Universitat Oberta de Catalunya
uoc.edu

MEMÒRIA DOBILE

Estudiant: Macià Sintes Andreu

Grau Enginyeria Informàtica
TFG Desenvolupament aplicacions dispositius mòbils (Android)

Consultors: David Escuer Latorre
Jordi Almirall López

Professor assignatura: Carles Garrigues Olivella

Data Lliurament: 05/06/2020

FITXA DEL TREBALL FINAL

Títol del treball:	*Dobile*
Nom de l'autor:	*Macià Sintes Andreu*
Nom dels consultors:	*David Escuer Latorre* *Jordi Almirall López*
Nom del PRA:	*Carles Garrigues Olivella*
Data de lliurament (mm/aaaa):	*03/2020*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	*TFG Desenvolupament aplicacions dispositius mòbils (Android)*
Idioma del treball:	*Català*
Paraules clau	*Domòtica, Android, Arduino*

Resum del Treball (màxim 250 paraules): *Aplicació mòbil desenvolupada en Android de tipus domòtic. La finalitat és el control centralitzat de diferents aparells elèctrics com per exemple llums, motors, etc. Així com també el control d'equips d'aire condicionat. El control es fa mitjançant targetes Arduino o similars.*

Abstract (in English, 250 words or less): *Mobile application developed on Android intended at home automation. The purpose is the centralized control of different electrical appliances such as lamps, motors, etc. As well as the control of air conditioning machines. The control is done using Arduino boards.*

Índex

Llista de figures

1. Introducció

1.1 Context i justificació del Treball

Es tracta d'una aplicació desenvolupada en Android anomenada Dobile. La necessitat a cobrir és el poder controlar diversos aparells elèctrics de casa des de un sol lloc. Encendre o apagar llums, ventiladors, etc. Controlar aparells des del mòbil sense la necessitat del comandament a distància. La principal finalitat és la comoditat, al poder manejar els aparells de manera centralitzada.

També en el cas de l'aire condicionat es té l'avantatge de poder arrancar l'aparell amb prou temps per trobar la temperatura adequada quan arribem a casa, sobre tot en dies de molt de fred o molt de calor.

Ja existeixen diferents opcions semblants en el mercat. La majoria, a més de l'app pròpia, tenen el seu hardware, el qual rep les comandes des de l'app. Principalment es tracta de control de interruptors remots.

A continuació es mostra una comparativa amb diferents aplicacions ja existents al mercat.

Smart Home

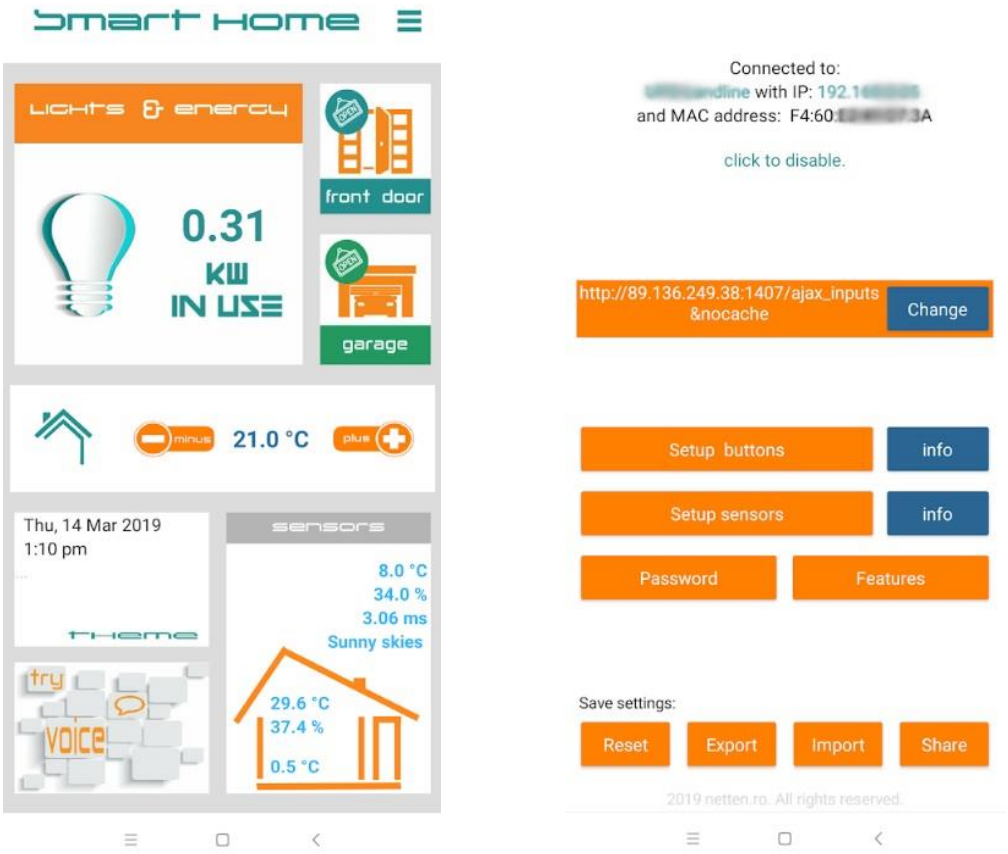


Figura 1 – Captures Smart Home

https://play.google.com/store/apps/details?id=com.nettenro.SmartHome
http://www.netten.ro/

La finalitat d'aquesta aplicació és la mateixa que la nostra, controlar diversos aparells i sensors mitjançant dispositius Arduino, Raspberry, Esp 8266, etc. L'usuari també ha de configurar una IP per cada dispositiu, com a la nostra.

Els dispositius remots contenen un fitxer XML/JSON amb tots els paràmetres. L'usuari ha de configurar els botons que vol, quin paràmetre vol llegir, quin paràmetres ha d'enviar. La configuració pot ser una mica confusa, comparat amb la nostra i l'usuari ha de conèixer exactament quins paràmetres enviar o quins paràmetres ha de llegir. Es necessita un coneixement tècnic per utilitzar-la.

Home Automation Projects

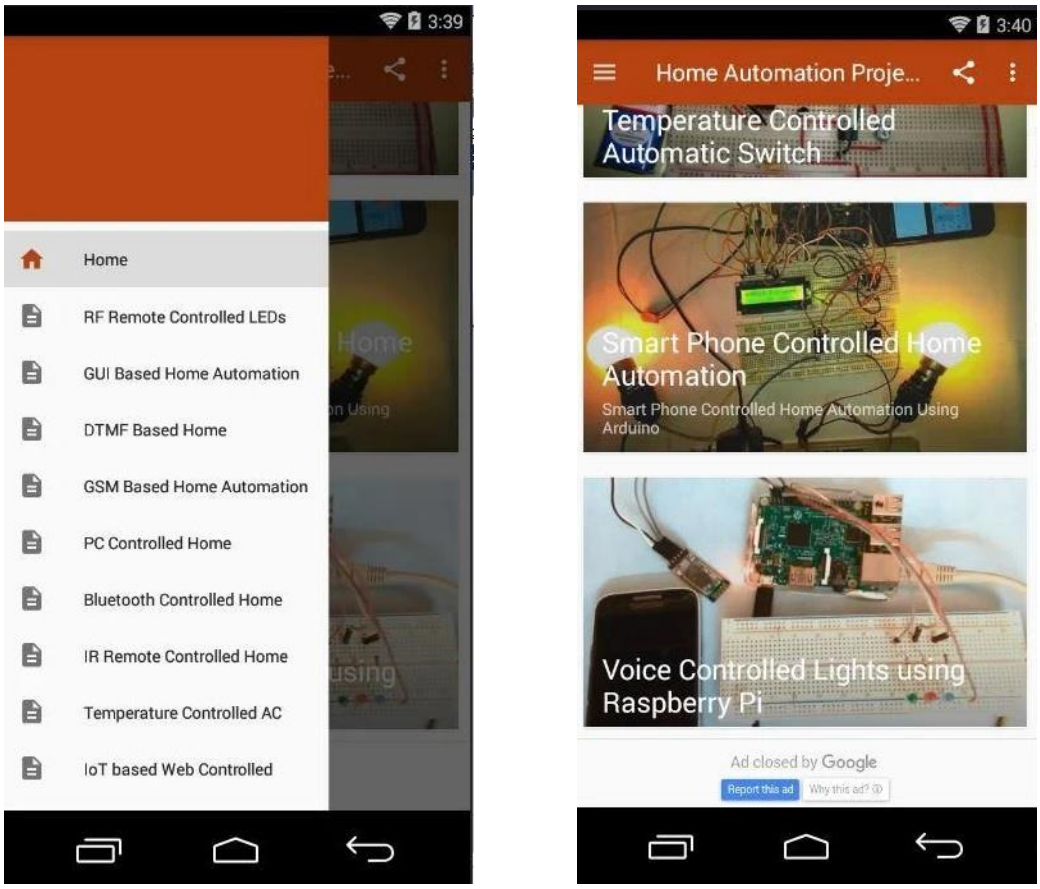


Figura 2 - Captures AutomationProjects

https://play.google.com/store/apps/details?id=com.andromo.dev614880.app773078

Aquesta aplicació és una mostra de vàries apps existents dins la temàtica de control remot mitjançant Arduino, però realment enfocades a mostrar com construir els dispositius amb vídeos i explicacions. No es tracta realment d'una

aplicació per controlar dispositius remots, sinó més be per ensenyar a construir aquests dispositius

ArduController

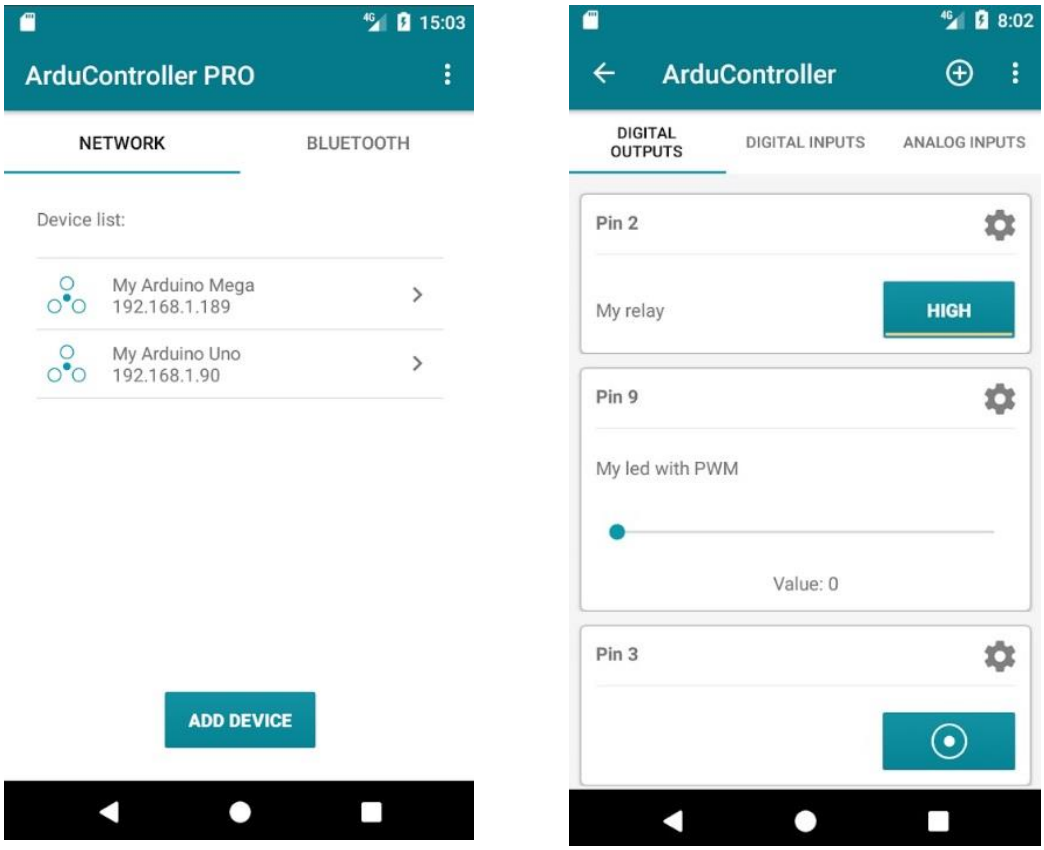


Figura 3 - Captures ArduController

https://play.google.com/store/apps/details?id=it.Ettore.arducontroller
https://www.gallinaettore.com/android_apps/arducontroller/

Aquesta aplicació també permet controlar dispositius Arduino. Disposa de les seves pròpies llibreries Arduino per crear els programes. La principal diferència amb la nostra és que també es necessita un coneixement tècnic per configurar els dispositius a l'app. Es necessita saber a quin pin de la targeta Arduino estan connectades les entrades i sortides, i quin sensor s'utilitzarà. Comparat amb la nostra també té un nombre limitat d'opcions. No permet enviar comandes infraroges per controlar aparells. Tampoc permet la creació d'escenaris com a la nostra.

Arduino Home Automation

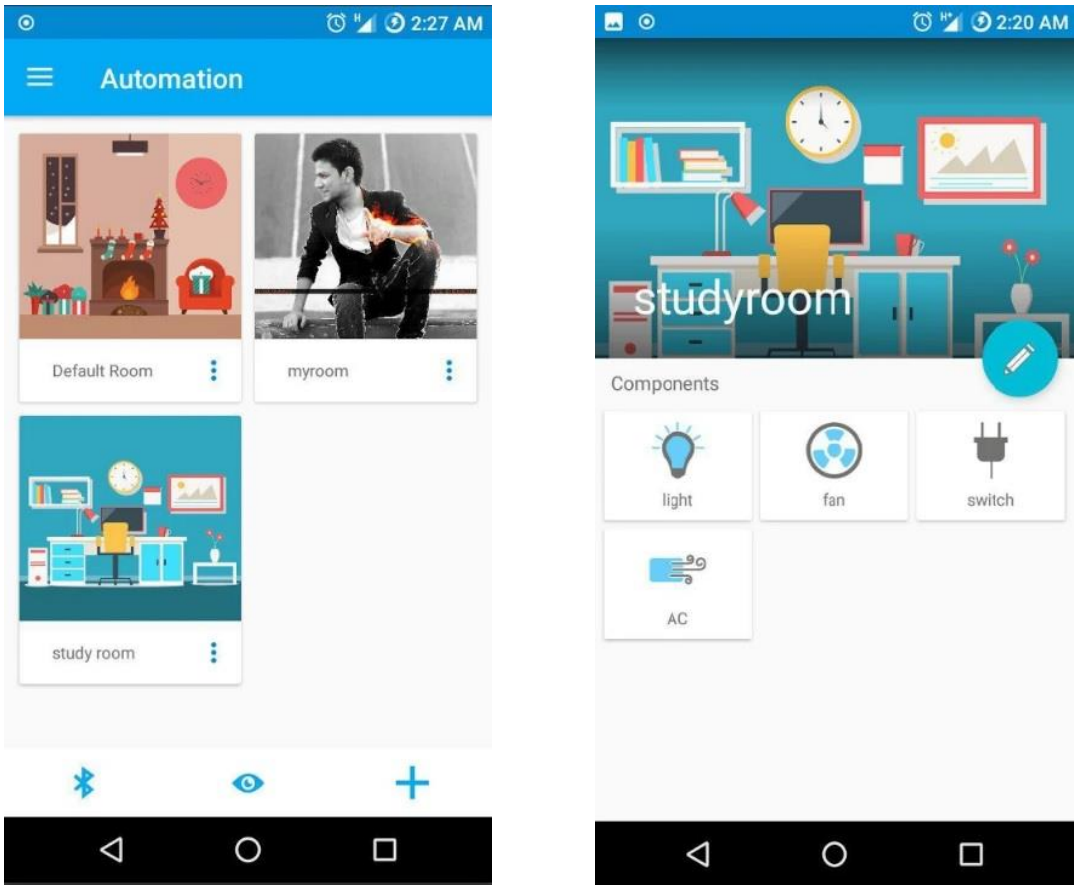


Figura 4 - Captures Arduino Home Automation

https://play.google.com/store/apps/details?id=com.darkbrothers.automation
http://unciarobotics.com/

Arduino Home Automation permet la creació de diferents escenaris i a dintre de cada escenari permet crear diferents dispositius. La principal desavantatge és que només es poden enviar comandes per Bluetooth, per tant es necessita tenir un receptor de Bluetooth. Per configurar els dispositius a l'app es necessita també un cert coneixement tècnic. Es tenen que configurar els codis que envia cada comanda.

Logitech Harmony

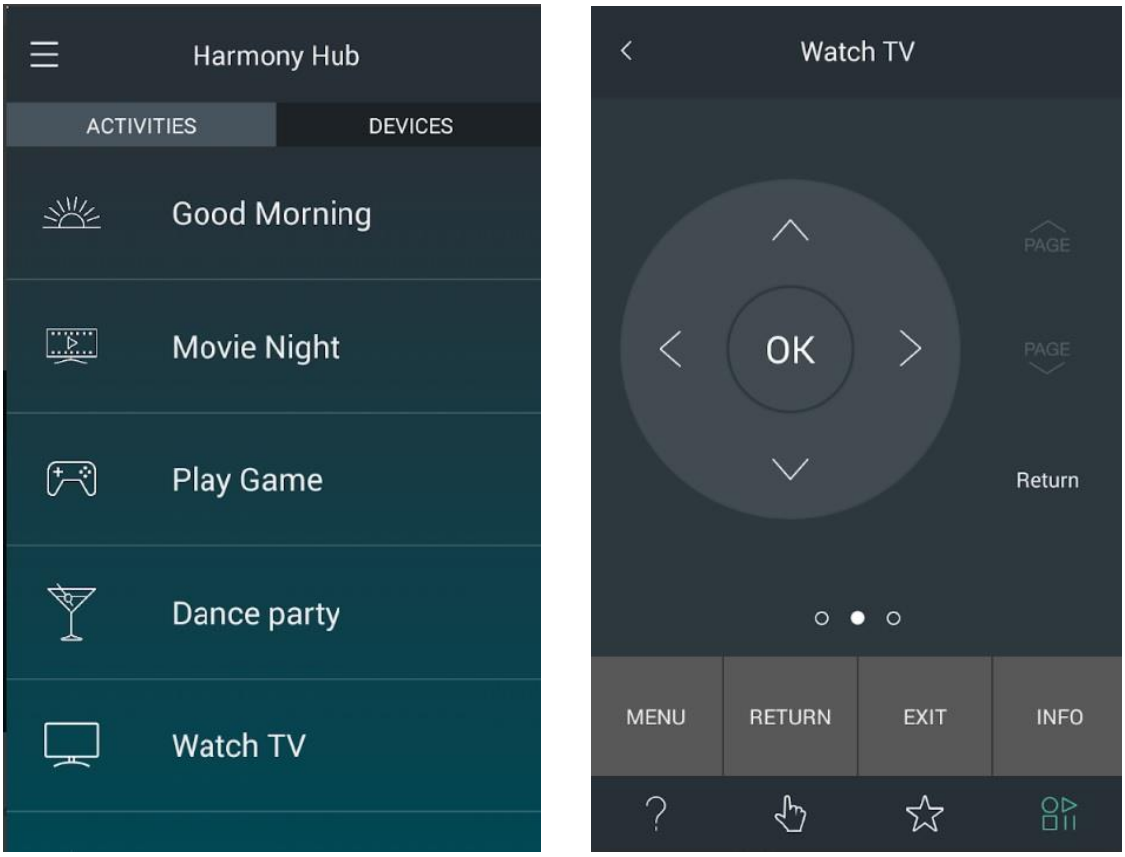


Figura 5 - Captures Logitech Harmony

https://play.google.com/store/apps/details?id=com.logitech.harmonyhub
https://www.myharmony.com/es-es/

És un exemple d'altres aplicacions que controlen aparells mitjançat el seu propi hardware. Aquesta és una de les més completes. Pot enviar comandaments per infrarojos, per Bluetooth i per RF. Disposa de una base de dades de una gran quantitat de marques amb els codis dels comandaments a distància. També té un comandament, a més de l'aplicació, que pot aprendre comandes directament dels comandaments a distància. L'aplicació envia les comandes per wifi a un aparell anomenat Hub, el qual és el que envia les comandes realment als diferents dispositius.

La principal desavantatge de sistemes com aquest és el preu del hardware propi. També necessita connexió a Internet per poder configurar-lo, ja que necessita connectar-se als seus propis servidors per baixar la configuració de cada marca.

1.2 Objectius del Treball

Es tracta d'una aplicació mòbil per Android que implementarà un sistema domòtic de control de dispositius a un habitatge. Interactuarà amb targetes Arduino o similars, enviarà ordres per activar o desactivar dispositius. En principi tindrà dos tipus de dispositius, un dispositiu simple que només rebrà ordres per activar o desactivar, encendre o apagar, com pot ser un llum o el motor d'una persiana. Un segon tipus de dispositiu per controlar equips d'aire condicionat. Aquests dispositius rebran ordres més complexes. En cada ordre se li envia a la targeta Arduino la temperatura, el mode de funcionament (fred o calor) i l'acció a realitzar, apagar, arrancar l'equip, canviar de temperatura. També es disposarà d'un sensor de temperatura, es mostrarà la temperatura de la habitació a l'aplicació Android.

En la part d'Arduino, que rebrà les comandes de l'aplicació Android, en els dispositius simples només es tracta d'activar o desactivar un o més relés, mentre que en els dispositius per control d'aire condicionat la targeta Arduino envia les comandes rebudes des de la app Android cap els equips amb leds infrarojos. A la targeta es guarden els codis de cada ordre disponible del comandament a distància de l'equip. És a dir que es simulen les ordres del comandament a distància. Per capturar els codis del comandament a distància es disposarà d'un altre dispositiu Arduino amb un led receptor d'infrarojos. Aquest sistema es podria utilitzar amb qualssevol aparell que funcioni amb comandament a distància, com per exemple un televisor, equip de música, etc. En principi s'utilitzarà per controlar els equips d'aire condicionat.

L'aplicació estaria pensada per utilitzar-se principalment a casa, es comunicaria per wifi amb les targetes Arduino. Però també tindrà la possibilitat de utilitzar-se des de fora de casa. Per exemple si es vol connectar l'aire condicionat abans d'arribar a casa. Per seguretat s'implementarà un servidor VPN en una targeta Raspberry per poder fer la connexió des de l'exterior.

La part de Arduino pel control dels dispositius ja està implementada en part. Es tractaria de crear l'aplicació Android que controla tots els dispositius Arduino.

Requisits funcionals

- **RF1**. L'usuari crea diferents escenaris. Cada escenari té un nom i contindrà un grup de dispositius. Un escenari es correspon amb una part de l'habitatge. Pot ser una habitació, la cuina, etc.
- **RF2**. L'usuari crea dispositius de tipus interruptor dins un escenari determinat. Cada dispositiu de tipus interruptor pot controlar entre 1 i 8 aparells. Cada dispositiu té una IP pròpia.
- **RF3**. L'usuari crea dispositius de tipus aire condicionat dins un escenari determinat. Cada dispositiu controla un aparell d'aire condicionat i té una IP pròpia.
- **RF4**. L'usuari rep informació de la temperatura ambient al lloc a on està l'aparell d'aire condicionat.

- **RF5**. L'usuari pot modificar o esborrar els escenaris. Al esborrar un escenari també s'esborren els dispositius que contenia.
- **RF6**. L'usuari pot modificar o esborrar els dispositius.
- **RF7**. Cada dispositiu permet enviar comandes per controlar els aparells
- **RF8**. Els dispositius de tipus aire condicionat mostren la temperatura ambient de l'habitació

Requisits no funcionals

- **RNF1**. No es necessiten coneixements tècnics per configurar l'aplicació. Només es necessita saber la IP de cada dispositiu.
- **RNF2**. El funcionament serà simple de manera que la pugui emprar un gran nombre d'usuaris de tot tipus i edat.
- **RNF3**. Queda oberta la possibilitat d'anar ampliant en un futur i afegir altres tipus de dispositius diferents.
- **RNF4**. No es necessitarà connexió a internet per poder fer-la servir des de casa.

1.3 Enfocament i mètode seguit

Molts dels sistemes ja existents al mercat ja tenen el seu hardware propi, els aparells del fabricant. Aquí es tracta de crear els nostres propis aparells, mitjançant Arduino o targetes de desenvolupament semblants. La qual cosa ens permet anar més enllà de les limitacions imposades al utilitzar els aparells d'un fabricant determinat.

Per tant es tractaria de desenvolupar un producte nou. Encara que semblant a sistemes comercials ja existents, la estratègia seria desenvolupar com si fos un producte nou. Això també obre la possibilitat a anar creixent i oferir prestacions que no ofereixen els sistemes ja existents.

He decidit desenvolupar l'aplicació mòbil en Android, amb el llenguatge Java. La base de dades que s'utilitzarà serà SQLite. Els dispositius es controlaran mitjançant targetes Arduino o semblants.

La implementació es farà amb l'IDE Android Studio ja que és l'IDE oficial de Google. La majoria d'informació a la xarxa i els darrers tutorials de Google per Android són per aquest IDE.

Android [1]

Es tracta d'un conjunt de programari per a dispositius mòbils que inclou sistema operatiu, programari intermediari i aplicacions, comprat per Google l'any 2005. El codi font d'Android s'anomena Android Open Source Project (AOSP) el qual va ser llençat inicialment amb llicència Apache. Això ha permès treure diferents

variants d'Android utilitzades en altres dispositius electrònics com consoles de videojocs, càmeres digitals, PCs, etc.

Android és el Sistema Operatiu més venut en mòbils des del 2011 i en tauletes des del 2013. El maig de 2017 tenia 2 bilions d'usuaris actius. El gener de 2020 Google Play Store tenia uns 2.9 bilions d'aplicacions.

Em vaig decidir per Android ja que trobo que és un sistema prou madur avui en dia, amb un gran nombre d'usuaris, molt bona documentació i suport. El sistema més utilitzat en dispositius mòbils.

També té una gran comunitat de desenvolupadors. Una gran nombre de llibreries. A més el codi font està publicat avui en dia amb Llicència Lliure.

Des de l'any 2017 es poden desenvolupar aplicacions Android en llenguatge Kotlin, a més de Java. He decidit utilitzar el llenguatge Java per desenvolupar l'aplicació ja que és el que més conec i el que he utilitzat durant els estudis del Grau.

SQLite

Es tracta de una base de dades de llicència lliure, lleuger, autònom, sense servidor, que guarda les dades a un fitxer de text en el mateix dispositiu [2].

He elegit aquesta base de dades perquè en la majoria de casos els usuaris no necessitaran crear un gran nombre d'escenes i dispositius. No es necessitarà desar un gran nombre de dades. Per tant aquesta base de dades és suficient per l'aplicació. A més de la simplicitat i facilitat de ús que aporta, sense la necessitat d'un servidor exterior.

Arduino [3]

Arduino és una plataforma d'electrònica amb llicència lliure, basada amb hardware i software pensats per ser fàcils d'utilitzar. Amb una gran comunitat al darrera. Principalment es tracta de targetes electròniques amb un microcontrolador i diferents entrades i sortides. Disposa de moltes llibreries per facilitar el desenvolupament.

He decidit utilitzar Arduino per controlar els diferents aparells degut a que es tracta d'un projecte de llicència lliure, amb una gran comunitat, amb molta informació, moltes llibreries, i per la versatilitat. Pot ser utilitzat en projectes molt diversos.

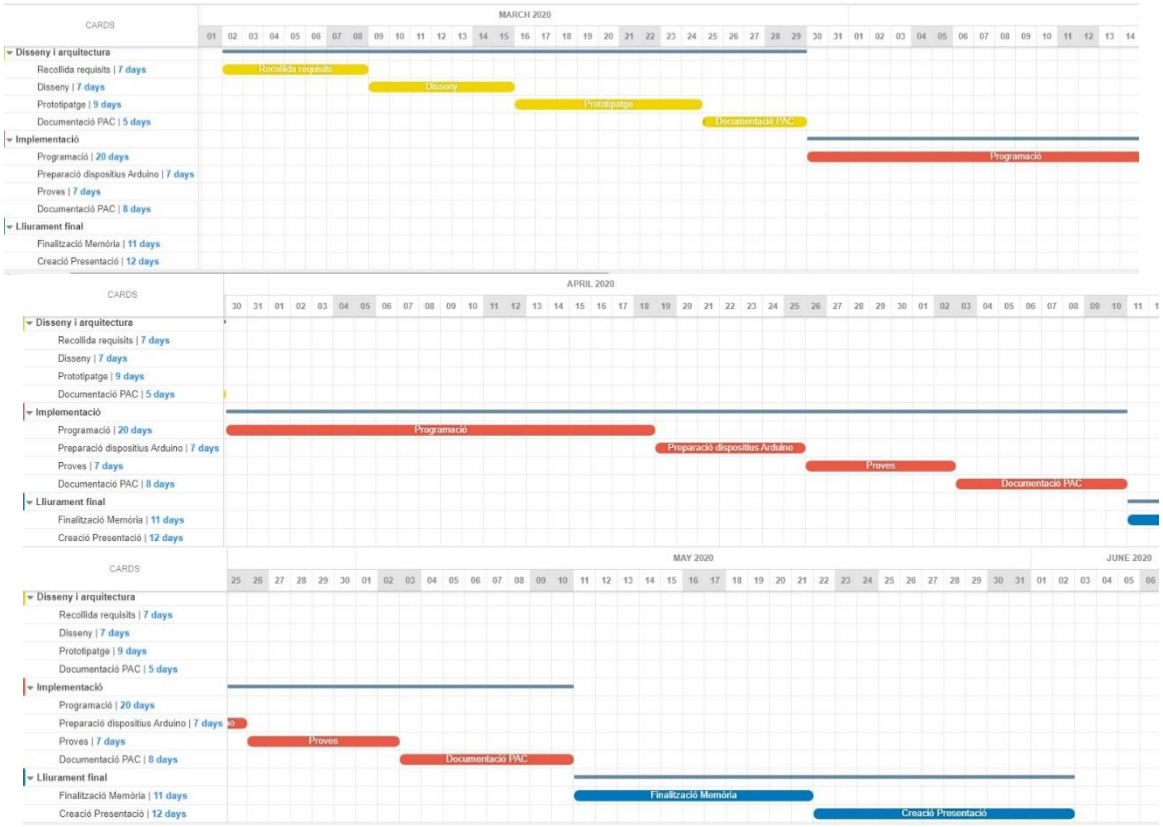
També existeix la possibilitat d'emprar targetes NodeMCU amb avantatges sobre Arduino. Ja tenen wifi integrat, tamany més petit i millors prestacions.

1.4 Planificació del Treball

De dilluns a divendres dedicaria 4 dies, un mínim de 2 hores. Els caps de setmana dedicaria entre 2 i 8 hores. Aquesta planificació és un mínim, en cas de sorgir problemes hi podria dedicar més temps.

Per fer la planificació utilitzo Trello (trello.com). He creat les llistes To Do, In Progress, Testing i Done. També una targeta per cada tasca. Mitjançant Elegantt, des del mateix Trello, he creat el diagrama de Gantt.

Adjunto imatges del diagrama:



Com que el diagrama de Elegantt no mostra les hores, a continuació es mostra una taula amb els dies i les hores per cada tasca.

Tasca	Dies	Hores
Disseny i Arquitectura		
Recollida requisits	7	10
Disseny	7	10
Prototipatge	9	14
Documentació PAC	5	8
Implementació		
Programació	20	28
Preparació dispositius Arduino	7	10
Proves	7	10

Documentació PAC	8	12
Lliurament Final		
Finalització Memòria	11	16
Creació Presentació	12	18

1.5 Breu sumari de productes obtinguts

A part de la memòria i la presentació, el lliurament final inclourà l'app anomenada Dobile, en Android, amb el seu codi. També el codi utilitzat a les targetes Arduino.

1.6 Breu descripció dels altres capítols de la memòria

Els altres capítols de la memòria seran:

- **Disseny i Arquitectura**. Recollida de requisits, disseny tècnic, disseny de la interfície gràfica i creació d'un prototip
- **Implementació.** Programació i proves de l'aplicació
- **Arduino**. Explicació dels dispositius utilitzats
- **Servidor VPN**. Explicació de la implementació d'un servidor VPN en una Raspberry per la connexió des de l'exterior de la casa.

2. Disseny DCU

Aplicant el disseny centrat en l'usuari es segueixen les diferents fases per la recollida de requisits i anàlisi dels possibles usuaris de l'aplicació, així com el context d'us

2.1 Usuaris i Context d'us

2.1.1 Indagació

Mètodes d'indagació
Descarto el mètode de shadowing perquè l'aplicació s'utilitzarà principalment a casa, en el dia a dia. Considero que és un mètode massa intrusiu, i a més penso que la presència de l'observador pot influir en alguns usuaris.

També descarto el logging ja que es tracta de un sistema encara no existent, no ens serveix en el nostre cas.

Tampoc s'utilitzarà l'anàlisi competitiva perquè es tracta d'analitzar aplicacions ja existents. Considero que podria desviar-nos de la intenció que tenim en aquesta fase, que és recollir les necessitats dels usuaris.

La indagació es farà amb el mètode de l'entrevista per recollir possibles necessitats, com volen els usuaris utilitzar una aplicació d'aquest tipus. Quins aparells volen controlar, etc

Es farà un qüestionari a una mostra de possibles usuaris de diferents tipus. El qüestionari serà presencial. No és un qüestionari tancat, està obert a noves preguntes i qüestions.

Questionari
1. **Si tingués un sistema de domòtica a casa, quins aparells voldria controlar des del mòbil?**

2. **Quina informació voldria que es mostrés en pantalla de la casa?**

3. **Des de on voldria controlar els aparells? (Des de casa, feina, vacances)**

4. **Com voldria que fos l'aplicació? (Senzilla (només un control bàsic), molt completa (moltes possibilitat de control)**

Resultats qüestionari
El qüestionari s'ha realitzat a una mostra de 6 persones de característiques molt variades, diferents edats, sexes, diferent nivells d'estudi.

Respecta a la pregunta 1, tots coincideixen en que voldrien apagar i encendre llums de la casa, ventiladors i persianes si tinguessin cases domòtiques. 5 dels enquestats voldrien poder controlar la temperatura de la calefacció o aire condicionat. Dos dels enquestats també voldrien poder controlar la televisió i música des de la mateixa app. Un dels enquestats considera que no és necessari controlar la música perquè ja existeixen apps dedicades a aquesta funció molt completes.

En la pregunta 2 tots coincideixen en que volen veure la temperatura i humitat de la casa.

En la pregunta 3 tots volen controlar els aparells des de casa. Dos dels enquestats treballen a torns i voldrien poder arrancar la calefacció o aire condicionat des del treball per poder trobar la temperatura adequada al arribar a casa. No els hi serveixen els programadors ja que tenen un horari de treball molt irregular. Dos dels enquestats comenten que seria útil encendre i apagar els llums mentre estan de vacances a la segona residència per donar la impressió que la seva casa està habitada i dissuadir els possibles lladres.

A la pregunta 4 ningú té cap interès especial. En general voldria que fos una aplicació senzilla, semblant a un comandament a distància.

Usuaris

L'aplicació pretén realitzar unes tasques molt senzilles, que tothom fa a casa diàriament. Per tant no hi ha cap limitació en quant al tipus d'usuari. El perfil de possibles usuaris és molt obert. Qualsevol persona, de qualsevol edat, sexe, nivell d'estudis, nivell de coneixement tècnic podria ser un usuari.

Un sol perfil és suficient per englobar totes les característiques dels possibles usuaris:

Perfil usuari	
Edat	Mínim de 8 anys
Sexe	Qualssevol
Estudis	Qualssevol
Coneixement tècnic	Cap
Context d'ús	Principalment a casa. Ocasionalment fora de casa, en qualssevol lloc, qualssevol moment.

Anàlisi de tasques

La llista de tasques que necessitaran els usuaris és:

- Crear un nou escenari
- Editar, esborrar un escenari
- Crear un nou dispositiu
- Editar, esborrar un dispositiu

- Mostrar temperatura dels escenaris amb aire condicionat o calefacció
- Encendre o apagar llums, ventiladors, persianes, motors en general
- Encendre o apagar aire condicionat o calefacció
- Modificar temperatura o mode funcionament de l'aire condicionat

Conclusions indagació

Després de la indagació es conclou que el possible usuari té un perfil molt ample. Per tant no es necessitaran coneixements tècnics per utilitzar l'aplicació i el seu funcionament haurà de ser molt simple. Semblant a comandaments a distància als quals la majoria d'usuaris ja utilitzen en la seva vida diària.

Les característiques són molt simples, només es pretén controlar interruptors i altres aparells que normalment es controlen amb comandaments a distància. Es tractaria de simular aquests comandaments a distància. Els control d'aire condicionat o calefacció mostraran la temperatura real en pantalla.

2.2 Disseny Conceptual

Escenaris d'us

Escenari 1	
Descripció	L'usuari arriba a casa i necessita encendre els llums del saló
Perfil usuari	Qualssevol
Context	A casa, qualssevol moment
Objectius	Encendre els llums de casa
Tasques	Activar un o més interruptors
Necessitats informació	Saber quin interruptor correspon a cada llum
Funcionalitats	Control llums de la casa
Desenvolupament tasques	Selecciona l'escenari que correspon a l'estància de la casa a on es troba i després seleccionar la làmpara que vol encendre

Escenari 2	
Descripció	L'usuari es desplaça a l'habitació i vol encendre els llums
Perfil usuari	Qualssevol
Context	A casa, qualssevol moment
Objectius	Encendre els llums d'una habitació
Tasques	Activar un o més interruptors
Necessitats informació	Saber quin interruptor correspon a cada llum
Funcionalitats	Control llums de la casa

Desenvolupament tasques	Selecciona l'escenari que correspon a l'estància de la casa a on es troba i després seleccionar la làmpara que vol encendre

Escenari 3	
Descripció	L'usuari surt d'una habitació i necessita apagar el llum
Perfil usuari	Qualssevol
Context	A casa, qualssevol moment
Objectius	Apagar els llums d'una habitació
Tasques	Desactivar un o més interruptors
Necessitats informació	Saber quin interruptor correspon a cada llum
Funcionalitats	Control llums de la casa
Desenvolupament tasques	Selecciona l'escenari que correspon a l'estància de la casa a on es troba i després la làmpara que vol apagar

Escenari 4	
Descripció	L'usuari surt de casa i necessita apagar tots els llums de la casa
Perfil usuari	Qualssevol
Context	A casa, qualssevol moment
Objectius	Apagar els llums de casa
Tasques	Apagar tots els llums encesos
Necessitats informació	Cap
Funcionalitats	Control llums de la casa
Desenvolupament tasques	Apaga tots els llums encesos

Escenari 5	
Descripció	L'usuari arriba a casa i vol arrancar l'aire condicionat del saló
Perfil usuari	Qualssevol
Context	A casa, qualssevol moment
Objectius	Aconseguir la temperatura de confort a casa
Tasques	Arrancar l'aire condicionat
Necessitats informació	Saber la temperatura ambient de l'estància de la casa
Funcionalitats	Control aire condicionat de la casa

Desenvolupament tasques	Seleccionar l'estància de la casa que vol i seleccionar l'aparell que vol arrancar

Escenari 6	
Descripció	L'usuari surt de la feina i vol arrancar l'aire condicionat del saló
Perfil usuari	Qualssevol
Context	Fora de casa, a qualssevol lloc i en qualssevol moment
Objectius	Aconseguir la temperatura de confort quan arribi a casa
Tasques	Arrancar l'aire condicionat
Necessitats informació	Saber la temperatura ambient de l'estància de la casa
Funcionalitats	Control aire condicionat de la casa
Desenvolupament tasques	Seleccionar l'estància de la casa que vol i seleccionar l'aparell que vol arrancar

Escenari 7	
Descripció	L'usuari surt de casa i vol apagar tots els aparells d'aire condicionat en marxa
Perfil usuari	Qualssevol
Context	A casa, qualssevol moment
Objectius	Aturar tots els equips d'aire condicionat en marxa
Tasques	Aturar l'aire condicionat
Necessitats informació	Cap
Funcionalitats	Control aire condicionat de la casa
Desenvolupament tasques	Aturar tots el aparells d'aire condionat

Fluxos d'interacció

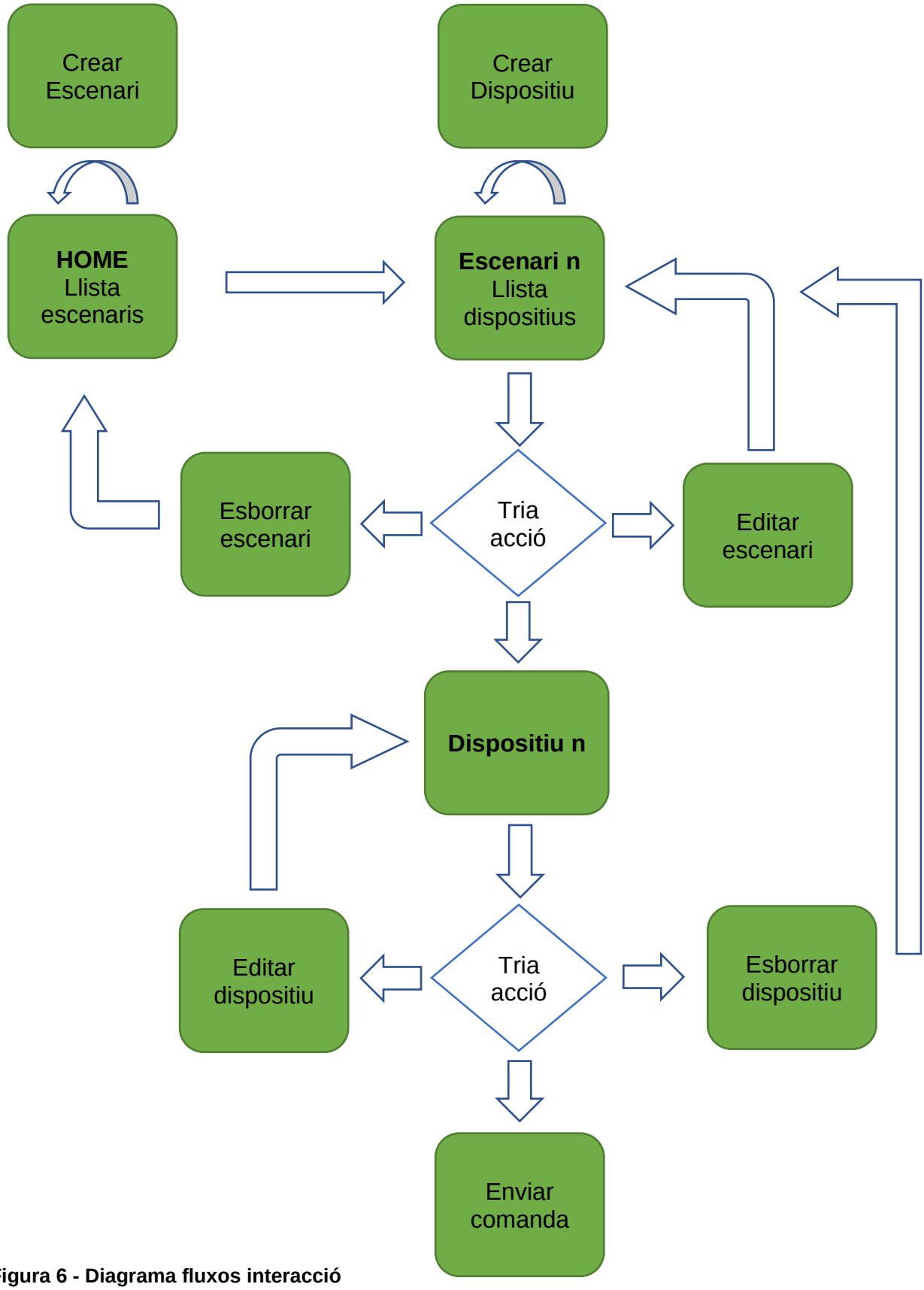


Figura 6 - Diagrama fluxos interacció

2.3 Prototipatge

Per crear el prototipus primer he realitzat un sketch a ma alçada de les pantalles.

Sketch inicial

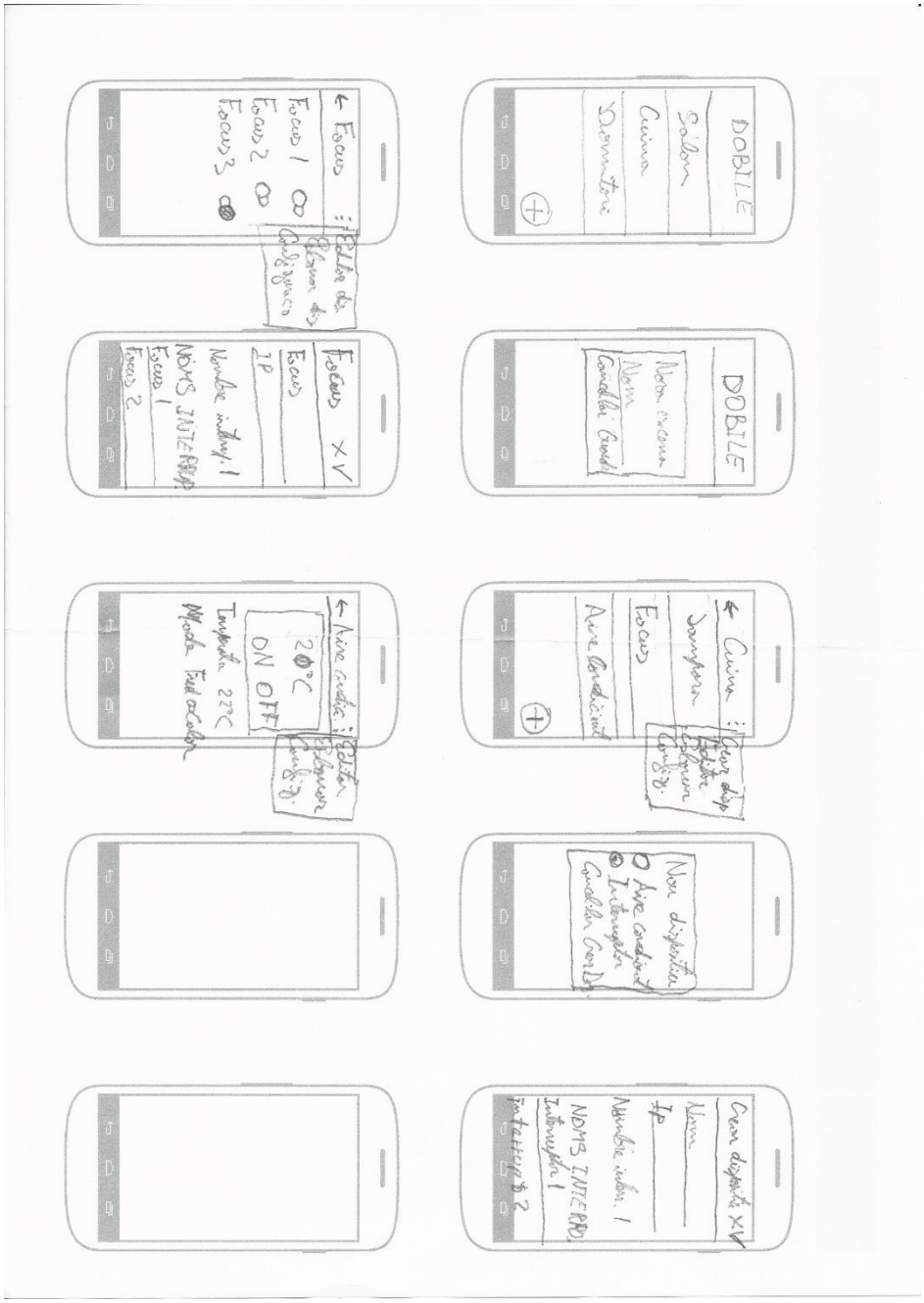


Figura 7 - Sketch inicial

Prototipus d'alta fidelitat

El prototipus d'alta fidelitat està creat amb Figma. He utilitzat el tema Dark Theme de Material Design [9] per intentar crear una semblança amb el típic comandament a distància que tothom coneix. Per que l'usuari trobi una certa familiaritat amb un aparell real que ja coneix.

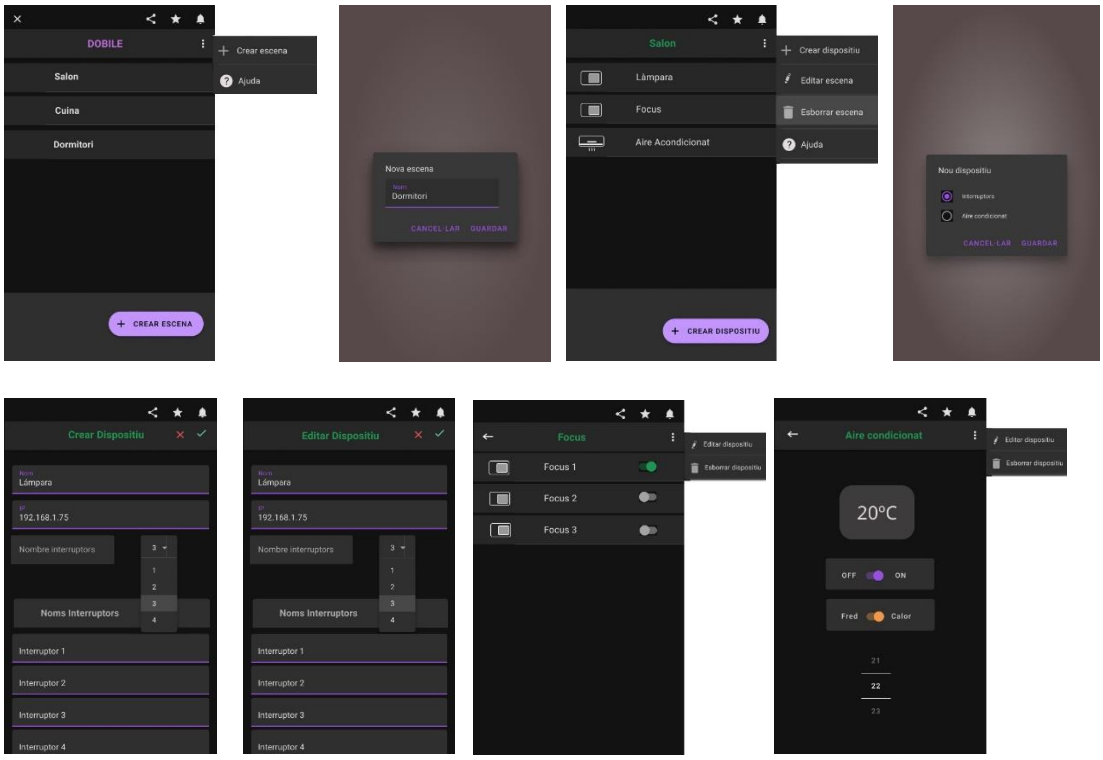


Figura 8 - Prototipus alta fidelitat

2.4 Avaluació

Es realitzarà l'avaluació del prototipus mitjançant la tècnica del test amb usuaris. Primer se'ls hi lliurarà un qüestionari per recollir les dades personals de cada usuari com edat, sexe, estudis, us que fa del mòbil, nivell de coneixement tècnic.

Qüestionari informació usuari

1. Edat
2. Sexe
3. Estudis
4. Professió
5. Utilitza mòbil habitualment?
6. Si és així, quant de temps al dia utilitza el mòbil?

7. Quin tipus d'aplicacions sol utilitzar amb més freqüència (jocs, navegació web, xarxes socials, etc)?
8. En quant al ús que fa del mòbil, quin tipus d'usuari es considera (bàsic, mitjà, avançat)?
9. A part del mòbil, té PC a casa? Si és així quin us en sol fer habitualment (jocs, navegació web, altres)?

Per fer l'avaluació es demanarà als usuaris realitzar una sèrie de tasques amb el prototipus per comprovar el seu funcionament i la seva usabilitat

Tasques a realitzar per part dels usuaris

- Crear un mínim de 4 escenaris
- Editar un escenari
- Esborrar dos escenaris
- Entrar en un escenari i crear un mínim de 3 dispositius de tipus interruptor, amb diferent nombre d'interruptors
- Entrar en un escenari i crear un mínim de 3 dispositius de tipus aire condicionat
- Entrar en varius dels escenaris i fer diferents modificacions. Canviar el nom, canviar la IP, canviar el nombre d'interruptors i els seus noms
- Esborrar 2 o 3 dispositius
- Entrar en un dispositiu de tipus interruptor i encendre o apagar vàries vegades les làmpares
- Entrar en un dispositiu de tipus aire condicionat i arrancar l'equip, modificar la temperatura, canviar el mode de funcionament a fred o calor, aturar-lo
- Retornar des de un dispositiu al seu escenari
- Retornar des de un escenari a la llista d'escenaris

Una vegada realitzades les tasques es passarà als usuaris un qüestionari per avaluar la seva experiència amb el prototipus

Qüestionari avaluació tasques

- Com avaluaria la seva experiència en la creació i edició d'escenaris (fluida, fàcil, complicat, etc)?
- Modificaria alguna cosa en aquestes tasques?
- Com avaluaria la seva experiència en la creació i edició de dispositius (fluida, fàcil, complicat, etc)?
- Modificaria alguna cosa en aquestes tasques?
- Com avaluaria el control d'interruptors (fluid, complicat, fàcil)?
- Com avaluaria el control d'aire condicionat (fluid, complicat, fàcil)?
- Modificaria alguna cosa en el control de dispositius?
- Com avaluaria la navegació per l'aplicació?
- Com avaluaria en general l'ús de l'aplicació?
- Modificaria o afegiria alguna cosa a l'aplicació

Amb els resultats dels qüestionaris es decidiria si es modifica alguna cosa del disseny, s'afegeix alguna funcionalitat nova, etc

3. Disseny tècnic

3.1 Casos d'us

Diagrama Casos d'ús

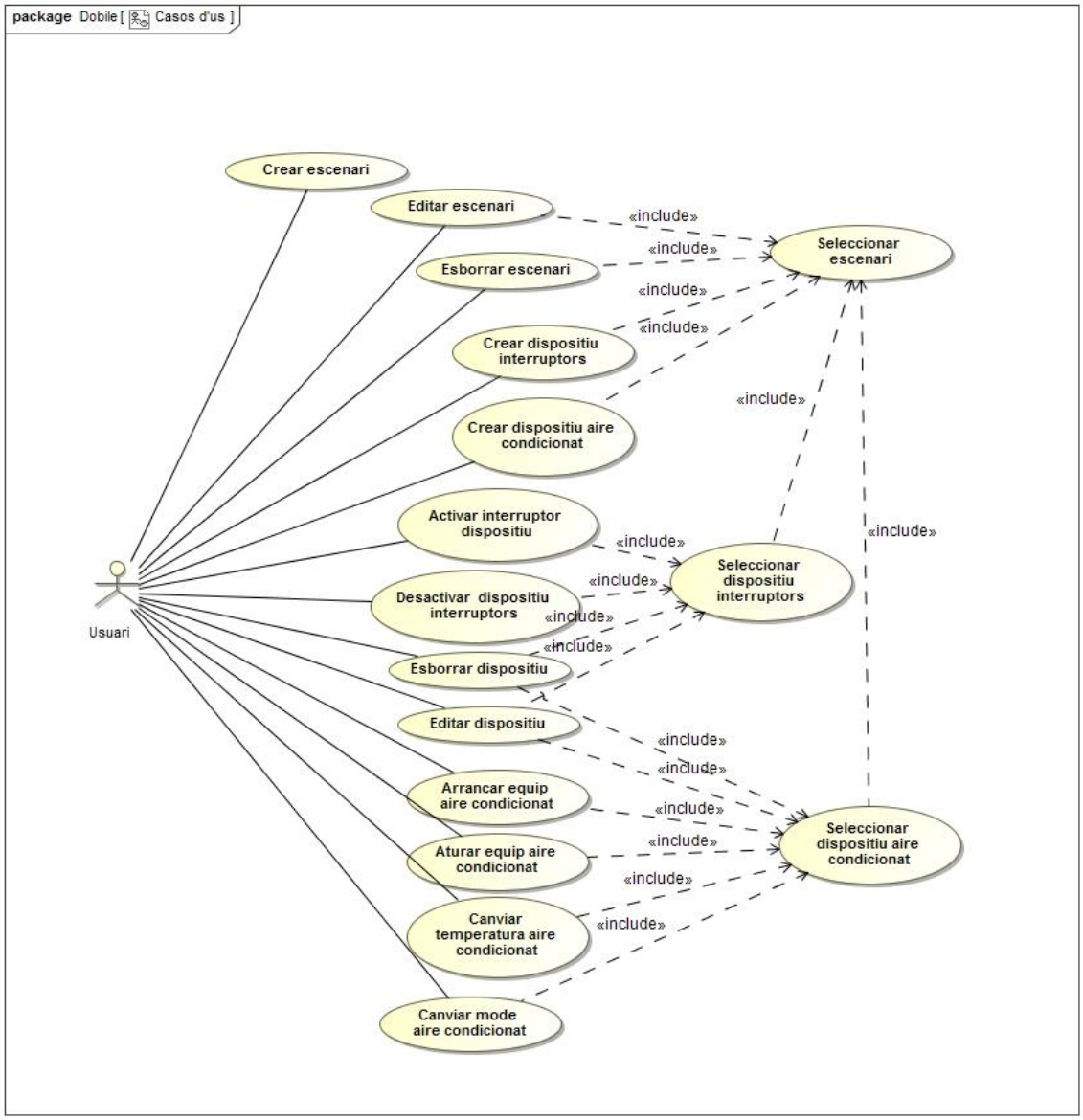


Figura 9 - Diagrama de Casos d'us

Casos d'ús

Identificador	CU-01
Nom	Crear escena
Prioritat	Mitja
Descripció	Crea nova escena
Actors	Usuari
Pre-Condicions	Cap
Iniciat per	Usuari
Flux	
Post-Condicions	S'ha creat una nova escena
Notes	

Identificador	CU-02
Nom	Seleccionar escena
Prioritat	Alta
Descripció	L'usuari selecciona una escena de la llista d'escenes
Actors	Usuari
Pre-Condicions	Cap
Iniciat per	Usuari
Flux	
Post-Condicions	Es mostra l'escena seleccionada
Notes	

Identificador	CU-03
Nom	Editar escena
Prioritat	Mitja
Descripció	Modificar el nom d'una escena
Actors	Usuari
Pre-Condicions	Existeix una escena a modificar
Iniciat per	Usuari
Flux	
Post-Condicions	S'ha modificat l'escena
Notes	

Identificador	CU-04
Nom	Esborrar escena
Prioritat	Mitja
Descripció	Esborrar una escena
Actors	Usuari

Pre-Condicions	Existeix una escena a modificar
Iniciat per	Usuari
Flux	
Post-Condicions	S'ha modificat l'escena
Notes	

Identificador	**CU-05**
Nom	Crear dispositiu aire condicionat
Prioritat	Mitja
Descripció	Crea un nou dispositiu de tipus aire condicionat en una determinada escena
Actors	Usuari
Pre-Condicions	L'usuari ha selecciona una escena a on es crearà el dispositiu
Iniciat per	Usuari
Flux	
Post-Condicions	S'ha creat un nou dispositiu de tipus aire condicionat a l'escena seleccionad
Notes	

Identificador	**CU-06**
Nom	Seleccionar dispositiu aire condicionat
Prioritat	Alta
Descripció	L'usuari selecciona un dispositiu de tipus aire condicionat de la llista de dispositius de l'escenari actual
Actors	Usuari
Pre-Condicions	L'usuari ha seleccionat un escenari
Iniciat per	Usuari
Flux	
Post-Condicions	Es mostra el dispositiu seleccionat
Notes	

Identificador	**CU-07**
Nom	Editar dispositiu aire condicionat
Prioritat	Mitja
Descripció	Modificar el nom i/o IP del dispositiu seleccionat
Actors	Usuari
Pre-Condicions	L'usuari ha seleccionat un dispositiu de tipus aire condicionat en una escena determinada
Iniciat per	Usuari
Flux	
Post-Condicions	S'han modificat les dades del dispositiu
Notes	

Identificador	CU-08
Nom	Esborrar dispositiu aire condicionat
Prioritat	Mitja
Descripció	S'esborrar un dispositiu de tipus aire condicionat
Actors	Usuari
Pre-Condicions	L'usuari ha seleccionat un dispositiu de tipus aire condicionat, en una escena determinada
Iniciat per	Usuari
Flux	
Post-Condicions	S'ha esborrat el dispositiu
Notes	

Identificador	CU-09
Nom	Crear dispositiu interruptors
Prioritat	Mitja
Descripció	Crea un nou dispositiu de tipus interruptors en una determinada escena
Actors	Usuari
Pre-Condicions	L'usuari ha selecciona una escena a on es crearà el dispositiu
Iniciat per	Usuari
Flux	
Post-Condicions	S'ha creat un nou dispositiu de tipus interruptor a l'escena seleccionada
Notes	

Identificador	CU-10
Nom	Seleccionar dispositiu interruptors
Prioritat	Alta
Descripció	L'usuari selecciona un dispositiu de tipus interrutptor de la llista de dispositius de l'escenari actual
Actors	Usuari
Pre-Condicions	L'usuari ha seleccionat un escenari
Iniciat per	Usuari
Flux	
Post-Condicions	Es mostra el dispositiu seleccionat
Notes	

Identificador	CU-11
Nom	Editar dispositiu interruptors
Prioritat	Mitja
Descripció	Modificar el nom, número d'interruptors i/o noms d'interruptors del dispositiu seleccionat
Actors	Usuari

Pre-Condicions	L'usuari ha seleccionat un dispositiu de tipus interruptors en una escena determinada
Iniciat per	Usuari
Flux	
Post-Condicions	S'han modificat les dades del dispositiu
Notes	

Identificador	**CU-12**
Nom	Esborrar dispositiu interruptors
Prioritat	Mitja
Descripció	S'esborrar un dispositiu de tipus interruptor
Actors	Usuari
Pre-Condicions	L'usuari ha seleccionat un dispositiu de tipus interruptor, en una escena determinada
Iniciat per	Usuari
Flux	
Post-Condicions	S'ha esborrat el dispositiu
Notes	

Identificador	**CU-13**
Nom	Activar interruptor
Prioritat	Alta
Descripció	L'usuari activa un interruptor per encendre un llum, arrancar un motor, etc
Actors	Usuari
Pre-Condicions	L'usuari ha seleccionat un dispositiu de tipus interruptor, en una escena determinada
Iniciat per	Usuari
Flux	
Post-Condicions	S'ha encés el llumo o s'activat el motor de l'aparell controlat
Notes	

Identificador	**CU-14**
Nom	Desactivar interruptor
Prioritat	Alta
Descripció	L'usuari desactiva un interruptor per apagar un llum, aturar un motor, etc
Actors	Usuari
Pre-Condicions	L'usuari ha seleccionat un dispositiu de tipus interruptor, en una escena determinada
Iniciat per	Usuari
Flux	
Post-Condicions	S'ha apagat el llumo o aturat el motor de l'aparell controlat

Notes	

Identificador	**CU-15**
Nom	Activar aire condicionat
Prioritat	Alta
Descripció	L'usuari arranca un aparell d'aire condicionat
Actors	Usuari
Pre-Condicions	L'usuari ha seleccionat un dispositiu de tipus aire condicionat. L'aparell estaba aturat
Iniciat per	Usuari
Flux	
Post-Condicions	Ha arrancat l'aparell d'aire condicionat controlat
Notes	

Identificador	**CU-16**
Nom	Desactivar aire condicionat
Prioritat	Alta
Descripció	L'usuari atura un aparell d'aire condicionat
Actors	Usuari
Pre-Condicions	L'usuari ha seleccionat un dispositiu de tipus aire condicionat. L'aparell estaba en funcionament
Iniciat per	Usuari
Flux	
Post-Condicions	Ha aturat l'aparell d'aire condicionat controlat
Notes	

Identificador	**CU-17**
Nom	Modificar temperatura
Prioritat	Alta
Descripció	L'usuari baixa o puja la temperatura a l'aparell d'aire condicionat
Actors	Usuari
Pre-Condicions	L'usuari ha seleccionat un dispositiu de tipus aire condicionat. L'aparell està en funcionament
Iniciat per	Usuari
Flux	
Post-Condicions	L'aparell d'aire condicionat ha rebut la comanda per modificar la temperatura
Notes	

Identificador	**CU-18**
Nom	Modificar mode aire condicionat (fred/calor)
Prioritat	Alta
Descripció	L'usuari el mode de funcionament de l'aparell d'aire condicionat (mode fred o mode calor)
Actors	Usuari

Pre-Condicions	L'usuari ha seleccionat un dispositiu de tipus aire condicionat. L'aparell està en funcionament
Iniciat per	Usuari
Flux	
Post-Condicions	L'aparell d'aire condicionat ha rebut la comanda per modificar el mode de funcionament
Notes	

Identificador	**CU-18**
Nom	Seleccionar dispositiu aire condicionat
Prioritat	Alta
Descripció	L'usuari selecciona un dispositiu de tipus aire condicionat de la llista de dispositius de l'escenari actual
Actors	Usuari
Pre-Condicions	L'usuari ha seleccionat un escenari
Iniciat per	Usuari
Flux	
Post-Condicions	Es mostra el dispositiu seleccionat
Notes	

3.2 Disseny base de dades

La base de dades conté una taula per les escenes, una taula pels dispositius de tipus interruptor, una taula pels dispositius de tipus aire condicionat i una taula pels noms d'interruptors.

Els dispositius interruptors, a més del nom i la IP, guarden el número de interruptors que controla cada dispositiu.

Els dispositius de tipus aire condicionat guarden també la darrera temperatura, el darrer mode de funcionament seleccionat i l'estat de l'equip (On/Off). Per evitar

que es perdi l'estat i les darreres comandes enviades quan es torna enrere o es surt de l'app

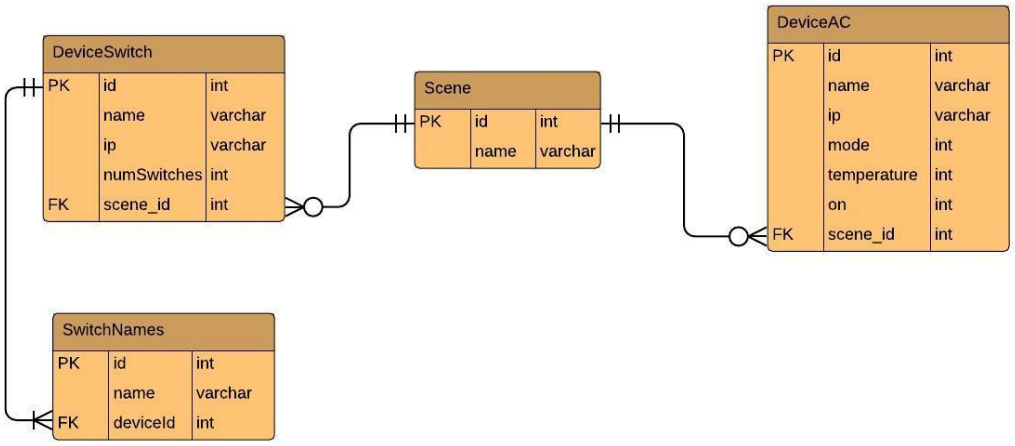


Figura 10 - Diagrama ER base de dades

3.3 Arquitectura MVVM

Seguint les recomanacions de Google a la seva Guia d'arquitectura d'aplicacions [4], s'implementarà amb el patrò MVVM (Model View ViewModel). La capa de View conté les Activities que formen la interfície gràfica. La capa de DataModel conté les operacions amb la base dades. Mentre que la capa ViewModel connecta les dades amb la vista.

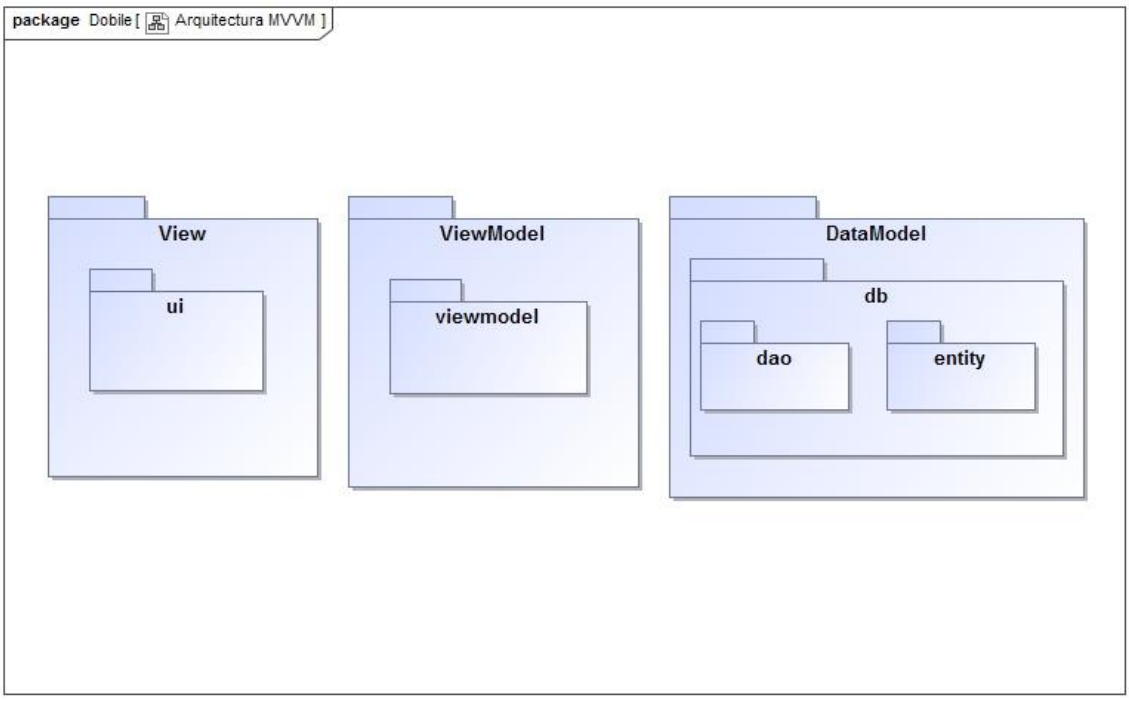


Figura 11 - Arquitectura MVVM

3.4 Diagrames de classes

Paquets ui i viewmodel

Les activities MainActivity i SceneActivity utilitzen ListAdapters per mostrar les llistes d'escenes i dispositius respectivamente, mitjançant llistes LiveData. Aquest sistema detecta canvis a la base de dades i actualitza automàticament la vista, sense que interfereixin els canvis d'estat de les activities.

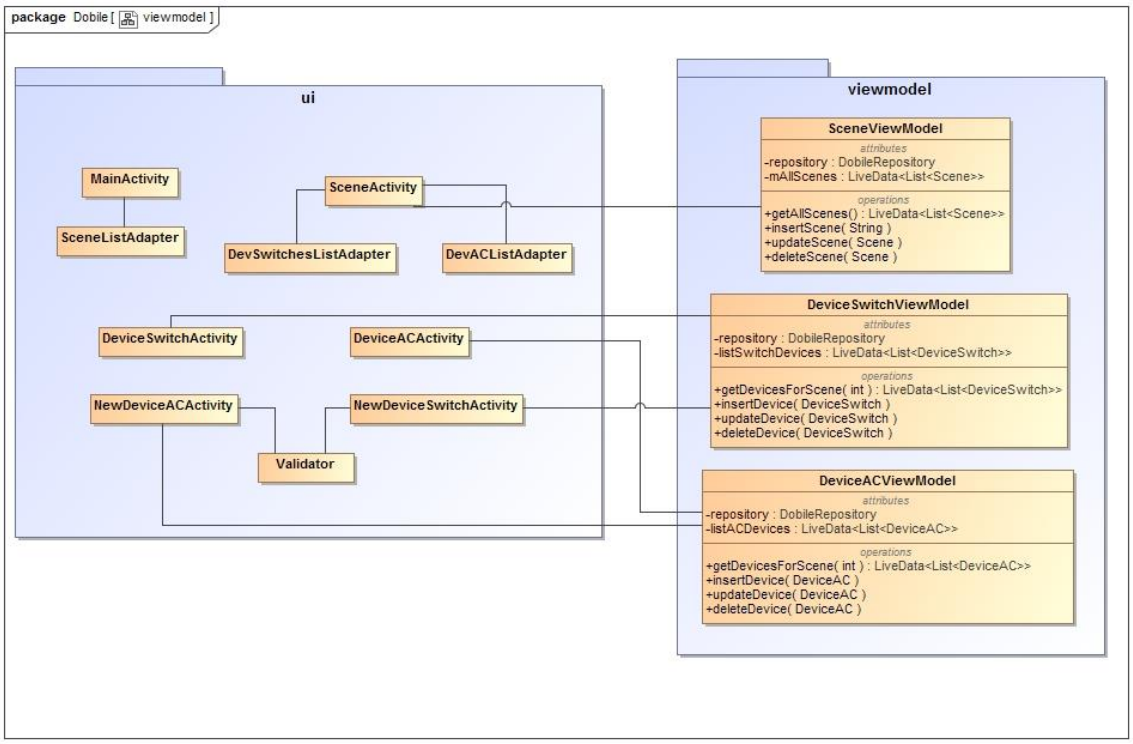


Figura 12 - Diagrama de classes ui i viewmodel

Paquet db

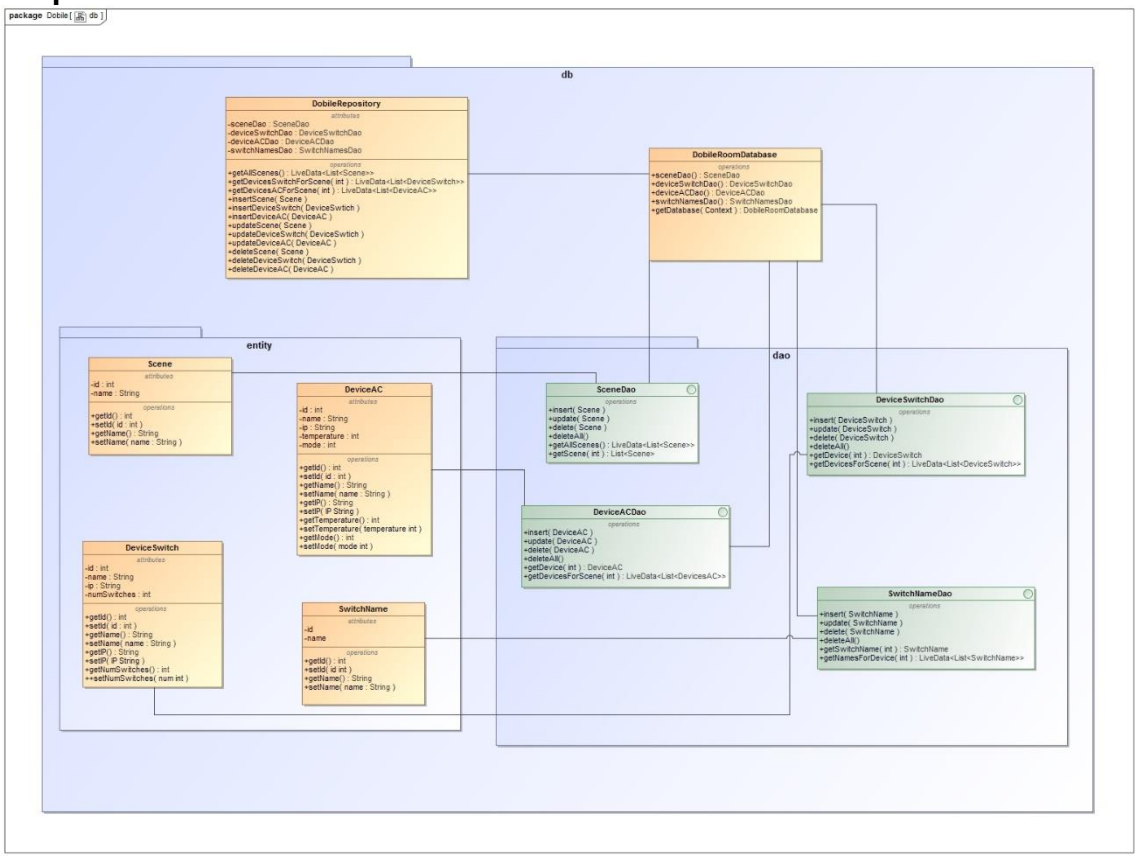


Figura 13 - Diagrama de classes – db

Android Test [5] [6]

També es disposarà de tests per comprovar les operacions amb la base de dades. Es comproven les operacions de les classes Dao. Conté una clase test per cada classe Dao. DaoTestSuite executa tots els tests alhora.

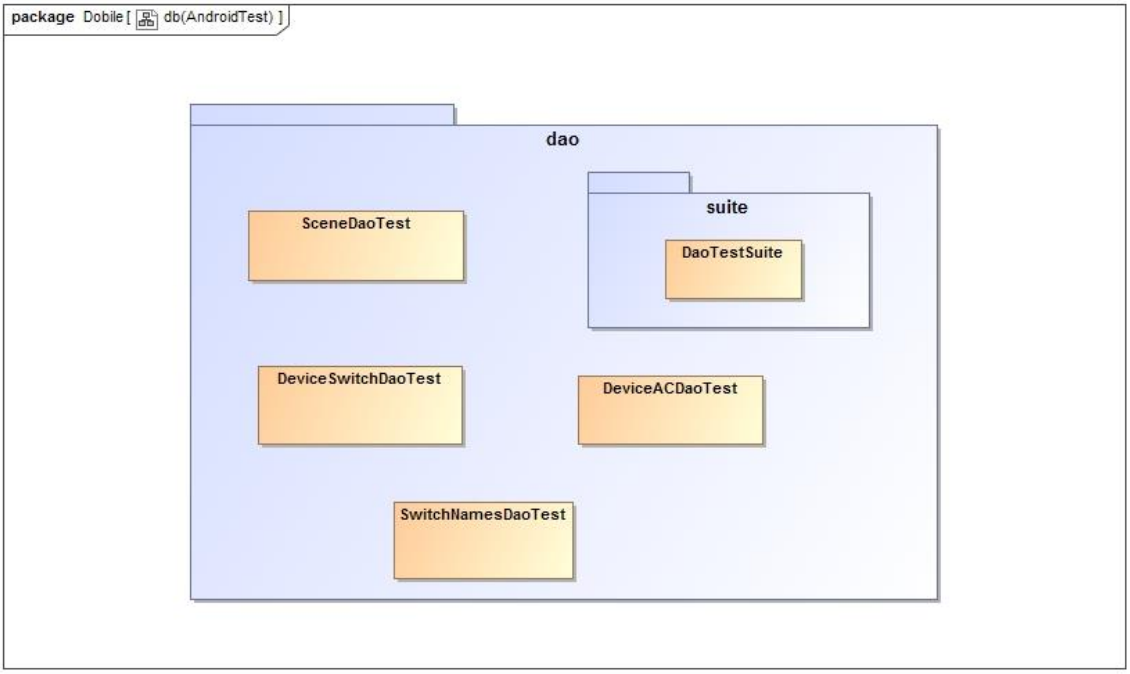


Figura 14 - Diagrama de classes - Android test

4. Implementació

La versió mínima de SDK que he elegit és la 23, que correspon a Android 6.0. Aquesta API va sortir el 2015. Amb aquesta API ens assegurem compatibilitat amb mòbils de fins a 5 anys.

Inicialment vaig començar la implementació amb la biblioteca de compatibilitat estàndard, però al final vaig decidir migrar les llibreries corresponents a Androidx, com es recomana des d'Android. A partir d'Android 9.0 tot el desenvolupament de biblioteques noves es fa des d'Androidx. Les biblioteques anteriors de Room i els components lifecycle (LiveData, AndroidViewModel) estan marcats com a obsolets actualment.

4.1 Persistència

La persistència s'implementa amb la llibreria Room. Aquesta llibreria forma part de l'Arquitectura de Components d'Android [4] i funciona sobre la base de dades Sqlite [2].

Els components principals de Room son la classe Database, les classes Entity i els DAO. Database conté la informació de la base de dades i és el principal punt d'acces a aquesta. Entity representa una taula de la base de dades. DAO conté els mètodes per accedir a la base de dades.

En el nostre cas la classe de tipus Database és DobileRoomDatabase. La base de dades conté quatre taules, cada una representada per una classe Entity: Scene, DeviceAC, DeviceSwitch i SwitchName. Aquestes classes són de tipus pojo, contenen un atribut per cada columna de la taula, amb els seus getters i setters. Cada atribut té una anotació @ColumnInfo que conté el nom de la columna de la base de dades que li correspon. L'atribut que correspon a la clau primària té l'annotació @PrimaryKey. També s'utilitzen altres anotacions com @NotNull que corresponen amb les columnes amb aquesta restricció.

Aquesta és la implementació de la Entity DeviceSwitch

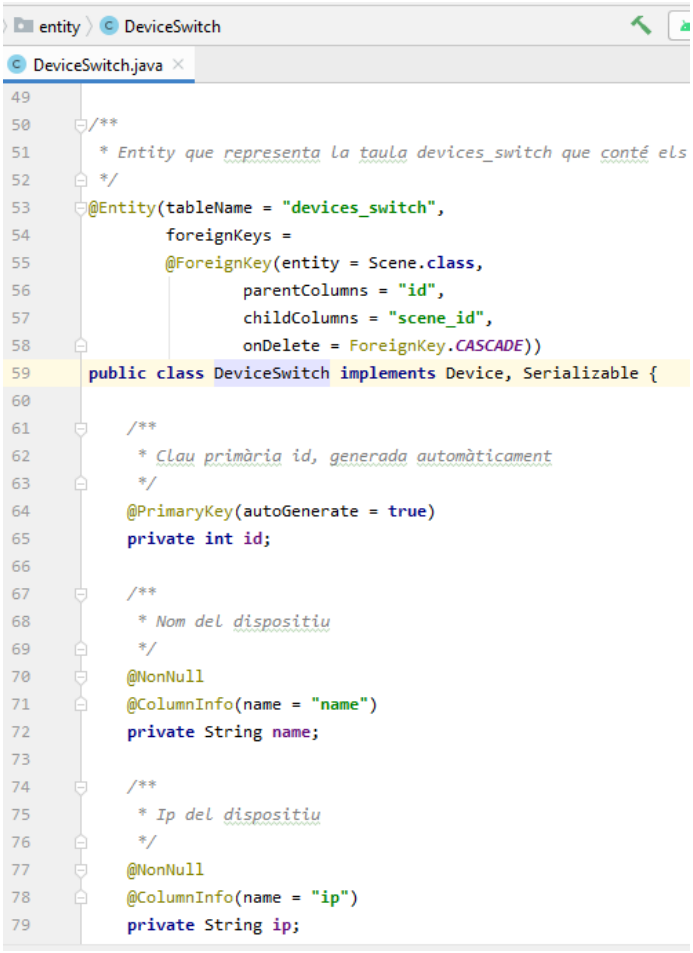
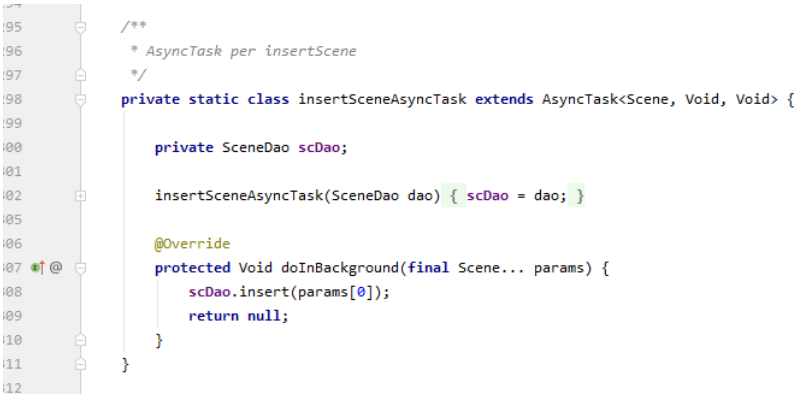


Figura 15 - Entity DeviceSwitch

A cada taula li correspon un DAO: SceneDao, DeviceSwitchDao, DeviceACDao i SwitchNamesDao. Es tracta de interfícies que contenen les operacions amb la base de dades.

Room també conté una classe repositori, en aquest cas anomenada DobileRepository i que és la que utilitza la capa ViewModel per realitzar les operacions amb la base de dades. Aquesta classe conté referències a tots els DAO. Les operacions amb la base de dades es fan en un thread separat del thread principal per no interferir en les operacions de la interfície gràfica i no deixar-la bloquejada. Cada operació amb la base de dades es realitza en un AsyncTask per aquest motiu, com es veu en aquest exemple per inserta una nova escena:



```
    /**
     * AsyncTask per insertScene
     */
    private static class insertSceneAsyncTask extends AsyncTask<Scene, Void, Void> {

        private SceneDao scDao;

        insertSceneAsyncTask(SceneDao dao) { scDao = dao; }

        @Override
        protected Void doInBackground(final Scene... params) {
            scDao.insert(params[0]);
            return null;
        }
    }
```

Figura 16 - Dobile Repositori. Inserció escena

4.2 ViewModel

La capa ViewModel conté les classes SceneViewModel, DeviceSwitichViewModel, DeviceACViewModel i DeviceNameViewModel. Aquestes són les encarregades de connectar la capa de UI amb la base de dades. Cada classe conté una referència a DobileRepositori que és el punt d'accés a les operacions de persistència. Ofereixen a la interfície gràfica, a les Activities totes les operacions que necessiten.

En aquesta imatge es veu com en el constructor de SceneViewModel es crea un objecte repositori i d'aquest objecte s'obté la llista de totes les escenes

```
public class SceneViewModel extends AndroidViewModel {

    /*Repositori de la base de dades*/
    private DobileRepository mRepository;

    /*Llista de totes les escenes*/
    private LiveData<List<Scene>> mAllScenes;

    /**
     * Constructor. Crea el repositori i obté la llista de totes les escenes
     *
     * @param application
     */
    public SceneViewModel (Application application) {
        super(application);
        mRepository = new DobileRepository(application);
        mAllScenes = mRepository.getAllScenes();

    }
```

Figura 17 - SceneViewModel constructor

El mètode insertScene de la mateixa classe també inserta l'escena mitjançant el repositori

```
    /**
     * Inserta una escena
     *
     * @param sceneName
     */
    public void insertScene(String sceneName) {
        Scene scene = new Scene(sceneName);
        mRepository.insertScene(scene);
    }
```

Figura 18 - SceneViewModel mètode insertScene

4.3 UI

Activities

Cada Activity conté una referència al ViewModel que necessita per realitzar les operacions amb la base de dades. A la classe SceneActivity, que mostra la llista de dispositius d'una escena, es crean objectes tres objectes ViewModel de tipus escena, dispositiu interruptor i dispositiu aire acondicionat

```
//ViewModels de les escenes i dispositius
mSceneViewModel = ViewModelProviders.of( activity: this).get(SceneViewModel.class);
deviceSwitchViewModel = ViewModelProviders.of( activity: this).get(DeviceSwitchViewModel.class);
deviceACViewModel = ViewModelProviders.of( activity: this).get(DeviceACViewModel.class);
```

Figura 19 - Creació Viewmodels a SceneActivity

A les activities corresponents es creen Observer sobre les llistes LiveData. Aplicant el patró Observer, quan a la base de dades es produeixen canvis, a les View s'actualitzen automàticament aquests canvis.

A la classe SceneActivity, per aplicar l'observer, sobre la llista de dispositius de tipus interruptor, es crida el mètode observe(), creant un nou objecte Observer. En aquest objecte s'implementa el mètode onChanged() que s'executa quan es produeixen canvis a la base de dades. En aquest cas el que es fa es aplica de nou la llista de dispositius modificada al ListAdapter

```
/**
 * Crea un nou observer sobre el LiveData que conté la llista de dispositius de tipus interruptor de l'escena.
 * Qualssevol canvi a la base de dades actualitza automàticament la View
 */
private void setDevSwitchesObserver(final DevSwitchesListAdapter adapterSwitches){

    deviceSwitchViewModel.getDevicesForScene(scene.getId()).observe( owner: this, new Observer<List<DeviceSwitch>>() {
        @Override
        public void onChanged(@Nullable final List<DeviceSwitch> devices) {

            adapterSwitches.setDevices(devices);
        }
    });
}
```

Figura 20 - Aplicació Observer a llista de dispositius

Les Activities implementades són:

- MainActivity: Entrada a l'aplicació, mostra la llista d'escenes
- SceneActivity: Mostra l'escena seleccionada
- DeviceACActivity: Dispositiu de tipus aire condicionat seleccionat. Permet enviar les comandes al dispositiu remot.
- DeviceSwitchActivity: Dispositiu de tipus interruptors. Permet enviar les comandes al dispositiu remot
- NewDeviceACActivity: Creació de nous dispositius de tipus aire condicionat. També permet la modificació dels ja existents
- NewDeviceSwitchActiviy: Creació de nous dispositius de tipus interruptor. També serveix per modificar els ja existents.

RecyclerView [8]

Per mostrar les llistes d'escenes i de dispositius s'utilitzen RecyclersView ja que són mes eficients per llistar grans llistes. Es reutilitzen els ítems en lloc de crear una gran llista. Potser no era necessari en una app com aquesta, el més normal és que la gent tingui un nombre limitat d'escenes, és a dir d'habitacions a la casa. I a cada escena és poc probable que els usuaris tinguin una gran llista de dispositius. Encara que podria existir algun cas en que sí fos possible, o que s'utilitzés l'app en un sistema industrial per exemple. Per tant he optat finalment per utilitzar RecyclerViews.

Els RecyclerView utilitzen ListAdapter, els quals implementen els ítems de la llista i contenen els Listener que realitzen l'acció quan es fa click a un ítem. S'ha implementat un ListAdapter per cada llista. Són tres: SceneListAdapter, DeviceSwitchAdapter i DeviceACListAdapter.

El constructor de cada ListAdapter crea un objecte ViewHolder (una classe interna del mateix ListAdapter i que conté un ítem de la llista). En aquest ViewHolder se li assigna la vista que representa el ítem. El mètode onBindViewHolder aplica les dades del dispositiu al ítem actual, en aquest cas el nom del dispositiu

```java
/**
 * Crea un nou ViewHolder inflant el layout device_switch_item que contindrà un sol ítem
 *
 * @param parent
 * @param viewType
 * @return
 */
@Override
public DeviceViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
    View itemView = mInflater.inflate(R.layout.device_switch_item, parent,  attachToRoot: false);
    return new DeviceViewHolder(itemView,  adapter: this);
}

/**
 * Assigna les dades, en aquest cas el nom del dispositiu, a l'ítem que està en una determinada posició
 *
 * @param holder ViewHolder de l'ítem
 * @param position Posició de l'ítem a la llista
 */
@Override
public void onBindViewHolder(DeviceViewHolder holder, int position) {
    if (mDevices != null) {
        DeviceSwitch current = mDevices.get(position);
        holder.deviceItemView.setText(current.getName());

    }
}
```

Figura 21 - Creacio del ViewHolder en un ListAdapter

Validator

S'ha creat una classe Validator per comprovar les entrades de text als formularis per crear i editar dispositius. Comprova que els camps obligatoris no estiguin buits i comprova que les IPs tinguin el format correcte.

Tema

D'acord amb el disseny DCU creat s'utilitza el tema Dark de Material Components (9), personalitzant els colors. Per aplicar el tema Dark he modificat al fitxer /res/values/styles.xml l'atribut parent de l'estil "APPTheme". L'estil per defecte que tenia l'he canviat a "Theme.MaterialComponents"

HelpFragment

La primera vegada que s'inicia l'aplicació es mostra una pantalla d'ajuda. També s'ha afegit als menús la opció d'ajuda per mostrar aquesta pantalla. La vista està implementada sobre un DialogFragment

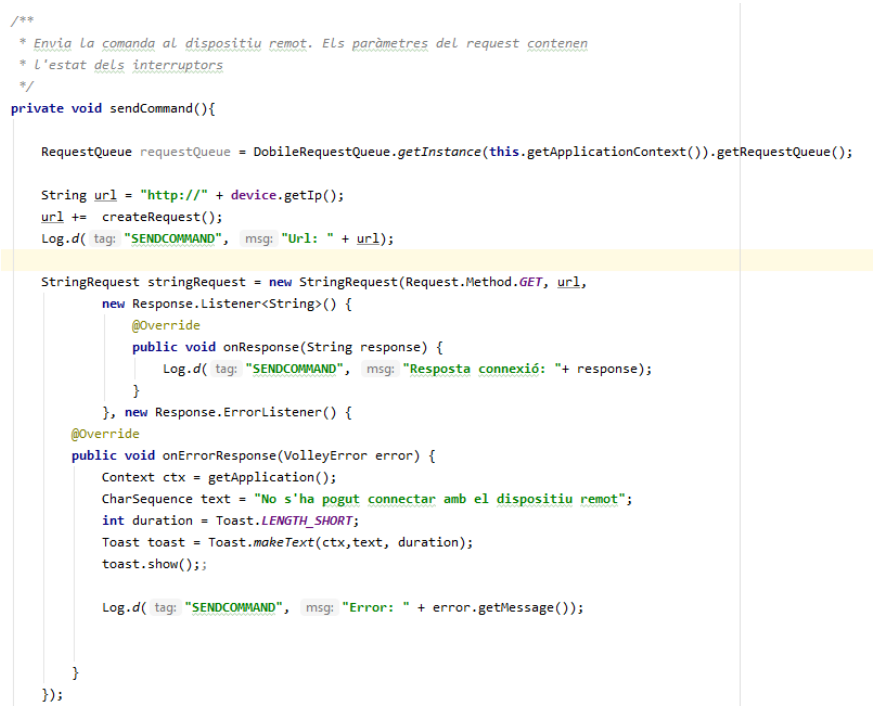
```
public class HelpFragment extends DialogFragment {
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
        // Use the Builder class for convenient dialog construction
        AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());

        LayoutInflater inflater = requireActivity().getLayoutInflater();
        // Inflate and set the layout for the dialog
        // Pass null as the parent view because its going in the dialog layout
        builder.setView(inflater.inflate(R.layout.dialog_help,  root: null));


        builder.setMessage("Ajuda Dobile")
                .setPositiveButton( text: "Acceptar", new DialogInterface.OnClickListener() {
                    public void onClick(DialogInterface dialog, int id) {

                    }
                });
```

Figura 22 - Implementació de HelpFragment

4.4 RequestQueue

Per enviar les comandes als dispositius remots s'ha creat la classe DobileRequestQueue. Implementa el patró singleton com es recomana per apps que realitzen connexions amb freqüència. D'aquesta manera es millora l'eficiència. En lloc de crear un nou request per cada connexió s'utillitza un sol request. Per realitzar les connexions s'utilitza la llibreria Volley [10].

L'activity DeviceSwitchActivity és la que envia les comandes als dispositius remots de tipus interruptor. Implementa el mètode sendCommand. Es crea un request amb l'estat dels interruptors i s'envia mitjançant DobileRequestQueue



```
/**
 * Envia la comanda al dispositiu remot. Els paràmetres del request contenen
 * l'estat dels interruptors
 */
private void sendCommand(){

    RequestQueue requestQueue = DobileRequestQueue.getInstance(this.getApplicationContext()).getRequestQueue();

    String url = "http://" + device.getIp();
    url +=  createRequest();
    Log.d( tag: "SENDCOMMAND",  msg: "Url: " + url);

    StringRequest stringRequest = new StringRequest(Request.Method.GET, url,
            new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {
                    Log.d( tag: "SENDCOMMAND",  msg: "Resposta connexió: "+ response);
                }
            }, new Response.ErrorListener() {
        @Override
        public void onErrorResponse(VolleyError error) {
            Context ctx = getApplication();
            CharSequence text = "No s'ha pogut connectar amb el dispositiu remot";
            int duration = Toast.LENGTH_SHORT;
            Toast toast = Toast.makeText(ctx,text, duration);
            toast.show();;

            Log.d( tag: "SENDCOMMAND",  msg: "Error: " + error.getMessage());


        }
    });
```

Figura 23 - Implementació de sendCommand()

4.5 Test [5] [6]

Es realitzen test AndroidJunit a les classes Dao per comprovar les operacions amb la base de dades. S'ha creat una classe test per cada classe Dao excepte per la classe DeviceACDao que encara no està implementat.

Abans de realitzar cada test s'executa el mètode createDb(). Aquest mètode crea una base de dades en memòria per fer els tests, sense afectar a les dades que puguin estar a la base de dades de treball. Es crea una instància de DobileRoomDatabase i d'aquesta instància es treuen els Dao que es necessiten.

```
@Before
public  void createDb(){

    MockitoAnnotations.initMocks( testClass: this);

    Context context = InstrumentationRegistry.getInstrumentation().getTargetContext();
    db = Room.inMemoryDatabaseBuilder(context, DobileRoomDatabase.class)
            .allowMainThreadQueries().build();
    deviceSwitchDao = db.deviceSwitchDao();
    sceneDao = db.sceneDao();
}
```

Figura 24 - Mètode createDb abans de realitzar cada test

Aquest mètode té l'anotació @Before per indicar que s'executa abans de cada test.

Els mètodes que realitzen els tests tenen l'anotació @Test. Existeix un mètode per cada operació amb la base de dades: insert(), delete(), update(), etc

```java
@Test
public void insert() throws Exception {
    Scene scene = new Scene( name: "Cuina");
    sceneDao.insert(scene);

    List<Scene> listScenes = sceneDao.getListAllScenes();
    scene = listScenes.get(0);

    List<SwitchNames> deviceNames = new ArrayList<>();

    SwitchNames devName1 = new SwitchNames("Focus 1");
    SwitchNames devName2 = new SwitchNames("Focus 2");
    SwitchNames devName3 = new SwitchNames("Focus 3");

    deviceNames.add(devName1);
    deviceNames.add(devName2);
    deviceNames.add(devName3);

    DeviceSwitch dev = new DeviceSwitch( name: "Llum cuina",  ip: "192.168.1.156",  numSwitches: 3, scene.getId(),deviceNames);

    deviceSwitchDao.insertDevice(dev);
```

Figura 25 - Test de inserció d'una escena a la base de dades

Aquí s'inserta un dispositiu i es comprova que existeix a la base de dades amb el mètode assertEquals:

```java
assertEquals(1, listDevices.size());
```

Per executar els test de cada classe es té que clicar amb el butó dret sobre la classe, a la finestra Project, i seleccionar Run "nom de la classe". Per exemple per executar els test de DeviceSwitchDaoTest s'ha de seleccionar Run DeviceSwitchDaoTest. Una vegada executats els tests, a la finestra inferior Run es mostra el nombre de tests que han passat i els que no. En aquest cas els 9 tests d'aquesta classe han passat correctament

Figura 26 - Execució de tests a la classe DeviceSwitchDaoTest

La classe DaoTestSuite, en el paquet suite, realitza tots els tests de cada classe DaoTest i mostra els que han passat i els que no. Per executar aquesta classe també es té que clicar amb el botó dret i seleccionar Run DaoTestSuite

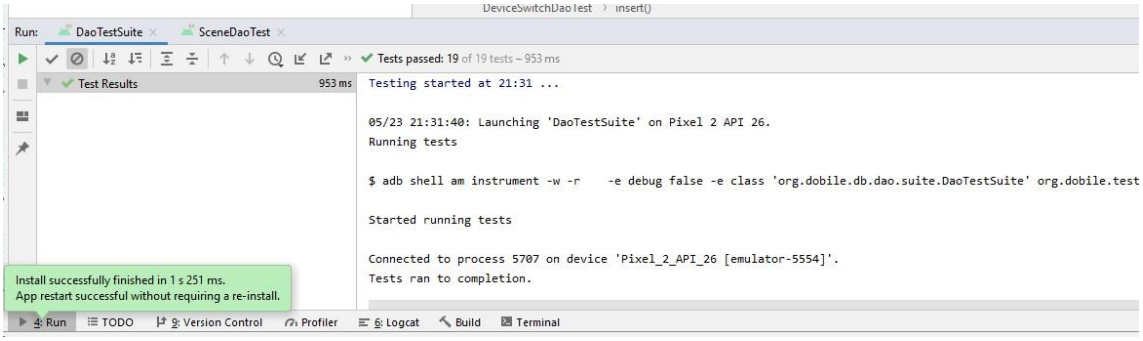


Figura 27 - Execució de tots els tests

Aquí han passat els 19 tests de totes les classes DaoTest.

5. Dispositius remots

5.1 Targetes de desenvolupament

Els dispositius remots es muntaran en targetes de desenvolupament o targetes de prototipatge. Aquestes targetes faciliten el prototipatge i muntatge de circuits electrònics. Existeixen diferents models. Unes de les més conegudes són les targetes Arduino, creades per l'empresa i comunitat del mateix nom.

Figura 28 - Arduino Uno

Es tracta de hardware lliure. Disposa de diferents models amb variacions de les seves característiques (diferències en el nombre d'entrades i sortides, en el processador, en la memòria, tamany de la targeta, etc).

Figura 29 - Arduino Mega

Algunes d'aquestes targetes de desenvolupament són Arduino Uno, Arduino Mega, Aduino Nano, etc [11].

5.2 Connexió wifi

Els dispositius remots necessitaran connexió wifi per poder rebre les comandes des de l'app. Una possible solució seria utilitzar Arduino Uno o Arduino Mega amb un mòdul wifi, com per exemple ESP8266 ESP01 [12]

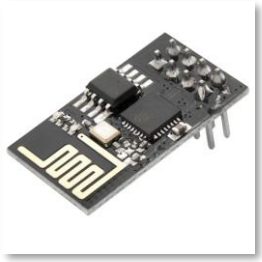


Figura 30 - ESP8266 ESP01

Figura 31 - ESP01 amb relé integrat

En els dispositius per controlar interruptors, en el cas de necessitar activar només un interruptor en tindríem prou amb el mòdul ESP01 connectat a un relé. Aquest exemple és un mòdul ESP01 amb relé ja integrat.

Si necessitem activar més de un interruptor la millor solució és utilitzar una targeta de desenvolupament amb wifi integrat. Existeixen vàries targetes d'aquest tipus com Arduino MKR1000, NodeMCU ESP32 o NodeMCU Lua Lolin V3 [13]. Les dues darreres són del fabricant Expressif i incorporen el mòdul ESP8266. La ESP32 a més té incorporat Bluetooth, que ens podria servir en un futur per enviar comandes a dispositius Bluetooth.

Figura 32 - NodeMCU V3

5.3 Control dispositius

Figura 33 - Mòdul de 8 relés

Per controlar els interruptors s'utilitzaran relés connectats a les sortides de la targeta de desenvolupament. Es venen mòduls amb diferents quantitats de relés ja integrats, des de un sol relé fins a 8 o més relés

Per enviar les comandes als equips d'aire condicionat s'empraran dos leds infrarojos controlats per un transistor [14]. Amb dos leds s'aconsegueix major poténcia de radiació i es pot arribar a una major distància.

Muntatge de dos leds i sensor de temperatura amb una targeta NodeMCU:

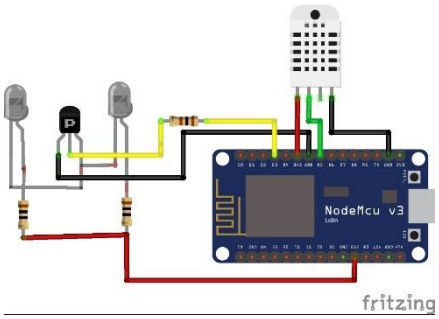


Figura 34 - Dispositiu aire acondicionat

5.4 VPN

Per poder controlar els dispositius des de fora de casa es farà mitjançant un servidor VPN per seguretat. Es pot instal·lar un servidor VPN a una Raspberry Pi seguint les indicacions d'aquesta pàgina https://www.pivpn.io/
Conté un script que facilita la instal·lació de manera segura.

Després és necessari instal·lar l'app OpenVPN [15] al mòbil i crear la nova connexió amb el fitxer de configuració proporcionat durant la instal·lació del servidor.

6. Conclusions

El treball m'ha servit sobre tot per refrescar i posar-me al dia amb la programació en Android. M'ha servit per aprendre molt sobre l'arquitectura MVVM i els seus components, com per exemple la llibreria Room. També ha estat una primera toma de contacte molt interessant amb el disseny DCU, temàtica totalment nova per mi, i la importància de dissenyar pensant en l'usuari.

En quant als objectius estan assolits en el que fa a la funcionalitat de l'app. La creació, modificació i esborrat d'escenes i dispositius funcionen correctament. També la part d'enviar les comandes als dispositius. Pot ser faltaria millorar la part estètica del disseny de la interfície gràfica.

El seguiment de la planificació penso que ha estat correcte en general. Només a la PAC 3, la part de la implementació, si que vaig estar apurat i al final vaig tenir el temps just. Volia deixar acabada la implementació el màxim possible en aquesta part. En general he anat seguint les etapes que m'havia fixat.

Les línies de treball futur serien seguir millorant la part estètica, afegir nous tipus de dispositius i fer un estudi per afegir afegir noves funcionalitats que no havia especificat als requisits inicials, i que podrien millorar la funcionalitat. Per exemple es podria afegir un botó per apagar tots els dispositius de la casa. Penso que seria útil quan l'usuari surt de casa. Una altra funcionalitat que es podria afegir seria crear una espècie de planificador que encengui els llums de manera aleatòria. Seria útil en segones residències per fer veure que la casa està habitada i evitar possibles robatoris.

6. Glossari

Arduino Companyia que desenvolupa software i hardware lliures. També es tracta d'una comunitat internacional que crea targetes de desenvolupament de hardware basades en microcontrolador.
Led (light emitting díode) Es tracta de un díode que emiteix llum quan se li aplica tensió als dos terminals, anomenats ànode i càtode. Fabricat amb material semiconductor,
Livedata Classe d'Android contenidora de dades que són observades mitjançant el patró Observer.
Microcontrolador (MCU) Petit ordinador integrat en un chip. Conté una o més CPU, la memòria i les entrades i sortides programables. Pot incloure la memòria flash o ROM i la memòria RAM. Pot formar part de un SOC (System on Chip).
NodeMCU Es tracta de firmware de codi obert que s'aplica a targetes de desenvolupament amb el mateix nom. L'avantatge sobre les típiques targetes Arduino Uno o Arduino Mega és que ja incorporen el mòdul ESP8266 que permet la connexió per wifi.
Observer Patró de disseny de software. Observa l'estat de l'objecte al qual s'aplica i quan es produeixen canvis ho notifica. S'utilitza a l'Arquitectura de Components d'Android. Quan es produeixen canvis a la base de dades els objectes LiveData ho notifiquen a les vistes mitjançant aquest patró.
Receptor infrarojos Sensor capaç de rebre les radiacions infrarojos. Utilitzats en els aparells electrònics que funcionen amb comandament a distància
Relé Dispositiu electromagnètic compost per un electroimant que obre o tanca uns contactes al rebre una tensió. Aquests contactes obren o tanquen un circuit elèctric. S'utilitzen com a interruptor electrònic.
Room Biblioteca de persistència d'Android que funciona sobre la base de dades SQLite.
SDK Conjunt d'eines de desenvolupament integrades a Android que conté un depurador, biblioteques, simulador de telèfon, documentació, exemples i tutorials.
Singleton Patró de disseny de software. Només es permet crear una sola instància de la classe que implementa aquest patró.
SOC (System on Chip) Circuit integrat o chip que integra la majoria de components d'un ordinador o sistema electrònic. Pot contenir el microcontrolador, la memòria, ports d'entrada i sortida, etc.
SQLite Motor de base de dades relacional.
Targeta de desenvolupament Targetes ideades per crear prototipus hardware de manera fàcil. Basades en microcontrolador. Contenen tots el elements necessaris a més del microcontrolador com la memòria flash, pins connectats a les entrades i sortides, connexió per l'alimentació, etc.
Tensió (elèctrica) Diferència de potencial elèctric entre dos punts. La unitat de mesura és el volti (V). Potser de tipus alterna (la polaritat canvia contínuament) o contínua, la qual té un pol positiu i un pol negatiu.

Transistor Component electrònic semiconductor d'estat sòlid que s'utilitza com a amplificador de senyals o com a commutador (o interruptor electrònic). Té tres terminals anomenats col·lector, base i emissor.
Transmissor infrarojos Led que emet llum en la gama dels infrarojos, no visible per l'ull humà i que s'utilitza en els comandaments a distància.

7. Bibliografia

1. **Android (operating system). *Wikipedia.* [En línea] https://en.wikipedia.org/wiki/Android_(operating_system).**

2. SQLite. [En línea] https://www.sqlite.org/index.html.

3. Wikipedia. Arduino. [En línea] https://en.wikipedia.org/wiki/Arduino.

4. Team, Google Developers Trainining. Android Developer Fundamentals (Version 2). *10.1: Room, LiveData, and ViewModel.* [En línea] https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/unit-4-saving-user-data/lesson-10-storing-data-with-room/10-1-c-room-livedata-viewmodel/10-1-c-room-livedata-viewmodel.html.

5. Bakshi, Kapil. Testing the Un-Testable With Android Architecture Components - Room Queries. *Proandroiddev.com.* [En línea] 04 de febrer de 2018. https://proandroiddev.com/testing-the-un-testable-and-beyond-with-android-architecture-components-part-1-testing-room-4d97dec0f451.

6. Birch, Joe. Android Architecture Components: Testing your Room DAO classes. *Medium.com.* [En línea] 18 de octubre de 2017. https://medium.com/exploring-android/android-architecture-components-testing-your-room-dao-classes-e06e1c9a1535.

7. Team, Google Developers Training. Android Developer Fundamentals (Version 2). *google-developer-training.github.io.* [En línea] Març de 2018. https://google-developer-training.github.io/android-developer-fundamentals-course-concepts-v2/.

8. Developers, Android. Cómo crear una lista con RecyclerView. *Android.com.* [En línea] https://developer.android.com/guide/topics/ui/layout/recyclerview.

9. Dark Theme. *Material Design.* [En línea] https://material.io/develop/android/theming/dark/.

10. Descripción general de Volley. *Developers.android.com.* [En línea] https://developer.android.com/training/volley.

11. Arduino Store - Most popular. [En línea] https://store.arduino.cc/arduino-genuino/most-popular.

12. Llamas, Luis. Conectar Arduino por Wifi con el módulo ESP8266 ESP01. *Lluisllamas.es.* [En línea] 27 de maig de 2017. https://www.luisllamas.es/arduino-wifi-esp8266-esp01/.

13. —. NodeMCU, la popular placa de desarrollo con ESP8266. *Luisllamas.es.* [En línea] 1 de juny de 2018. https://www.luisllamas.es/esp8266-nodemcu/.
14. Passerby. Powering mutiple Infrared LEDs through a single Arduino Uno pin? *Electronics.stackexchange.com.* [En línea] 11 de decembre de 2015. https://electronics.stackexchange.com/questions/205608/powering-mutiple-infrared-leds-through-a-single-arduino-uno-pin.

15. OpenVPN. [En línea] https://openvpn.net/.

16. Vogel, Lars. Developing Android unit and instrumentation tests - Tutorial. *Vogella.com.* [En línea] 10 de 04 de 2017. https://www.vogella.com/tutorials/AndroidTesting/article.html.

17. Arduino. Language Reference. *arduino.cc.* [En línea] https://www.arduino.cc/reference/en.

18. Ada, Lady. IR Remote Signals. *Adafruit.com.* [En línea] 29 de Juliol de 2012. https://learn.adafruit.com/ir-sensor/ir-remote-signals.

19. Getting Started. *Arduino.cc.* [En línea] https://www.arduino.cc/en/Guide/Introduction.

20. Android. Como probar tu base de datos. *Developers.android.com.* [En línea] https://developer.android.com/training/data-storage/room/testing-db#java.

21. Miu, Magda. Android Room Persistence Library: Relations. *Medium.com.* [En línea] 29 de juny de 2018. https://medium.com/@magdamiu/android-room-persistence-library-relations-75bbe02e8522.

22. Muntenescu, Florina. 7 Steps To Room. *Medium.com.* [En línea] 2017 de juliol de 2017. https://medium.com/androiddevelopers/7-steps-to-room-27a5fe5f99b2.

23. Nilanchala. Android Input Dialog Example. *StackTips.com.* [En línea] 24 de decembre de 2013. https://stacktips.com/tutorials/android/android-input-dialog-example.