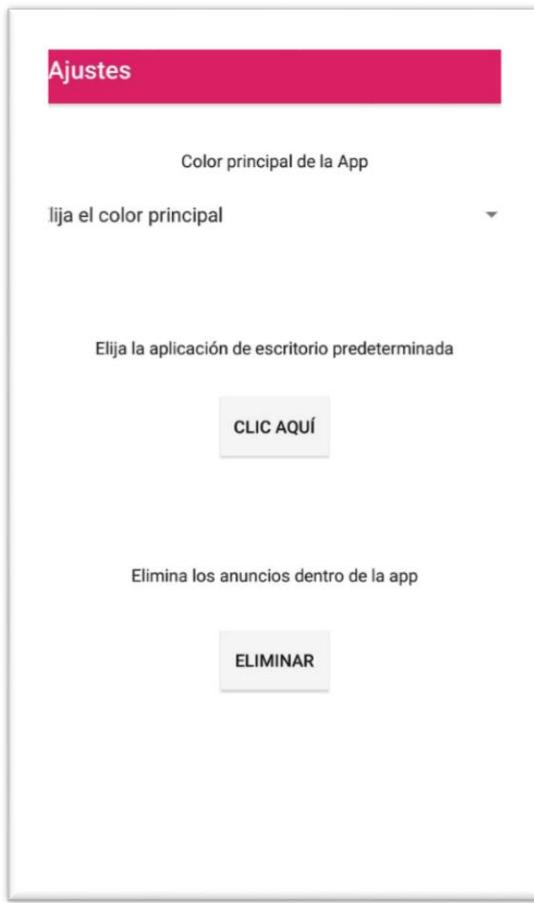


Figura 66, layout configuració

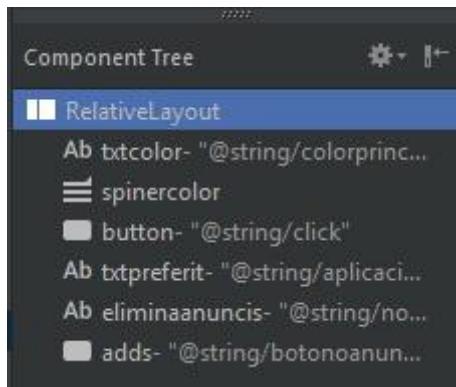


Figura 67, components del layout configuració

Per a l'addició dels botons i dels TextViews, s'ha utilitzat l'editor de layouts inclòs per Android Studio, mentre que per al component "Spinner", es va haver d'acudir a llenguatge XML (que gràcies a la guia de Android Studio, l'únic treball necessari va ser canviar el noms dels elements que es volia ficar l'"spinner").

Primerament, es va crear un element spinner en el layout, afegint les següents línies de codi (Figura 68)

```
<Spinner  
    android:id="@+id/spinercolor"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:layout_alignParentStart="true"  
    android:layout_alignParentTop="true"  
    android:layout_marginTop="78dp"  
    android:textColor="?attr/colortext"/>
```

Figura 68, atributs del spinner

Tot seguit, es va crear un arxiu anomenat “strings_array.xml” el qual es troava al directori res/values i incloïa tots els diferents colors que tenen que aparèixer a el “spinner” (Figura 69)

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string-array name="colors">  
        <item>@string/tria</item>  
        <item>@string/rosa</item>  
        <item>@string/blau</item>  
        <item>@string/gris</item>  
        <item>@string/cyan</item>  
        <item>@string/verd</item>  
        <item>@string/taronja</item>  
    </string-array>  
</resources>
```

Figura 69

Layout informació:

El layout informació està format per dos TextViews, dos botons i un botó flotant (Figura 70 i Figura 71). Un Botó flotant consisteix en un botó de forma circular que en aquest cas s'utilitza com a botó de compartir l'aplicació.

La implementació d'aquests components al layout s'ha fet mitjançant l'eina editor de layouts inclosa en Android Studio.

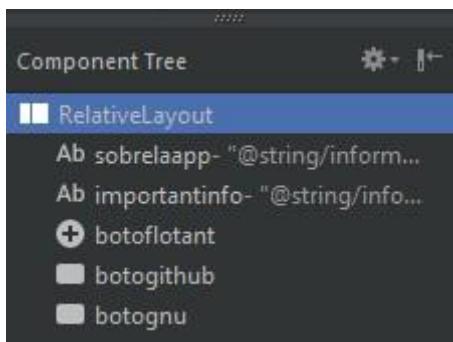


Figura 70, components del Layout informació

Sobre la App

Sobre la aplicación

Esta aplicación ha sido desarrollada por Pablo Fraile Alonso para el trabajo de investigación de Bachiller. Todo el trabajo e información recopilados están bajo la Licencia GNU, así que si usted quiere usar o modificar alguno de los parámetros del trabajo, sea consciente de este hecho.



Figura 71, layout informació

8.3 Explicació de les classes

• Classes incloses a Activitat_Principal.java:

```
package com.ordinador.pablo.accesdirecte;

import android.icu.util.Calendar;
import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;
import android.support.annotation.RequiresApi;
import android.view.View; // Classe necessària per
import android.content.Intent; // Classe necessàri
import android.content.Context; // Classe necessàr
import android.net.Uri; // Classe necessària per q
import android.content.SharedPreferences; // Class
import android.view.Menu; // Classe necessària per
import android.view.MenuInflater; // Classe necess
import android.view.MenuItem; // Classe necessària
import android.widget.Button; // Classe necessària
import android.widget.ToggleButton; // Classe Nec
import android.widget.CompoundButton; // Classe ne
import android.net.wifi.WifiManager; // Classe nec
import android.net.ConnectivityManager; // Classe
import android.net.NetworkInfo; // Classe que desc
import android.net.Network; // Classe necessària p
import android.provider.MediaStore; // Classe nece
import android.hardware.camera2.CameraAccessException;
import android.hardware.camera2.CameraManager; // /
import android.content.pm.PackageManager; // Class
import android.widget.Toast; // Classe necessària
import android.provider.Settings; // Classe necess
import android.os.Build; // Classe necessària per
import android.provider.AlarmClock; // Classe nece

import com.google.android.gms.ads.AdRequest; // Cl
import com.google.android.gms.ads.AdView;
import com.google.android.gms.ads.MobileAds;
```

Figura 72, classes importades a Activitat_Principal

Les classes “`android.support.v7.app.AppCompatActivity`” i “`android.os.Bundle`” ja han estat incorporades per Android Studio, donat que es va triar la opció de elegir un layout simple.

La classe “`android.icu.util.Calendar`” serà utilitzada per a saber l’hora en la que es troba el dispositiu.

La classe “`android.support.annotation.RequiresApi`” va ser afegida per Android Studio al veure que apareixia a una referència de classe superior a Android Lollipop (5.1). Per

tant, s'havia de ficar l'anotació de que aquella funcionalitat únicament funcionaria amb determinades versió d'Android.

La classe “`android.view.View`” serà utilitzada per a representar les diferents interfícies de usuari.

La classe “`android.content.Intent`” serà utilitzada per a iniciar una nova activitat o reiniciar l'actual.

La classe “`android.content.Context`” serà utilitzada per a saber informació global sobre l'aplicació i sobre el medi en el que es troba.

La classe “`android.net.Uri`” serà utilitzada per a guardar dades.

La classe “`android.content.SharedPreferences`” serà utilitzada per a guardar paràmetres en memòria, encara que l'aplicació tanqui el procés.

La classe “`android.view.Menu`” serà utilitzada per a mostrar i programar accions amb el menú.

La classe “`android.view.MenuItem`” serà utilitzada per a interaccionar amb els ítem del menú en objectes.

La classe “`android.view.MenuItem`” serà utilitzada per a interaccionar amb els ítem del menú.

La classe “`android.widget.Button`” serà utilitzada per a programar les diferents accions quan es premi un botó.

La classe “`android.widget.ToggleButton`” serà necessària per a programar els botons tipus toogle

La classe “`android.widget.CompoundButton`” serà utilitzada pel Switch que habilita el mode obscur, donat que es una classe que serveix per a mostrar l'estat de botons amb dos estats (habilitat i desactivat).

La classe “`android.net.wifi.WifiManager`” serà utilitzada per a la gestió del wifi del dispositiu.

La classe “`android.net.ConnectivityManager`” serà utilitzada per a comprovar l'estat de connexió de la xarxa, tant amb wifi com amb dades mòbils (GPRS)

La classe “`android.net.NetworkInfo`” serà utilitzada per a descriure el estat d'una interfície de xarxa

La classe “`android.net.Network`” serà utilitzada per a identificar una xarxa

La classe “`android.provider.MediaStore`” serà utilitzada per a accedir a la càmera.

La classe “`android.hardware.camera2.CameraAccessException`” serà utilitzada per a identificar quan el servei CameraManager no és vàlids (nulls)

La classe “`android.hardware.camera2.CameraManager`” serà necessària per a detectar, caracteritzar o connectar a dispositius amb càmera.

La classe “`android.content.pm.PackageManager`” serà necessària per a mirar les característiques del dispositiu.

La classe “`android.widget.Toast`” serà utilitzada per a mostrar missatges flotants per pantalla a l'usuari.

La classe “`android.provider.Settings`” serà utilitzada per a redirecciónlar a l'aplicació configuració del dispositiu.

La classe “`android.os.Build`” serà necessària per a mirar la API del dispositiu

La classe “`android.provider.AlarmClock`” serà necessària per a poder executar l'aplicació d'alarma predeterminada.

Les classes

“`com.google.android.gms.ads.AdRequest`”, “`com.google.android.gms.ads.AdView`” i “`com.google.android.gms.ads.MobileAds`” seran necessàries per a el funcionament dels anuncis.

• Classes incloses a configuracio.java:

La classe “android.view.Menu” serà necessària per tal de poder eliminar la opció de menú que ha estat definida en “Activitat_Principal.java”

La classe “android.view.View” serà utilitzada per a representar les diferents interfícies de usuari.

La classe “ android.content.Intent” serà utilitzada per a iniciar una nova activitat o reiniciar l’actual.

La classe “android.content.SharedPreferences” serà utilitzada per a guardar paràmetres en memòria, encara que l’aplicació tanqui el procés.

La classe “android.widget.Button” serà utilitzada per a programar les diferents accions quan es premi un botó.

La classe “android.widget.Spinner” serà utilitzada per a programar les diferents accions quan s’utilitzi un objecte spinner del layout.

La classe “android.widget.AdapterView” serà utilitzada per mostrar i programar els diferents elements del objecte Spinner.

La classe “android.widget.ArrayAdapter” serà utilitzada per identificar els diferents ítems necessaris per al objecte Spinner.

La classe “android.widget.Toast” serà utilitzada per mostrar missatges Toast.

Les classes “ com.anjlab.android.iab.v3.BillingProcessor”, “com.anjlab.android.iab.v3.TransactionDetails”, “android.support.annotation.NonNull” i “android.support.annotation.Nullable” són classes importades de la llibreria “Android In App Billing” que no està inclosa en Android Studio i que, per tant, s’inclourà pròximament al compilador Gradle. Les quatre classes seran utilitzades per tal d’incloure els pagaments dintre de l’aplicació

• **Classes incloses a info.java:**

La classe “ android.content.Intent” serà utilitzada per a iniciar una nova activitat o reiniciar l’actual.

La classe “android.support.design.widget.FloatingActionButton” serà utilitzada per tal de programar el botó “flotant” inclòs al layout.

La classe “android.view.Menu” serà utilitzada per a NO mostrar el menú.

La classe “android.view.View” serà utilitzada per a representar les diferents interfícies de usuari.

La classe “android.widget.Button” serà utilitzada per a programar les diferents accions quan es premi un botó.

La classe “android.net.Uri” serà utilitzada per a guardar dades de la pàgina web

8.4 Explicació del codi font

8.4.1 Activitat_Principal.java

1) Accions dels diferents botons.

* **Botó de càmera:**

Per al funcionament d’aquest, primerament es va haver de nombrar un objecte de tipus botó anomenat “camera”, que serà igual a el id de el botocamera definit en el layout. (Figura 73)

```
Button camera;  
camera = findViewById(R.id.botocamera);
```

Figura 73, botó càmera

Un cop definit l’objecte “camera”, es va programar un mètode anomenat “camera” que consisteix en crear un Intent, anomenat “i” el qual utilitzarà la classe MediaStore per tal d’executar l’aplicació predeterminada de la càmera (Figura 74).

```
// Mètode per a executar la càmera  
public void camera(){  
    Intent i = new Intent(MediaStore.INTENT_ACTION_STILL_IMAGE_CAMERA);  
    this.startActivity(i);  
}
```

Figura 74, configuració del mètode “camera”

Finalment, es crea una acció per quan el botó càmera es pressionat (Figura 75).

Aquesta acció executarà el mètode camera. Iniciant així l’aplicació.

```
camera.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        camera();  
    }  
});
```

Figura 75, configuració del botó càmera.

- **Botó de Galeria:**

Per al funcionament del botó Galeria, primerament es va crear un objecte de tipus Botó anomenat “galeria” que serà igual a el id de el botogaleria (Figura 76).

```
Button galeria;  
galeria = findViewById(R.id.botogaleria);
```

Figura 76, botó galeria

Un cop definit l’objecte “Galeria”, es va crear un mètode anomenat “galeria” que consisteix en crear un intent “i” de tipus ACTION_VIEW (per tant vol dir que el usuari pot veure i navegar pel content que es definirà) amb un proveïdor de continguts el qual obrirà el directori “/media/internal/images/media” que està relacionat amb els arxius multimèdia (Figura 77).

```
// Mètode per a executar la galeria
public void galeria(){
    Intent i = new Intent (Intent.ACTION_VIEW, Uri.parse("content://media/internal/images/media"));
    startActivity(i);
}
```

Figura 77, mètode galeria

Per finalitzar, s'assigna al botó galeria l'acció d'executar el mètode galeria quan hagi estat clicat.(Figura 78).

```
galeria.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        galeria();
    }
});
```

Figura 78, configuració del botó galeria

- **Botó Alarma**

Pel funcionament del botó rellotge, primerament es va crear un objecte de tipus Botó anomenat “rellotge” que serà igual a el id de el botoalarma (Figura 79).

```
Button rellotge;
rellotge = findViewById(R.id.botoalarma);
```

Figura 79, botó rellotge

Un cop definit l'objecte “rellotge”, es va crear un mètode anomenat “rellotge” que consisteix en crear i executar un intent “obrerellotge” que serà igual a un nou intent que executarà la classe AlarmClock i que farà l'acció d'obrir l'aplicació d'alarma (Figura 80).

```
// Mètode per a executar l'Alarma  
public void rellotge(){  
    Intent obrerellotge = new Intent(AlarmClock.ACTION_SET_ALARM);  
    startActivity(obrerellotge);  
}
```

Figura 80, mètode rellotge.

Per finalitzar, s'assigna al botó rellotge que quan es premi executi el mètode rellotge (Figura 81).

```
rellotge.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        rellotge();  
    }  
});
```

Figura 81, acció del botó rellotge.

- **Botó trucar.**

Pel funcionament del botó trucar, primerament es va crear un objecte de tipus Botó anomenat “trucar” que serà igual a el id del bototrucar (Figura 82) .

```
Button trucar;  
trucar = findViewById(R.id.bototrucar);
```

Figura 82, objecte trucar

Un cop definit l'objecte “trucar”, es va crear un mètode anomenat “marcador” que consisteix en crear i executar un intent “i” que serà igual a un nou intent de tipus ACTION_VIEW, la qual cosa indica que es mostrarà l'aplicació de trucades predeterminada per l'usuari (Figura 83).

```
public void marcador(){
    Intent i = new Intent(Intent.ACTION_DIAL);
    startActivity(i);
}
```

Figura 83, mètode marcador

Per finalitzar, s'assigna al botó trucar que quan es premi executi el mètode marcador (Figura 84).

```
trucar.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        marcador();
    }
});
```

Figura 84, botó trucar onClick

- **Botó contactes:**

Pel funcionament del botó contactes, primerament es va crear un objecte de tipus Botó anomenat “contactes” que serà igual a el id del botocontactes (Figura 85).

```
Button contactes;
contactes = findViewById(R.id.botocontactes);
```

Figura 85, botó contactes

Un cop definit l'objecte “contactes”, es va crear un mètode anomenat “contactes” que consisteix en crear i executar un intent “i” de tipus ACTION_VIEW (per tant vol dir que l'usuari pot veure i navegar pel content que es definirà) amb un proveïdor de continguts anomenat content://contacts/people/ que està relacionat amb els arxius multimèdia (Figura 86).

```
public void contactes(){
    Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("content://contacts/people/"));
    startActivity(i);
}
```

Figura 86, mètode contactes

Per finalitzar, s'assigna l'execució del mètode contactes quan es premi el botó contactes (Figura 87).

```
contactes.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        contactes();
    }
});
```

Figura 87, botó contactes onClick

- **Botó WhatsApp**

Per al funcionament del botó whatsapp , primerament es va crear un objecte de tipus Botó anomenat “whatsapp ” que serà igual al id del botowatsapp (Figura 88).

```
Button whatsapp;
whatsapp = findViewById(R.id.botowatsapp);
```

Figura 88, botó whatsapp

Un cop definit l'objecte “whatsapp”, es va crear un mètode anomenat “whatsapp” que consisteix en crear i executar un intent “i” que serà igual a l'acció d'executar el paquet amb la id “com.whatsapp” (Figura 89).

Per si un cas l'usuari no tingués instal·lada l'aplicació whatsapp, s'afegeix un estament if/else que consisteix en especificar que si el Intent “i” no és igual a “null” vol dir que l'aplicació whatsapp sí està instal·lada i per tant, l'acció pot ser executada. En canvi, si l'Intent “i” sí és “null”, es mostrerà un missatge de Tipus Toast amb la String anomenada nowatsapp.

```
public void whatsapp(){
    Intent i = getPackageManager().getLaunchIntentForPackage("com.whatsapp");
    if (i != null){
        startActivity(i);
    }
    else {
        Toast.makeText(this, R.string.nowatsapp, Toast.LENGTH_SHORT).show();
    }
}
```

Figura 89, mètode whatsapp.

Finalment, s'assigna l'execució del mètode whatsapp quan es premi el botó whatsapp (Figura 90).

```
whatsapp.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        whatsapp();
    }
});
```

Figura 90, botó whatsapp onClick

- **Botó Navegador**

Pel funcionament del botó navegador, primerament es va crear un objecte de tipus Botó anomenat “navegador” que serà igual al id del botonavegador (Figura 91).

```
Button navegador;
navegador = findViewById(R.id.botonavegador);
```

Figura 91, botó navegador

Un cop definit l'objecte navegador, es va crear un mètode anomenat “navegador” (Figura 92) que consisteix en crear un objecte Uri (proveïdor de contingut) anomenat “pagweb” que serà igual a la String “<https://google.es>” en un format adaptat per a Uri.

Després de definir l'objecte “pagweb”, es crea un Intent anomenat “google” que serà igual a un Intent de tipus ACTION_VIEW (per tant vol dir que l'usuari pot veure i navegar pel content

que es definirà), i que obrirà el contingut de l'objecte “pagweb. Finalment s'executa el Intent anomenat “google”.

```
public void navegador(){
    Uri pagweb = Uri.parse("https://google.es");
    Intent google = new Intent(Intent.ACTION_VIEW, pagweb);
    startActivity(google);
}
```

Figura 92, mètode navegador

Per finalitzar la programació del botó, s'assigna l'execució del mètode “navegador” quan es premi el botó navegador (Figura 93).

```
navegador.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        navegador();
    }
});
```

Figura 93, botó navegador onClick

- **Botó Configuració:**

Pel funcionament del botó configuració , primerament es va crear un objecte de tipus Botó anomenat “config” que serà igual al id del botoconf_android (Figura 94).

```
Button config;
config = findViewById(R.id.botoconf_android);
```

Figura 94, botó configuració

Un cop definit l'objecte “config”, es va crear un mètode anomenat “config” que consisteix en crear i executar un intent “i” que serà igual a l'acció d'executar el paquet amb la id “com.android.settings”, que és la id de l'aplicació configuració. Finalment s'executa l'Intent i (Figura 95).

```
public void config(){
    Intent i = getPackageManager().getLaunchIntentForPackage("com.android.settings");
    startActivity(i);
}
```

Figura 95, mètode config

Per finalitzar, s'assigna l'execució del mètode “config” quan es premi el botó configuració (Figura 96).

```
config.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        config();
    }
});
```

Figura 96, botó configuració onClick

- **Botó Mapes:**

Pel funcionament del botó mapes, primerament es va crear un objecte de tipus Botó anomenat “maps” que serà igual al id del botomaps (Figura 97).

```
final Button maps;
maps = findViewById(R.id.botomaps);
```

Figura 97, botó maps

Un cop definit l'objecte “maps”, es va crear un mètode anomenat “maps” que consisteix en crear i executar un intent “i” que serà igual a l'acció d'executar el paquet amb la id “com.google.android.apps.maps”, la qual és la id de l'aplicació Google Maps (Figura 98).

Per si un cas l'usuari no tingués instal·lada l'aplicació Google Maps, s'afegeix un estament if/else que consisteix en especificar que si el Intent “i” no és igual a “null” vol dir que l'aplicació Google Maps sí està instal·lada i per tant, l'acció pot ser executada. En canvi, si l'Intent “i” sí és “null”, es mostrarà un missatge de Tipus Toast amb la String anomenada nomaps.

```
public void maps(){
    Intent i = getPackageManager().getLaunchIntentForPackage("com.google.android.apps.maps");
    if (i != null){
        startActivity(i);
    } else {
        Toast.makeText(this,R.string.nosdk,Toast.LENGTH_LONG).show();
    }
}
```

Figura 98, mètode maps

Finalment, s'assigna l'execució del mètode maps quan es premi el botó mapes (Figura 99).

```
maps.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        maps();
    }
});
```

Figura 99, botó maps onClick

- **Botó Dades Mòbils:**

Per al funcionament del botó de Dades Mòbils, primerament es va crear un objecte de tipus Botó anomenat “dades” que serà igual a la id del btdades (Figura 100).

```
Button dades;
dades = findViewById(R.id.btdatos);
```

Figura 100, botó dades

Un cop definit l'objecte “dades”, es va crear un mètode anomenat “api” (Figura 101). Aquest mètode consisteix en un mètode if/else que consisteix en el següent:

- 1) Si la versió d'Android és igual o superior a Android P (API 28), es pot utilitzar la classe “android.provider.Settings” que serà l'encarregada d'executar l'apartat de dades mòbils dins de l'aplicació de configuració del dispositiu. Un cop l'aplicació sap que el dispositiu té la versió 28 del Android SDK o superior, es crea un intent “i” que serà

igual a un Intent d'obrir l'aplicació configuració i concretament l'apartat sobre l'ús de dades. Finalment s'obre l'activity.

- 2) Si la versió d'Android no és igual o inferior a Android P (API 28) es mostra un missatge Toast amb la String “nosdk”, la qual indica que el dispositiu no suporta aquesta funció.

```
public void api(){  
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.P) {  
        Intent i = new Intent(Settings.ACTION_DATA_USAGE_SETTINGS);  
        startActivity(i);  
    }  
    else {  
        Toast.makeText(this,R.string.nosdk,Toast.LENGTH_LONG).show();  
    }  
}
```

Figura 101, mètode API

Finalment, s'assigna l'execució del mètode api, quan es premi el botó dades (Figura 102).

```
dades.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        api();  
    }  
});
```

Figura 102, botó dades onClick

- **Botó tipus “Toggle” Wifi:**

Per al funcionament del botó Toggle wifi, primerament es va crear un objecte de tipus Botó Toggle anomenat “wifitoggle” que serà igual a la id del togglewifi (Figura 103).

```
ToggleButton wifitoggle;  
wifitoggle = findViewById(R.id.togglewifi);
```

Figura 103, Toggle Wifi

Un cop definit l'objecte “wifitoggle”, s'assigna el següent estament if/else quan es premi el toggle (Figura 104):

- 1) Si el botó ja ha estat clicat anteriorment, i per tal, està activat, es crearà un objecte de tipus WifiManager anomenat “wifi” que serà igual al context en el que es trobi el servei wifi. Finalment, dins de l'objecte “wifi”, s'assignarà el valor true a el boolean setWifiEnabled, el qual tindrà com a conseqüència encendre el wifi del dispositiu.
- 2) En canvi si el toggle no està encès, es crearà l'objecte de tipus WifiManager completament igual que a l'anterior on s'assignarà el valor false al boolean setWifiEnabled, el qual tindrà com a conseqüència apagar el wifi del dispositiu.

```
// Configura el toggle wifi
wifitoggle.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (isChecked){
            WifiManager wifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);
            wifi.setWifiEnabled(true);
        }
        else{
            WifiManager wifi = (WifiManager) getSystemService(Context.WIFI_SERVICE);
            wifi.setWifiEnabled(false);
        }
    }
});
```

Figura 104, Toggle Wifi onChecked

Finalment, es van escriure unes línies de codi mitjançant el codi d'exemple de la web Android Developers sobre comprovar si un dispositiu es troba connectat a la xarxa (<https://developer.android.com/training/basics/network-ops/managing>) per a saber si el dispositiu es troba connectat a la xarxa, per així, si ho està, activar el toogle wifi (Figura 105).

```
// Mira si el dispositiu està ja connectat a la xarxa i el tipus de connexió en què es troba
final ConnectivityManager connexio = (ConnectivityManager) getSystemService(Context.CONNECTIVITY_SERVICE);
for (Network network : connexio.getAllNetworks()) {
    NetworkInfo networkInfo = connexio.getNetworkInfo(network);
    if (networkInfo.getType() == ConnectivityManager.TYPE_WIFI) {
        wifitoggle.setChecked(true);
    }
}
```

Figura 105, ConnectivityManager

- **Botó tipus “Toggle” Llanterna:**

Pel funcionament del botó Toggle Llanterna, primerament es va crear un objecte de tipus Botó Toggle anomenat “llanternatoggle” que serà igual a la id del togglellanterna (Figura 106).

```
final ToggleButton llanternatoggle;
llanternatoggle = findViewById(R.id.togglellanterna);
```

Figura 106, Toggle Llanterna

Un cop definit el Toggle Llanterna, es van crear dues variables (Figura 107). Una variable anomenada flash de tipus boolean que serà igual a “false”, i una variable String [] (on les dues corxes indiquen que la String té un únic paràmetre) nomenada mCamerald.

```
public boolean flash = false;
private String[] mCameraId;
```

Figura 107, Variables

A continuació es va crear el mètode detectaflash (Figura 108). El qual és un mètode que retorna un paràmetre boolean (per tant true o false), el qual és la variable flash. Aquesta variable serà igual a el valor true o false de si el dispositiu té la funcionalitat de flash.

```
private boolean detectaflash() {
    flash = this.getPackageManager().hasSystemFeature(PackageManager.FEATURE_CAMERA_FLASH);
    return flash;
}
```

Figura 108, mètode detectaflash

Tot seguit, es va crear un objecte de tipus CameraManager anomenat “cam” que es igual a la String resultant de el servei CAMERA_SERVICE (servei que executa al càmera). Un cop creat l’objecte, es crea una excepció try/catch (la qual és utilitzada per tal de fer funcionar primera l’opció de try, i si no funciona executar les línies de codi que estiguin sota el catch), que s’utilitzarà primer per a guardar els diferents Id de la càmera o càmeres en la variable String[] mCamerald creada anteriorment. Finalment, l’excepció catch, s’utilitzarà per si en el cas de que l’accés a la càmera falli, ajudar a diagnosticar el que fa fallar l’excepció try (Figura 109).

```
final CameraManager cam = (CameraManager) getSystemService(Context.CAMERA_SERVICE);
try {
    mCameraId = cam.getCameraIdList();
}
catch (CameraAccessException e){
    e.printStackTrace();
}
```

Figura 109, Objecte cam

Per finalitzar, es programarà la funció de la llanternatoggle (Figura 110), la qual inclourà múltiples estaments if/else. El primer estament if/else, serà l'encarregat de comprovar si la versió d'Android és superior o igual a Android M (SDK 23), ja que és l'API mínima pel funcionament del cameraservice . Si és compleix aquest requisit, es continuarà amb el procediment, si no es compleix, es durà a terme el mètodenoflash, que consisteix en fer sortir per pantalla un missatge Toast amb la String “noflash”. Si l'API és superior o igual , es mirarà si el toggle està activat o desactivat. En el cas que estigui activat es mirarà si el mètode detectaflash és igual a true, i en el cas que ho sigui s'implementarà una excepció try/catch, on primerament es ficarà el TorchMode (mode llanterna) de la cameraID guardada a mCamerald a true, activant així la llanterna. En l'excepció catch, però, s'incorporarà la funció e.printStackTrace(); per a poder diagnosticar la falla de l'excepció try. En el cas que detectaflash no sigui igual a true, s'executarà el mètodenoflash, i es ficarà el llanternatoggle com a false (apagat).

```

// Configura el Toggle
llanternatoggle.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
    @RequiresApi(api = Build.VERSION_CODES.M)
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M){
            if (isChecked){
                if (detectaflash()) {
                    try {
                        cam.setTorchMode(String.valueOf(mCameraId), true);
                    } catch (CameraAccessException e) {
                        e.printStackTrace();
                    }
                } else {
                   noflash();
                    llanternatoggle.setChecked(false);
                }
            }else{
                if (detectaflash()){
                    try{
                        cam.setTorchMode(String.valueOf(mCameraId),false);
                    }catch (CameraAccessException e){
                        e.printStackTrace();
                    }
                } else {
                   noflash();
                    llanternatoggle.setChecked(false);
                }
            }
        } else {
           noflash();
        }
    });
});

```

Figura 110, llanternatoggle onChecked

En cas de que el toggle no estigui encès, es durà a terme el mateix procediment que amb el estament if, amb l'excepció de canviar el TorchMode de la camerald per el valor false, per tal de desactivar la llanterna (Figura 111).

```

public voidnoflash(){
    Toast.makeText(this,R.string.noflash, Toast.LENGTH_LONG).show();
}

```

Figura 111, mètode noflash

2) Configuració del menú

Per a mostrar i programar el menú, s'ha utilitzat la mateixa metodologia que amb l'aplicació calculadora.

Primerament es va crear un mètode anomenat “onCreateOptionsMenu” (Figura 112) el qual retorna un valor Boolean (true o false). Aquest mètode transforma arxius xml en objectes. Per tant, troba el menú gràcies a la id donada anteriorment, i el nombra menuconf. Finalment, retorna true, ja que així s'indica que existeix un menú i el mostra.

```
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.menu, menu);  
    return true;  
}
```

Figura 112, mètode onCreateOptionsMenu

Per poder mostrar les opcions del menú, es crea el mètode anomenat “onOptionsItemSelected” (Figura 113) el qual troba la id dels diferents ítems del menú, els assigna un nombre int anomenat id, i crea un switch on hi ha diferents accions dependent de el id que el usuari hagi triat.

En el cas que es triï el ítem configuració (conf) , s'executarà el mètode “conf” que executarà l'activity configuració (Figura 114). En canvi, en el cas que l'usuari triï el ítem informació (info), s'executarà el mètode “info”, el qual executarà l'activity informació (Figura 115) . Finalment, retorna true o false dependent si s'ha triat alguna opció del menú o no.

```
public boolean onOptionsItemSelected(MenuItem item) {
    // Handle item selection
    switch (item.getItemId()) {
        case R.id.config:
            conf();
            return true;
        case R.id.info:
            info();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Figura 113, mètode OnOptionsItemSelected

```
public void conf(){
    Intent conf = new Intent(this,configuracio.class);
    startActivity(conf);
}
```

Figura 114, mètode conf

```
public void info(){
    Intent i = new Intent(this, info.class);
    startActivity(i);
}
```

Figura 115, mètode info

3) Mode Obscur Automàtic

A més d'una opció per a poder canviar el tema predeterminat per l'aplicació, es va decidir afegir un mode obscur automàtic que s'encengués des de les 22:00 fins a les 6:00 .

Per aquesta tasca va ser necessari afegir el “style” fonsnegre i fonsblanc en els arxius xml relacionats amb els colors (trobat a l'apartat 8.2 explicació de la interfície gràfica). Un cop definits aquests paràmetres, es va crear el mètode “obscurautomatic”, el qual serà el gestionador del mode obscur (Figura 116).

```

public void obscurautomatic(){
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N){
        Calendar hora = Calendar.getInstance();
        int HORA = hora.get(Calendar.HOUR_OF_DAY);
        if (HORA <= 6){
            setTheme(R.style.FonsNegre);
        } else if (HORA >= 22) {
            setTheme(R.style.FonsNegre);
        }else{
            setTheme(R.style.FonsBlanc);
        }
    }
}

```

Figura 116, mètode obscurautomatic.

El mètode “obscurautomatic” consisteix en un conjunt d'estaments if i else on, primerament l'aplicació determina si la versió d'Android del dispositiu és superior o igual a Android N (SDK 24), per tal de continuar el procés, si aquest valor no és correcte, no s'executarà el procés dintre d'aquest if. En el cas que el primer estament if es compleixi, es va crear un objecte de tipus Calendar anomenat hora, on aquest tindrà tota la informació sobre la data i l'hora del dispositiu. Un cop creat l'objecte hora, es crea una variable int anomenada “HORA” la qual serà igual a una hora en format 24 hores . Seguidament, es va crear un estament if/else if/else el qual definia que si la variable HORA era menor o igual a 6 o superior o igual a 22 (les deu de la nit), el tema canviaria automàticament a el style FonsNegre. En canvi, si la variable hora no complia aquests paràmetres, s'utilitzaria el style predeterminat (FonsBlanc).

Finalment, es va afegir la línia de codi per tal d'executar el mètode obscurautomatic (Figura 117). Aquesta línia va ser afegida abans de carregar la interfície gràfica del layout per tal de que l'aplicació determinés primerament el style predeterminat i després carregués el layout correcte.

```

● obscurautomatic();
super.onCreate(savedInstanceState);
setContentView(R.layout.activitat_principal);

```

Figura 117, execució del mètode obscurautomatic.

4) Implementació d'anuncis

Per obtenir l' implementació d'anuncis dintre de l'aplicació, es va utilitzar la SDK de Google Admob, la qual és l'encarregada de mostrar i gestionar els anuncis dintre dels layouts d'aplicacions Android. Per tal d'importar la SDK dintre de l'aplicació, es va seguir la guia ràpida d'implementació de la SDK de la pàgina web oficial de Google Admob.

Primerament, es va haver d'implementar el repositori de google a el compilador Gradle, per tal de que aquest podés accedir a tots els diferents projectes d'aquest, i per tant donar la possibilitat d'incorporar a l'aplicació noves classes (Figura 118).

```
implementation 'com.android.support:design:28.0.0'
implementation 'com.google.android.gms:play-services-ads:17.1.0'
//noinspection GradleCompatible
```

Figura 118, Google Admob incorporat a Gradle

Un cop incorporada la funcionalitat del Google Admob, es va haver de crear un compte de Google Admob (Figura 119) per tal de tenir accés a una identificació pròpia (ID) per a que l'aplicació sàpiga a quina conta de google ha d'enviar les estadístiques i les recompenses econòmiques dels anuncis.

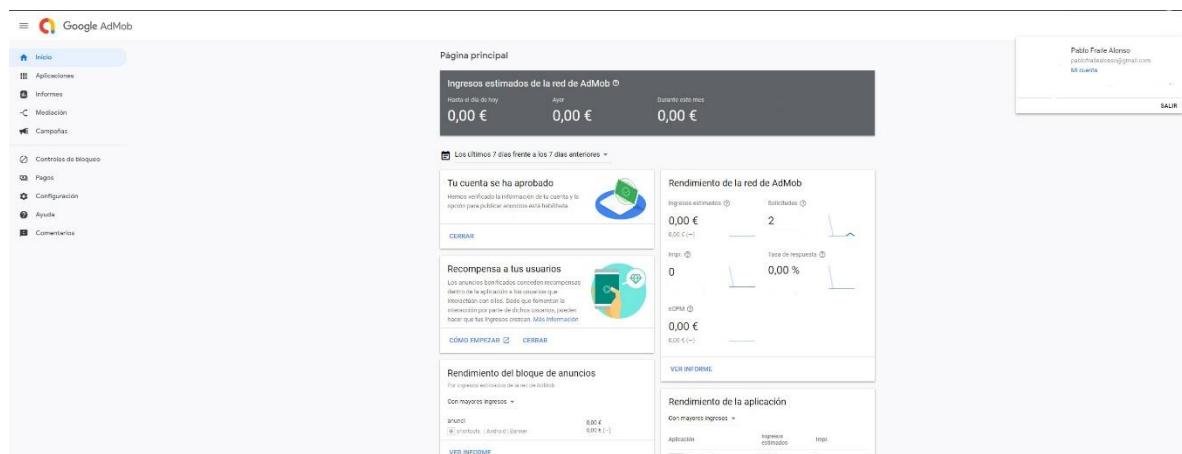


Figura 119, Pàgina principal de Google Admob

A posteriori de crear la conta de Google Admob, ja es va poder accedir a una ID, la qual va ser necessària per a configurar el AndroidManifest.xml, on es va haver d'especificar el nom de l'aplicació (on en aquest cas es va utilitzar el nom utilitzat a la guia d'Android Developers), i el valor de la ID de Google Admob.

En quant a la part del layout, es va decidir incorporar un anunci de tipus banner, el qual consisteix en una imatge rectangular que està situada en una part del layout. Per tant, es va copiar el codi XML de la guia de Google Admob (Figura 120) i es va desplaçar fins a la part inferior del layout.

```
<com.google.android.gms.ads.AdView xmlns:ads="http://schemas.android.com/apk/res-auto"  
    android:id="@+id/adView"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignParentBottom="true"  
    android:layout_centerHorizontal="true"  
    android:layout_marginBottom="0dp"  
    ads:adSize="BANNER"  
    ads:adUnitId="ca-app-pub-8364082387419453/8228162085" />
```

Figura 120, banner d'anuncis en el layout

Per finalitzar, es van incorporar quatre noves línies de codi a l'Activitat_Principal.java (Figura 121), per tal del funcionament i variació dels diferents anuncis dintre de l'aplicació.

Primerament, s'inicialitza el SDK de Google Admob, després es crea un objecte AdView anomenat mAdView que serà igual a la ID del objecte del layout anomenat adView. A continuació es va crear un objecte AdRequest (sol·licitud d'anuncis) anomenat adRequest,, el qual contindrà informació sobre una sol·licitud d'un anunci. Finalment, s'indica al objecte mAdView que ha de mostrar el anunci del objecte adRequest.

```
MobileAds.initialize(this, "ca-app-pub-8364082387419453~4975312786");  
AdView mAdView = findViewById(R.id.adView);  
AdRequest adRequest = new AdRequest.Builder().build();  
mAdView.loadAd(adRequest);
```

Figura 121, objecte mAdView

5) Programació del botó enrere.

Com l'aplicació “dreceres” és ara un llançador d'aplicacions i, per tant, pot ser elegida com l'aplicació predeterminada per aquesta acció, si es polsava el botó enrere del dispositiu no es podia anar al llançador principal ja que la pròpia aplicació era la predeterminada per a generar aquesta acció, i per tant, hi havia un error on únicament es mostrava una pantalla en negre.

Per a poder solucionar aquest problema, es va haver d'afegir un mètode per al botó enrere (Figura 122) , on s'indicava que quan es premés, es tornaria a iniciar l'Activitat_Principal, i per tant, l'aplicació es carregaria de nou.

```
public void onBackPressed(){
    Intent i = new Intent(this,Activitat_Principal.class);
    startActivity(i);
}
```

Figura 122, mètode onBackPressed

8.4.2 Explicació de “configuració.java”

1) Mètode per no mostrar el menú.

Com l'arxiu “configuracio” no es més que una extensió de la classe “Activitat_Principal”, implementa tots els canvis ja programats en aquest arxiu. Per tant, el layout “configuracio” continuarà mostrant el menú d'opcions, donat que el mètode seguirà retornant “true” .

Per tal d'eliminar aquest menú, es configura el mètode boolean per a mostrar el menú, i es retorna un valor “false”, el qual indicarà que no es mostri el menú (Figura 123).

```
public boolean onCreateOptionsMenu(Menu menu) { return false; }
```

Figura 123, mètode onCreateOptionsMenu

2) Configuració del Spinner per al canvi de color.

Per tal de poder canviar el color principal de l'aplicació, va ser necessari tenir en compte la durabilitat d'aquest canvi, donat que en l'anterior aplicació “Calculadora”, es va incorporar un canvi de tema que no era definitiu. En canvi, en l'aplicació “Dreceres” es va decidir incorporar l'opció d'un canvi de tema definitiu (es a dir, un canvi de tema que quan es tanqués el procés de l'aplicació, el tema predeterminat sigue l'elegit pel usuari). Per aconseguir aquest objectiu, va ser totalment necessària la classe “SharedPreferences” .

Primerament, es va definir l'objecte de tipus Spinner anomenat “color” (Figura 124) , el qual serà l'encarregat de mostrar els diferents colors al usuari.

```
Spinner color;
color = findViewById(R.id.spinnercolor);
```

Figura 124, objecte color

Seguidament, es van crear tres variables a l'arxiu “Activitat_Principal.java” que seran necessàries per aquest procediment.

```
public static final String SHARED_PREFS = "sharedPrefs";
public static final String TEXT = "text";
public int nombre = 0;
```

Figura 125, variables

A continuació, seguint la guia d'Android Developers, es va definir un objecte ArrayAdapter anomenat adapter (el qual indica els diferents objectes que es mostraran a la fila del spinner), i es definirà com l'adapter predeterminat per l'objecte color. Finalment, es va programar l'acció del objecte color quan l'usuari pren alguna opció del array (Figura 126)

```

ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this, R.array.colors, android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
color.setAdapter(adapter);
color.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        switch (position){
            case 0:
                break;

            case 1:
                guardadadesrosa();
                reinicia();
                break;

            case 2:
                guardadadesblau();
                reinicia();
                break;

            case 3:
                guardadadesgris();
                reinicia();
                break;

            case 4:
                guardadescyan();
                reinicia();
                break;

            case 5:
                guardadesverd();
                reinicia();
                break;

            case 6:
                guardadestaronja();
                reinicia();
                break;
        }
    }
    @Override
    public void onNothingSelected(AdapterView<?> parent) {}
});
```

Figura 126, configuració del Spinner Color

Un cop l'usuari ha elegit un objecte del Spinner, aquest crida a un mètode anomenat “guardadades + el color triat” (Figura 127), el qual crearà un objecte SharedPreferences anomenat “sharedPreferences” el qual serà igual a la String SHARED_PREFS creada anteriorment, la qual serà assignada d'un mode privat (cosa que indica que les dades que es guardaràn dintre de les diferents variables d'aquesta SHARED_PREFS, seran accessibles únicament per la pròpia aplicació Dreceres, i no seran visibles per altres aplicacions). Dintre del objecte SharedPreferences anomenat SHARED_PREFS, s'editarà la variable TEXT creada a “Activitat_Principal.java” per tal de que sigui relacionada amb un nombre int, que depenen del color serà 1,2,3,4,5 o 6.

Finalment, s'aplicarà el canvi del editor, i es mostrarà un missatge Toast el qual indicarà al usuari que el canvi de tema s'ha efectuat correctament.

```
public void guardadadesrosa(){
    final SharedPreferences sharedPreferences = getSharedPreferences(SHARED_PREFS, MODE_PRIVATE);
    final SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putInt(TEXT, 1);
    editor.apply();
    Toast.makeText(this,R.string.rosaaplicat, Toast.LENGTH_LONG).show();
}
public void guardadadesblau(){
    final SharedPreferences sharedPreferences = getSharedPreferences(SHARED_PREFS, MODE_PRIVATE);
    final SharedPreferences.Editor editor = sharedPreferences.edit();
    editor.putInt(TEXT, 2);
    editor.apply();
    Toast.makeText(this,R.string.blauaplicat, Toast.LENGTH_LONG).show();
}
```

Figura 127, mètode guardadadesrosa i guardadadesblau.

Al finalitzar l'execució del mètode “guardadades + el color triat”, es crida al mètode “reinicia”, el qual reinicia l'activitat “configuracio” creant un nou Intent anomenat “i” que carrega la classe “configuracio.class”. Aquest mètode serà crític per tal de poder aplicar el tema en temps real, sense que l'usuari tingui que tancar l'aplicació manualment per tal de que el layout es torni a reiniciar.

Com els paràmetres relacionats amb la variable “TEXT” per si sols no indiquen res, es va crear un mètode anomenat “cargatema” a l'activitat “Activitat_Principal.java” (Figura 128), el qual carga l'objecte SharedPreferences i dona el valor int de la variable “TEXT” a la variable “nombre”. Un cop definit la variable “nombre”, es creen un conjunt d'estaments “if” els quals indiquen que si el nombre és igual a 1, es carrega el tema predeterminat, si el nombre és igual a 2, es carrega el tema blau, i així successivament fins arribar a 6, on es carregarà el tema taronja.

```

public void cargatema(){
    SharedPreferences sharedpreferences = getSharedPreferences(SHARED_PREFS, MODE_PRIVATE);
    nombre = sharedpreferences.getInt(TEXT,0);
    if (nombre == 1){
        setTheme(R.style.AppTheme);
    }
    if (nombre == 2){
        setTheme(R.style.TemaBlau);
    }
    if (nombre == 3){
        setTheme(R.style.TemaGris);
    }
    if (nombre == 4){
        setTheme(R.style.TemaCyan);
    }
    if (nombre == 5){
        setTheme(R.style.TemaVerd);
    }
    if (nombre == 6){
        setTheme(R.style.TemaTaronja);
    }
}

```

Figura 128, mètode *cargatema*

Per finalitzar es carregarà el mètode “*cargatema*” justament al iniciar l’aplicació (Figura 129).

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    ● cargatema();
    obscurautomatic();
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activitat_principal);
}

```

Figura 129, execució del mètode *cargatema*.

3) Configuració del botó per eliminar anuncis

Per tal de poder eliminar els anuncis del layout principal, es va decidir incorporar un botó amb l’habilitat de fer compres dintre de l’aplicació, per tal de poder pagar una quantitat significativa per poder eliminar completament els anuncis. Per poder realitzar aquest procés, es va decidir utilitzar una llibreria aliena a Android Studio, anomenada “com.anjlab.android.iab.v3”, la qual inclou instruccions per a ser implementada dintre del projecte d’Android Studio.

Primerament, es va incorporar aquesta llibreria dintre del compilador Gradle (Figura 130), per tal de poder treballar amb les diferents classes que aquesta inclou.

```
dependencies {  
    ...  
    implementation 'com.anjlab.android.iab.v3:library:1.0.44'  
}
```

Figura 130, implementació de la llibreria “com.anjlab.android.iab.v3”

Seguidament, es va declarar el permís necessari a l’AndroidManifest.xml, per tal que l’aplicació tingui el permís del sistema per a poder llançar els missatges de compra dintre de l’aplicació.

```
<uses-permission android:name="com.android.vending.BILLING" />
```

Figura 131, Declaració del permís a l’AndroidManifest.xml

A continuació, es van afegir diferents línies de codi per tal d’incorporar la llibreria a l’arxiu “configuracio.java”. Primerament, es va indicar que la classe “configuracio”, que a la vegada era una extensió de la classe “Activitat_Principal”, incorporava una interfície (un conjunt de mètodes sense codi programat dintre) anomenada “BillingProcessor.IBillingHandler”, la qual serà la causant de tots els mètodes de pagament.

```
public class configuracio extends Activitat_Principal implements BillingProcessor.IBillingHandler {  
}
```

Figura 132, creació classe “configuracio”

Un cop implementada la interfície, es va crear dintre de la classe principal un objecte BillingProcessor anomenat “bp” (Figura 133) . Seguidament, al mètode “onCreate”, es va definir l’objecte “bp” com un nou procés de pagament el qual conté la llicència de l’aplicació (la qual va ser proporcionada per la Google Play Console a l’hora de penjar l’aplicació).

```
BillingProcessor bp;
```

Figura 133, objecte bp

```
bp = new BillingProcessor(this, "MIIBIjANBgkqhkiG9w0BAQEAAQCAQ8AMIIIBCgKCAQEApS0SU8
```

Figura 134, Definició de l’objecte “bp”.

Al finalitzar l'assignació del objecte “bp”, es va crear el botó “anuncis”, el qual era igual a la id del botó “anuncis” dintre del layout (Figura 135).

```
Button anuncis;  
anuncis = findViewById(R.id.anuncis);
```

Figura 135, botó anuncis

En cas de que es cliqués el botó “anuncis” (Figura 136), es va programar un estament “if/else” que indicava si l'aplicació complia el mètode de tipus boolean anomenat “playstore” el qual és necessari per saber si el telèfon consta de l'aplicació PlayStore (Figura 137).

```
anuncis.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        if (playstore()) {  
            bp.purchase(configuracio.this, "com.shortcuts.noanuncis");  
        } else {  
            Toast.makeText(configuracio.this, R.string.noplaystore, Toast.LENGTH_LONG).show();  
        }  
    }  
});
```

Figura 136, anuncis onClick.

```
public boolean playstore(){  
    boolean playstore = BillingProcessor.isTabServiceAvailable(this);  
    boolean si = true;  
    if (!playstore){  
        si = false;  
    }  
    return(si);  
}
```

Figura 137, mètode playstore

Aquest mètode crea dues variables boolean, una anomenada playstore, que serà igual al mètode “BillingProcessor.isTabServiceAvailable” el qual s'inclou en la llibreria anteriorment incorporada i retorna un valor “true” si el dispositiu inclou l'aplicació Google Play Store, i retorna un valor “false” si el dispositiu no inclou l'aplicació. Finalment, es crea un estament “if” el qual indica que si la variable playstore retorna “false”, l'aplicació dona el valor “false” a la variable “si”. Per finalitzar, es retorna el valor de si.

Quan ja s'ha executat el mètode “playstore”, amb els estaments “if/else”, es comprova si el mètode ha retornat un valor “true”. En aquest cas, l'aplicació donaria a l'aplicació l'acció d'executar el procediment de compra, el qual estaria format per el nom de la classe executada en el moment(configuracio.class) i en la id del producte definida anteriorment a la Google Play Console. En el cas de que el valor retornat per el mètode playstore fos “false”, es mostraria per pantalla un missatge de tipus Toast amb la String noplaystore.

Per a poder controlar quan l'aplicació ha estat comprada i quan no, es va decidir utilitzar la classe “SharedPreferences”. Per això, primerament es va haver de crear dues variables a la classe “Activitat_Principal” (ja que és la classe on es mostraran els anuncis) anomenades COMPRA (la qual es de tipus String) i noanuncis (la qual és de tipus boolean) (Figura 138).

```
public static final String COMPRA = "compra";
public boolean noanuncis = false;
```

Figura 138, variables COMPRA i noanuncis.

Una vegada creades les variables, es van haver de configurar els diferents mètodes que es van crear automàticament a l'hora d'implementar la interfície “BillingProcessor.IBillingHandler”.

El mètode principal per al funcionament serà l'anomenat “onProductPurchased”, el qual serà l'indicat de realitzar una acció quan el pagament hagi estat realitzat amb èxit. En aquest mètode s'utilitzarà la variable “SHARED_PREFS” utilitzada anteriorment pel canvi de color de l'aplicació per tal de relacionar la variable COMPRA amb el nom “true” (Figura 139). Finalment s'aplicaràn els canvis.

```
@Override
public void onProductPurchased(@NonNull String productId, @Nullable TransactionDetails details) {
    final SharedPreferences sharedpreferences = getSharedPreferences(SHARED_PREFS, MODE_PRIVATE);
    final SharedPreferences.Editor editor = sharedpreferences.edit();
    editor.putBoolean(COMPRA,true);
    editor.apply();
    Toast.makeText(this,R.string.comprat, Toast.LENGTH_LONG).show();
}
```

Figura 139, mètode onProductPurchased

Als altres mètodes els quals són:

- “onPurchaseHistoryRestored”: El qual serà cridat quan l’usuari demani el reemborsament del capital pagat per eliminar els anuncis dintre de l’aplicació. Com no s’indica que el usuari hagi comprat l’aplicació, es marca la variable COMPRA com a “false”.
- “onBillingError”: El qual serà cridat quan passi quelcom error dintre del mètode pagament. Com no s’indica que el usuari hagi comprat l’aplicació, es marca la variable COMPRA com a “false”.
- “onBillingInitialized”: El qual serà cridat quan el procediment de comprar ha estat inicialitzat i està llest per admetre les dades de compra. Com no s’indica que el usuari hagi comprat l’aplicació, es marca la variable COMPRA com a “false”.
- “onDestroy”: El qual serà l’encarregat de finalitzar el servei de pagament un cop acabat, útil per a no perjudicar el rendiment del dispositiu.
- “onActivityResult”: El qual s’executarà quan l’activitat la qual s’estava realitzant es finalitza. Dona dades sobre el resultat de l’activity.

```

@Override
public void onPurchaseHistoryRestored() {
    final SharedPreferences sharedpreferences = getSharedPreferences(SHARED_PREFS, MODE_PRIVATE);
    final SharedPreferences.Editor editor = sharedpreferences.edit();
    editor.putBoolean(COMPRA,false);
    editor.apply();
}
@Override
public void onBillingError(int errorCode, @Nullable Throwable error) {
    final SharedPreferences sharedpreferences = getSharedPreferences(SHARED_PREFS, MODE_PRIVATE);
    final SharedPreferences.Editor editor = sharedpreferences.edit();
    editor.putBoolean(COMPRA,false);
    editor.apply();
}
@Override
public void onBillingInitialized() {
    final SharedPreferences sharedpreferences = getSharedPreferences(SHARED_PREFS, MODE_PRIVATE);
    final SharedPreferences.Editor editor = sharedpreferences.edit();
    editor.putBoolean(COMPRA,false);
    editor.apply();
}
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (!bp.onActivityResult(requestCode, resultCode, data)) {
        super.onActivityResult(requestCode, resultCode, data);
    }
}
@Override
public void onDestroy(){
    if(bp != null){
        bp.release();
    }
    super.onDestroy();
}

```

Figura 140, mètodes de compra ocasionals

Per finalitzar la configuració de tot el botó anuncis, es va crear el mètode “cargaanuncis” (Figura 141) el qual consistia en carregar les dades de la variable COMPRA i fer que la variable “noanuncis” fos igual a el valor en format boolean de la variable COMPRA. Un cop definit el mètode, aquest s’executa, i en el cas de que “noanuncis” sigui “false” es mostraran els anuncis (Figura 142).

```

public void cargaanuncis(){
    SharedPreferences sharedpreferences = getSharedPreferences(SHARED_PREFS,MODE_PRIVATE);
    noanuncis = sharedpreferences.getBoolean(COMPRA,false);
}

```

Figura 141, mètode cargaanuncis.

```
cargaanuncis();
if (!noanuncis){
    MobileAds.initialize(this,"ca-app-pub-8364082387419453~4975312786");
    AdView mAdView = findViewById(R.id.adView);
    AdRequest adRequest = new AdRequest.Builder().build();
    mAdView.loadAd(adRequest);
}
```

Figura 142, estament if per decidir si mostrar o no els anuncis.

Com a últim afegit, es va decidir afegir una línia de codi a l'arxiu “configuracio.class” la qual indicava que si la variable “noanuncis” era igual a “true”, el botó “anuncis” quedaria des habilitat (Figura 143).

```
if (noanuncis){ ads.setEnabled(false);}
```

Figura 143, des habilitar e botó anuncis.

4) Configuració del botó per elegir l'aplicació d'inici predeterminada.

Per tal de poder elegir l'aplicació d'escriptori predeterminada, es va decidir incorporar un botó que fes d'accés directe a l'apartat de l'aplicació “configuració” del dispositiu.

Primerament, es va definir l'objecte de tipus botó anomenat “aplicacionspredeterminades”, el qual és igual a la id “button” (que indica el botó d'aplicacions predeterminades en el layout) (Figura 144).

```
Button aplicacionspredeterminades;
aplicacionspredeterminades = findViewById(R.id.button);
```

Figura 144, botó aplicacionspredeterminades

A continuació, es va configurar l'objecte “aplicacionspredeterminades”(Figura 145) per tal de que quan es cliqués, iniciés un estament “if/else” que indica el següent:

- Si la versió d'Android és igual o superior a la versió d'Android “N” (SDK 24), es crea un Intent anomenat “i” el qual crearà un intent que anirà directament a l'apartat d'elecció d'aplicacions predeterminades de l'aplicació configuració. Aquesta funció és únicament compatible amb versions d'Android N o superior, es per això que es necessari el estament “if/else”.

- Si la versió d'Android no és igual o superior de la SDK 24, es mostra a l'usuari un missatge de tipus Toast amb la String “APInoN”, la qual indica que la versió d'Android no és compatible

```
    aplicacionespredeterminadas.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.N){
                Intent i = new Intent(android.provider.Settings.ACTION_MANAGE_DEFAULT_APPS_SETTINGS);
                startActivity(i);
            } else {
                Toast.makeText(configuracio.this,R.string.APInoN,Toast.LENGTH_LONG).show();
            }
        }
    });
});
```

Figura 145, *aplicacionespredeterminadas onClick*

8.4.3 Explicació de “info.java”

El arxiu info.java es aquell que serà executat en el format .class quan l'usuari hagi clicat l'opció “sobre l'aplicació” en el menú. Aquest arxiu, és una extensió de la “activitat_principal.java”, per tant, cal recalcar que totes les línies de codi escriptes anteriorment en l'activitat principal, seran també executats. Tenint en compte aquest fenomen, caldrà doncs, desactivar tant el mètode per a mostrar el menú (Figura 146).

```
public boolean onCreateOptionsMenu(Menu menu) { return false; }
```

Figura 146, mètode *OnCreateOptionsMenu*

Per la configuració dels botons, primerament es van declarar com a botons mitjançant la id nombrada anteriorment (Figura 147), i posteriorment es van configurar amb l'acció “setOnClickListener”. L'acció consistia en crear un uri.parse (que consisteix en un proveïdor de continguts, simplement guarda dades) amb les pàgines web que es volien executar. Finalment, s'iniciava un Intent amb una ACTION_VIEW, que vol dir que es vol mostrar dades a l'usuari (Figura 148).)

```

Button github;
github = findViewById(R.id.botogithub);

Button llicencia;
llicencia = findViewById(R.id.botognu);

```

Figura 147, botons github i llicencia.

```

github.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Uri pagweb = Uri.parse("https://github.com/Pablit02020/Treball-de-Reerca");
        Intent githubweb = new Intent(Intent.ACTION_VIEW, pagweb);
        startActivity(githubweb);
    }
});

llicencia.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Uri pagweb = Uri.parse("https://raw.githubusercontent.com/Pablit02020/Treball-de-Reerca/master/LICENSE");
        Intent gnu = new Intent(Intent.ACTION_VIEW, pagweb);
        startActivity(gnu);
    }
});

```

Figura 148, botons github i llicencia onClick

Finalment, per la programació del botó compartir, es va crear un mètode anomenat “share”, basat en la guia de la web Android Developers sobre el funcionament dels Intents.

Primerament, es va crear un objecte Intent anomenat “i” de tipus “send”, el qual permet enviar dades entre una aplicació i una altra.

A continuació, es van crear tres strings anomenades:

- “titol”(la qual és la String indicada de mostrarà la frase: com parteix l’aplicació”).
- “cos: la qual mostrarà la frase: Siusplau, descarrega l’aplicació dreceres, és molt útil! ”
- “comparteixamb: la qual mostrarà la frase: Compartir l’aplicació amb”.

Un cop creades les variables, es va definir el títol del Intent com la String “títol” i el cos del intent es va definir amb la variable cos. Finalment es va programar l’inici de la activity que mostrarà com a títol principal la frase inclosa dintre de la String “compartiramb”.

```
public void share(){
    Intent i = new Intent(Intent.ACTION_SEND);
    i.setType("text/plain");

    String titol = getString(R.string.titol);
    String cos = getString(R.string.cos);
    String comparteixamb = getString(R.string.comparteixamb);

    i.putExtra(Intent.EXTRA_SUBJECT, titol);
    i.putExtra(Intent.EXTRA_TEXT, cos);
    startActivity(Intent.createChooser(i, comparteixamb));
}
```

Figura 149, mètode share

Per finalitzar, es va crear un objecte de tipus FloatingActionButton anomenat “compartir” el qual era igual a la id del “botoflotant” del layout. Un cop creat l’objecte, es va definir la seva acció a la hora de ser clicat, la qual seria accionar el mètode “share”.

```
compartir.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        share();
    }
});
```

Figura 150, botó compartir onClick.

8.5 Penjar l’aplicació a la Google Play.

El procediment de penjar l’aplicació a la Google Play va ser el mateix que en l’aplicació calculadora (apartat 7.5).

Els únics canvis notables van ser l’arxiu APK, la descripció de l’aplicació i les imatges que es mostren a la Google Play (Figura)



Figura 151, imatge Dreceres de la Google Play Store

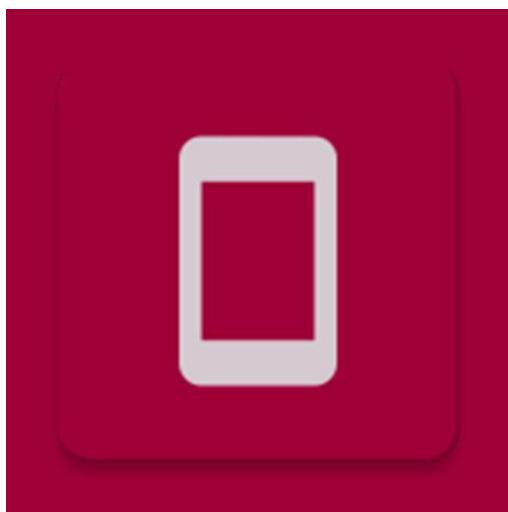


Figura 152, logo Dreceres.

Conclusions

Un cop assolits els objectius dels dos projectes s'ha arribat a les següents conclusions:

- Per tal de programar en Java, s'ha de tenir en compte les paraules reservades del codi font i les diferents classes que aquest pot tenir.
- La màquina virtual és necessària per a que qualsevol dispositiu sigui capaç d'executar un programa amb un codi escrit en Java.
- Les llibreries externes poden incorporar noves interfícies i mètodes que no es troben en el java JDK.
- Un entorn de desenvolupament integrat (IDE) és indispensable per tal de tenir un control total sobre el codi font i els recursos de l'aplicació.

WebGrafia

- Steven S.Muchnick (1997). Advanced Compiler Design Implementation. Recuperat 13 agost, 2018, de
https://books.google.es/books?hl=es&lr=&id=Pq7pHwG1_OkC&oi=fnd&pg=PA1&dq=compiler&ots=4Z6_Npb6nO&sig=ZYKLNEtyd7srae5M19YnLtRUjqQ#v=onepage&q&f=false
- Oracle. (s.d.), Lesson: Common Problems (and Their Solutions). Recuperat 16 maig, 2018, de <https://docs.oracle.com/javase/tutorial/getStarted/problems/index.html>
- Javier Smaldone, J.S. (s.d.). Software Libre versus Software Propietario. Recuperat 20 maig, 2018, de <https://smaldone.com.ar/opinion/docs/slvssp.html>
<https://developer.android.com/guide/topics/resources/providing-resources?hl=es-419>
- Free Software Fundation, INC. (2007) .A Quick Guide to GPL v3. Recuperat 20 maig, 2018, de <https://www.gnu.org/licenses/quick-guide-gplv3.html>
- Elivar Largo, E.L. (2016, 4 octubre). 4 Elementos claves para entender POO en Java [Publicació en un blog]. Recuperat 14 maig, 2018, de <https://www.ecodeup.com/4-elementos-claves-para-entender-la-programacion-orientada-objetos-en-java/>
- AnjLab. (2014, 28 febrer). Android In-App Billing v3 Library [Publicació en el fòrum de codi github]. Recuperat 16 novembre, 2018, de <https://github.com/anjlabs/android-inapp-billing-v3>
- Android Developers. (2018, 7 agost). Configurar el ID de aplicación. Recuperat 26 setembre, 2018, de <https://developer.android.com/studio/build/application-id?hl=es-419>
- Android Developers. (2018, 4 octubre). Wi-Fi scanning overview. Recuperat 8 novembre, 2018, de <https://developer.android.com/guide/topics/connectivity/wifi-scan#java>
- Android Developers. (2018, 25 abril). Iniciar otra activity. Recuperat 30 agost, 2018, de <https://developer.android.com/training/basics/firstapp/starting-activity?hl=es-419>
- Android Developers. (2018, 25 abril). Creación de una interfaz de usuario sencilla. Recuperat 28 agost, 2018, de <https://developer.android.com/training/basics/firstapp/building-ui?hl=es-419>
- Android Developers. (2018, 25 abril). Build and run your app. Recuperat 14 agost, 2018, de <https://developer.android.com/studio/run/>

- Android Developers. (2018, 25 abril). Biblioteca de compatibilidad. Recuperat 21 agost, 2018, de <https://developer.android.com/topic/libraries/support-library/?hl=es-419>
- Android Developers. (2018, 25 abril). Arquitectura de la plataforma. Recuperat 6 agost, 2018, de <https://developer.android.com/guide/platform/?hl=es-419>
- Android Developers. (2018, 13 novembre). Configurar tu compilación. Recuperat 1 desembre, 2018, de <https://developer.android.com/studio/build/?hl=es-419>
- Android Developers, (2018, 8 novembre). Device compatibility overview. Recuperat 15 novembre, 2018, de <https://developer.android.com/guide/practices/compatibility?hl=es-419>
- Android Developers, (2018, 8 novembre). Add a Floating Action Button. Recuperat 16 novembre, 2018, de <https://developer.android.com/guide/topics/ui/floating-action-button>
- Android Developers, (2018, 7 setembre). Google Admob Get Started. Recuperat 2 novembre, 2018, de <https://developers.google.com/admob/android/quick-start>
- Android Developers, (2018, 7 setembre). Banner Ads. Recuperat 2 novembre, 2018, de <https://developers.google.com/admob/android/banner>
- Android Developers, (2018, 28 agost). Intentos y filtros de intents. Recuperat 6 novembre, 2018, de <https://developer.android.com/guide/components/intents-filters?hl=es-419>
- Android Developers, (2018, 25 abril). Provisión de recursos. Recuperat 30 agost, 2018 de
- Android Developers, (2018, 25 abril). Permissions Overview. Recuperat 31 agost, 2018, de <https://developer.android.com/guide/topics/permissions/overview?hl=es-419>
- Android Developers, (2018, 25 abril). Manifiesto de la app. Recuperat 31 agost, 2018, de <https://developer.android.com/guide/topics/manifest/manifest-intro?hl=es-419>
- Android Developers, (2018, 25 abril). Firmar tu aplicación. Recuperat 6 novembre, 2018, de <https://developer.android.com/studio/publish/app-signing?hl=es-419>
- Android Developers, (2018, 25 abril). Controles de números. Recuperat 15 agost, 2018, de <https://developer.android.com/guide/topics/ui/controls/spinner?hl=es-419>
- Android Developers, (2018, 25 abril). Aspectos fundamentales de la aplicación. Recuperat 30 agost, 2018, de <https://developer.android.com/guide/components/fundamentals?hl=es-419>
- Android Developers, (2018, 23 abril). Camera. Recuperat 10 octubre, 2018, de <https://developer.android.com/training/camera/?hl=es-419>

- Android Developers, (2018, 17 abril). Introduction to Activities. Recuperat 10 octubre, 2018, de <https://developer.android.com/guide/components/activities/intro-activities?hl=es-419>
- ¿Qué es Java? (2013, 5 febrer). Recuperat 14 maig, 2018, de http://aularagon.catedu.es/materialesaularagon2013/POO-Tecnologia/M1/qu_es_java.html