Sergi Domingo Basora

SOUNDLESS: PLATAFORMA DE CIÈNCIA CIUTADANA PER AUDITAR ELS EFECTES DE LA CONTAMINACIÓ ACÚSTICA

TREBALL DE FI DE GRAU

dirigit per Pedro García López

Grau de Tècniques de Desenvolupament d'Aplicacions Web i Mòbils

UNIVERSITAT ROVIRA I VIRGILI

Tarragona

2021

Índex

1 Introducció

Quin impacte té la contaminació acústica en el nostre benestar?

Un ciutadà que viu al costat d'un aeroport o en el si d'una ciutat concorreguda, i que es veu sotmès a quantitats ingents de soroll ambiental, ¿té sistemes per demostrar de manera empírica les afectacions que té en la qualitat del seu son o en la seva tranquil·litat?

El problema de la contaminació acústica s'ha estudiat i acotat estrictament. L'OMS[1], que ha determinat que la contaminació acústica és la 2a causa de morts prematures a Europa [1], estableix uns límits acústics a partir dels quals el so resulta perjudicial per l'oïda humana a llarg termini i resulta en afectacions psicològiques i fisiològiques. L'OMS ha elaborat diversos documents al llarg dels anys per tal d'acompanyar amb evidència científica les estratègies públiques que mitiguin l'impacte de la contaminació acústica als països europeus [2], la primera dels quals, *Guidelines for Community Noise*, s'emet l'any 1999. Posteriorment, el 2002, la Unió Europea emet una directiva que serviria com a base per moltes legislacions futures i acabaria permeant a nivells més baixos, com lleis estatals, lleis autonòmiques i ordenances municipals des d'aleshores. Podem citar, per exemple, un extracte de l'*Ordenança de Medi Ambient de Barcelona* del 2011 [3]:

"En el desenvolupament de les diferents tasques de planejament urbà i d'organització de tot tipus d'activitats i serveis, l'Ajuntament tindrà present la contaminació acústica [...]. Ha de garantir que les actuacions municipals comportin, sempre que sigui possible, la disminució dels nivells sonors ambientals."

Les polítiques públiques s'han vist influenciades en alguna mesura per totes aquestes recomanacions i això ha acabat repercutint en un desenvolupament més sostenible de les ciutats. A dia d'avui, les ciutats de més de 100.000 habitants de l'estat espanyol publiquen periòdicament mapes de soroll i els mapes publicats a l'àrea metropolitana de Barcelona mostren una millora substancial respecte a les dades de fa 10 anys [4]. Ciutats com Reus han elaborat importants plans d'actuació per frenar l'impacte de la contaminació acústica [5].

I tanmateix, sense anar gaire lluny, un estudi de la Universitat de Vic del 2018 anunciava que el 88% dels pisos de la ciutat de Barcelona es veuen exposats a alts nivells sonors [6]. Un informe 3 anys posterior de l'Agència de Salut Pública de Barcelona [7] rebaixa aquesta xifra al 57%, però també xifra en 130 les víctimes mortals de la contaminació acústica *cada any* a la ciutat. En qualsevol dels casos, l'exposició al soroll és un problema molt vigent i que afecta a moltes persones.

A més de les estadístiques, que agreguen i homogeneïtzen la situació de milers de llars, cal considerar tot un seguit de circumstàncies que poden quedar invisibilitzades en

[1] Organització Mundial de la Salut.

estudis i polítiques d'abast generalista. No totes les ciutats o regions tenen les eines per dur a terme anàlisis amb prou periodicitat i prendre accions concretes al respecte, i fins i tot les que sí que ho fan poden menystenir o infravalorar els efectes de l'estacionalitat turística, l'aïllament acústic dels edificis, l'ubicació d'espais de treball i espai de descans dins els mateixos edificis, la major quantitat de trànsit en contextos concrets i barris concrets, l'impacte del soroll de tipus diferents de persones segons edat i condició, etcètera.

És a dir: si volem respondre la pregunta que ens hem fet al principi d'aquest apartat, "Quin impacte té la contaminació acústica en el nostre benestar?", la tasca pot resultar enormement complicada.

La idea de Soundless és molt clara: la implementació d'una plataforma de ciència ciutadana (*citizen science*) a través de la qual ciutadans puguin mesurar i indicar en el clic d'un botó quin és l'impacte de la contaminació acústica en la **seva** salut, mesurat dins la **seva** pròpia llar, i fer-ho de manera que les problemàtiques de cada usuari puguin manifestar-se en informes individuals i col·lectius. L'objectiu és, doncs, descobrir situacions que s'estiguin produint realment a les llars de les persones i determinar com els afecten a nivell personal.

La idea neix en el grup de recerca Cloudlab, del Departament d'Enginyeria i Matemàtiques (DEIM-URV), i prové originalment d'una situació concreta. A la ciutat de Tarragona, coneixem comunitats de veïns afectades per sorolls excessius derivats del pas de vies de tren per alguns barris i localitzacions. A partir d'aquesta situació real, volem posar-nos en la pell d'un ciutadà que, a tall d'exemple, vegi afectada la qualitat del seu son per factors relacionats amb el trànsit de trens, i facilitar-li eines perquè sigui ell mateix qui pugui emprendre accions i reclamar canvis a nivell local basant-se en les seves pròpies dades (interrupcions en el son, excessos de decibels quan passa un tren, afectacions en el ritme cardíac…). Volem donar-li eines per auditar específicament quins sons han afectat a la seva salut, com ho han fet i en quin moment, i oferir aquest servei de manera gratuïta, fàcil, pràctica, fiable i no esbiaixada.

En aquest sentit, el grup de recerca m'ha assignat per al desenvolupament del meu Treball Final de Grau les tasques d'iniciar aquesta iniciativa amb la vocació de relacionar la vessant de la recerca i les eines creades en el grup amb la creació d'aquesta plataforma amb objectius de caire social i ciutadà.

L'aplicabilitat de la recerca i la innovació m'interessa molt a nivell personal i és una iniciativa que he emprès amb orgull. A partir d'aquí, el procés que ja hem iniciat no culmina amb la meva presentació del Treball Final de Grau, sinó que preveu tenir una continuïtat en el temps per seguir reforçant el que, a dia d'avui, ja és un prototip funcional d'aplicació. L'objectiu final o la missió del projecte, tanmateix, és clara, i és conèixer el que hem repetit ja en aquesta introducció: quin impacte té la contaminació acústica en el nostre benestar.

En els següents apartats s'explorarà com s'ha decidit materialitzar i portar a terme el projecte utilitzant tecnologies apreses al llarg del meu grau i el coneixement de les quals

he anat ampliant al llarg de la meva estada com a becari a Cloudlab. Tot el procés constitueix el meu Treball Final de Grau.

2 Descripció general del projecte

Soundless és una plataforma digital de ciència ciutadana (*citizen science*) que té per objectiu auditar i analitzar els efectes de contaminació acústica en les persones, amb dades recollides per dispositius mòbils dels usuaris.

Soundless permet relacionar lectures d'excessos de soroll amb dades de salut reals dels usuaris des de dispositius *wearable* i la detecció de tipus de soroll per comprendre millor les fonts causants d'excessos. També permet la visualització de dades mitjançant diversos tipus d'informes i mapes per facilitar el monitoratge de les dades i la presa de decisions.

2.1 Objectius

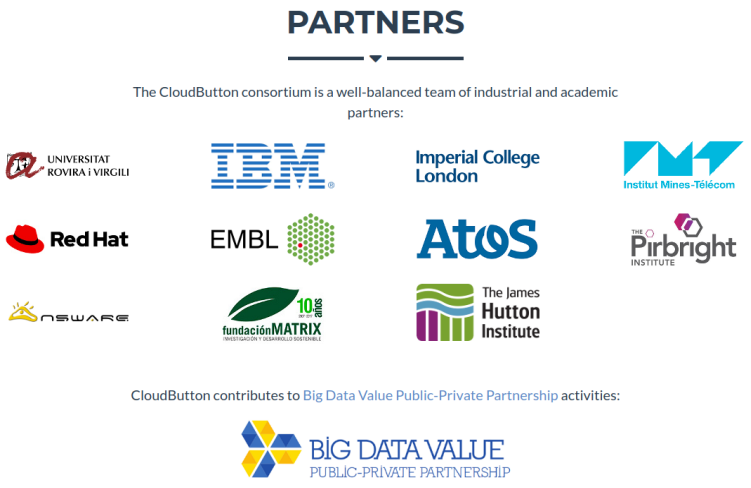
En el procés de conceptualització de Soundless s'identifiquen els següents objectius:

- Captació de dades d'intensitat de so ambiental
- Correlació de les dades d'intensitat de so amb mètriques de salut dels usuaris
- Classificació de sons per a la identificació de fonts de soroll
- Anàlisi de l'impacte de la contaminació acústica en els usuaris

2.2 Antecedents: inicis i projecte Cloudbutton

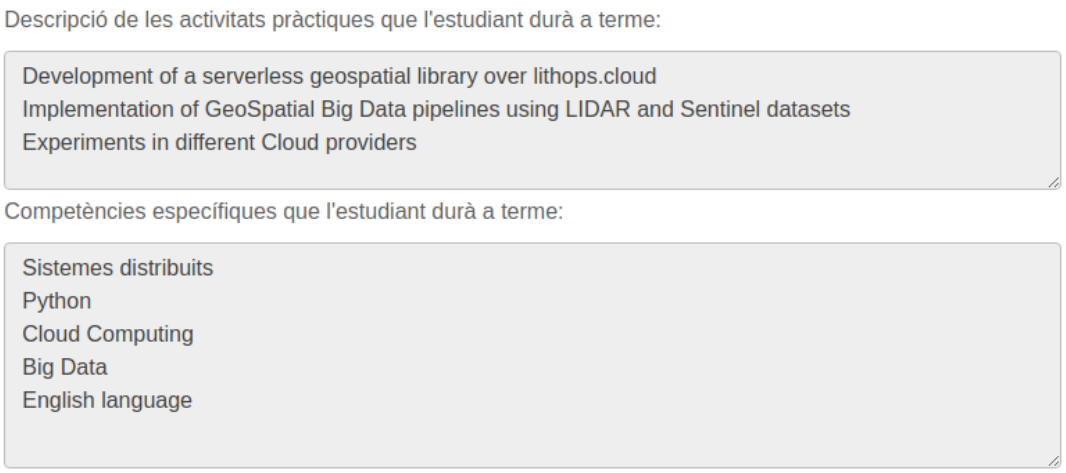
Aquest apartat s'endinsa en la relació de Soundless amb la meva tasca personal en el grup de recerca Cloudlab i, més concretament, en el projecte europeu Cloudbutton.

El grup de recerca Cloudlab ha sigut un dels actors que ha encapçalat el projecte Cloudbutton [8], una iniciativa d'àmbit europeu que pren com a objectiu democratitzar i simplificar els *workflows* Big Data. El projecte, estrenat oficialment els dies 22-25 de gener de 2019, rep finançament del programa de recerca i innovació Horizon 2020.



Imatge 2.2.1: entitats que constitueixen el consorci Cloudbutton. Font: [8]

A finals de 2020, el grup de recerca Cloudlab publica una oferta de beca per a estudiants de grau anomenada ***Serverless and Data analytics for GeoSpatial data***, a la qual m'inscric i sóc admès. La beca, que es convoca per treballar en els objectius de Cloudbutton i més concretament en objectius descrits en la imatge 2.2.2, comença a principis de 2021. Aquesta suposa la meva entrada al grup de recerca Cloudlab.



Imatge 2.2.2

Al llarg de la primera meitat de 2021, la beca em permet formar-me en tecnologies Cloud i relacionades (serverless/FaaS, Cloud Object Storage, Docker, IBM Cloud, formats i particionament de dades geospacials, llibreries geospacials en Python, etc). Especialment, em permet emprar el *framework* Lithops, desenvolupat en el marc del projecte Cloudbutton, per executar *workflows* Big Data entorns tan diversos com IBM Cloud Functions o Kubernetes i estudiar mètriques relacionades amb el rendiment de la llibreria (KPIs), detectar errors, i dur a terme tasques relacionades.

El projecte Soundless s'adiu amb els objectius de Cloudbutton de donar ús a les tecnologies Cloud per tal d'optimitzar tasques geospacials. **Soundless es percep com una opció per compatibilitzar diverses tecnologies i objectius**: captació i emmagatzemament de dades massives geolocalitzades (dades de so freqüents d'un gran nombre de dispositius); execució de tasques geospacials complexes en paral·lel (generació de grans quantitats de mapes, estudis de correlació amb altres fonts de dades geolocalitzades, etc) i computació d'anàlisis a temps real (amb l'augment de rendiment que Lithops proporciona).

Un dels resultats que Soundless aspira a proporcionar, en fases avançades, és la publicació de grans conjunts de dades sobre contaminació acústica *en conjunció amb eines com Lithops i tecnologies Cloud / GeoSpatial* per tal que científics, analistes de dades i fins i tot persones del carrer i associacions de veïns puguin estudiar l'impacte de la contaminació acústica de manera fàcil i molt eficient, sense que hi hagi grans costos d'infraestructura com a contrapartida ("democratitzar els *workflows* Big Data").

Per tant, Soundless no és un producte de coincidències, sinó una **continuació de la meva tasca a Cloudlab** per tal de contribuir a la missió del projecte Cloudbutton i, així

doncs, també incorpora implícita els valors i objectius de Cloudbutton i les tecnologies que s'hi han desenvolupat per tal d'esdevenir un cas d'ús de referència del projecte.

Soundless comença la seva fase de conceptualització i disseny el juny-juliol de 2021. A partir d'aquest moment, es defineix que Soundless serà el meu Treball Final de Grau. Vegeu apartat 2.3.1.Sobre la tria del dispositiu mòbil com a eina vehicular de recol·lecció de dades per més informació sobre el concepte inicial.

2.3 Informacions derivades del procés iteratiu de recerca

En aquest apartat s'identificaran una sèrie d'*insights* relacionats amb la descripció del projecte Soundless i els objectius de l'apartat 2.1.Objectius, i que justifiquen l'elaboració del llistat de requisits de l'aplicació, donen cos i estructura a tots els objectius del projecte i n'esclareixen la complexitat. La recol·lecció d'aquesta informació ha format part del Treball Final de Grau i és per tant part del procés d'investigació que dut a terme.

2.3.1 Sobre la tria del dispositiu mòbil com a eina vehicular de recol·lecció de dades

L'elecció de fer servir una aplicació mòbil com a client de la plataforma Soundless per als usuaris té una justificació. En una fase prèvia del Treball Final de Grau (juny-juliol de 2021; vegeu apartat 2.2.Antecedents: inicis i projecte Cloudbutton) estudio l'opció de fer construir sensors mitjançant la marca de microcontroladors Raspberry Pi. Es treballa per a utilitzar una nova *suite* d'eines d'orquestració de flotes IoT anomenada **balena** (balena.io)**,** amb **balenaOS** com a sistema operatiu per als dispositius, **balenaEngine** per gestionar tasques mitjançant contenidors, i **openBalena** per gestionar flotes. A excepció d'aquesta darrera, totes les eines es proven en dispositius reals adquirits pel grup de recerca Cloudlab.

Al cap d'unes setmanes, quan ja he implementat i provat diverses funcionalitats amb un prototip real, formulem diverses raons de pes per fer aquest canvi de plataforma:

1. L'únic element de maquinari afegit que s'introduïa en el sensor era el micròfon, així que no aportava cap element físic addicional de valor respecte a un mòbil.
2. Hi havia complexitat afegida a l'hora de detectar quina era la ubicació del dispositiu, connectar el dispositiu a una xarxa Wi-fi, i tot d'altres facilitats de què disposen els mòbils nativament.
3. L'experiència de l'usuari a l'hora de configurar el seu dispositiu era complicada i no disposava d'una interfície gràfica per interactuar-hi (fins i tot s'hauria hagut de disposar d'una app mòbil per a configurar-lo).
4. Es volia que el sensor fos tan barat com fos possible per facilitar l'adopció del dispositiu, així que no hi havia cap opció més barata que emplear dispositius "universals" que tothom té a casa com són els mòbils.
5. Amb el pressupost per cada sensor no podíem assolir un augment de qualitat destacable de micròfon en comparació amb els micròfons que ja es troben en els mòbils.

6. No hi havia raons per pensar que un mòbil actual es veuria mancat de capacitat de processament respecte a un sensor.
7. Els meus coneixements s'adaptaven més al desenvolupament d'aplicacions mòbils.

A partir d'aquestes raons es va decidir pivotar per tal de fer que el client fos una app mòbil. D'aquesta manera, el disseny de Soundless canvia substancialment, i el concepte posterior és el que es treballarà i es mostrarà en aquest document (si bé moltes parts són aplicables igualment a totes dues fases).

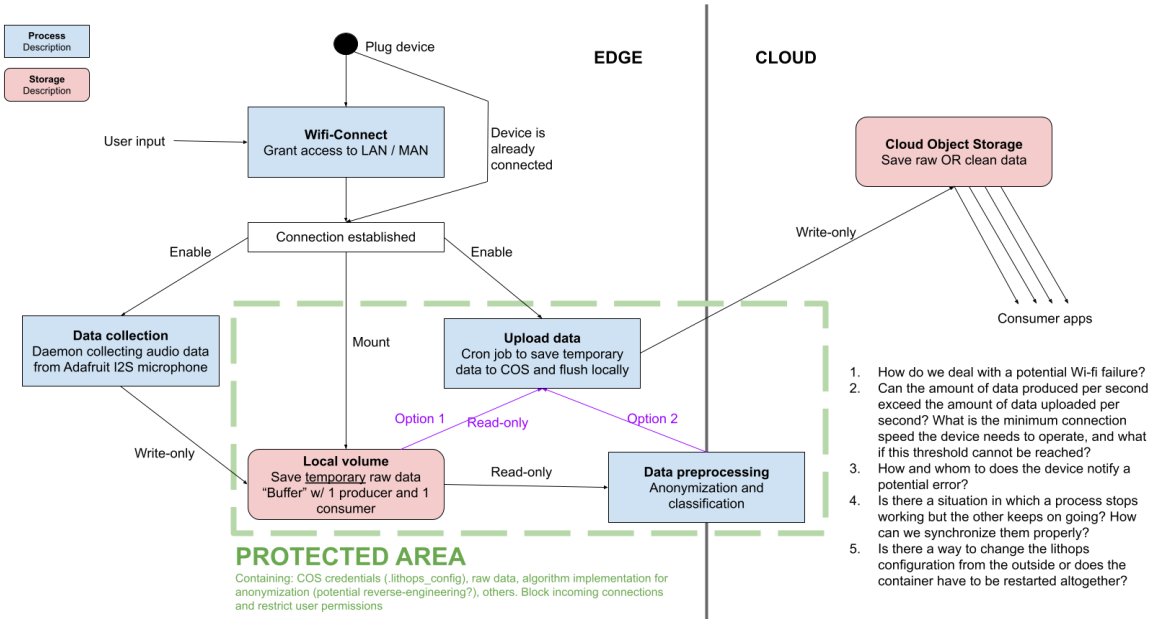
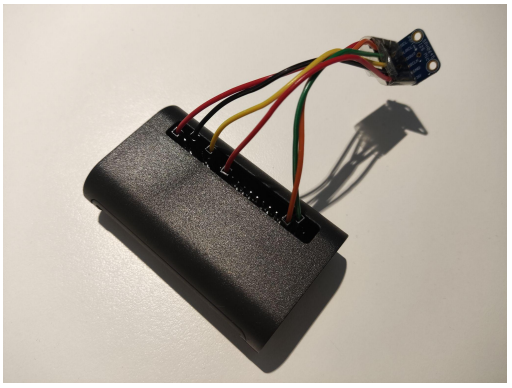


Diagrama 2.3.1.1: diagrama il·lustratiu de la fase anterior del projecte Soundless en què es treballava en el concepte d'un sensor Raspberry Pi



Imatge 2.3.1.2: prototip aconseguit del sensor físic

2.3.2 Sobre la mesura de decibels

La base per conèixer la contaminació acústica és captar distintivament les etapes d'excessos de soroll i mesurar-les en decibels de manera automatitzada. Es presenten diversos reptes:

En primer lloc, la **qualitat** de la medició de decibels depèn del **maquinari** (*hardware*) utilitzat, és a dir, en la qualitat dels micròfons. L'equipament de millor qualitat per al càlcul de decibels en interiors i (especialment) en exteriors pot variar entre els centenars i els diversos milers d'euros. A més, com a desenvolupadors de l'aplicació, no disposem control sobre la qualitat del micròfon dels dispositius mòbils dels usuaris ni sobre el context o espai on es duran a terme les gravacions; en conseqüència, no podem garantir uns estàndards de qualitat complets. Aquest problema és inherent a la plataforma i no té solució efectiva.

En un segon lloc, el **càlcul de decibels** (que es materialitza més endavant en l'annex 1) en si no és un procés trivial, ni tan sols realitzable en alguns casos. Els decibels (com a mesura de la intensitat de les ones de so) són una unitat comparativa, s'empren escales de decibels diferents per a situacions diferents i es necessita tenir parametritzat un valor de referència. Una de les dificultats afegides és l'absència d'un mapeig 1:1 de l'amplitud d'una ona física de so amb una xifra digital, cosa que, unit a la condició anterior, fan necessari un procés de calibració del maquinari per al càlcul de decibels. Per acabar, cal seleccionar una mesura de decibels que tingui sentit per la tasca concreta; per exemple, en el cas de voler conèixer l'impacte del so en l'oïda humana, s'acostuma a utilitzar un filtre sobre l'escala de decibels SPL (Sound Pressure Level) anomenat A-weighting, que dona pes addicional a unes freqüències de so determinades i redueix el pes d'unes altres per adaptar millor l'ona de so a la comprensió que en té l'oïda. Tanmateix, depenent del format de les dades d'entrada, tenim restringida la varietat de càlculs que podem aplicar sobre aquestes dades.

En un tercer lloc, la manera com un receptor interpreta una ona de so es veu modificat per **condicions d'entorn**. Com a exemple, un dispositiu sobre un trípode en un balcó exterior farà una mesura diferent que un dispositiu situat en contacte amb una taula, en un interior. La orientació del dispositiu, la distància, etc. són altres factors a considerar. Una plataforma de ciència ciutadana que obtingui dades d'usuaris independents té la capacitat de **recomanar però no obligar** al compliment de determinades condicions d'entorn.

En un quart lloc, el sistema on Soundless s'executi ha de tenir una sèrie d'**eines** i APIs que facilitin la **manipulació** i l'estudi del so recopilat, i ha de tenir la potència de processament suficient per tal d'efectuar les operacions de captació de decibels de manera òptima.

Complexitat

És important, pels motius anteriors, informar que la medició de la intensitat d'ones sonores en dispositius mòbils amb la màxima precisió és un problema que pot ben bé formar part d'un Treball Final de Grau independent en d'altres especialitzacions universitàries.

En aquest treball hem passat per un procés de recerca conscient i hem determinat que aquestes consideracions havien de ser enteses i reflectides sobre el producte però que, sobretot a nivell de prototip, la prioritat era captar tendències, increments i excessos en la intensitat sonora i el seu efecte en el benestar de les persones, *més que no pas* un procés rigorós de càlcul que requerís la intervenció de molts agents especialitzats. Soundless no és un servei de sonometria. Així doncs, Soundless ha de calcular la intensitat sonora però ha d'impedir que un excés de rigor dificulti el desenvolupament del projecte.

2.3.3 Sobre l'obtenció de mètriques de salut

Un dels objectius inicials de Soundless és que ha de ser capaç de relacionar les dades d'excessos de soroll amb dades de salut (canvis en patrons de son, alteracions en el ritme cardíac, etc) d'usuaris de la plataforma de manera que es pugui complir un dels principis de Soundless: la detecció dels *efectes* de la contaminació acústica (i no la detecció de contaminació acústica *i prou*).

D'on podem treure dades de monitorització de mètriques de salut que ens permetin aquest anàlisi? Durant una bona part del desenvolupament del Treball Final de Grau s'ha investigat l'opció d'utilitzar *només el dispositiu mòbil com a captador de dades*, una possibilitat instigada per l'aparició recent de **Sleep API**, una API de detecció de son a través dels sensors del mòbil [9] i altres possibilitats secundàries [10]. Tanmateix, per una millor adequació amb el cas d'ús de Soundless, s'acaba optant per l'ús de dispositius *wearable* externs, i és que es conclou que es tracta d'una manera més confiable, amb un ventall més ampli de dades a captar i que pot donar una millor experiència als usuaris.[2]

Per afavorir la consistència en el procediment i en les dades recollides, Soundless vol recomanar als usuaris l'ús d'un dispositiu específic, la polsera Xiaomi Mi 6, pel seu reduït preu i la quantitat de dades que pot recollir [11]. Tot i això, també es vol donar una certa flexibilitat a l'usuari perquè aprofiti dispositius que ja empri en el seu ús personal.

[2] Malgrat la decisió d'utilitzar *wearables*, en el procés d'implementació es proven diverses maneres d'implementar la connexió i la recollida de dades amb Sleep API per obtenir probabilitats que l'usuari dormi a temps real i obtenir les franges de temps en què l'usuari ha estat dormint anteriorment. Aquest procés es realitza a través de subscripcions a l'API i s'acaben obtenint dades reals. Ara bé, com que finalment es descarta l'opció i s'opta per iterar en el procés de disseny, no es presenta en aquest document cap altra referència directa a l'ús de Sleep API.

Imatge 2.3.3.1: polsera Mi Band 6. Font: botiga oficial de Xiaomi

Això implica que cal permetre que molts dispositius diferents de proveïdors diferents connectin amb Soundless mitjançant APIs, un repte que tampoc és fàcil d'acomplir pel gran ventall de proveïdors que rarament disposen d'APIs dedicades. Després d'estudiar les alternatives al llarg del procés de recerca, tot indica que cal utilitzar un magatzem de dades de salut amb la màxima interoperabilitat amb d'altres APIs i aplicacions com és **Google Fit**. En resum, el que això permet és que qualsevol dispositiu o aplicació interoperable amb Google Fit és utilitzable per Soundless per llegir les dades, només connectant el dispositiu o aplicació particular amb Google Fit. Un cas particular és el de l'esmentada polsera Xiaomi Mi 6, que així ho permet. Lamentablement, però, Google Fit no rep dades a temps real dels dispositius sinó que triga un temps en realitzar aquesta actualització periòdica.

Una consideració addicional és que malgrat que les aplicacions relacionades amb e-Health cada vegada són més nombroses i hi ha una tendència creixent a analitzar i estudiar comportaments de salut, les dades de salut resulten particularment sensibles i el seu ús indegut o les filtracions de dades poden resultar enormement perjudicials per als usuaris. És per això que cal adoptar des del principi una política restrictiva en l'ús de les dades de salut i estudiar bé les conseqüències d'aquest ús. Aquesta consideració s'articula millor en l'apartat 3.3.1.Requisits no funcionals del producte.

2.3.4 Sobre la classificació de soroll

La classificació o detecció dels tipus de soroll és una funcionalitat que es creu no prioritària, però que pot ajudar molt a la validació de dades provinents dels usuaris i a conèixer quines són les fonts que provoquen efectes als usuaris de l'aplicació. La classificació de sons és un problema de Machine Learning que, per tant, requereix llibreries adequades que permetin classificar aquests sons en la plataforma que es triï.

La realització d'aquesta part de l'aplicació també es veu subjecta a coneixements de Machine Learning que no he pogut obtenir en la formació curricular del meu grau. Ara bé, a nivell personal, he dut a terme formacions prèvies en alguns d'aquests aspectes a través de plataformes online que m'han permès entendre els conceptes necessaris.

En la fase de conceptualització es planteja l'opció de dur a terme la classificació en el dispositiu mòbil en comptes de fer-lo en el servidor pels problemes que pot comportar l'enviament de dades de so sensibles per Internet (un exemple pot ser una conversa entre dues persones). Aquesta arquitectura passaria a ser un exemple del que podríem dir "**Edge Machine Learning**" ja que passaríem a estar classificant en el mateix dispositiu mòbil, és a dir, el més a prop possible de la font de dades.

Es planteja també que, idealment, la classificació d'un so s'ha de produir a temps real: identificant els tipus de soroll *mentre* s'escolten.

El problema és que, en aquest cas (classificació en el mòbil a temps real) un dispositiu que capti les dades des d'un micròfon ha d'utilitzar les dades de so captades per a diverses finalitats: la de lectura de decibels i la de classificació. Hi ha diverses maneres de compaginar ambdues funcionalitats, però ens podem trobar amb el cas que els sistemes operatius i els dispositius mòbils tinguin restringits els accessos al micròfon de tal manera que només una font pot estar llegint les dades a temps real. Si bé això implica que (en teoria) una aplicació lectora hauria de poder estar redirigint l'*input* "en brut" de micròfon a dos processos diferents que efectuin la lectura de decibels per una banda i la classificació, per una altra, això no sempre es pot complir i es podria donar el cas que no es puguin executar els dos procediments en paral·lel.

Aquest és un problema identificat ja en el procés de recerca i que a la pràctica d'acaba demostrant i acaba afectant el desenvolupament del treball, tal com es veurà més endavant en l'apartat 6.Implementació i en l'annex 2. En aquests apartats també es discutiran alternatives com la classificació a través de fitxers.

2.4 Col·laboració amb alumne de màster

Durant el desenvolupament del Treball Final de Grau, un alumne del **Màster URV-UOC en Enginyeria Computacional i Matemàtica (ECiM)** va entrar a formar part del grup de recerca Cloudlab per tal de cercar una línia de recerca que li permetés realitzar el seu Treball Final de Màster (TFM). Al llarg del document es faran algunes referències a l'alumne com a "l'**alumne de màster ECiM**".

Des del grup de recerca se li van proposar diversos projectes actius en relació als quals podria dur a terme el seu TFM i un d'ells va ser Soundless, que ja estava en una fase de desenvolupament adequada per identificar i supervisar possibles tasques en les quals l'alumne de màster ECiM podria contribuir. L'alumne accepta aquesta proposta i passa a centrar el seu TFM en un projecte compatible amb Soundless que pugui incorporar-se en l'aplicació final. Els avantatges d'això són múltiples: l'alumne de màster ECiM pot treballar en un projecte que tingui un impacte en una aplicació real, el projecte Soundless es veu beneficiat d'una expertesa i coneixements de què no gaudia, i el grup de recerca Cloudlab pot concentrar les tasques de tots dos estudiants en un projecte i disposar d'una plataforma potencialment millorada.

Al llarg de tot el procés de desenvolupament de Soundless, em reuneixo amb l'alumne de màster ECiM en un total de 7 reunions de treball, duc a terme un seguiment aproximadament setmanal de les tasques que realitza en relació a Soundless, assessoro l'alumne a l'hora de determinar la viabilitat o no de les propostes per al seu TFM, li facilito recursos i documentació sobre les eines emprades, etcètera.

En concret, l'alumne de màster ECiM efectua amb el meu suport directe o indirecte les següents tasques:

- Recerca sobre les dades de so recollides per l'aplicació per tal de calcular els nivells d'intensitat de so en decibels
- Creació d'un *script* Python per generar dades simulades d'usuari que l'aplicació utilitza al llarg de la fase de proves
- Investigació sobre la viabilitat o no d'algorismes propis per tal de classificar les dades de soroll
- Utilització de llibreries de dades geospacials per al tractament i visualització de les dades recopilades per l'aplicació en mapes
- Agrupació (*clustering*) de dades per millorar les representacions en mapes de soroll

A inicis de novembre de 2021 l'alumne de màster ECiM, després d'un temps de participar en el projecte, emet un informe-proposta de realització de TFM en què, a més de les tasques mencionades, centra la línia de treball futura en una nova proposta: una xarxa neuronal capaç de detectar possibles afectacions en la salut mental derivats de la contaminació acústica.

3 Requisits

Nota: en el moment en què es finalitza el Treball Final de Grau, la recollida de requisits d'algunes de les funcionalitats de l'aplicació no ha estat definida encara, degut a la naturalesa canviant del projecte i a la seva projecció a llarg termini, i és per això que només s'han especificat les funcions que han passat per una fase de disseny completa.

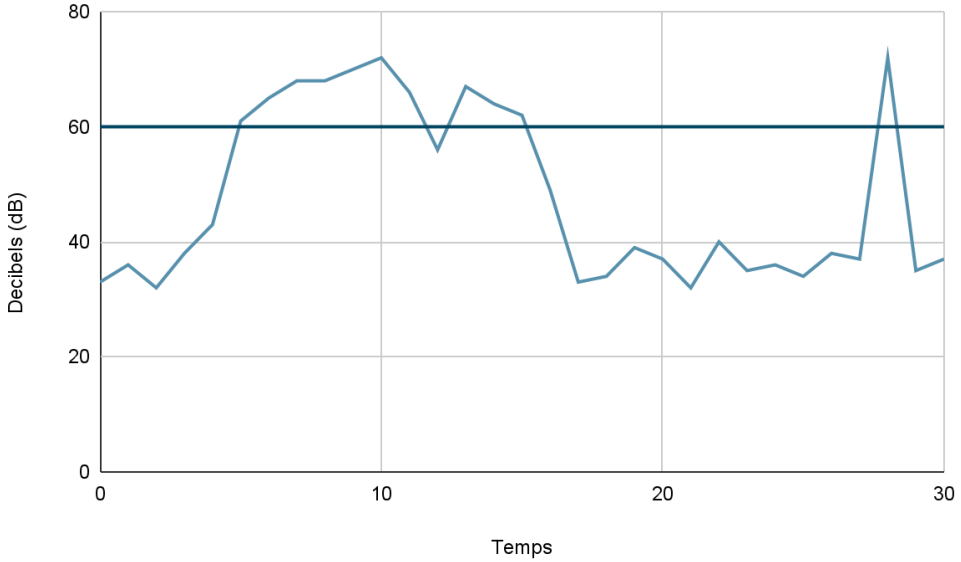
3.1 Informació recollida sobre els requisits

L'aplicació Soundless ha de permetre que un usuari d'un dispositiu Android inicií una **gravació** des de la pantalla principal, en el moment que ho vulgui. Una gravació consisteix en una detecció a temps real, mitjançant el micròfon del mòbil, de les variables de *so*: la intensitat de so (en decibels SPL) i la classificació de so (etiquetes com "Trànsit", "Conversa", "Animals" o "Crits"). La classificació de so s'haurà d'obtenir en l'idioma del dispositiu.

Un cop iniciada una gravació, l'usuari ha de poder interrompre la gravació quan vulgui. La interrupció ha de poder fer-se de dues maneres: **aturant** la gravació, tot causant que es generin i desin informes en base a les dades detectades, o bé **cancel·lant** la gravació, fet que no causa que es generi cap informe.

Durant la gravació, es volen desar dades únicament de les etapes on hi hagi hagut excessos de soroll, o bé ***etapes d'exposició***. Cal implementar un algorisme per detectar aquestes etapes en què hi hagi hagut soroll continuat.

El gràfic 3.1.1 representa un exemple de gravació. Tenint en compte (en aquest exemple) que a partir de 60 dB considerem que es produeix "excés de soroll", voldríem estudiar l'etapa d'exposició entre l'instant temps = 5 i temps = 15, en què es produeix un excés de decibels de manera consistent. No pas l'etapa en l'instant temps = 28; en què es produeix un pic aïllat.



Gràfic 3.1.1: exemple conceptual de gravació

Quan la gravació s'atura i per tant cal generar els informes (si es cancel·la, les dades es descartaran), cal crear un llistat de ***resums d'etapes d'exposició***, que també anomenarem indistintament ***informes***: un resum per cada etapa d'exposició. En aquest resum s'hi desaran els càlculs d'aquella etapa d'exposició, amb dades agregades sobre el que s'hagi recol·lectat: màxim de decibels, mitjana de decibels, etiquetes de classificació més freqüents, geolocalització en el moment en què s'ha iniciat la gravació, marques de temps i possibles atributs addicionals.

Els resums de les etapes d'exposició es **desaran** automàticament a 1) la base de dades del **dispositiu mòbil** (en el seu sistema de fitxers) i 2) a una base de dades en el **servidor**. Cada resum serà a partir d'aleshores representat en els mapes que el servidor generi (les dades hauran de ser anònimes). Els resums seran les úniques dades que es desaran en el servidor.

L'usuari podrà consultar en qualsevol moment un **llistat** amb els resums d'etapes d'exposició de gravacions que s'hagin fet en el seu mòbil, i **mapes** on es representin els resums d'etapes d'exposició de gravacions en general, del seu propi mòbil i del d'altres usuaris. Tant el llistat com els mapes podran filtrar-se segons (almenys) data i nivell de soroll, a més d'altres variables que es decideixin.

Asíncronament, és a dir, en algun moment posterior a la gravació, l'usuari podrà fer un **anàlisi de les variables de *salut*** (ritme cardíac, mètriques de son, i d'altres que es puguin considerar d'interès). Aquest procés es durà a terme només en cas que l'usuari ho sol·liciti, sempre i quan hagi demanat els permisos corresponents i hagi facilitat una font de dades de salut provinent de ***wearables*** que sigui compatible.

Quan l'usuari faci ús d'aquesta funció, Soundless connectarà amb la font de dades de salut per tal d'extreure totes les variables de salut i correlacionar-les estadísticament amb *tots* els resums d'etapes d'exposició disponibles. En aquest cas, els resums d'etapes d'exposició **s'actualitzaran** (tant en el dispositiu mòbil com en el servidor) amb les noves dades i, a partir d'aleshores, cada resum contindrà un apartat addicional amb les variables de salut.

És recomanable que l'usuari pugui veure un tutorial que li expliqui quines funcionalitats té l'aplicació i com utilitzar-les.

3.2 Requisits funcionals

3.2.1 Diagrama de casos d'ús

El diagrama 3.2.1.1 mostra el diagrama de casos d'ús corresponent a un usuari de la plataforma Soundless.

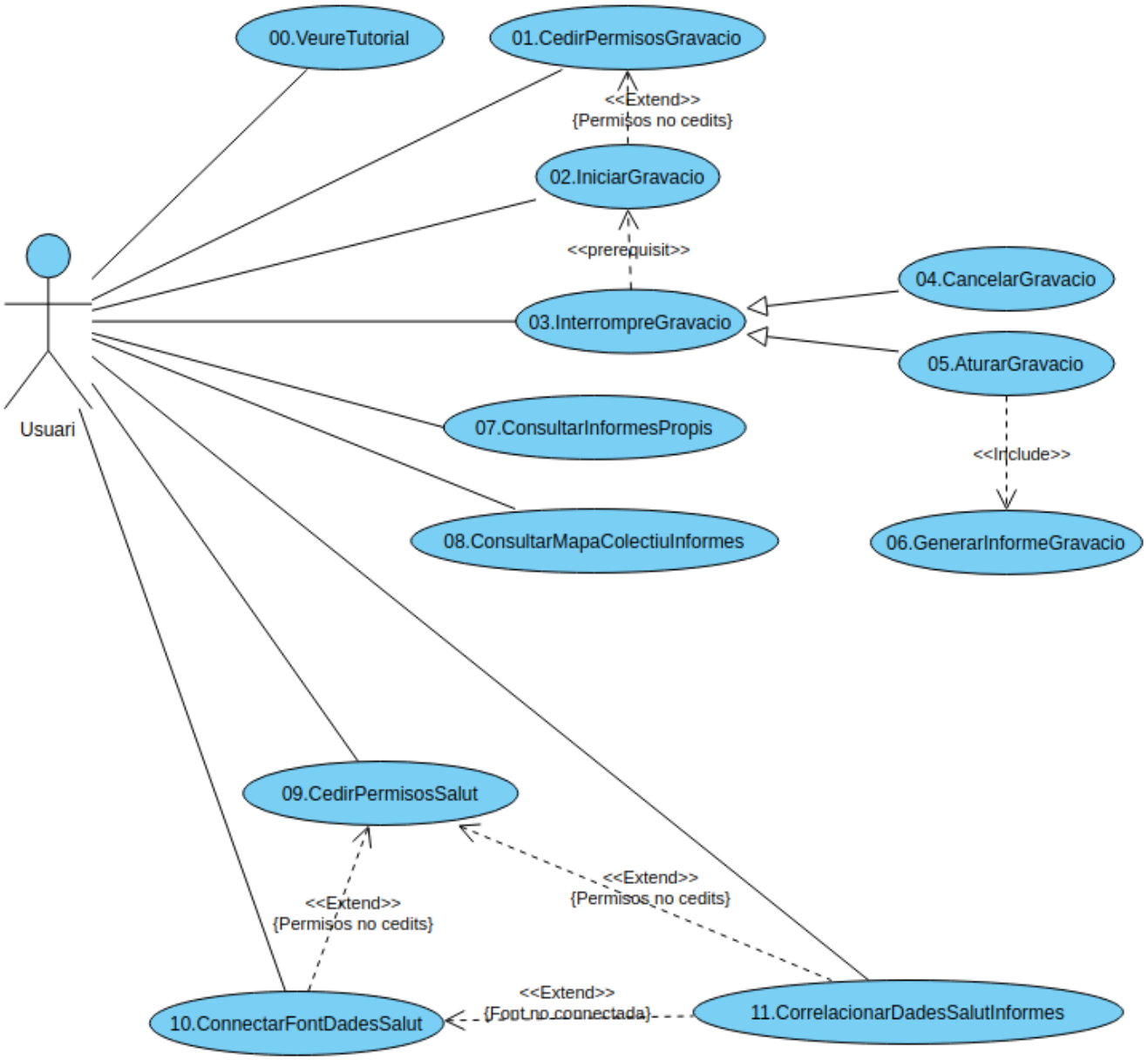


Diagrama 3.2.1.1

3.2.2 Especificació textual dels casos d'ús principals

Per la rellevància que tenen, es detallarà textualment el funcionament dels casos d'ús 02, 05, 06 i 11. El cas d'ús 04 tindrà un funcionament igual al cas d'ús 05 però deixant de banda la part de guardar les dades.

Cas d'ús 02.IniciarGravacio

Resum de la funcionalitat: s'engeguen els processos per a la captació de so i la classificació a temps real i l'usuari comença a rebre informació constant en la interfície d'usuari sobre els resultats de la captació de dades.

Paràmetres d'entrada: cap.

Paràmetres de sortida: cap.

Usuaris: Usuari.

Precondició: cap.

Postcondició: cap.

Procés normal principal:

1. El sistema mostra una pantalla amb un botó per engegar la gravació.
2. L'Usuari prem el botó.
3. El sistema inicialitza els components que interactuen amb el micròfon i activa els processos per realitzar la lectura de les dades periòdica.
4. El sistema amaga el botó de gravació i mostra el botó de cancel·lar i el botó d'aturar la gravació.
5. El sistema mostra periòdicament les dades recollides durant la gravació.

Alternatives de procés i excepcions:

3a. El sistema no disposa dels permisos d'accés al micròfon (per la lectura de dades de so) i/o de detecció de la geolocalització del dispositiu (per la generació d'informes posteriors).

3a1. El sistema executa el **Cas d'ús 01.CedirPermisosGravacio**.

3a2. El sistema torna al punt 3.

Cas d'ús 05.AturarGravacio

Resum de la funcionalitat: a petició de l'usuari, el sistema finalitza una gravació activa, genera un resum per cadascuna de les etapes d'exposició detectades i desa els resums.

Paràmetres d'entrada: cap.

Paràmetres de sortida: llistat de resums d'etapes d'exposició.

Usuaris: Usuari.

Precondició: cal haver executat el **Cas d'ús 02.IniciarGravacio**.

Postcondició: cal deixar l'aplicació en l'estat en què es trobava abans d'iniciar la gravació, és a dir, cal alliberar recursos de micròfon, eliminar processos de gravació, tornar a l'estat anterior de la interfície d'usuari, etc.

Procés normal principal:

1. El sistema mostra un botó per tal d'aturar la gravació activa en la pantalla de gravació.
2. L'Usuari prem el botó.
3. El sistema executa el **Cas d'ús 06.GenerarInformeGravacio**.
4. El sistema allibera tots els recursos associats a la gravació i torna a mostrar el botó per iniciar una gravació nova.
5. El sistema mostra un missatge informant de la finalització de la gravació.

Alternatives de procés i excepcions: cap.

Cas d'ús 06.GenerarInformeGravacio

Resum de la funcionalitat: crear un resum (informe) per a cada etapa d'exposició detectada durant la gravació que s'acaba i desar les dades.
Paràmetres d'entrada: les dades de la gravació.
Paràmetres de sortida: llistat de resums per cada etapa d'exposició.
Usuaris: cap.
Precondició: cal haver aturat una gravació activa (és a dir, cal haver executat el **Cas d'ús 05.AturarGravacio**).
Postcondició: cap.
Procés normal principal:

1. El sistema rep les dades de la gravació que contenen les etapes d'exposició i les lectures associades a cadascuna.
2. El sistema agrega les dades, efectua els càlculs i emet un resum per a cada etapa d'exposició.
3. El sistema desa les dades a la base de dades de l'aplicació en l'emmagatzematge del mòbil.
4. El sistema desa les dades al servidor.

Alternatives de procés i excepcions:

1a. No s'ha detectat cap etapa d'exposició.
1a1. El sistema omet els punts 2, 3 i 4.
3a. El dispositiu no disposa d'emmagatzematge.
3a1. El sistema mostra un missatge a l'usuari.
4a. El dispositiu no disposa de connexió a Internet.
4a1. El sistema mostra un missatge a l'usuari.

Cas d'ús 11.CorrelacionarDadesSalutInformes

Resum de la funcionalitat: l'usuari cedeix permís a Soundless perquè accedeixi a la font de dades de salut, llegeixi dades corresponents als moments en què s'han dut a terme etapes d'exposició al soroll, i correlacioni excessos de so amb possibles afectacions en la salut. En cas que no hi hagi dades disponibles, el sistema no correlaciona cap magnitud.
Paràmetres d'entrada: llistat de resums d'etapes d'exposició.
Paràmetres de sortida: cap.
Usuaris: Usuari.
Precondició: cap.
Postcondició: cap.
Procés normal principal:

1. El sistema mostra un botó per analitzar les dades de salut en els intervals on hi ha hagut etapes d'exposició.
2. L'Usuari prem el botó.
3. El sistema llegeix les dades i localitza possibles relacions entre esdeveniments.
4. El sistema actualitza els resums d'exposició en les bases de dades amb els índexs de correlació que hagi calculat.

Alternatives de procés i excepcions:

3a. El sistema no detecta cap font de dades de salut connectada.
3a1. S'executa el **Cas d'ús 10.ConnectarFontDadesSalut.**
3a2. El sistema torna al punt 3.

3b. El sistema detecta una font de dades de salut però no els permisos necessaris. (Possiblement, l'Usuari ha revocat els permisos a posteriori a l'aplicació o han estat revocats pel sistema operatiu)

3b1. S'executa el **Cas d'ús 09.CedirPermisosSalut.**

3b2. El sistema torna al punt 3.

3.3 Requisits no funcionals

3.3.1 Requisits no funcionals del producte

- Totes les dades han de ser anònimes.
- Cal demanar el mínim de dades de salut possibles a l'usuari i que siguin poc invasives.
- Cal garantir que les dades de salut llegides no abandonin el dispositiu mòbil a través d'Internet excepte en situacions en què la funcionalitat ho requereixi.
- Cal prioritzar que els informes col·lectius agreguin dades (p.ex. per localització) en comptes de mostrar-les individualment, per impedir la traçabilitat no desitjada d'aquestes dades.

3.3.2 Requisits no funcionals del procés

- La plataforma a utilitzar de la banda del client (app per a usuaris) serà Android.
- L'aplicació ha d'orientar-se als següents nivells de l'SDK d'Android:
  - minSdk 29 (Android 10) - degut a la implementació d'algunes de les funcionalitats obligatòries com la captura de decibels (v. annex 1)
  - targetSdk 30 (Android 11) - degut a la necessitat d'exportar l'aplicació a la botiga d'aplicacions Play Store d'Android que imposa aquest requisit per a publicar, a més de la previsible adopció massiva en el mercat d'Android 11 en el moment del llançament respecte a l'adopció encara reduïda d'Android 12
  - compileSdk 30 (Android 11) - per compatibilitat amb targetSdk
- El llenguatge de programació a utilitzar serà Kotlin.
- Se seguiran, en la mesura de les possibilitats, les recomanacions d'Android Jetpack pel que fa a arquitectures, llibreries, guies d'estil i eines relacionades.
- No es contempla la portabilitat a altres plataformes com iOS en les primeres etapes.
- Caldrà utilitzar els servidors disponibles al laboratori del grup de recerca Cloudlab per implementar l'aplicació de la banda del servidor.
- El servidor rebrà les dades de l'aplicació a través d'una API REST.
- L'API REST es crearà en el llenguatge Python fent servir Django i Django REST Framework.
  - Aquest requisit es deu al fet que possibles noves funcionalitats de la plataforma s'implementaran utilitzant el framework Lithops creat pel grup de recerca Cloudlab i que utilitza el llenguatge Python.
- La base de dades que s'utilitzarà és PostgreSQL.
- El codi ha de ser prou eficient per permetre l'execució de l'aplicació a una gran majoria de dispositius Android.

4 Anàlisi de requisits

En aquesta fase convertim les informacions recollides en l'apartat 3.Requisits a objectes. No s'estudiarà el comportament de les classes de frontera ni les seqüències d'interaccions entre classes i actors.

4.1 Diagrama de classes d'entitat

El diagrama 4.1.1 mostra una estructura amb les principals entitats dins l'estructura de l'aplicació (de l'aplicació client Android). En iteracions posteriors s'ha refactoritzat la base de codi per incorporar aspectes funcionals lleugerament diferents, però el nucli de la lògica de l'aplicació és el mateix i les classes descrites en el diagrama es mantenen.

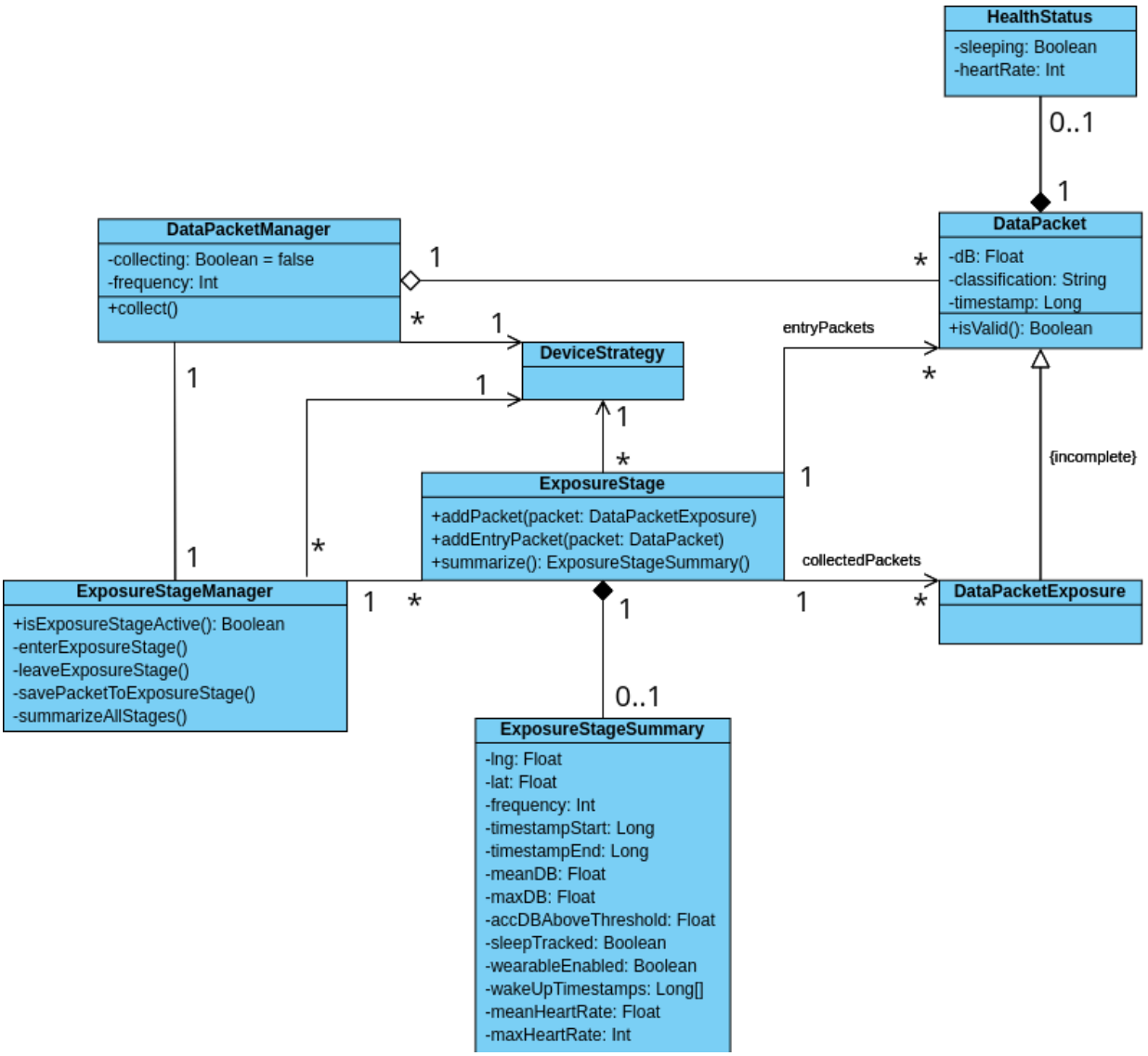


Diagrama 4.1.1

4.2 Explicació del diagrama de classes d'entitat

1. La unitat bàsica d'informació és **DataPacket**. Els objectes d'aquesta classe contindran informació associada a un moment puntual del temps durant una gravació.

2. Els DataPackets contenen **HealthStatus**, que és són objectes que encapsulen dades i operacions relacionades amb les dades de salut sol·licitades a l'usuari.

3. La classe **DataPacketManager** recol·lecta dades i, amb aquestes dades, crea i emmagatzema DataPackets.

4. A mesura que un DataPacketManager va creant DataPackets, segons el nivell de decibels, es pot produir un excés de soroll. En aquest cas, s'instanciarà una **ExposureStage** (etapa d'exposició).

5. L'**ExposureStageManager** és l'encarregat d'instanciar l'ExposureStage, i ho farà quan el DataPacketManager li ho indiqui. L'ExposureStageManager també mantindrà totes les ExposureStage que s'hagin produït en qualsevol moment de la gravació.

6. Quan una ExposureStage està activa (ho sap l'ExposureStageManager), en comptes de DataPackets, es crearan **DataPacketExposure**s. Aquestes entitats són DataPackets especialitzats.

7. Una ExposureStage contindrà els DataPacketExposures que s'hagin creat mentre la fase d'exposició està activa (**collectedPackets**); però no només això, també contindrà els DataPackets *que han fet que s'entri* en fase d'exposició (**entryPackets**). Això es deu al fet que els DataPackets d'entrada també són rellevants per determinar l'excés de so que s'ha produït.

8. Una ExposureStage pot ser resumida. L'ExposureStageManager s'encarrega de resumir totes les seves ExposureStages amb el mètode summarizeAllStages(). Típicament això es produirà al final de la gravació activa. Quan es resumeix una ExposureStage, s'obté un **ExposureStageSummary** que conté tota càlculs i informació agregada dels collectedPackets i entryPackets que contenia l'ExposureStage. Els ExposureStageSummary seran els objectes que després es desaran en les bases de dades i s'analitzaran en els informes; en altres paraules, són una abstracció per als informes / resums d'etapes d'exposició.

9. **DeviceStrategy** és un objecte que defineix el comportament de la resta de classes. Per exemple, defineix com es recullen els DataPackets, amb quina freqüència, si es fa ús de *wearables* o no, o detalls relacionats amb el mètode de classificació. DeviceStrategy es pot modificar en temps d'execució seguint el patró *Strategy*.

5 Disseny

5.1 Arquitectura de l'aplicació (client)

El model d'arquitectura triat per tal de satisfer els estàndards de qualitat del producte i les recomanacions d'Android Jetpack (un requisit que es detalla en l'apartat 3.3.2.Requisits no funcionals del procés) és l'arquitectura MVI (Model-View-Intent), una extensió de l'arquitectura MVVM (Model-View-ViewModel) popular per a aplicacions Android, amb funcionalitats afegides per tal de facilitar i sistematitzar la comunicació entre l'objecte ViewModel i els elements de la vista (View).

En l'apartat 6.1.Implementació de l'arquitectura triada (client) es fa referència a com aquesta decisió es materialitza en el codi.

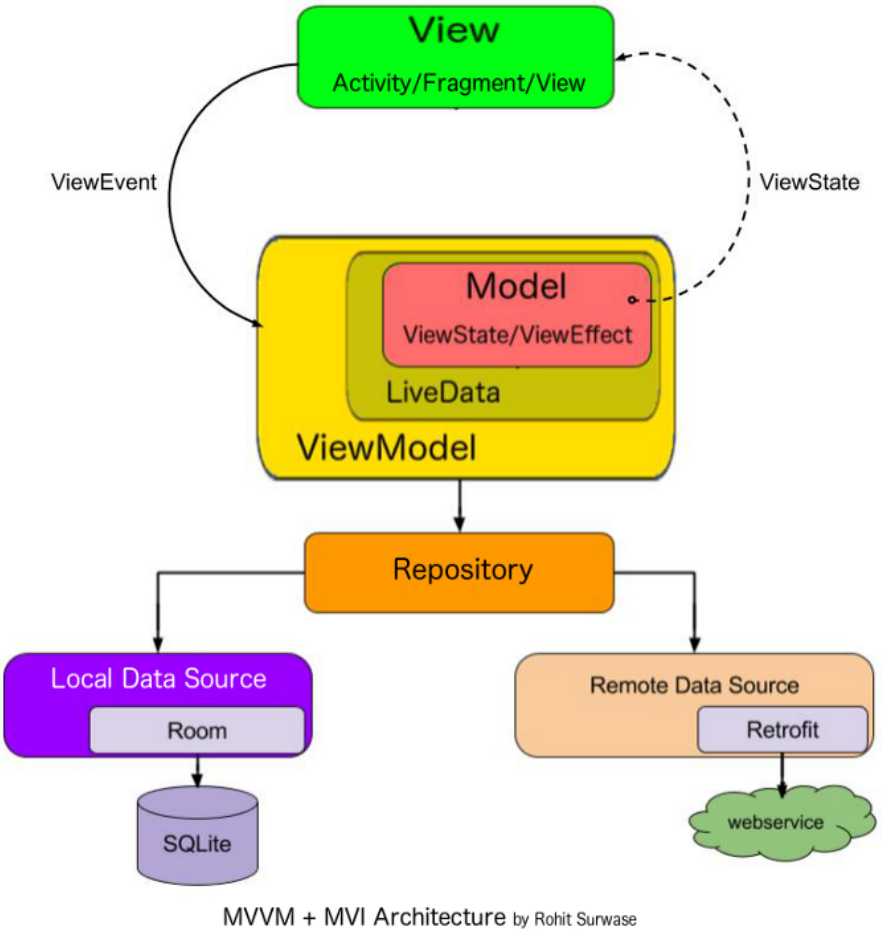


Diagrama 5.1.1: Model d'arquitectura MVI. Font: ProAndroidDev

5.2 Disseny de la persistència de dades

En el conjunt de l'aplicació, **l'única entitat que persisteix en bases de dades són els resums de les etapes d'exposició**. Aquestes entitats contenen dades agregades corresponents a una etapa d'exposició al soroll que s'hagi produït al llarg d'una gravació.

Aquestes dades es desaran, en base als requisits, a:

- La base de dades SQLite de l'aplicació en el dispositiu mòbil
- La base de dades PostgreSQL del servidor a través de l'API REST

5.2.1 Ús d'ORMs i implicacions en la creació de les taules

En la creació de les taules i el procés de disseny s'ha fet ús de les següents eines:

- ORM[3] de la llibreria Room per al client Android
- ORM de Django en el servidor

És per això que en el disseny de la persistència el que s'ha fet no és realitzar l'especificació textual en SQL de les taules (aquesta part va a càrrec dels ORM corresponents) sinó crear els objectes que fossin després traslladats a les taules, és a dir, determinar els *models*.

A continuació es mostra la configuració que s'ha fet d'aquests models.

Servidor (API)
(Python)

```
from django.db import models
from django.contrib.gis.db import models as geomodels

class Summary(models.Model):
   location = geomodels.PointField(srid=4326)  # 4326: Lng, Lat
   frequency = models.PositiveSmallIntegerField(default=1000)  # ms
   timestamp_start = models.BigIntegerField()
   timestamp_end = models.BigIntegerField()

   mean_dB = models.DecimalField(max_digits=7, decimal_places=4)
   max_dB = models.DecimalField(max_digits=7, decimal_places=4)
   acc_dB_above_threshold[4] = models.DecimalField(max_digits=12,
decimal_places=4)

   sleep_tracked = models.BooleanField(default=False)
   sleep_api_enabled[5] = models.BooleanField(default=False)
   wearable_enabled = models.BooleanField(default=False)
   wakeup_timestamps[6] = models.CharField(max_length=1000, blank=True)
```

[3] Object-Relational Mapper, que tradueix els models definits en el codi a taules de la base de dades.
[4] `acc_dB_above_threshold`: decibels acumulats per damunt del límit establert com a perjudicial
[5] `sleep_api_enabled`: Sleep API activada? Aquest camp es manté en el model per compatibilitat
[6] `wakeup_timestamps`: casos en què l'usuari s'hagi despertat (obtingut a partir de les dades de salut)

Client
(Kotlin)

```
import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "summaries")
class SummaryDatabaseEntity
(
   @field:PrimaryKey(autoGenerate = true)
   @ColumnInfo(name = "id")
   var id: Int,

   @ColumnInfo(name = "lng")
   var lng: Float,

   @ColumnInfo(name = "lat")
   var lat: Float,

   @ColumnInfo(name = "frequency")
   var frequency: Int,

   @ColumnInfo(name = "timestampStart")
   var timestampStart: Long,

   @ColumnInfo(name = "timestampEnd")
   var timestampEnd: Long,

   @ColumnInfo(name = "meanDB")
   var meanDB: Float,

   @ColumnInfo(name = "maxDB")
   var maxDB: Float,

   @ColumnInfo(name = "accDBAboveThreshold")
   var accDBAboveThreshold: Float,

   @ColumnInfo(name = "sleepTracked")
   var sleepTracked: Boolean,

   @ColumnInfo(name = "sleepApiEnabled")
   var sleepApiEnabled: Boolean,

   @ColumnInfo(name = "wearableEnabled")
   var wearableEnabled: Boolean,

   @ColumnInfo(name = "wakeupTimestamps")
   var wakeupTimestamps: String = ""
)
```

Els dos models gestionen de maneres diferents les coordenades del dispositiu. En el servidor, com que necessitem processar i representar gràficament punts en formats geogràfics per la creació de mapes, el que farem *en el moment de serialitzar* les dades és

passar longitud i latitud a format WKT[7] (en una String de tipus: "`POINT (lng, lat)`"). Les llibreries geogràfiques que utilitzem s'encarregaran de processar aquesta informació.

En el mòbil simplement desem la longitud i latitud numèricament, per mantenir la informació neta i agnòstica sobre futures altres eines de serialització, possibles altres serveis que necessitin les dades en diferent format, etc.

5.3 Disseny de la interfície gràfica (client)

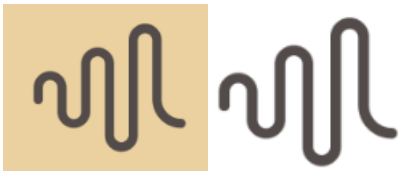
Aquest apartat contindrà alguns dels aspectes més rellevants de l'aspecte gràfic de les pantalles i finestres de l'aplicació Android.

5.3.1 Sistema de disseny

Soundless utilitza Material Design 2. (No pas Material Design 3, també anomenat Material You, anunciat el maig de 2021 per Google [12])

5.3.2 Logotip, fonts i colors

Logotip
El logotip de Soundless s'ha dissenyat per a tenir una aparença relacionada amb els conceptes "ones de so" i "anàlisi".



Imatge 5.3.2.1 (esquerra): logotip de Soundless
Imatge 5.3.2.2 (dreta): logotip de Soundless en fons blanc

Fonts
Les fonts triades per als textos de l'aplicació són les següents:

PT Sans Bold (imatge 5.3.2.3)

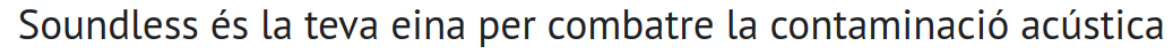
- Per títols
- Contorns arrodonits similars als del logotip
- Imatge de confiança

Imatge 5.3.2.3

[7] Well Known Text, sintaxi popular per definir objectes geospacials.

PT Sans (imatge 5.3.2.4)

- Per subtítols i altres contextos en què es vulgui ressaltar un altre missatge principal
- Versió de l'anterior font, sense negreta



Imatge 5.3.2.4

Nunito light (imatge 5.3.2.5)

- Per tutorials i informació sobre el projecte, missatges de caire més comunicatiu
- Font estilitzada i moderna

Imatge 5.3.2.5

Roboto (imatge 5.3.2.6)

- Per la resta de missatges
- Font estàndard d'Android
- Funcional i neutra

Imatge 5.3.2.6

Colors

Els colors es divideixen en dues sub-gammes:

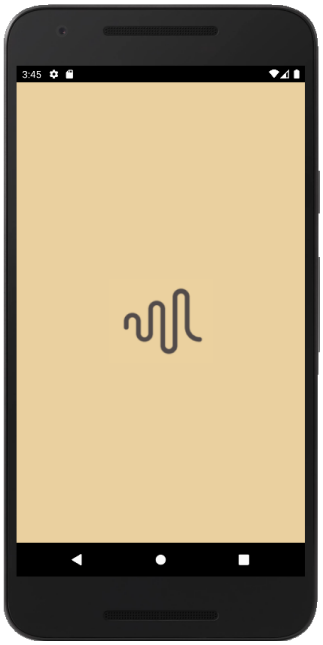
```
Color primari: #FF524B4A
Color primari-fosc: #FF292322
Color primari-clar: #FF7E7776

Color secundari: #FFEAD09F
Color secundari-fosc: #FFB79F70
Color secundari-clar: #FFFFFFD0
```

El color **primari** i el **secundari** són els dos colors disponibles en el logotip i que tenen un pes més rellevant en tot el disseny de pantalles. La resta de colors són iteracions sobre aquests dos per tal de configurar una gamma més àmplia i poder afavorir la llegibilitat, contrastar parts de la interfície, millorar l'estètica d'alguns components, etc.

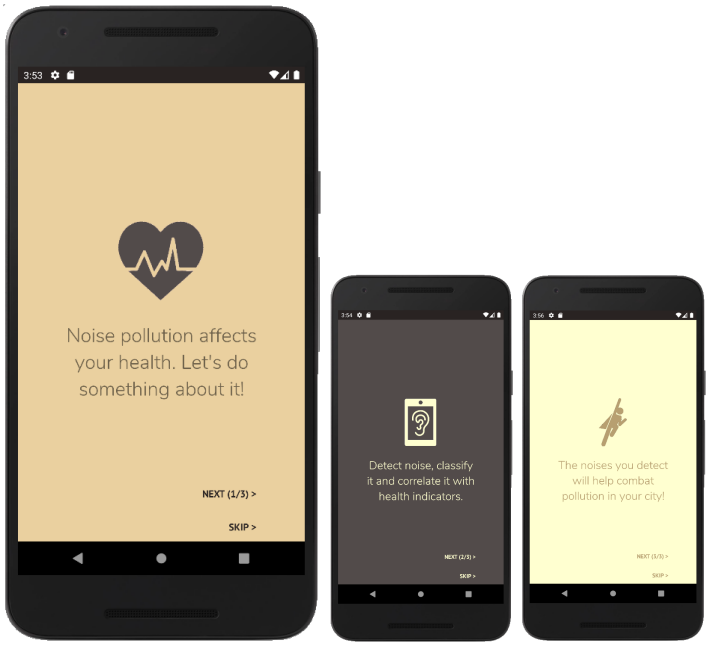
Tota la gamma de colors s'ha seleccionat per la seva calidesa i induir una sensació de comfort-benestar.

5.3.3 Pantalles

En aquest apartat es mostra el disseny de les pantalles principals de l'aplicació. No es mostren tots els quadres de diàleg ni els missatges Toast. Notem que els missatges són en anglès però s'han elaborat traduccions per a tots els dissenys i missatges en tres idiomes: català, castellà i anglès.

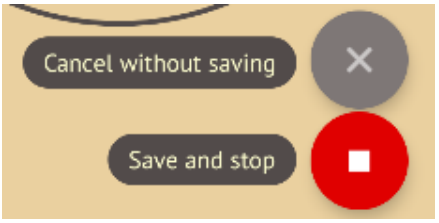


Imatge 5.3.3.1: pantalla Splash

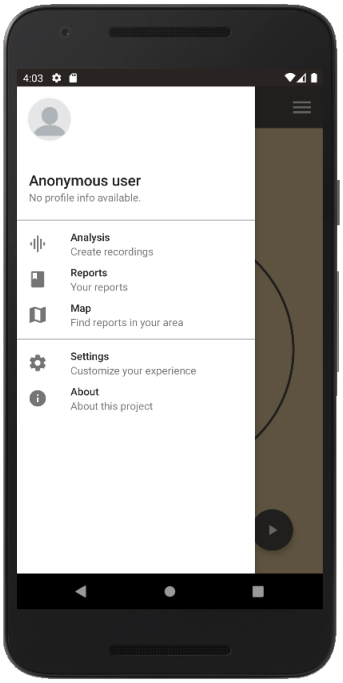


Imatges 5.3.3.2, 5.3.3.3 i 5.3.3.4: pantalles de tutorial (amb missatges de prova)

Imatge 5.3.3.5: pantalla principal de gravació

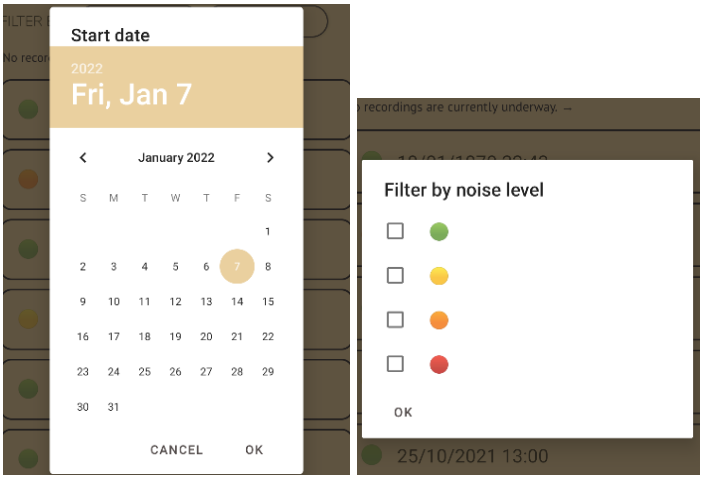


Imatge 5.3.3.6: botons en iniciar una gravació

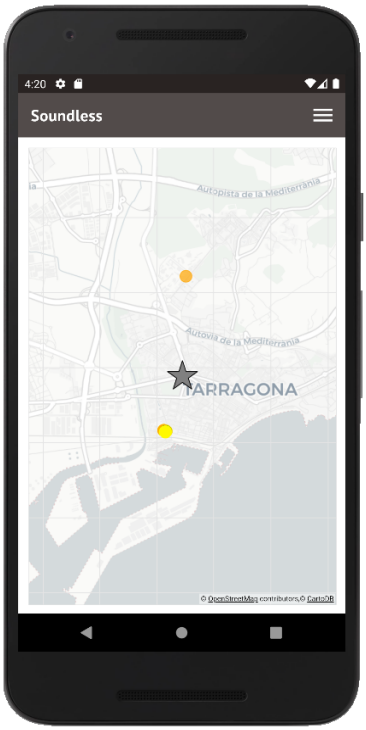


Imatge 5.3.3.7: menú drawer

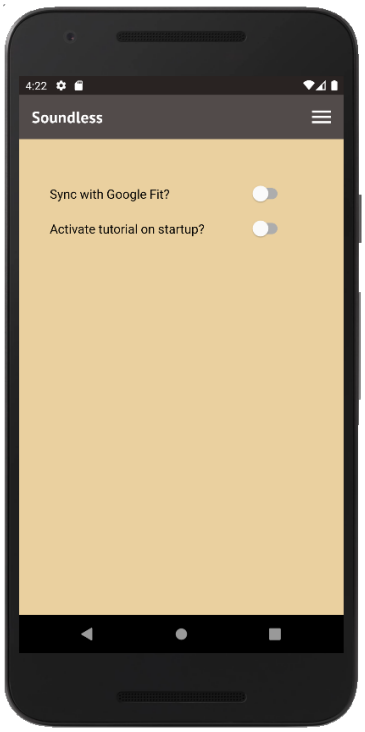
Imatge 5.3.3.8: pantalla informes



Imatges 5.3.3.9 i 5.3.3.10: quadres de diàleg per filtrar informes



Imatge 5.3.3.11: mapa d'informes en l'àrea propera a l'usuari



Imatge 5.3.3.12: pantalla de configuració

Imatge 5.3.3.13: pantalla "Sobre el projecte"

5.3.4 Consultoria d'Experiència d'Usuari realitzada

El novembre de 2021 sol·licito a una professional d'experiència d'usuari amb molts anys d'experiència laboral en disseny de producte i disseny UX/UI[8], professora de la Universitat Oberta de Catalunya, una breu auditoria per determinar en quins aspectes pot millorar el disseny de pantalles i la usabilitat de l'aplicació.

Aquesta trobada resulta molt profitosa i aclareix aspectes que posteriorment es traduiran en tasques específiques:

- Reduir desordre de la pantalla principal
- Substituir icones i modificar posició d'elements
- Millorar la comprensibilitat dels informes amb gradients de color
- Demanar permisos quan l'usuari entra en l'aplicació
- Implementar swipe en les pantalles de tutorial i eliminar possibilitat de saltar-lo
- (altres…)

[8] User eXperience design / User Interface design: Disseny d'experiència d'usuari i d'interfícies d'usuari

6 Implementació

6.1 Implementació de l'arquitectura triada (client)

Mitjançant l'arquitectura Model-View-Intent (MVI):

- Centralitzem **l'execució de tota la lògica** de l'aplicació en una o més classes de tipus ViewModel. En el nostre cas disposem d'una única classe `MainViewModel`, malgrat que es contempla en futures iteracions separar funcionalitats en diferents classes.

  Els ViewModels mantenen un cicle de vida independent al dels elements de la vista.
  - Això facilita, per exemple, que l'usuari pugui estar gravant mentre gira la pantalla i es reinicia el cicle de vida de l'Activity. La gestió dels cicles de vida, en general problemàtica en projectes Android, s'ha vist molt beneficiada per aquesta possibilitat al llarg del treball.

  També disposen de maneres fàcils d'accedir a les mateixes instàncies de ViewModel des de diferents punts del codi per mantenir la consistència respecte als estats de l'aplicació.

  Dins del ViewModel, **la lògica s'estructura en diferents objectes** per tal de facilitar la separació d'interessos i encapsular operacions i dades. Aquests objectes s'estructuren de la manera citada en l'apartat 4.1.Diagrama de classes d'entitat.

- Els elements de la vista (View), com Activities i Fragments, mantenen el mínim de lògica possible i se centren en organitzar el comportament dels elements gràfics, cridar *callbacks* i, especialment, en emetre ***estats***.

- Quan un element de la vista com pot ser un Fragment emet un *estat*, el ViewModel **observa** aquest canvi i executa tota la lògica associada a aquell estat.
  - Per exemple, quan l'usuari clica el botó per gravar, la vista emet l'estat "Gravant" i aleshores el ViewModel s'encarregarà d'inicialitzar objectes, crear processos, sol·licitar l'accés al micròfon, etc. Notem que el ViewModel no sap per què cal gravar ni qui ha emès l'estat corresponent a gravar.

- Igualment, les classes corresponents a la vista **observen** els *estats* que emet el ViewModel.
  - Per exemple, a l'hora de carregar les dades corresponents als informes, el ViewModel emet diversos estats (Carregant / Èxit / Error). Així doncs, quan un Fragment observa un estat d'error, pot imprimir un missatge per pantalla

notificant que hi ha hagut un error. Notem que el Fragment no sap per què s'ha produït l'error ni qui ha emès l'estat d'error.

- La clau d'aquest sistema de comunicació és disposar d'objectes **observables**, i en el nostre cas utilitzem els objectes **LiveData**, elements d'Android Jetpack. Els objectes LiveData contenen un estat (DataState) que és la variable que s'observa i dins l'estat es contenen les dades corresponents a aquell estat, com pot ser un mesurament de decibels. En alguns casos utilitzem subclasses com MutableLiveData i també estenem LiveData per adaptar-lo a les nostres necessitats.

```
open class StatelessLiveEvent<T> : MutableLiveData<T>() {
    private val mPending = AtomicBoolean( initialValue: false)

    @MainThread
    override fun observe(owner: LifecycleOwner, observer: Observer<in T>) {
        // Observe the internal MutableLiveData
        super.observe(owner, {  it: T!
            if (mPending.compareAndSet( expect: true,  update: false)) {
                observer.onChanged(it)
            }
        })
    }

    @MainThread
    override fun setValue(t: T?) {
        mPending.set(true)
        super.setValue(t)
    }

    /**
     * Util function for Void implementations.
     */
    fun call() {
        value = null
    }
}
```

Imatge 6.1.1: Exemple del codi en què estenem LiveData amb una classe pròpia per notificar des del ViewModel alguns esdeveniments

A continuació es mostra la declaració típica d'una variable LiveData extreta del nostre ViewModel:

```
// Variable privada per modificar els continguts dins la classe
private val _packetDataState: MutableLiveData<DataState<DataPacket>>
= MutableLiveData()

// Variable pública per passar a altres classes, de manera que la
puguin observar, garantint alhora que no es modifiquin les dades
val packetDataState: LiveData<DataState<DataPacket>>
      get() = _packetDataState  // Getter personalitzat
```

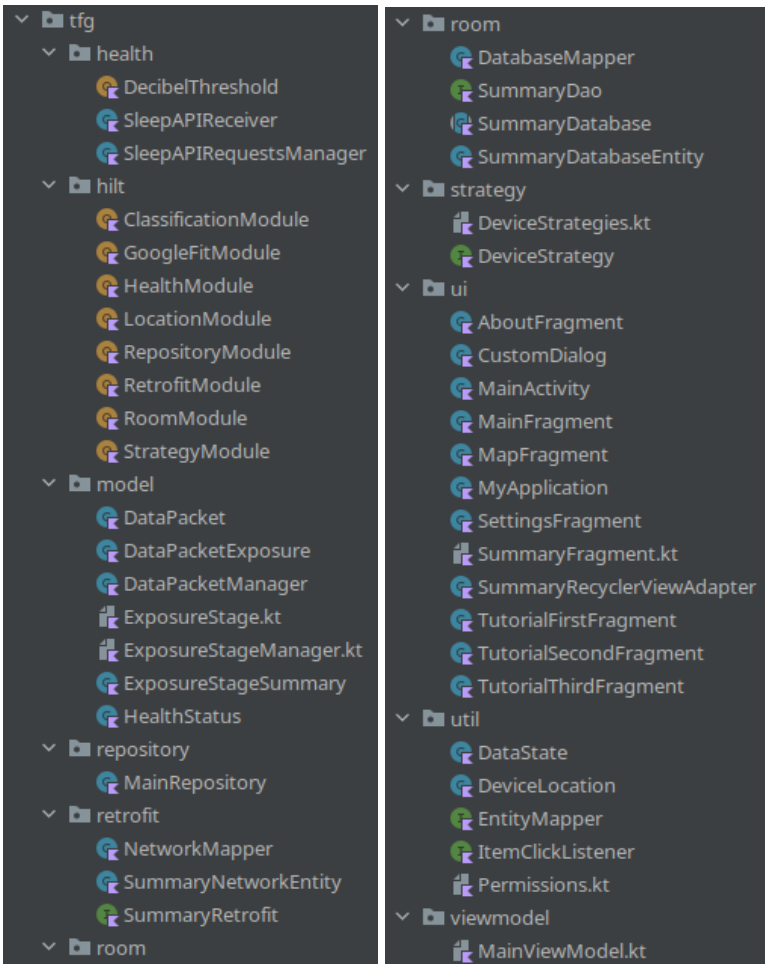
- L'aplicació també centralitza tota la gestió de les dades en una classe Repository (en el nostre cas anomenada `MainRepository`) que serveix com a abstracció per totes les operacions relatives a la persistència de dades:

    - Operacions CRUD sobre la base de dades
    - Serialització
    - Enviament de dades a l'API

    En aquesta part del codi (el Repository) es fa un ús intensiu de les llibreries que es detallen en l'apartat 6.4.Ús de llibreries Hilt, Room i Retrofit (client).

6.2 Estructura de fitxers (client)

En les imatges 6.2.1 i 6.2.2 s'exposen captures de l'entorn de desenvolupament Android Studio per mostrar com s'ha organitzat l'estructura de fitxers.

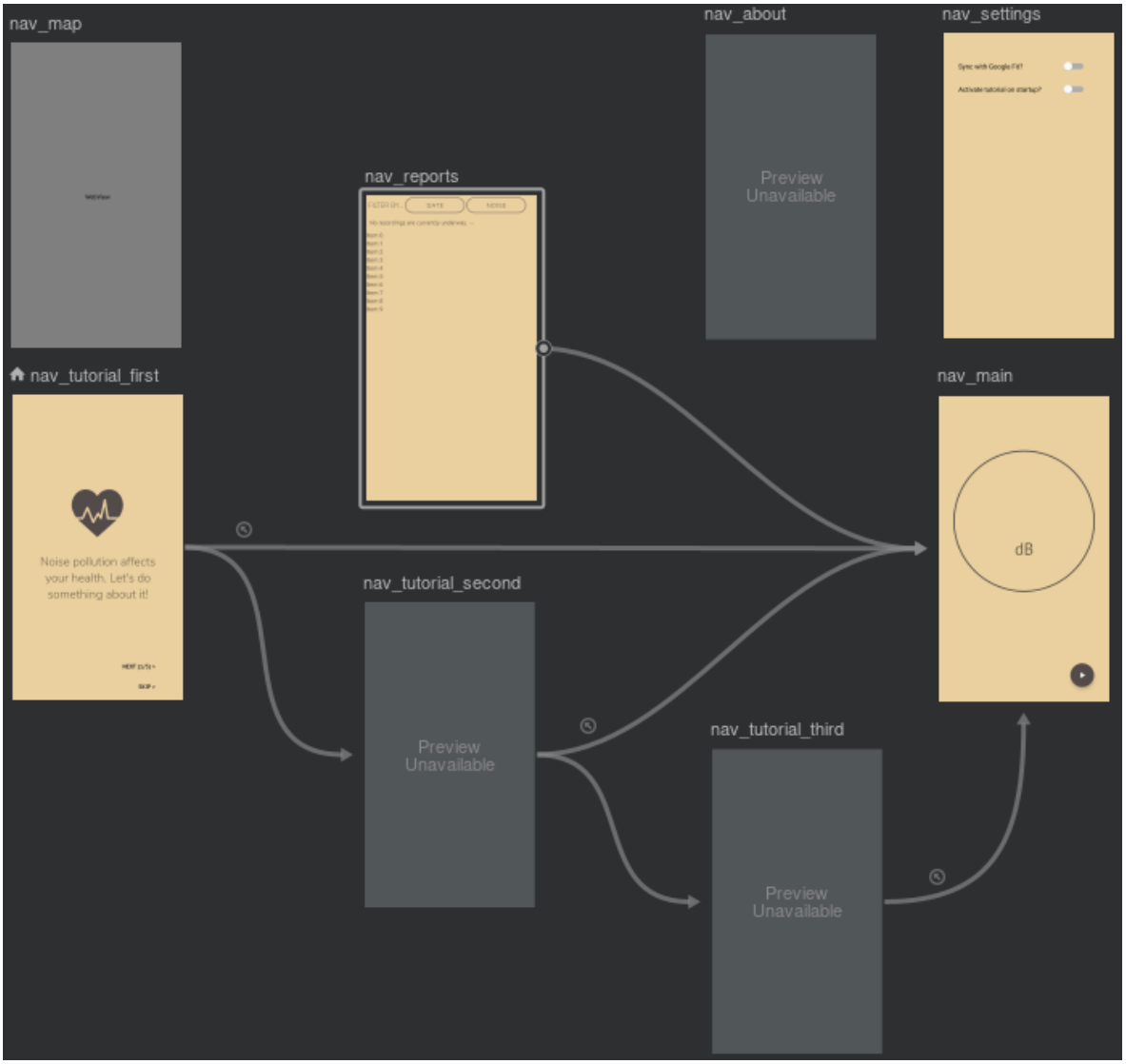


Imatge 6.2.1 (esquerra)
Imatge 6.2.2 (dreta)

6.3 Sistema d'interfície gràfica i sistema de navegació (client)

Pel que fa a les classes que recullen la lògica de la interfície gràfica, s'ha utilitzat una sola Activity compresa de molts Fragments diferents. Un Fragment representa una pantalla de l'aplicació.

Per navegar entre els Fragments s'ha utilitzat una API d'Android anomenada **Navigation Component** que millora la navegabilitat entre components de la vista en comptes d'utilitzar el sistema tradicional d'Intents. La Navigation Component també facilita maneres de dur a terme transicions gràfiques entre pantalles i gestiona internament la *back stack*.



Imatge 6.3.1: graf autogenerat per Android Studio gràcies a Navigation Component per representar els esdeveniments que condueixen d'un Fragment a un altre

6.4 Ús de llibreries Hilt, Room i Retrofit (client)

En el projecte s'han utilitzat diverses llibreries que generen codi en temps de compilació per a tasques concretes i que milloren tant el desenvolupament com l'estructura del codi.

6.4.1 Hilt (injecció de dependències)

La injecció de dependències permet centralitzar la instanciació de variables per utilitzar-les a tot arreu de la mateixa manera, reduint la duplicació de codi, entre molts altres avantatges. Hilt està construït sobre la llibreria Dagger d'injecció de dependències i n'és l'alternativa més moderna. La llibreria s'utilitza en molts punts, i en les imatges 6.4.1.1 i 6.4.1.2 en podem veure un exemple.

```
@Module
@InstallIn(SingletonComponent::class)
object ClassificationModule {

    private const val MODEL_FILE = "yamnet.tflite"

    @Singleton
    @Provides
    fun provideAudioClassifier(@ApplicationContext context: Context): AudioClassifier {
        return AudioClassifier.createFromFile(context, MODEL_FILE)
    }

}
```

```
class DataPacketManager @Inject constructor(
    private val exposureStageManager: ExposureStageManager
) {
    // AudioRecord variables
    @Inject lateinit var audioClassifier: AudioClassifier
```

Imatge 6.4.1.1 (superior): objecte Hilt per instanciar i injectar instàncies d'AudioClassifier
Imatge 6.4.1.2 (inferior): classe que rep la dependència injectada

6.4.2 Room (persistència a SQLite en el mòbil)

La llibreria Room s'utilitza per desar les dades en una base de dades SQLite local. La llibreria ens permet generar codi automàticament mitjançant l'ús d'anotacions per a classes, interfícies i DAOs.

6.4.3 Retrofit (enviament de dades serialitzades a l'API)

La llibreria Room s'utilitza per serialitzar i enviar les dades a l'API. Ens permet generar codi automàticament mitjançant l'ús d'anotacions per a classes, interfícies i funcions per a l'ús de mètodes HTTP. La serialització es fa utilitzant KotlinX Serialization.

6.5 Recol·lecció de dades i paral·lelisme (client)

El procés de recol·lecció de dades és periòdic, és a dir, cada X segons es realitza una lectura (mentre la gravació està activa). Per defecte, aquesta freqüència és d'1 segon, però la xifra es pot modificar.

A l'hora de recollir les dades durant una gravació, hem de saber:

- Com captem els decibels a temps real: annex 1
- Com i quan classifiquem els sons a temps real: annex 2

Inicialment, es preveia iniciar tot de processos en paral·lel durant el procés de gravació, fent ús de les corutines de Kotlin, threads lleugers d'execució amb una empremta baixa de recursos. Per tant, quan s'iniciava una gravació, s'havien d'iniciar:

- Corutina d'obtenció de decibels a temps real
- Corutina de classificació de sons a temps real
- Corutina de captació de dades de salut a temps real
- Corutina de construcció de paquets de dades (que va llegint les dades de les corutines anteriors a través de **channels**, objectes que permeten la comunicació entre fils d'execució en una lògica productor-consumidor, i els compila en una unitat atòmica anomenada **DataPacket**)
- Corutina principal (que rep els DataPackets de la corutina de construcció de paquets a través de channels i els gestiona, per exemple, detectant si s'entra o se surt d'etapes d'exposició, descartant o desant aquestes dades, etc.)

Malgrat tot, el plantejament s'ha vist afectat per:

- Obligatorietat de captar decibels i classificar en la mateixa corutina, per raons exposades en l'annex 2
- No possibilitat de captar dades de salut a temps real

De manera que, finalment, s'executen:

- Corutina d'obtenció de decibels i classificació a temps real
  (context d'execució de la corutina: `kotlinx.coroutines.Dispatchers.Default`*)*
- Corutina de construcció de paquets de dades
  (context d'execució de la corutina: `kotlinx.coroutines.Dispatchers.Default`*)*
- Corutina principal
  (context d'execució de la corutina: `kotlinx.coroutines.Dispatchers.IO`*)*

Que segueixen la mateixa lògica que en el plantejament inicial i es comuniquen a través de channels (canals) de Kotlin. En la corutina d'obtenció de decibels i classificació, les dues tasques s'alternen mitjançant un channel de tipus "ticker" que determina en quin moment una tasca ha de deixar pas a l'altra i viceversa.

En la corutina d'obtenció de decibels i classificació també es duu a terme la traducció de les etiquetes de so a l'idioma del dispositiu mitjançant la **Translation API de MLKit**, que descarrega un model Machine Learning a través de la xarxa per tal de poder adaptar la traducció a temps real en tots els idiomes (altrament hauríem de tenir un diccionari per les 521 categories en tots els idiomes possibles).

```
CoroutineScope(Default).launch {  this: CoroutineScope
    // Preparing translation API
    var translate = true
    val sourceLanguage = TranslateLanguage.ENGLISH
    val targetLanguage = Locale.getDefault().language
    if (sourceLanguage == targetLanguage) translate = false

    if (translate) {
        val options = TranslatorOptions.Builder()
            .setSourceLanguage(sourceLanguage)
            .setTargetLanguage(targetLanguage)
            .build()
        translator = Translation.getClient(options)
        val conditions = DownloadConditions.Builder()
            .build()

        var blocked = true
        translator.downloadModelIfNeeded(conditions)
            .addOnSuccessListener {  it: Void!
                blocked = false
                translate = true
            }
            .addOnFailureListener {  it: Exception
                blocked = false
                translate = false
            }
```

```
if (translate) {
    translationTask = translator.translate(mostRelevantLabel)
}
classification =
    if (translate and ::translationTask.isInitialized) {
        while (!translationTask.isComplete and !translationTask.isCanceled) delay( timeMillis: 50)
        if (translationTask.isSuccessful) {
            translationTask.result
        } else {
            mostRelevantLabel
        }
    } else {
        mostRelevantLabel
    }
```

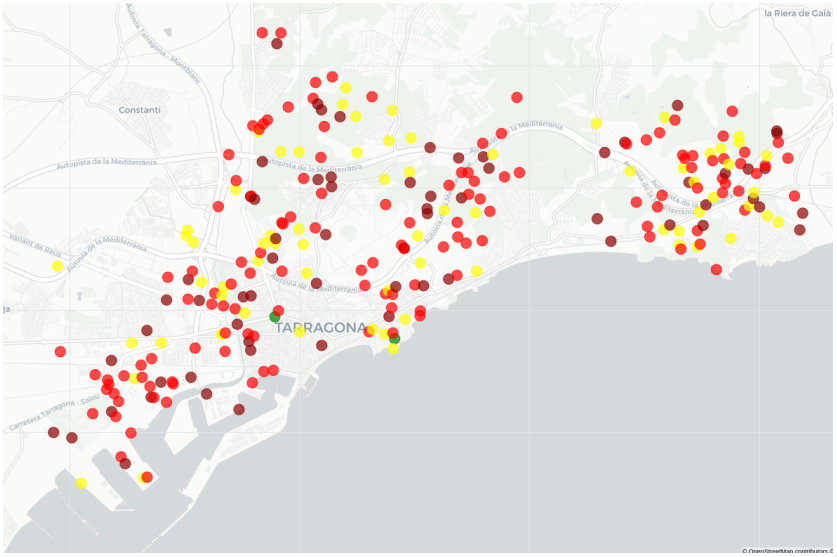
Imatge 6.5.1 (superior) i Imatge 6.5.2 (inferior): captures del codi en què es mostra com es fa la traducció d'etiquetes

6.6 Estructura de l'API (servidor)

En el desenvolupament del Treball Final de Grau he creat una API que segueix l'estàndard REST per a les operacions CRUD[9], fent servir Django i Django REST Framework. Personalment ja havia utilitzat aquestes eines anteriorment en projectes propis i projectes del grau. Els *endpoints* segueixen l'estructura habitual generada per Django Rest Framework i s'han fet servir ViewSets.

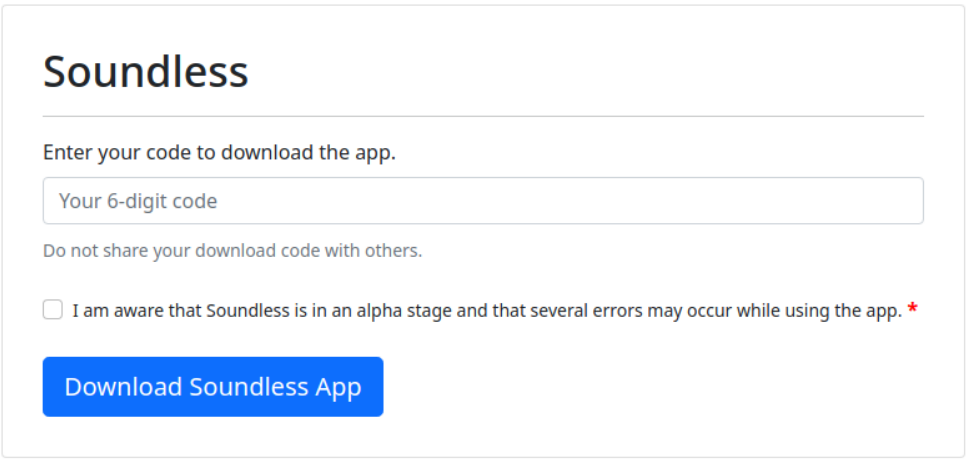
També s'han definit altres enllaços a Django per a altres funcions que generen pàgines HTML (a diferència dels *endpoints* REST, que retornen objectes JSON):

1. Mapa interactiu (zoom, punts clicables, etc) generat amb la llibreria Bokeh que s'incrusta com a WebView al client: /



Imatge 6.6.1: captura del mapa generat pel servidor

2. Pàgina de descàrrega de l'aplicació (.apk[10] signada): /download/

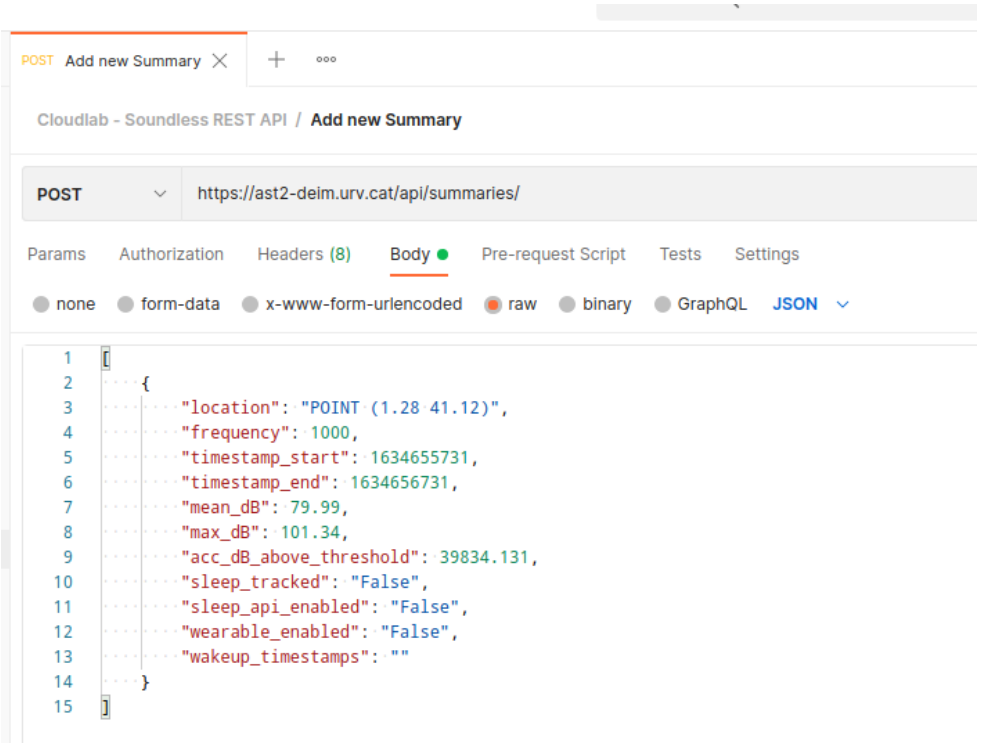


Imatge 6.6.2: captura del formulari per descarregar l'aplicació

[9] Create, Read, Update, Delete (Crear, Llegir, Actualitzar i Esborrar).

[10] Android Package, format de fitxer per la distribució d'aplicacions Android. A partir d'agost de 2021, Google Play requereix migrar els .apk a un format diferent per publicar aplicacions, anomenat Android App Bundle.

Al llarg de tot el procés s'han anat provant totes les funcionalitats de l'aplicació Django mitjançant l'eina Postman tal i com es menciona en l'apartat 7.Avaluació.



*Imatge 6.6.3: exemple de prova d'*endpoint *de l'API REST en el client de Postman*

6.7 Gestió del servidor

Aquest apartat menciona aspectes breus sobre la gestió de la infraestructura.

6.7.1 Connexió amb el servidor

La connexió amb el servidor s'ha fet en tot moment via SSH[11]. Gràcies a extensions dels editors de codi també s'ha dut a terme una part important del desenvolupament de l'API a través de SSH tot utilitzant l'editor de codi en local, cosa que ens ha permès una bona experiència de desenvolupament.

6.7.2 Com servim l'API

L'API Django se serveix amb:

- Guincorn com a servidor WSGI[12]
- Nginx com a *proxy* invers

S'ha treballat en la configuració i la seguretat del servidor per tal que fos *production-ready*, si bé les capacitats de l'alumne en matèria de seguretat i gestió de servidors són limitades i s'ha tractat d'arribar a un límit acceptable.

[11] Secure SHell, programa per accedir a màquines en remot.
[12] Web Server Gateway Interface, especificació de servidor web.

6.7.3 Obtenció de certificats de seguretat

Per compatibilitat amb la llibreria Retrofit d'Android, que obliga a realitzar connexions a través d'HTTPS[13], entre moltes d'altres raons, ha estat necessari implementar un certificat de seguretat al servidor.

A través del servei tècnic de la universitat s'ha sol·licitat un certificat TLS[14] i s'ha instal·lat al servidor amb Nginx. El certificat està signat per a la Universitat Rovira i Virgili com a organització.

Imatge 6.7.3.1: informació del certificat

6.7.4 Autorització de comptes a través de Google Cloud OAuth

Per tal que els usuaris puguin connectar el seu compte de Google amb Soundless, s'ha creat un projecte de Google Cloud i s'utilitza el seu sistema OAuth per autoritzar comptes específics a utilitzar l'aplicació. Fins ara, només s'ha utilitzat un sol correu de proves.

[13] HyperText Transfer Protocol Secure.

[14] Transport Layer Security, protocol de seguretat de la capa de transport per garantir la transmissió encriptada dels continguts web.

7 Avaluació

Malgrat que l'aplicació ha passat per diverses etapes de proves, l'avaluació del funcionament de l'aplicació (entès de manera estructurada, repetible i parametritzable) és el que menys ha evolucionat durant el projecte i que té un marge més gran de millora.

En part, això es deu a la necessitat d'avaluar les funcions de captació de so i de classificació en dispositius físics reals i manualment, per tal de 1) comparar les mètriques de decibels amb un lector de decibels calibrat, 2) estudiar les incompatibilitats o problemes del maquinari, 3) comprovar el rendiment del dispositiu executant tasques potents com la classificació+traducció a temps real; unes funcions que no es poden dur a terme, per exemple, des d'un emulador.

Així doncs, moltes tasques de proves s'han dut a terme de manera manual.

El grup de recerca Cloudlab ha adquirit per a Soundless diversos dispositius mòbils des dels quals s'han pogut executar proves, llistats en la taula 7.1.

Mòbil	Descripció
Xiaomi Mi A3	Model estàndard - dispositiu de proves de referència
Alcatel 1SE	Mòbil de baixes prestacions, amb versió antiga d'Android no actualitzable, per provar l'app en terminals de baix rendiment
Samsung Galaxy A52s	Mòbil Samsung de gamma mitjana-alta
Samsung Galaxy A02s	Mòbil Samsung de gamma mitjana-baixa

Taula 7.1

Entre d'altres factors, com l'adopció en el mercat de les marques, s'han tingut en compte les recomanacions de portals com *Don't Kill My App!* [13] a l'hora d'adquirir els telèfons, per provar funcions en background en marques amb polítiques agressives d'estalvi de bateria que suposessin un repte per les funcionalitats de l'aplicació.

També s'ha provat l'aplicació en un mòbil personal de l'alumne.

Xiaomi POCO F3	Mòbil personal dirigit especialment a tests per mòbils de pantalla gran

En la taula 7.2 s'exposen diverses de les proves d'avaluació que s'han dut a terme.

Concepte	Accions	Eina utilitzada	Resum	Futures accions
Proves de components del codi	Tests unitaris a les funcions del ViewModel (lògica)	JUnit (*framework*)	Funcionament correcte dels tests. Implementació insuficient de tests al llarg de la 2a meitat del Treball. Aspecte a millorar.	Reforçar code coverage, automatitzar l'execució de tests mitjançant CD/CI[15]
Funcionament de tots els elements de la interfície gràfica	Testeig manual utilitzant dispositius físics. Proves en modes landscape/portrait, amb/sense barra de tasques, menú lateral, elements clicables, funcionament notificacions i quadres de diàleg, etc. en totes les pantalles	Mòbils Android	Errors en la interfície gràfica que s'han anat depurant iterativament.	Generar centenars d'interaccions pseudo aleatòries amb l'eina Monkey. Crear tests d'instrumentació amb l'eina Espresso
Funcionament de la captació de decibels	Validesa dels decibels gravats	Mesurador de decibels SPL calibrat Wensn WS1361. Exemple de les proves: annex 3	Es descobreix que alguns mòbils antics no podran executar la captació de dB. S'identifica que els micròfons valoren excessivament alguns sons.	Extreure arxius de decibels de les gravacions i comparar-los programàticament
Funcionament de la classificació	Anàlisi del temps que es triga en iniciar una classificació amb AudioClassifier.	Mòbils Android	Es decideix descartar etiquetes de so durant un període inicial	-

[15] Continuous Integration / Continuous Delivery (Integració Contínua / Desplegament Continu)

	Anàlisi de la correcció de les etiquetes de so		de la gravació i es fa una tria de per reduir el pes de les etiquetes “silenci”, que són massa predominants.	
Proves automatitza des dels *endpoints* de l’API	Ús de client HTTP per fer peticions automatitzades GET, POST, PUT, DELETE i pings al servidor	Postman	Molt útil per descobrir problemes en la serialització i adaptació entre els models de Retrofit i de Django, abans de programar en el client i depurar.	-
Avaluació de la funcionalitat de connexió amb Google Fit	Obtenció de dades amb polsera Xiaomi Mi 6 adquirida pel grup de recerca, connexió app Mi Fit - Google Fit - Soundless, *logging* de les dades a Soundless i comparativa amb dades a Mi Fit	*Logging* d’Android Studio	Dades reals i vàlides.	Implementació funció de correlació de les dades de salut amb dades acústiques

Taula 7.2

8 Conclusions

La plataforma Soundless neix amb l'objectiu d'auditar els efectes de la contaminació acústica en les persones. Neix com a una eina col·laborativa, gratuïta i desinteressada per ajudar les persones a combatre un problema social amb repercussions sobre la salut.

Per adhesió a aquests valors i als objectius de la plataforma, ha estat per mi una satisfacció assolir uns resultats que considero esperançadors. Soundless evoluciona favorablement i es pot seguir plantejant moltes fites en un futur.

En cinc mesos, Soundless ha passat de ser una idea indefinida sobre el paper a estar en una fase sòlida de desenvolupament i acomplir molts dels objectius originals. A efectes pràctics, la plataforma encara no ha vist la llum ni s'ha pogut fer arribar a mans d'usuaris reals que contribueixin a la plataforma amb les seves dades, però s'està a prop d'arribar a aquest punt. El grup de recerca Cloudlab ha decidit seguir apostant pel desenvolupament de Soundless en els propers mesos i seguir oferint-me una contractació per poder-ho realitzar.

El prototip implementat i entregat en finalitzar el Treball Final de Grau:

- Grava correctament decibels SPL
- Realitza una classificació del so entrant per micròfon que s'alterna amb la lectura de decibels, i obté etiquetes de tipus de so
- Tradueix les etiquetes de so a temps real
- Obté dades de comptes de Google Fit
- Detecta i genera etapes d'exposició al soroll en funció d'un algorisme implementat
- Pot gravar durant períodes prolongats; no produeix errors greus en temps d'execució en els dispositius de prova
- Resumeix, serialitza, envia i desa les dades correctament
- Actualitza i mostra correctament les dades per pantalla; disposa d'una interfície gràfica funcional, en 3 idiomes (català, castellà i anglès) i en formats *portrait* i *landscape*
- Genera informes de dades de gravació; genera mapes de soroll col·lectius en la banda del servidor i els mostra com a WebViews en el client
- Es basa en arquitectures i patrons de software robustos, i el codi està documentat
- Disposa d'una infraestructura adequada per producció de la banda del servidor
- L'aplicació és descarregable i instal·lable en mòbils Android

En canvi, no s'ha aconseguit encara:

- Implementar la funció de correlació de dades de salut
- Compatibilitzar l'aplicació amb dispositius Android antics
- Implementar filtres a les ones de so i en general millorar la qualitat de la captació de decibels
- Mostrar mapes de calor (*heatmaps*) de contaminació acústica i filtres per a mapes

- Classificar mentre es capten decibels en paral·lel
- Fer públic el prototip a Google Play en format App Bundle
- Captar les dades en *background* mentre es faci ús d'altres aplicacions

S'han albirat també nous objectius en què s'està començant a treballar poc a poc:

- Transformació dels processos de generació de mapes de la banda del servidor (que actualment es realitza sota demanda per cada petició) a una arquitectura de generació de pàgines estàtiques que s'emmagatzemin en un proveïdor Cloud Object Storage, a fi de blindar la seguretat i augmentar el rendiment
- Implementació de la xarxa neuronal de l'alumne de màster ECiM en l'aplicació
- Preparació de tasques d'anàlisi de dades generades per l'aplicació mitjançant el *framework* Lithops, desenvolupat pel grup de recerca Cloudlab

Com a alumne del Treball Final de Grau, l'aprenentatge que he rebut en aquests mesos ha estat molt enriquidor. La taula 8.1 mostra un llistat de tecnologies, eines i tècniques en què he percebut una millora substancial dels meus coneixements i que m'han servit per incrementar la meva comprensió general del desenvolupament de software.

Taula d'evolució de l'aprenentatge al llarg del Treball Final de Grau		
Tecnologia	Coneixements previs al TFG	Resultat d'aprenentatge del TFG
Llenguatge de programació Kotlin	Cap	Desenvolupada una base de codi 100% Kotlin (vegeu annex 4) per al client Android. Aprenentatge sobre funcionalitats avançades de Kotlin, paral·lelisme i comunicació entre corutines, immutabilitat, programació funcional, injecció de dependències, serialització…
Desenvolupament Android	Coneixements intermedis adquirits en 2 assignatures del grau	Bones pràctiques, arquitectura MVI/MVVM, llibreries Jetpack, permisos, signatura i publicació d'apps, navegació i transicions, problemes en cicles de vida dels components, *LiveData*, Material Design 2…
Deployment d'aplicacions Django	Mai dut a terme en servidors reals	Coneixements sobre configuració de servidors WSGI (Gunicorn) i proxy (Nginx), configuració de Django i Django REST Framework per a producció, implementació de seguretat bàsica i certificats HTTPS…
Machine Learning	Conceptes essencials i execució de models simples en local	Com utilitzar el *runtime* Tensorflow Lite en un projecte

		real, com fer ús de models a Android, novetats i evolució d'Edge Machine Learning, tipologies de *clustering* i de *labeling*…
Tractament de dades geospacials en aplicacions web	Cap	Com i per què implementar les extensions PostGIS en la base de dades i GeoDjango per models i ORM; llibreria Bokeh per generació de mapes interactius en format JSON, nous formats de codificació de dades geospacials…
Tractament de dades de so	Cap	Conceptes d'encodings, mètriques i escales d'intensitat de so, formats de fitxers, APIs de gravació de so per a Android, projectes similars, límits de tolerabilitat i normativa, calibració de maquinari, reptes i complexitat…
Fase prèvia del Treball Final de Grau		
Raspberry Pi	Cap	Sistemes operatius per a microcontroladors, *suite* d'eines Balena, models de plaques, com soldar maquinari amb cablejat en la placa, llibreries de codi obert per llegir dades del maquinari, execució de Python 3 i Lithops en entorns de poca potència, escriptura de fitxers en un Cloud Object Storage…
Tecnologies Cloud	Serverless - FaaS, Cloud Object Storage, IBM Cloud, Google Cloud, *framework* Lithops, conceptes Kubernetes…	Aprofundiment en els coneixements i posada en pràctica
Anàlisi de dades geospacials	Pandas, GeoPandas, *framework* Lithops, formats de fitxers, dades Sentinel, *open data*...	Aprofundiment en els coneixements i posada en pràctica

Taula 8.1

És sobretot per aquest aprenentatge que agraeixo l'experiència que m'ha proporcionat aquest treball i el suport proporcionat el grup de recerca Cloudlab i, més en general, la Universitat Rovira i Virgili. El Treball Final de Grau ha estat una oportunitat

immillorable per posar en pràctica les meves habilitats i entrar al món professional amb decisió i entusiasme.

Annexos

Annex 1: obtenció de decibels

Nota: Es recomana llegir prèviament l'apartat 2.3.2.Sobre la mesura de decibels que tracta aspectes sobre la complexitat de la medició de decibels. Un dels reptes que s'hi menciona és el propi càlcul de decibels, que és del que tracta aquest apartat.

La obtenció de decibels ha estat present durant moltes setmanes en tot el treball afectant la viabilitat de la tria de plataforma i la viabilitat de l'aplicació. Durant un llarg període de temps es va considerar la possibilitat que no hi hagués manera factible de captar decibels. Afortunadament, de manera conjunta amb l'alumne de màster ECiM es va trobar un sistema per calcular aquests decibels SPL[16] de manera consistent, si bé no rigorosa. La següent descripció conté imprecisions i no s'ha de considerar com una referència.

Els decibels són una unitat comparativa que es calcula en base a un valor de referència. El càlcul de decibels es basa en la fórmula descrita en la imatge A1.1. La línia superior detalla el càlcul que es realitzaria en cas que disposéssim de potència en Watts mentre que la línia inferior correspon al mateix càlcul per al voltatge o el corrent. Notem que la 2a línia es dedueix del fet que $P = V^2/R$ i $P = I^2R$.

$$dB = 10\log\left(\frac{power}{reference\ power}\right)$$

$$dB = 20\log\left(\frac{voltage}{reference\ voltage}\right) = 20\log\left(\frac{current}{reference\ current}\right)$$

Imatge A1.1

Per obtenir les mètriques que apareixen en la fórmula i que ens permeten calcular els decibels, llegirem el so provinent del micròfon a través de la classe MediaRecorder (Android API 1). Un cop inicialitzada la instància de MediaRecorder i començada la gravació, fem una crida al mètode `MediaRecorder.getMaxAmplitude()`, que calcula la màxima amplitud (és a dir, la màxima intensitat de so) de l'ona sonora *des de la darrera crida que s'hagi fet al mètode*, de manera que, si cridem el mètode cada segon, MediaRecorder ens donarà el valor màxim de la pressió de so en cada segon.

El mètode retorna valors entre 0 i 32.767 (rang de valors de 16 bits sense signe), que representen una conversió lineal dels Pascals (pressió de so) que entren pel micròfon a ona digital, en una escala en relació amb el valor màxim del corrent que pot circular pel micròfon. Aquests valors no corresponen a cap unitat estàndard i el valor es pot veure alterat entre dos dispositius diferents.

[16] Sound Pressure Level, o nivell de pressió de l'ona sonora en la seva propagació per l'aire.

A partir d'aquí s'empra la 2a fórmula de la imatge A1.1 (ja que disposem de valors de corrent elèctric) comparant-lo amb un valor estàndard de referència. El resultat són decibels en una escala dBFS, és a dir, la magnitud digital de la intensitat de so en base al valor màxim de referència digital.

(Kotlin)

```
val maxAmplitude = mediaRecorder.maxAmplitude
val dBFS: Float = 20 * log10(maxAmplitude / 255.0F)
```

Aleshores, es fa una traducció dels decibels a escala completa (dBFS) a una estimació dels decibels SPL mitjançant un offset de calibració. Aquest offset l'obtenim amb `MicrophoneInfo.getSensitivity()`, un mètode que ens permet accedir a la sensibilitat del micròfon i obtenir el nivell en dBFS produït per un to de 1000 Hz **a 94 dB** SPL. La xifra és representativa de la quantitat de voltatge elèctric que genera el micròfon en base a aquest *input* de pressió de so i, per tant, ens serveix per conèixer quants dBFS s'obtindrien en cas que s'haguessin escoltat a través del micròfon aquests 94 dB SPL. Aquesta xifra ens serveix per fer el càlcul invers i calibrar els càlculs, de manera diferent a cada dispositiu i cada micròfon, per deduir una estimació dels dB SPL que s'haurien produït donats els dBFS de què disposàvem.

La sensitivitat acostuma a ser un valor negatiu i en el dispositiu principal de proves el seu valor és de -37.

(Kotlin)

```
val dBSPL: Float = 94.0F + dBFS +
mediaRecorder.activeMicrophones[0].sensitivity    // offset
```

El mètode `getSensitivity()`:

- Es va afegir a la classe MicrophoneInfo en l'API 28 d'Android, per això necessita almenys que el dispositiu tingui Android 9. Això afecta els requeriments de minSdk de l'aplicació durant el treball.
- Depèn que el dispositiu estigui adaptat per llegir aquestes dades, és a dir, depèn del maquinari. En algun cas identificat en la fase de prova, el mètode retorna `MicrophoneInfo.SENSITIVITY_UNKNOWN` i per tant no es poden calcular els decibels. Això, però, ha de succeir només en casos aïllats.

El sistema no estima els dB SPL amb una gran precisió, però sí que ho fa amb un marge de tolerabilitat de +-5dB en comparació amb un dispositiu calibrat de detecció de decibels SPL, cosa que compleix amb els estàndards de qualitat fixats per al prototip. (Vegeu annex 3)

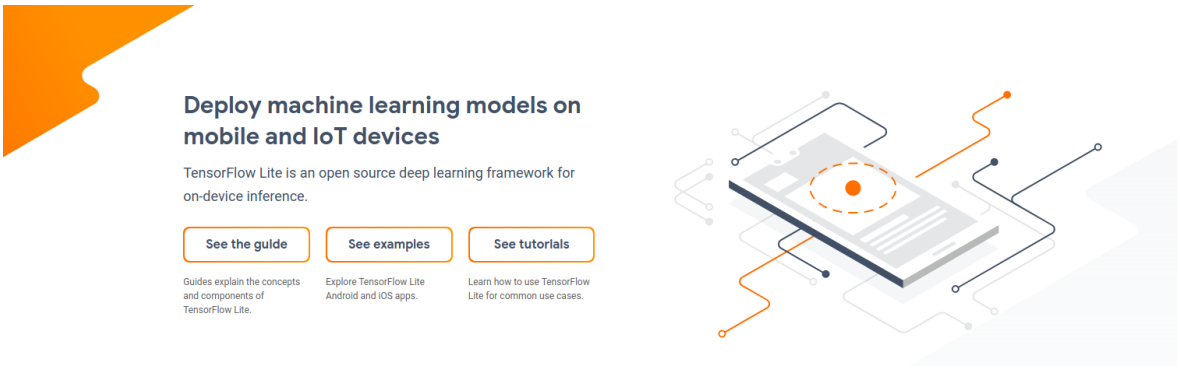
Aplicar filtres sobre les dades

Les característiques de l'aplicació farien molt recomanable aplicar un filtre A-weighting ja que aleshores les dades estarien en una escala adaptada especialment a l'oïda humana. Això, però, no ha estat possible fins al moment perquè MediaRecorder no

permet l'accés a les dades "en brut" i, en conseqüència, no s'ha trobat la manera d'implementar aquest filtre.

Annex 2: classificació i estratègia per compaginar amb la lectura de decibels

La classificació de decibels ha estat present en el projecte des de l'inici gràcies a la presència d'un *runtime* anomenat TensorFlow Lite i que permet executar tasques Machine Learning a Android i altres plataformes amb una empremta baixa de recursos.



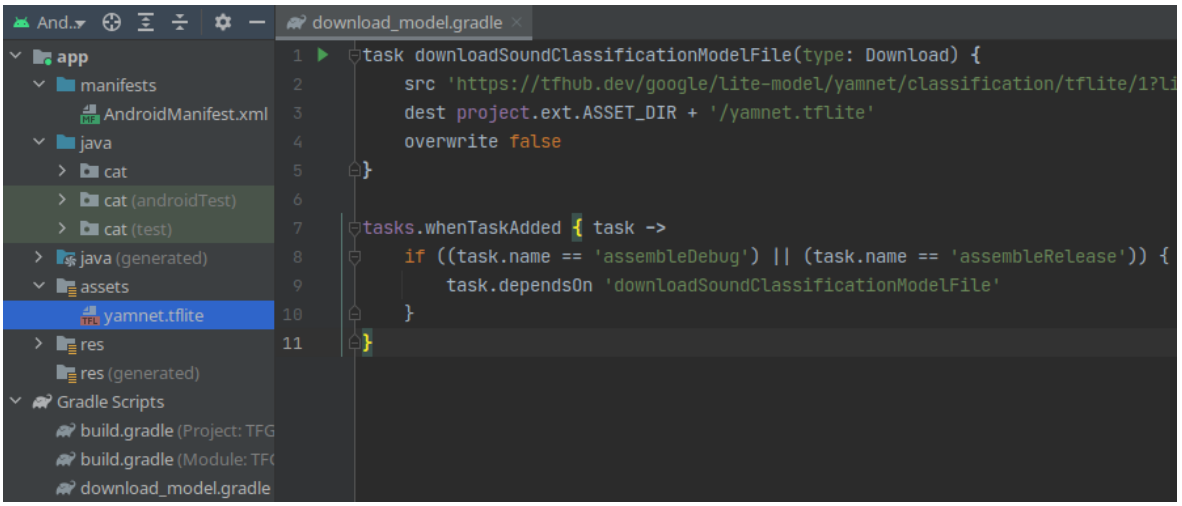
Imatge A2.1: pàgina principal de TensorFlow Lite. Font: [14]

TensorFlow Lite proporciona una API anomenada AudioClassifier, part de la llibreria Task Library, que permet executar classificacions de so a temps real a través d'ones de so provinents de micròfon. És a dir: donada una ona de so, AudioClassifier identifica patrons en l'ona per tal de reconèixer si el so que l'ha originada prové d'una font o una altra. Per tal d'aconseguir-ho, AudioClassifier ha de fer ús d'un **model** estadístic que hagi estat entrenat amb moltes dades d'àudio prèviament etiquetades. Els models per a TensorFlow Lite són fitxers amb l'extensió **.tflite**.

L'API proporciona eines per emprar models propis, però l'equip de TensorFlow també facilita un model generalista anomenat YAMNet en format compatible amb TensorFlow Lite. D'aquesta manera, és fàcil adequar el nostre cas d'ús a aquest model altament optimitzat **sense** haver de recórrer a:

1. Generar un *dataset* propi.
2. Entrenar i optimitzar el model.

El model YAMNet classifica sons entre 521 categories [15] i és, per tant, prou complet per identificar tots els sons que entrin per micròfon. Com a contrapartida, no tenim la capacitat de personalitzar la nostra classificació, per exemple, per especialitzar-la en un conjunt més acotat i controlat de categories. (Potser diferenciar entre "Canari" i "Lloro" ens és indiferent, però podríem necessitar més precisió per diferenciar "Cotxe" de "Tren").



Imatge A2.2: descarreguem YAMNet a través de la xarxa mitjançant un script *de Gradle. El model queda guardat en la carpeta "assets".*

Malgrat que les eines que TensorFlow ens posa a l'abast són molt útils, la classificació de sons s'ha de compaginar també amb la captació de decibels, implementada amb la classe MediaRecorder (v. annex 1). **Aquí és on trobem dificultats substancials.**

L'API AudioClassifier de TensorFlow Lite és *utilitzable* en els següents casos:

1. Si es realitza la classificació mitjançant un objecte AudioRecord (API d'Android) que accedeixi al micròfon a temps real i n'extregui les dades de so. Però ens trobem que:

   a) Un objecte AudioRecord i un objecte MediaRecorder (que s'utilitza en la captació de dB) no poden accedir al micròfon al mateix temps.

      a1) Una alternativa seria fer servir AudioRecord tant per la captació de decibels com per la classificació, i cridar les funcions del mateix objecte per les dues coses. Però, en aquest cas, caldria implementar un mètode propi `.getMaxAmplitude()` per a AudioRecord que doni uns resultats iguals als que dóna `MediaRecorder.getMaxAmplitude()`. Aquesta opció no s'ha pogut tirar endavant (tot i intentar-ho de manera conjunta amb l'alumne de màster ECiM) degut a la poca interpretabilitat de les dades en brut donades per AudioRecord.

   b) Tampoc hi ha cap manera de forçar que, internament, MediaRecorder utilitzi un objecte AudioRecord per a la recol·lecció de dades ni viceversa. Ambdues APIs són independents i "incompatibles".

2. Si es proporcionen dades de so al classificador en format `AudioRecord.ENCODING_PCM_FLOAT`, no necessàriament a temps real.

Una possibilitat seria que MediaRecorder desi el so guardat en un fitxer o en una "pipe" d'Unix en format `AudioRecord.ENCODING_PCM_FLOAT`, que consumeixi l'AudioClassifier per dur a terme la classificació. Totes dues opcions s'han intentat, però ens trobem amb l'inconvenient que MediaRecorder fa servir un encoding i *a més fa servir un format de fitxer per comprimir i estructurar les dades*, una consideració que fa -possiblement sumat a altres factors- que AudioClassifier no sigui capaç d'entendre aquestes dades. No s'ha trobat la manera d'impedir que MediaRecorder dugui a terme aquest processament. Una solució seria utilitzar APIs per descomprimir els fitxers i revertir les dades al format original però no s'ha trobat la manera de fer-ho.

Per les raons exposades, la classificació i la captació de decibels no es poden dur a terme a temps real al mateix temps ni a través de fitxers (de moment, amb els coneixements / eines actuals).

Per tal d'entendre millor el problema, mostrarem en els següents diagrames les diverses opcions que s'han anat estudiant:

Opció 1 (diagrama A2.3)

No factible perquè només pot accedir al micròfon un dels processos.

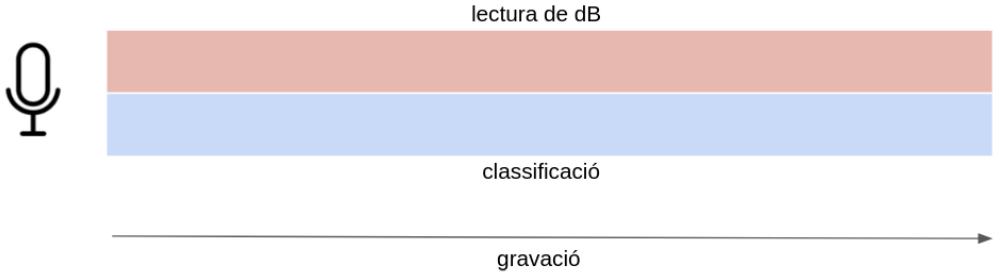


Diagrama A2.3

Opció 2 (diagrama A2.4)

No factible perquè viola els requisits de seguretat i privacitat de les dades.

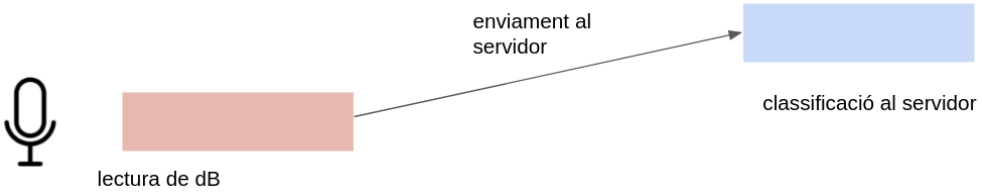


Diagrama A2.4

Opció 3 (diagrama A2.5)

No factible perquè AudioClassifier no entén l'output de MediaRecorder.

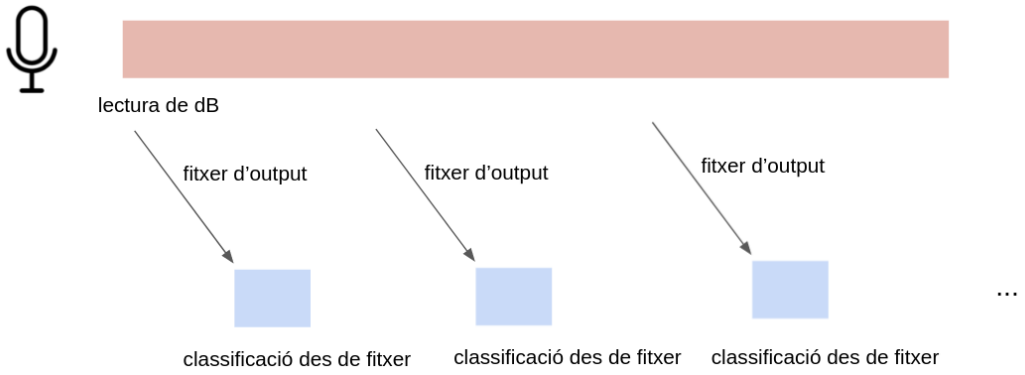


Diagrama A2.5

Opció 4 (diagrama A2.6)

Ídem a Opció 3.

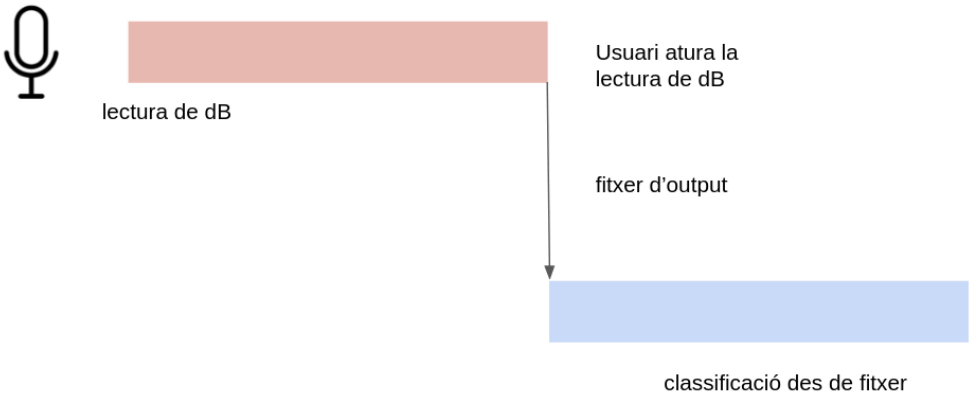


Diagrama A2.6

Opció 5 (diagrama A2.7)

Factible. MediaRecorder i AudioRecord s'anirien alternant per l'ús del micròfon. D'aquesta manera, la lectura de dB es pot fer al llarg d'un període i en acabat MediaRecorder cedeix el seu accés al micròfon a AudioRecord perquè classifiqui quin so està causant els dB (trànsit, persones, animals, trens…).

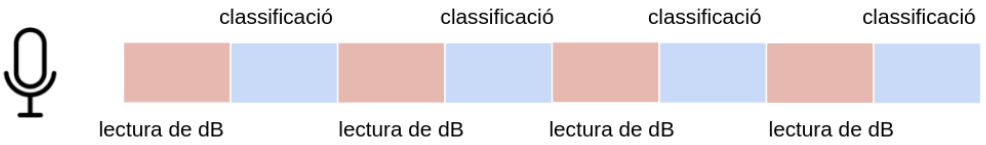


Diagrama A2.7

Finalment, doncs, **s'ha implementat l'opció 5**. El codi resultant és funcional i s'obtenen etiquetes de so (exemple: "sons de teclat", "explosió", "discurs") que després es passen per una API de traducció amb la llibreria MLKit per obtenir una etiqueta en l'idioma del dispositiu.

Clarament aquesta solució té l'inconvenient que no es reben dades cada instant sinó que s'alterna la lectura d'unes dades en uns moments i en d'altres es classifica. **La finestra de temps en què es classifica és petita, mentre que la majoria de temps es reben dB**, i és que es prioritza un seguiment dels dB per sobre de classificacions de tipus de so. Malgrat tot això permet **tenir mostres dels tipus de so** de manera que si una font de so s'allarga a través dels minuts es podrà saber que aquest so ha estat tenint lloc durant un cert període.

En diversos moments l'alumne de màster ECiM ha estudiat l'opció de desenvolupar per al seu Treball Final de Màster un model propi de classificació i que fos després adaptable a Soundless amb una petita llibreria complementària que s'adaptés millor al cas d'ús –en resum, substituir AudioClassifier per una llibreria pròpia–, però s'ha acabat desestimant aquesta alternativa.

Annex 3: test d'avaluació de decibels comparant dispositius de prova

El següent text s'ha copiat directament del document original redactat després del test:

TEST 01

Data: 18 d'octubre 2021 (commit a Github) / 19 d'octubre 2021 (realització del test)

Dispositius: Xiaomi A3 (smartphone) comparat amb Sound Level Meter digital Wensn WS1361 (mesurador de decibels calibrat de cost aprox 30€)

Commit: `8cd71d66039f96e444f4f5c77005a1d631666d6b`

Situació del codi: els mesuraments actuals tenen aparença de ser correctes. Obtenim els dB amb la classe MediaRecorder. La classificació a temps real s'ha deixat de banda per incompatibilitat amb MediaRecorder (llibreria TFLite Audio Classifier requereix AudioRecord). La part clau del codi actual és:

```
val maxAmplitude = mediaRecorder.maxAmplitude
dBFS = 20 * log10(maxAmplitude / 255.0F)
dBSPL = 94.0F+dBFS+mediaRecorder.activeMicrophones[0].sensitivity
```

Casos analitzats:

1. Oficina en ambient de treball. Dispositius agafats en mà, sense cap objecte en contacte amb el micròfon, i càlculs fets en diversos punts de l'oficina.

   Hi ha sons ambientals estables i continuats: veus en una sala adjacent, sorolls de teclat, calaixos, passos de persones dins l'oficina, sons de cotxes provinents d'una finestra oberta, veus del carrer, veus d'un edifici adjacent, sons mecànics llunyans.

2. Ídem a situació 1 però amb els dispositius en contacte físic amb una superfície llisa i plana (taula).

3. Els dispositius es troben en contacte físic amb una superfície llisa i plana (taula). Es duen a terme cops secs sobre la superfície de la taula on hi ha els dispositius. Tot seguit es duen a terme cops a un objecte proper que no està en contacte amb la taula. Aquesta prova es fa tot traient i posant en diversos moments el recobriment d'espuma que protegeix el micròfon del SLM (Sound Level Meter).

4. Es fa una lectura des d'una finestra d'un segon pis. Sons de dues motocicletes i converses fortes entre dues persones a una distància de 10 metres. Hi ha una ràfega d'aire sostinguda durant tot el mesurament (1min).

5. Una persona bufa directament al micròfon del dispositiu (a tots els dispositius amb la mateixa intensitat).

6. Mesuraments fets en l'interior d'un portal que dóna a un carrer transitat. Dispositius agafats en mà, sense cap objecte en contacte amb el micròfon, i càlculs fets en una posició estàtica.

   Sons d'ambient: vehicles circulant a uns 25-30 km/h a 3 metres del cotxe, persones parlant a una distància propera (5 metres).

7. Conversa continuada entre dues persones, a una distància d'un metre, en una sala oberta i diàfana. Conversa a un volum normal. Dispositius en mà.

8. Conversa continuada entre dues persones, a una distància de dos metres, en una sala oberta i diàfana. Conversa a un volum baix. Dispositius sobre una taula propera.

9. Conversa continuada entre dues persones, a una distància d'un metre, en una sala tancada i petita. Conversa a un volum normal. Dispositius sobre una taula propera.

10. Conversa continuada entre dues persones, a una distància de dos metres, en una sala tancada i petita. Conversa a un volum baix. Dispositius en mà.

Resultats:

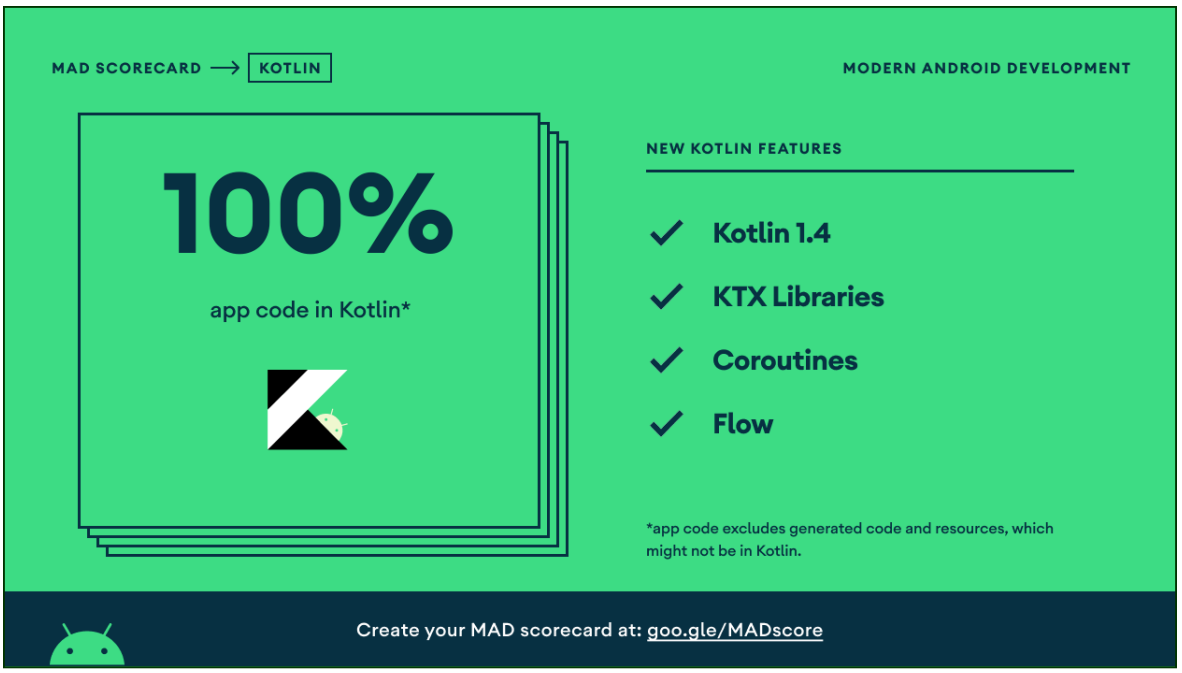
Cas	Resultats dels mesuraments utilitzant Xiaomi A3 i Sound Level Meter (SLM)	Comentaris
1	So d'ambient: 35-45 dB El mòbil mostra alta sensibilitat respecte al SLM (+10, +20 dB) quan es produeixen sons curts i bruscos. Davant un so persistent o un nivell de so ambiental estable, els mesuraments són molt iguals i entren en un rang molt proper (es mouen contínuament a +- 5dB). En general, tant el mòbil com el SLM tenen més sensibilitat davant sons propers que sons llunyans (sons del carrer…).	Si es vol que els mesuraments siguin més propers al SLM (nota: potser no cal i preferim que es detectin sempre els màxims), s'hauria d'aplicar alguna de les següents correccions: ● Que el mòbil obtingui no només el pic més elevat d'amplitud sinó a una mitjana de les amplituds més elevades (podria no ser possible degut a la manca de mètodes de MediaRecorder) ● Aplicar un factor de correcció que faci menys vulnerable al dispositiu davant de pics sobtats de so o que tingui en compte l'amplitud màxima dels segons anteriors per calcular l'actual.
2	Mateixos resultats que en el cas 1	Veure cas 1
3	Capta de seguida els cops i mostra un pic molt alt (per exemple, 70 dB quan el so d'ambient és ~40 dB). El SLM reacciona de manera molt més conservadora i mostra mesures de 45-55 dB. Quan els cops no són tan secs i es van fent amb una freqüència més ràpida i regular (és a dir, el so passa de ser un pic sobtat a ser una seqüència continuada), els mesuraments del mòbil i el SLM s'assemblen molt més i les xifres són pràcticament equiparables.	Veure cas 1
4	So d'ambient: 65-75 dB	Cap.

	Els mesuraments del mòbil i el SLM són pràcticament idèntics.	
5	Quan es bufa molt fort a una distància molt curta del micròfon, el so no ultrapassa els 90-90.2 dB i es manté molt estable en aquest punt. A la mateixa distància i intensitat, el micròfon del SLM registra volums més forts, de 110-120 dB. Quan es bufa més suaument i a una distància més gran, en condicions equiparables, el mòbil capta prop de 80 dB i el SLM també.	Clarament el mòbil no és capaç de registrar sons per sobre de 90 dB, com apuntaven algunes fonts d'Internet. Tot i que aquesta restricció és problemàtica, no hauria de suposar un problema greu ja que mesuraments en l'interior d'una casa de més de 90 dB seran extremadament rars i tota mesura propera a aquest llindar és clarament denunciable de totes maneres.
6	Davant el pas d'un vehicle (transcorrent uns 5-7 segons des que el so del cotxe és proper, passa per davant, i s'allunya), el SLM registra de manera consistent uns quants dB (al voltant de 4-8 dB) més que el mòbil. Davant les converses de les persones que circulen a peu pel carrer, no hi ha cap diferència perceptible entre mòbil i SLM.	L'experiment s'ha repetit orientant el mòbil i el SLM en direccions diferents (sense canviar la distància a la font de soroll), amb el mateix efecte. Una hipòtesi molt bàsica és que sons de vehicles, que són forts, plens i arriben al micròfon des de molts angles diferents, en un ambient a l'exterior, no són tan ben captats pel dispositiu degut a la mala qualitat del micròfon. Una idea és optar per un micròfon extern connectat al mòbil o bé aplicar un factor corrector davant de sons o situacions que previsiblement puguin tenir les mateixes característiques.
7	No es detecten diferències perceptibles que no s'hagin comentat en la resta de casos.	
8		
9		
10		

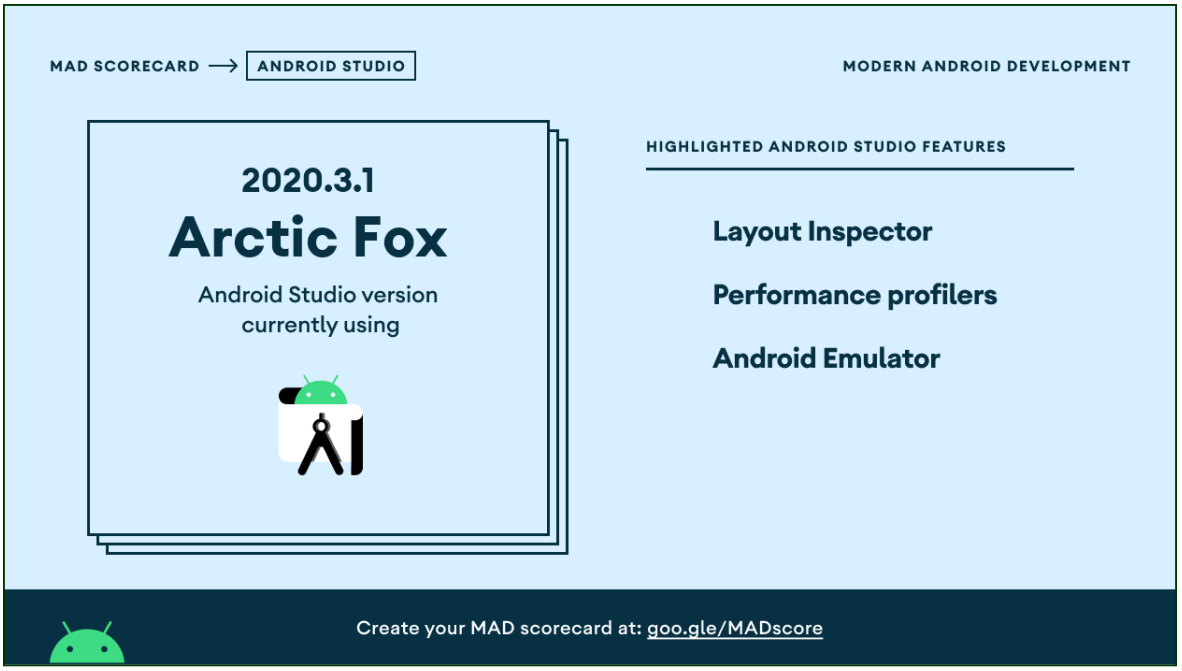
Annex 4: avaluació de pràctiques innovadores MAD (Modern Android Development)

En aquest apartat es mostra el resultat molt positiu d'una avaluació del *plugin* MAD Scorecard per determinar quin és el grau d'adopció de tecnologies noves i eficients en una aplicació Android. L'eina va fer-se pública el desembre de 2020 i és una manera original d'avaluar les àrees en què l'aplicació segueix les tendències de desenvolupament recomanades per l'equip d'Android.

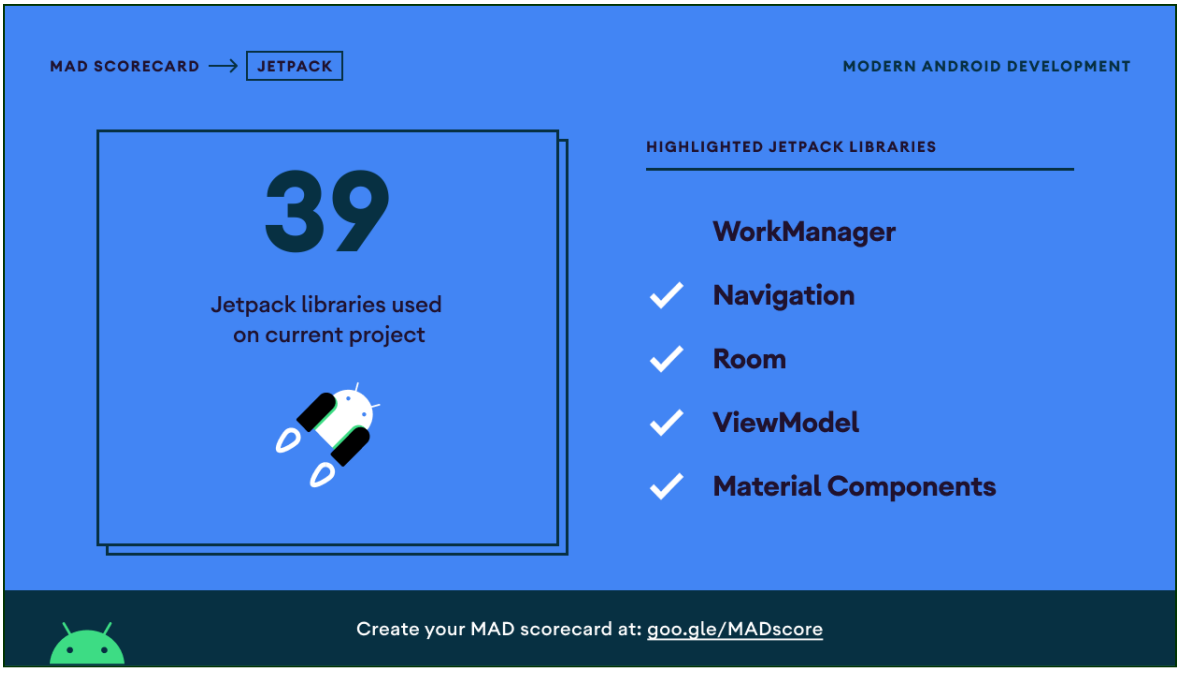
Imatge A4.1: resum MAD Scorecard



Imatge A4.2: utilització de codi en Kotlin



Imatge A4.3: utilització de l'entorn de desenvolupament



Imatge A4.4: utilització de llibreries d'Android Jetpack

Referències

[1] Corporació Catalana de Mitjans Audiovisuals
https://www.ccma.cat/324/la-contaminacio-acustica-es-la-segona-causa-de-morts-prematures-a-europa/noticia/3053306/. 16 d'octubre de 2020.

[2] Oficina Regional Europea de l'Organització Mundial de la Salut
https://www.euro.who.int/en/health-topics/environment-and-health/noise/publications. [Consulta: desembre de 2021].

[3] Repositori Obert de Coneixement de l'Ajuntament de Barcelona
https://bcnroc.ajuntament.barcelona.cat/jspui/handle/11703/86859 [Consulta: desembre de 2021]

[4] Congrés AcustiCat (congrés d'acústica de Catalunya)
https://www.congresacusti.cat/post/milloren-els-indicadors-d-exposici%C3%B3-al-soroll-a-l-%C3%A0rea-metropolitana-de-barcelona-en-els-darrers-10. 15 d'octubre de 2020.

[5] Ajuntament de Reus
https://www.reus.cat/sites/reus/files/Fitxers/arees/medi_ambient/documents/4.pla_daccio_en_materia_de_contaminacio_acustica._fase_iii.pdf. [Consulta: gener de 2022]

[6] Universitat de Vic
https://www.uvic.cat/noticies/un-88-dels-blocs-de-pisos-de-barcelona-estan-exposats-a-alts-nivells-sonors. 24 de juliol de 2018.

[7] Agència de Salut Pública de Barcelona https://www.aspb.cat/documents/sorollambientalisalut-bcn2021/. [Consulta: desembre de 2021]

[8] Cloudbutton https://cloudbutton.eu/. [Consulta: 2021]

[9] Google Developers https://developers.google.com/location-context/sleep. [Consulta: agost de 2021]

[10] Google Blog https://blog.google/technology/health/take-pulse-health-and-wellness-your-phone/ [Consulta: setembre de 2021]

[11] Pàgina web oficial de Xiaomi Mi https://www.mi.com/es/mi-smart-band-6/. [Consulta: setembre de 2021]

[12] Material Design Blog https://material.io/blog/announcing-material-you. 18 de maig de 2021.

[13] Don't Kill My App! https://dontkillmyapp.com/ [Consulta: octubre de 2021]

[14] TensorFlow Lite https://www.tensorflow.org/lite/ [Consulta: gener de 2022]

[15] Repositori de Github de YAMNet
https://github.com/tensorflow/models/tree/master/research/audioset/yamnet [Consulta: gener de 2022]