Universitat de Girona
Escola Politècnica Superior

Projecte fi de grau

Estudi: Grau en Enginyeria Informàtica

Títol: My Student Agenda - Aplicació per estudiants

Document: Memòria

Alumne: Sergi Villa Millan

Tutor: Dr. Martin Campos, Ignacio Clemente
Departament: Informàtica, Matemàtica Aplicada i Estadística
Àrea: Llenguatges i Sistemes Informàtics

Convocatòria (mes/any): Juny 2022

Universitat de Girona
Escola Politècnica Superior

PROJECTE FI DE GRAU

My Student Agenda - Aplicació per estudiants

Autor:

Sergi VILLA MILLAN

Juny 2022

Grau en Enginyeria Informàtica

Tutor:

Ignacio Clemente Martín Campos

"Todos los grandes inventos tecnológicos

creados por el hombre - el avión,

el automóvil, el ordenador - dicen

poco acerca de su inteligencia,

pero dicen mucho acerca de su pereza"

Mark Kennedy

Agraïments

Als meus pares l'esforç que han fet perquè jo hagi pogut arribar fins aquí.

A la meva àvia pel suport incondicional i la confiança que sempre m'ha demostrat.

Als amics que tenia quan vaig començar aquesta etapa i als que ha fet durant el camí.

Al meu tutor Ignacio Clemente Martín Campos i en general a tots els docents amb els quals he tingut el plaer de treballar.

Índex

Índex de figures

CAPÍTOL 1

Introducció

Davant d'un món que no deixa d'evolucionar, la major part de les feines i tasques s'han digitalitzat parcial o totalment. Activitats tan quotidianes com comprar o treballar es poden fer des de casa sense cap problema i cada vegada són més les facilitats i avantatges que ens proposen les alternatives digitals.

Avui en dia els nounats ja es veuen amb un telèfon mòbil entre les mans quan arriben al món i els joves i adults no concebem la vida sense ells. Degut a la importància que els *smartphones* han obtingut, s'ha creat un mercat molt ampli i competitiu en el qual les aplicacions en són l'objecte principal. No només es busca poder fer qualsevol cosa sinó fer-la d'una manera còmoda i fàcil.

En aquest treball desenvoluparé una aplicació per dispositius amb el sistema operatiu Android que pugui substituir a la tradicional agenda d'estudiant que tots en algun moment hem utilitzat.

1.1. Motivacions

Vaig entrar al Grau d'Enginyeria Informàtica sense tenir gaire idea de com programar ni crear aplicacions, només tenia la convicció que en volia

aprendre. Durant aquests darrers quatre anys he cursat diverses assignatures, però sens dubte les que més m'han agradat han sigut les relacionades amb la programació i les bases de dades. N'hi ha hagut una en especial, *Projecte de Desenvolupament de Software*, la qual he gaudit molt al poder sintetitzar els dos aspectes alhora. En aquesta assignatura vam desenvolupar una aplicació en Android en equips de 4 a 6 persones i va ser la manera d'apropar-nos més al món laboral real, ja que ens vam haver de coordinar com a equip per tirar endavant un projecte en el qual cadascú era responsable de diferents parts.

De fet, em va agradar tant l'experiència que he decidit realitzar les meves estades en entorn laboral com a programador d'Android a Additio, on he après moltes coses que he pogut aplicar en aquest projecte i els coneixements adquirits m'han ajudat a programar de manera més professional.

1.2. Propòsit

El propòsit d'aquest projecte és digitalitzar un element tan necessari i fonamental com són les agendes personals, en aquest cas per als estudiants. En general les aplicacions que he pogut trobar són molt carregoses i no són còmodes d'utilitzar, ja que tenen un munt d'opcions i de pestanyes que a l'usuari mitjà no li aporten gran cosa, ja que normalment tots busquem facilitat i velocitat. El que he buscat ha estat implementar un *software* intuïtiu i senzill, sense omplir l'aplicació de funcions poc útils, per

aconseguir menús nets i que els usuaris entenguin què fa cadascuna de les opcions i/o pestanyes disponibles.

1.3. Objectius del projecte

L'objectiu final és desenvolupar una aplicació que qualsevol estudiant de qualsevol nivell pugui utilitzar. Aquesta, doncs, no ha de ser gaire restrictiva i els serveis que ofereixi han de ser prou generals com per adaptar-se a diferents condicions.

Per poder fer ús de l'aplicació, caldrà que l'usuari es registri. Un cop registrat, en termes generals, podrà crear assignatures, establir quines són les activitats avaluables d'aquestes i quin pes tenen a la nota final, indicar quina puntuació va obtenint a cadascuna, definir-ne els horaris i els professors, programar recordatoris per a diferents esdeveniments (exàmens, entregues, presentacions...) i saber en tot moment com porta les assignatures. No es pretén incloure la funcionalitat de calendari ni de gestió d'horaris, ja que ja hi ha diverses aplicacions molt potents dedicades únicament a aquests aspectes.

Per poder assolir aquest objectiu principal hi ha una sèrie d'objectius secundaris que s'han de cobrir:

- Creació d'una base de dades per poder guardar les dades dels usuaris, assignatures, professors, qualificacions i recordatoris, entre d'altres.

- Desenvolupar una aplicació en Java amb *Android Studio* amb *Retrofit* com a client HTTP.
- Implementar una API REST[1] amb *Spring Boot* per poder comunicar-se amb la base de dades.
- A l'hora d'entrar les qualificacions, s'ha de trobar un sistema perquè els usuaris puguin personalitzar els sistemes de puntuacions, ja que aquestes poden estar en diferents formats i representar diferents valors (per exemple, en comptes del sistema 0-10 potser es fa servir un sistema F-A o un Insuficient-Excel·lent, i cal definir quin valor numèric representa cada puntuació).

[1] Una API (*Application Programming Interface*) REST és una interfície que especifica com han d'interaccionar diferents sistemes, cumplint els principis de disseny REST (*Representational State Transfer*).

CAPÍTOL 2

Estudi de viabilitat

Per portar a terme aquest projecte es requereixen una sèrie de coneixements i recursos que són necessaris. Tot seguit es realitza un estudi d'aquests requisits i quin cost econòmic suposarien en el supòsit d'un cas real al món empresarial.

2.1. Coneixements necessaris

Hi ha uns coneixements que són indispensables per poder construir l'aplicació correctament:

- Calen uns coneixements avançats de programació, principalment Java i SQL.
- És imprescindible entendre com estructurar una base de dades i com funcionen aquestes per aconseguir eficiència i bon temps de resposta.
- Nocions d'enginyeria del *software* per poder dissenyar una aplicació confiable i de qualitat.
- Entendre com funciona Android, pel que fa a sistema operatiu, cicles de vida dels processos, crides al sistema, versions, etc. Aquest punt es troba més detallat a l'apartat 4.2.

2.2. Recursos tecnològics

Per desenvolupar el projecte es requereixen certs recursos i programari:

- Indispensablement, cal un ordinador sobre el qual poder treballar.
- Un entorn de desenvolupament integrat (*IDE*) per muntar i executar un servidor *backend*, en aquest cas *IntelliJ IDEA*.
- Un entorn sobre el qual testejar les crides del servidor, com ara *Postman*.
- Un programa que permeti visualitzar les dades que hi ha al servidor gràficament, com *HeidiSQL*.
- Un IDE per desenvolupar l'aplicació Android, com per exemple *Android Studio*.
- Un dispositiu Android per executar l'aplicació. En el cas d'*Android Studio* es poden crear múltiples dispositius emuladors amb diferents versions d'Android i diferents mides, però l'execució d'aquests té un cost de recursos molt gran per l'ordinador i, per tant, l'experiència pot no ser del tot real (temps de processament lent, transicions i moviments tallats, etc).
- Un programa d'edició d'imatges per crear i dissenyar el logo, la icona i altres continguts, tal com *GIMP*.

2.3. Recursos humans

Per dur a terme l'aplicació caldrien:

- Un programador d'Android treballant mitja jornada durant tres mesos.
- Un programador *backend* i dissenyador de base de dades treballant mitja jornada durant dos mesos.
- Un dissenyador gràfic amb, com a molt, una setmana de feina.

2.4. Recursos econòmics

Un cop definits tots els recursos que es necessiten per tirar endavant aquest projecte, es pot fer una aproximació del cost econòmic que s'hauria d'enfrontar:

Recurs	Descripció	Cost
Ordinador	Dispositiu amb prou memòria i capacitat per executar el servidor i els emuladors simultàniament.	1000€
IntelliJ IDEA	IDE per programar el servidor. Requereix llicència de JetBrains.	150€/any
Postman	Programa per testejar el funcionament del servidor. Programari lliure.	0€
HeidiSQL	Programa per visualitzar l'estat i la informació de la base de dades.	0€
Android Studio	IDE per programar l'aplicació Android. Programari lliure.	0€
Dispositiu Android	Dispositiu per testejar el funcionament de l'aplicació Android.	300€

GIMP	Programa d'edició d'imatges per dissenyar el logo i la icona de l'aplicació, entre d'altres. Programari lliure.	0€
TOTAL		1450€

Figura 2.4.1 - Taula de costos dels recursos tecnològics

Per calcular els costos econòmics que suposaria la contractació dels professionals he consultat la guia Hays[2] per saber quins són els sous aproximats per cada empleat que requereix aquest projecte. En el cas del programador Android s'estimen uns 2400€ bruts/mes, pel programador *backend* uns 2750€ bruts/mes i pel dissenyador uns 1800€ bruts/mes.

Perfil	Temps	Sou/mes	Cost
Programador Android	3 mesos a mitja jornada = 1.5 mesos	2400€	3600€
Programador *backend*	2 mesos a mitja jornada = 1 mes	2750€	2750€
Dissenyador gràfic	1 setmana = 0.25 mesos	1800€	450€
TOTAL			6800€

Figura 2.4.2 - Taula de costos dels recursos humans

Si fem la suma dels costos dels recursos tecnològics i humans obtenim un pressupost inicial de 1450€ + 6800€ = 8250€.

[2] Guía Hays 2022 - Informe de salarios

CAPÍTOL 3

Metodologia

Al tractar-se aquest d'un projecte de desenvolupament de *software* és imprescindible seguir una metodologia estructurada i ben definida per a aconseguir un resultat correcte. Tot i això, s'ha de permetre un cert grau de flexibilitat, ja que durant el transcurs del projecte poden sorgir problemes inesperats, modificacions o hi poden haver tasques que s'allarguin. La metodologia escollida en aquest cas ha estat SCRUM.

3.1. Metodologia SCRUM

La metodologia SCRUM és un marc de treball per al desenvolupament, entrega, manteniment i gestió de productes/projectes. Està dissenyada per a equips de deu persones o menys que es divideixen la feina i treballen col·laborativament per a assolir un objectiu comú.

Aquest model de treball va ser identificat i definit per Hirotaka Takeuchi i Ikujiro Nonaka l'any 1986 al seu article *The New New Product Development Game*[3]. En aquest, van definir una nova metodologia de treball que comparaven amb l'avanç en formació a melé, ja que els integrants de l'equip avancen junts per a aconseguir un propòsit major.

Hi ha una sèrie de punts que la caracteritzen:

[3] Hirotaka Takeuchi i Ikujiro Nonaka, "The New New Product Development Game", *Harvard Business Review*

- A diferència d'altres mètodes de treball, SCRUM es basa en un procés iteratiu i incremental on els membres d'un equip treballen en paral·lel, en contra del plantejament tradicional de treball seqüencial i en cascada.
- Es dóna molta importància a la comunicació interna i es valora que aquesta sigui presencial. Es fan reunions diàries durant les quals cada membre de l'equip exposa com va avançant en les tasques que té assignades. A més, en tot moment qualsevol persona pot saber en què està treballant cadascú gràcies a un tauler o pissarra.
- Al final de cada iteració (*sprint*) es mostren els resultats al client perquè aquest els valori i proposi els canvis que cregui convenients.
- Es dóna prioritat al que més valor té pel client.
- S'assumeix que el producte no pot ser completament entès i definit per avançat, ja que les demandes del client van canviant durant el transcurs del projecte i que sorgeixen problemes inesperats i imprevisibles. S'adopta, doncs, una perspectiva tolerant i els esforços es centren en maximitzar les capacitats de l'equip per poder fer front ràpidament a les variacions dels requisits, a l'evolució de les tecnologies i els canvis de condicions del mercat.

Es podria dir que aquesta és una metodologia indicada per a projectes en entorns complexos, on és necessari obtenir resultats regularment, on els requisits són canviants o poc definits i on la flexibilitat i productivitat són fonamentals.

Seguint aquestes bones pràctiques, entre d'altres, s'aconsegueixen uns beneficis molt destacats:

- **Reducció de riscos**. El fet de conèixer la velocitat a la qual l'equip avança amb el projecte i de desenvolupar en primer lloc les funcionalitats amb més valor, es poden dissuadir riscos efectivament i de forma anticipada.
- **Reducció del temps d'espera per sortir a producció**. El client pot començar a utilitzar les funcionalitats amb més valor abans que el projecte estigui completament acabat.
- **Millor qualitat de software**. La manera de treballar tan metòdica que implica el model juntament amb la necessitat d'obtenir una versió funcional al final de cada iteració ajuden a l'elaboració d'un software d'alt nivell.
- **Flexibilitat davant canvis**. S'aconsegueix una gran capacitat de reacció davant dels requeriments que es generen a partir de les necessitats o modificacions del client o l'evolució del mercat. El sistema està dissenyat per adaptar-se fàcilment als canvis.
- **Millor productivitat**. En eliminar la jerarquia que impliquen altres models de treball i permetre als equips gestionar i estructurar la feina de manera autònoma, s'aconsegueix una motivació que afecta notablement a la productivitat.
- **Prediccions de temps**. Com que es coneix la velocitat a la qual l'equip treballa, és possible estimar la quantitat de feina que es pot fer en

cada iteració i, per tant, calcular aproximadament quan estaran disponibles les diferents funcionalitats que demana el client.

3.2. SCRUM aplicat a aquest projecte

Des d'un principi tenia clar que la metodologia amb la qual anava a treballar seria SCRUM, ja que l'experiència que havia obtingut tant en algunes assignatures com, sobretot, durant la meva estada en l'entorn laboral, havia sigut molt satisfactòria.

Al tractar-se d'un projecte individual seré jo mateix el que faci els tres rols que hi ha dins d'un equip; planificaré els requeriments i les tasques que en derivin (vegeu apartat 6), em preocuparé per la correcta aplicació de la metodologia i desenvoluparé l'aplicació.

Un altre aspecte clau en l'elecció de la metodologia va ser que es contempla el fet que hi hauran imprevistos i aquests són acceptats i entesos com a una part del procés, no són rebutjats ni percebuts negativament. És molt important aquesta filosofia tenint en compte que seré jo sol qui ha de tirar tot el projecte endavant, coneixent que hi ha coses que sé fer i coses que hauré d'aprendre.

CAPÍTOL 4

Marc de treball i conceptes previs

4.1. Marc de treball

En el desenvolupament d'aplicacions és molt important conèixer quin és el teu públic objectiu per tal de maximitzar el teu producte, ja que el mercat és molt competitiu i són els detalls els que marquen la diferència. Saber quin és el perfil mitjà dels usuaris pels quals està pensada l'aplicació és crucial per poder incloure els detalls que els faci optar per la teva aplicació en comptes d'una altra.

MyStudentAgenda està orientada als estudiants, per tant, es pot intuir un perfil econòmic baix, i és per això que l'aplicació serà completament gratuïta. A més, pressuposarem que els usuaris no volen destinar gaire temps al seu ús, per tant, calen menús senzills i ràpids.

Si el que es vol es obrir-se al màxim nombre d'usuaris possible, Android és la plataforma indicada. Com es pot veure a la següent imatge, el nombre de vendes de dispositius amb aquest sistema operatiu és significativament major a iOS, el seu competidor directe.

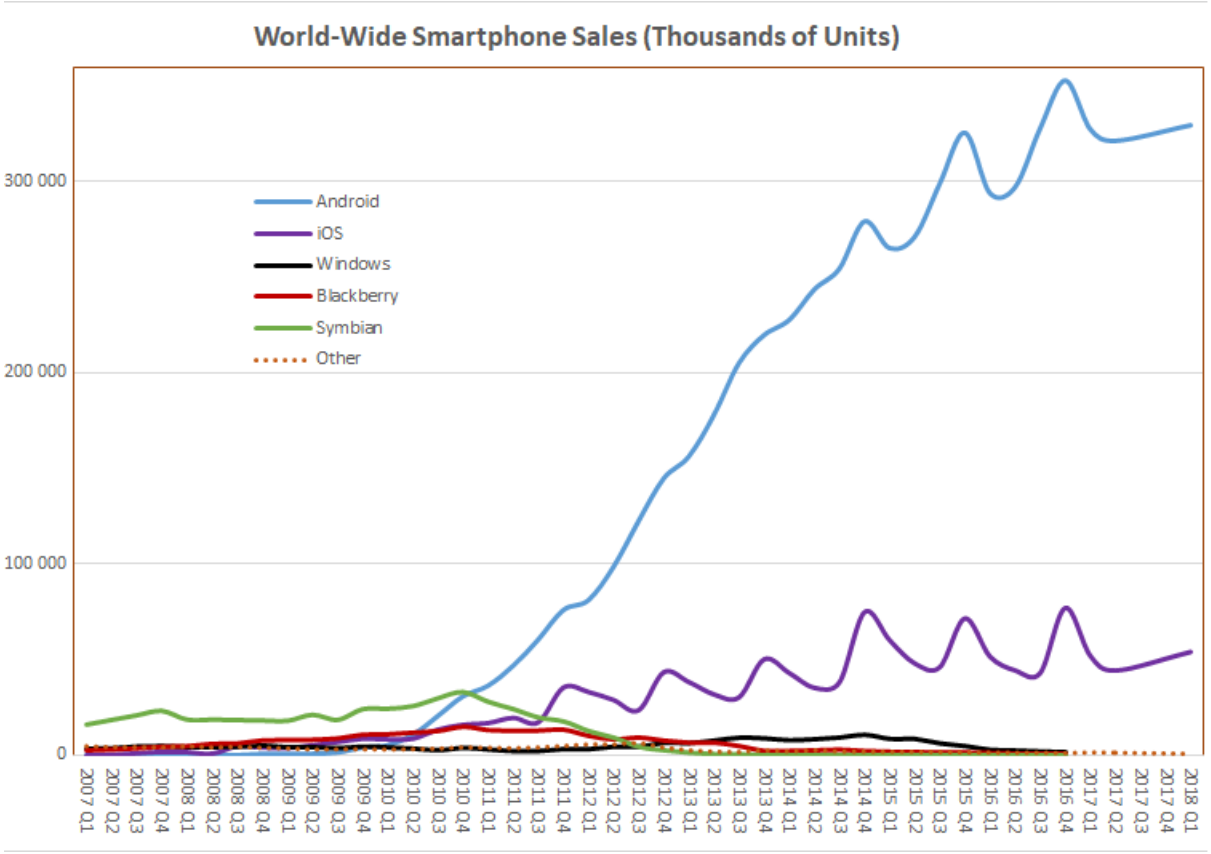


Figura 4.1.1 - Evolució de les vendes de dispositius mòbils segons el seu sistema operatiu

4.2. Conceptes previs

Per poder entendre algunes explicacions que es troben més endavant hi ha certs conceptes que seria interessant que quedessin clars:

4.2.1. Conceptes d'Android

Versions

El sistema operatiu Android ha tingut diverses versions al llarg del temps, aproximadament es treu una versió nova cada any. Fins a la versió 9.0, que va sortir l'agost del 2018, aquestes s'identificaven per un nom codi on cada versió tenia nom de postres o dolços (per exemple la versió 4.0 es va publicar amb el nom d'*Ice Cream Sandwich*). Això es feia per unificar petites versions sota un sol identificador, ja que abans era comú que es publiqués una versió i durant els següents mesos es despleguessin versions amb l'objectiu d'anar solucionant errors que sorgien.

Això va canviar amb la publicació d'Android 10, ja que des de llavors ja no hi ha *releases (publicacions)* de petites versions solucionant *bugs* (errors), sinó que s'aposta tot a una única versió ara sí identificada pel nombre d'aquesta.

A l'hora de desenvolupar una aplicació és molt important decidir quina és la versió mínima que han de tenir els dispositius per poder executar-la, ja que això definirà quin *SDK*[4] es farà servir durant el procés de desenvolupament. Es sol establir una versió mínima molt baixa per permetre a tots els usuaris l'accés a l'aplicació, pel fet que les versions més altes poden usar la gran majoria de llibreries que usen les versions més baixes.

[4] *Software Development Kit*. Són les eines que es fan servir per desenvolupar aplicacions Android. S'inclouen llibreries, documentació, API, etc.

Views

La classe *View* és la base de tots els components gràfics que es poden crear i utilitzar a Android. Aquests components, d'ara en endavant "vistes", són tots els textos, botons, imatges i en general qualsevol element que apareix a la pantalla. La unificació d'aquestes vistes sota una única classe pare permet una estandardització del seu ús, ja que comparteixen les propietats principals (alçada, amplada, marges, visibilitat...) i la majoria de mètodes (se'ls pot assignar *listeners*[5] per a definir el seu comportament davant diferents situacions).

La classe *ViewGroup* és una subclasse de *View* que actua de contenidor per guardar vistes dins seu i permet estructurar components. El conjunt d'aquestes vistes, filles, dins d'un (com a mínim) *ViewGroup* pare permet dissenyar les pantalles de l'aplicació en fitxers XML, els anomenats *layouts*.

Activity

Les *Activities* són els elements bàsics d'una aplicació. Cada *Activity* conté una part lògica (un fitxer *Java*) i una part gràfica (un *layout*), i solen ser l'equivalent a una pantalla o un conjunt de pantalles de l'aplicació.

Activity Lifecycle

Les *Activities* tenen un cicle de vida, l'*Activity Lifecycle*. Totes passen per determinats estats i aquests poden ser controlats des del fitxer *Java* de

[5] Mecanisme que permet definir quin comportament ha de tenir una vista en funció dels events que es llancin durant l'execució de l'aplicació.

l'*Activity* per definir-ne el comportament. Aquests estats s'invoquen a través de les següents crides:

- *onCreate():* Es crida sempre que es crea una *Activity*.
- *onStart()*: Es crida quan l'*Activity* comença a ser visible per l'usuari.
- *onResume()*: Es crida quan l'usuari ja pot interactuar amb els components (vistes) de l'*Activity*.
- *onPause()*: Es crida abans que l'*Activity* deixi de ser visible per l'usuari.
- *onStop()*: Es crida quan l'usuari deixa de veure l'*Activity*, per exemple quan se n'ha iniciat una altre.
- *onRestart()*: Es crida quan es recupera una *Activity* que s'havia pausat.
- *onDestroy()*: Es crida quan una *Activity* està a punt de ser destruïda, ja sigui perquè la funció d'aquesta s'ha acabat i el programador la destrueix o perquè el sistema l'elimina per tal d'alliberar memòria.

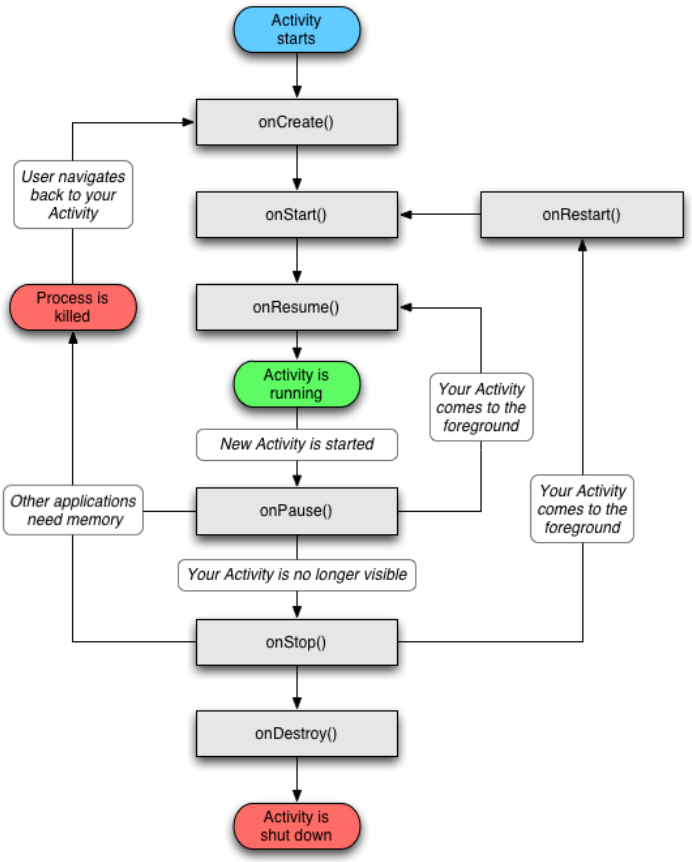


Figura 4.2.1.1 - Activity Lifecycle

4.2.2. Conceptes de la metodologia de treball

Sprints

Un *sprint* és el període durant el qual un equip SCRUM intenta desenvolupar, implementar, corregir o millorar algunes funcionalitats. Aquests tenen una durada d'entre dues i quatre setmanes, habitualment. Al final de cada sprint s'ha d'obtenir un producte funcional, s'hagin complert o no els objectius.

User Stories

Les històries poden ser definides i associades com a funcionalitats que aporten valor al producte que demana el client. Les històries solen ser desenvolupades per diferents persones, ja que engloben tasques de disseny, programació *backend* i *frontend*, testing, etc. A causa d'això, les històries normalment estan subdividides en *tasks*.

Tasca

Les tasques són les parts en què poden ser descompostes les històries. Aquesta divisió ajuda a construir un producte més ben treballat i definit. No cal que siguin comprensibles pel client, ja que normalment són definides usant tecnicismes, sinó pels desenvolupadors que les implementaran. L'ideal és que cada tasca la implementi un sol programador.

CAPÍTOL 5

Requisits del sistema

Per dur a terme aquest projecte s'han establert una sèrie de requisits, tant funcionals com no funcionals. Cada requisit té assignat un nombre i un nivell de prioritat, ja que n'hi ha alguns que són indispensables per desenvolupar l'aplicació i n'hi ha d'altres que afegeixen funcionalitats o accessoris, però que no són imprescindibles per al correcte funcionament d'aquesta.

5.1. Requisits funcionals

Els requisits funcionals, és a dir, les funcionalitats i serveix que ofereix l'aplicació, són:

1. Els usuaris s'han de registrar i/o iniciar sessió per poder accedir als menús de l'aplicació. ***[Prioritat 1]***
2. L'usuari ha de poder crear assignatures. ***[Prioritat 1]***
3. L'usuari ha de poder veure, editar i accedir a totes les assignatures que ha creat (i que no ha eliminat). ***[Prioritat 1]***
4. L'usuari ha de poder crear grups d'activitats[6] per cada assignatura. Cada grup representa un percentatge del total de nota. ***[Prioritat 1]***
5. L'usuari ha de poder crear activitats per cada assignatura. Cada activitat representa un percentatge del total de la nota. Les activitats poden pertànyer a un grup d'activitats. ***[Prioritat 1]***

[6] Els grups d'activitats serveixen per crear conjunts d'activitats que formen un sol bloc. Un bon exemple seria un grup de "Pràctiques" amb les activitats "Pràctica 1" i "Pràctica 2".

6. L'usuari ha de poder assignar una qualificació a cada activitat i aquesta ha de modificar la nota global de l'assignatura a la que pertany en conseqüència. ***[Prioritat 1]***
7. L'usuari ha de poder definir un sistema d'avaluació propi. ***[Prioritat 1]***
8. A l'hora d'entrar la qualificació d'una activitat l'usuari ha de poder decidir quin sistema de puntuació vol fer servir, per permetre diferents maneres d'avaluar. ***[Prioritat 1]***
9. L'usuari ha de poder veure, editar i accedir a tots els grups i totes les activitats que ha creat (i que no ha eliminat) per una assignatura. ***[Prioritat 1]***
10. L'usuari ha de poder veure de manera gràfica quin percentatge de l'assignatura ja ha estat avaluat i la nota que té segons les qualificacions que ha obtingut a les diferents activitats. ***[Prioritat 1]***
11. L'usuari ha de poder crear professors. ***[Prioritat 1]***
12. L'usuari ha de poder veure, editar i accedir a tots els professors que ha creat (i que no ha eliminat). ***[Prioritat 2]***
13. L'usuari ha de poder ocultar professors per a que aquests no apareguin quan es vulgui assignar un professor a una assignatura. ***[Prioritat 3]***
14. L'usuari ha de poder crear tasques per a les diferents assignatures. ***[Prioritat 1]***
15. L'usuari ha de poder veure, editar i accedir a totes les tasques que ha creat (i que no ha eliminat). ***[Prioritat 1]***
16. L'usuari ha de poder programar com a molt un recordatori per cada tasca. ***[Prioritat 1]***

17. L'usuari ha de poder ocultar assignatures per a que aquestes no apareguin quan es vulgui assignar una activitat o una tasca a una assignatura. ***[Prioritat 2]***
18. L'usuari ha de poder assignar professors, activitats i tasques a les assignatures. ***[Prioritat 1]***
19. L'usuari ha de poder canviar l'idioma de l'aplicació des del menú d'opcions. Els idiomes disponibles seran el català, l'anglès i l'espanyol. ***[Prioritat 2]***
20. L'usuari ha de poder modificar el seu nom d'usuari, correu i contrasenya. ***[Prioritat 2]***
21. L'usuari ha de poder esborrar totes les seves dades. ***[Prioritat 1]***

5.2. Requisits no funcionals

Els requisits no funcionals, és a dir, les propietats i restriccions del sistema, són:

- L'aplicació ha de poder mantenir iniciada la sessió d'un usuari, no pot demanar autentificació cada cop que es vol usar aquesta. Caldrà, doncs, un sistema de gestió de sessions.
- Les operacions d'inserció de dades (com per exemple crear una assignatura o inserir la qualificació d'una activitat) seran operacions atòmiques. Això vol dir que aquestes operacions s'han de fer completes o els canvis es descartaran. Això és per prevenir que a la base de dades no hi hagin dades inconsistents a causa de possibles interrupcions durant el procés d'inserció o actualització de dades.

- L'aplicació ha de ser compatible amb tots els dispositius amb una versió d'Android 5.0 o superior. Aquesta decisió s'explica en més detall a l'apartat 7.2.1.

5.3. Matriu de dependències

Per veure les relacions entre requisits es pot consultar la matriu de dependències:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
1																					
2	X																				
3		X																			
4			X																		
5			X	X																	
6					X																
7	X																				
8						X	X														
9				X	X																
10									X												
11	X																				
12											X										
13												X									
14	X	X																			
15														X							
16															X						
17			X																		
18			X		X						X			X							
19	X																				
20	X																				
21	X																				

Figura 5.3.1 - Taula de dependències entre requisits funcionals

CAPÍTOL 6

Planificació

Per estar en concordança amb la metodologia escollida, la planificació del desenvolupament d'aquest projecte es va fer amb *sprints* amb una durada de dues setmanes cada un. A cada *iteració* l'objectiu era obtenir una versió funcional de l'aplicació amb, com a mínim, una funcionalitat nova completament acabada a cada iteració.

Com que hi havia dues grans parts del projecte clarament diferenciades, el servidor i l'aplicació Android, vaig començar per la primera, ja que havia de ser la base del projecte. La intenció era disposar d'un servidor completament preparat abans de començar amb la implementació de l'app, cosa que no va ser possible, ja que al llarg del projecte hi va haver canvis que van comportar modificacions sobre aquest. Aquestes alteracions, però, van ser en la majoria lleus i no van suposar gaire desviació.

Els sprints es van plantejar de la següent manera:

- Sprint 1: Inicialització del servidor i implementació de les *Entities*. L'objectiu era obtenir un servidor funcional amb les configuracions pertinents per poder treballar. Es van implementar també les entitats bàsiques (objectes) de la base de dades.
- Sprint 2: Implementació dels *Services* i *Repositories*. Durant el segon *sprint* es van implementar els Serveis (classes on es troba la lògica) i els Repositoris (classes que interactuen directament amb la base de dades) per a cada entitat.

- Sprint 3: Implementació dels *Controllers*. Al final del tercer *sprint* es volia aconseguir una API REST la qual estigués preparada per lliurar tota la informació que necessitaria al llarg de la implementació de l'app.
- Sprint 4: Inicialització de l'aplicació i gestió de sessions. El propòsit de la quarta iteració era obtenir una aplicació Android amb les configuracions a punt i un control de sessió d'usuaris, és a dir, poder iniciar sessió (*login*) i poder-la tancar (*logout*), i que els canvis es mantinguin fins i tot quan es tanca l'aplicació.
- Sprint 5: Implementació pantalles Configuració i Professors. L'objectiu era implementar els *fragments*[7] per a les pestanyes de Configuració (El meu perfil, idioma, etc.) i Professors (veure, crear i editar).
- Sprint 6: Implementació Assignatures. Obtenir una versió la qual permetés la creació, edició i visualització de les Assignatures de l'usuari.
- Sprint 7: Implementació Activitats i Grups. L'objectiu era implementar tots els *fragments* necessaris per a la creació, edició i visualització de les Activitats i dels Grups.
- Sprint 8: Implementació Tasques i Recordatoris. Implementació de les pantalles de creació, edició i visualització de Tasques. Implementació també d'un sistema de programació i gestió de notificacions per a aquestes.

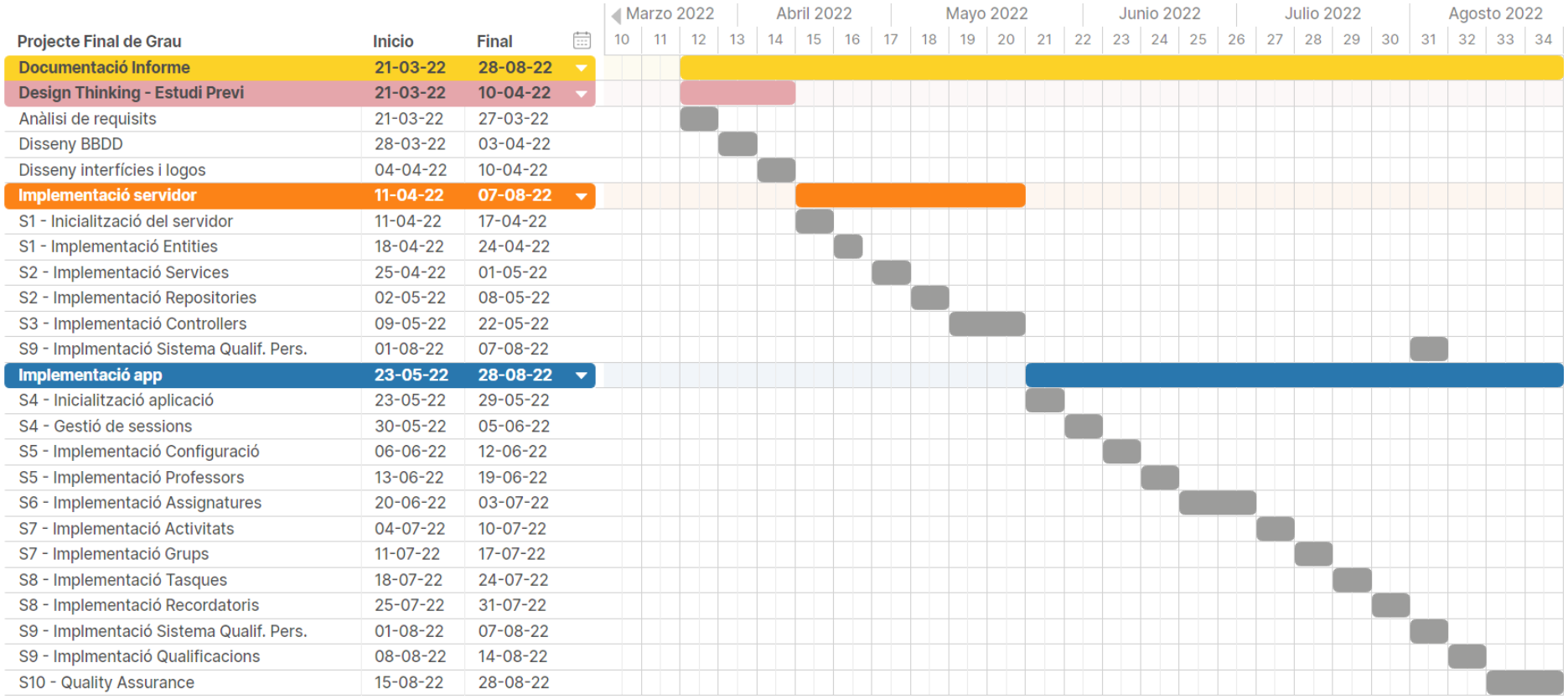
[7] Els *fragments* són parts de la interfície d'usuari que estan lligats a una *Activity*. S'usen per reaprofitar segment de codi i lògiques, passar informació entre "pantalles" i adaptar-se a dispositius de diferents configuracions.

- Sprint 9: Implementació Sistema de Qualificació Personalitzat i Qualificacions. Implementació d'un sistema per a permetre a l'usuari dissenyar un sistema de qualificació personalitzat. Implementació també dels *fragments* per a crear, editar i eliminar Qualificacions per a les Activitats.
- Sprint 10: Quality Assurance. L'objectiu d'aquest últim *sprint* era fer una repassada completa i exhaustiva de l'aplicació (*testing*) per detectar baixades de rendiment, corregir errors ortogràfics, polir detalls estètics i preparar una versió formal.

Prèviament al procés de programació de l'aplicació, hi va haver un procés d'anàlisi de requisits, disseny de la base de dades, disseny de les interfícies de l'aplicació i logos d'aquesta i investigació de les tecnologies a usar. També, paral·lelament i al llarg de tot el procés de desenvolupament, s'ha anat documentant l'evolució del projecte en aquest document.

A la pàgina següent es pot apreciar un diagrama de Gantt[8] de com va acabar essent la temporalització del projecte.

[8] Un diagrama de Gantt és una eina de planificació/temporalització de treball, que presenta les activitats de manera gràfica sobre un eix temporal, indicant-ne l'inici, el fi, la durada, etc.



Projecte Final de Grau	Inicio	Final
Documentació Informe	**21-03-22**	**28-08-22**
Design Thinking - Estudi Previ	**21-03-22**	**10-04-22**
Anàlisi de requisits	21-03-22	27-03-22
Disseny BBDD	28-03-22	03-04-22
Disseny interfícies i logos	04-04-22	10-04-22
Implementació servidor	**11-04-22**	**07-08-22**
S1 - Inicialització del servidor	11-04-22	17-04-22
S1 - Implementació Entities	18-04-22	24-04-22
S2 - Implementació Services	25-04-22	01-05-22
S2 - Implementació Repositories	02-05-22	08-05-22
S3 - Implementació Controllers	09-05-22	22-05-22
S9 - Implmentació Sistema Qualif. Pers.	01-08-22	07-08-22
Implementació app	**23-05-22**	**28-08-22**
S4 - Inicialització aplicació	23-05-22	29-05-22
S4 - Gestió de sessions	30-05-22	05-06-22
S5 - Implementació Configuració	06-06-22	12-06-22
S5 - Implementació Professors	13-06-22	19-06-22
S6 - Implementació Assignatures	20-06-22	03-07-22
S7 - Implementació Activitats	04-07-22	10-07-22
S7 - Implementació Grups	11-07-22	17-07-22
S8 - Implementació Tasques	18-07-22	24-07-22
S8 - Implementació Recordatoris	25-07-22	31-07-22
S9 - Implmentació Sistema Qualif. Pers.	01-08-22	07-08-22
S9 - Implmentació Qualificacions	08-08-22	14-08-22
S10 - Quality Assurance	15-08-22	28-08-22

Figura 6.1 - Diagrama de Gantt del projecte

CAPÍTOL 7

Estudis i decisions

L'elecció de les tecnologies que s'empraran és clau per assolir un projecte exitós. En aquest apartat es documenten quines han sigut les llibreries i tecnologies usades.

7.1. Seguiment del projecte

7.1.1. Github

Github és un servei de *hosting* de repositoris que afavoreix el control de versions i la col·laboració entre usuaris. S'ha utilitzat per guardar dos projectes, servidor i aplicació. D'aquesta manera el codi sempre ha estat guardat i protegit al núvol, oferint una gran disponibilitat i confiança.

7.1.2. YouTrack

YouTrack és un sistema de gestió de projectes de *software* i programació de tasques. Ha sigut l'eina que s'ha fet servir per dissenyar els *sprints*, definir *user stories* i *tasks* i comptabilitzar-ne el temps emprat.

7.1.3. Postman

Postman és una plataforma sobre la qual els desenvolupadors poden dissenyar, construir, provar i iterar les seves API. Ha sigut l'aplicació usada per *testejar* les crides implementades al servidor.

7.2. Android

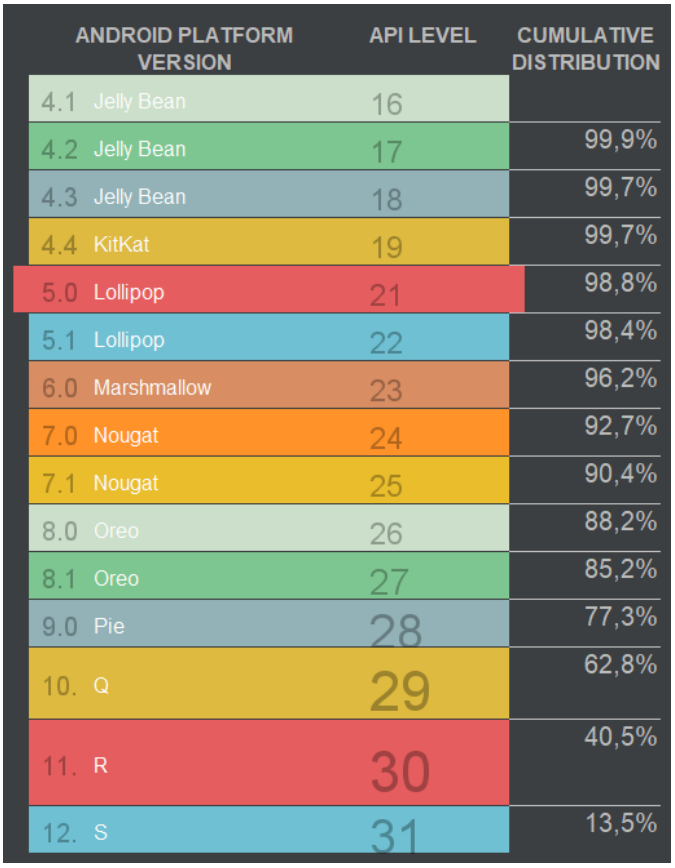
7.2.1. Versió d'Android

Com s'ha explicat prèviament en aquest document, la decisió de quina és la versió mínima que han de tenir els dispositius per poder executar una aplicació és molt important, ja que aquesta defineix el mercat potencial d'usuaris i el ventall de funcionalitats que es poden fer servir al programar.

El mateix IDE amb el que s'ha creat l'aplicació, Android Studio, et mostra el percentatge d'usuaris que podran fer servir la teva aplicació en funció de la versió que escullis, i t'indica també quins canvis i funcionalitats aporta cada una.

Després d'investigar les tecnologies que afegia cada versió, es va optar per definir que la versió d'Android mínima per executar l'aplicació fos la *Lollipop* (5.0). A part de ser una versió amb un públic objectiu molt gran (vegeu figura 7.2.1.1), també inclou compatibilitat amb *Material Design*[9] i incorpora el *RecyclerView*, que és un *widget* (vista) dinàmic que permet mostrar llistes d'elements i que s'ha usat molt durant el procés d'implementació. A més, és la versió que va introduir la presentació de notificacions als dispositius amb la pantalla bloquejada, funcionalitat molt útil per a les notificacions que llança l'app.

[9] *Material Design* és una llibreria de codi obert que ofereix eines i components de disseny enfocats a enriquir l'experiència de l'usuari, millorant les interfícies, animacions, transicions i en general qualsevol aspecte gràfic.



ANDROID PLATFORM VERSION	API LEVEL	CUMULATIVE DISTRIBUTION
4.1 Jelly Bean	16	
4.2 Jelly Bean	17	99,9%
4.3 Jelly Bean	18	99,7%
4.4 KitKat	19	99,7%
5.0 Lollipop	21	98,8%
5.1 Lollipop	22	98,4%
6.0 Marshmallow	23	96,2%
7.0 Nougat	24	92,7%
7.1 Nougat	25	90,4%
8.0 Oreo	26	88,2%
8.1 Oreo	27	85,2%
9.0 Pie	28	77,3%
10. Q	29	62,8%
11. R	30	40,5%
12. S	31	13,5%

Figura 7.2.1.1 - Taula de percentatges d'usuaris que suporten cada versió d'Android

7.2.2. PersistentCookieJar

PersistentCookieJar és una llibreria que usa les *SharedPreferences* (vegeu apartat 9.1.1) per gestionar les sessions dels usuaris. La versió emprada és una implementació per *OkHttp3*, que actúa de client HTTP.

7.2.3. ButterKnife

ButterKnife és una llibreria que serveix per lligar (*bind*) les *views* d'un *layout* des d'un fitxer *Java* i així poder modificar-ne el comportament. Per entendre millor els seus avantatges:

- Si la vinculació es fes "tradicionalment", usant els mètodes bàsics que proporciona Android, aquesta es faria al mètode *onCreate()* o al mètode *onCreateView()* (en funció de si estem tractant una *Activity* o un *Fragment*, respectivament) i sempre després d'establir quin és el *layout*:

```
// ATTRIBUTES
ImageView ivIcon;

TextView tvTitle;

// PUBLIC METHODS
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstance) {
    rootView = inflater.inflate(R.layout.subject_fragment, container,  attachToRoot: false);

    ivIcon = rootView.findViewById(R.id.subject_fragment_icon);
    tvTitle = rootView.findViewById(R.id.subject_fragment_title);

    return rootView;
}

@Override
public void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setHasOptionsMenu(true);
}
```

Figura 7.2.3.1 - View binding tradicional

- En canvi, si s'utilitza aquesta llibreria, es pot vincular la vista al moment de declarar l'atribut:

```
// ATTRIBUTES
@BindView(R.id.subject_fragment_icon)
ImageView ivIcon;

@BindView(R.id.subject_fragment_title)
TextView tvTitle;

// PUBLIC METHODS
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstance) {
    rootView = inflater.inflate(R.layout.subject_fragment, container,  attachToRoot: false);

    ButterKnife.bind( target: this, rootView);

    return rootView;
}
```

Figura 7.2.3.2 - View binding mitjançant la llibreria ButterKnife

Com es pot intuir, quan es treballa amb desenes de vistes l'ús de *ButterKnife* ajuda a obtenir un codi més senzill, ordenat i entenedor.

7.2.4. QuadFlaskColorPicker

Mitjançant la llibreria *QuadFlaskColorPicker*, els usuaris poden escollir el color de les seves Assignatures des d'un selector de colors molt original i intuïtiu, ja que a diferència dels selectors de colors clàssics, aquest només té dues barres per definir lluminositat i transparència. Segons els valors d'aquestes, es mostra una paleta de colors o una altra:

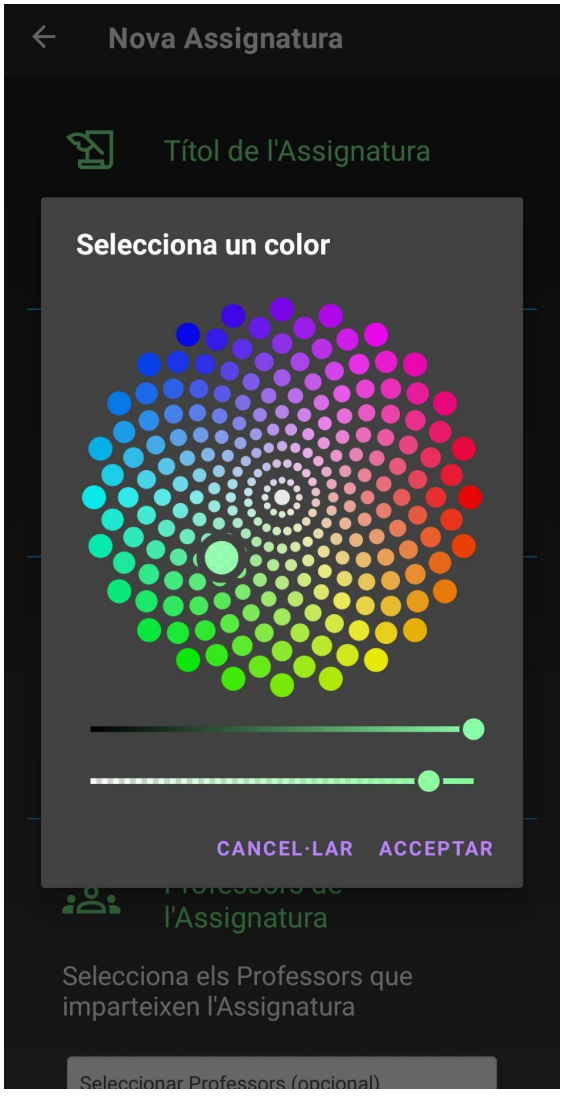


Figura 7.2.4.1 - Diàleg per escollir el color d'una Assignatura

7.2.5. MPAndroidChart

La llibreria *MPAndroidChart* ha sigut l'escollida per representar gràficament l'estat de cada Assignatura. Aquest mòdul proporciona objectes amb els

quals es poden dissenyar gràfics de molts tipus amb un control total sobre els aspectes gràfics.

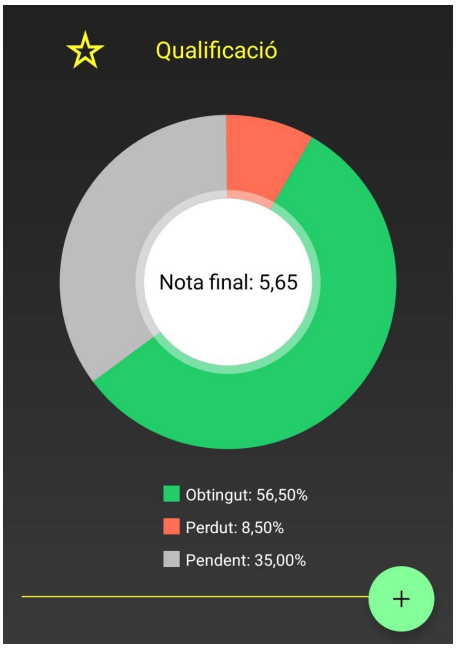


Figura 7.2.5.1 - Exemple d'un PieChart que mostra l'app

7.2.6. Retrofit

Retrofit és un client HTTP per Android i Java que garanteix una connexió segura amb un servidor REST. Més concretament s'ha empleat la versió *Retrofit 2*, que aprofita per defecte *OkHttp* com a capa de xarxa. Aquesta tecnologia serialitza[10] (i desserialitza) els objectes *JSON*[11], que és el format amb el qual el servidor envia (i rep) la informació, i d'aquesta manera el

[10] En *Java*, serialitzar un objecte vol dir convertir un objecte en una seqüència binària, de manera que aquesta pugui ser revertida per obtenir l'objecte inicial.
[11] Els objectes *JSON* (JavaScript Object Notation) és una forma de representar objectes en forma de text. És una de les tecnologies més usades a l'hora d'intercanviar informació entre aplicacions.

servidor i l'aplicació poden intercanviar informació d'una manera molt ràpida i segura.

7.3. Servidor

7.3.1. Spring

Spring és un *framework*[12] que permet crear servidors escalables, modulars, segurs i robustos. Ajuda als programadors a obtenir un codi de molt rendiment, fàcil de testejar i reutilitzable.

La tecnologia més identificativa d'*Spring* és la *Dependency Injection*, un patró de disseny orientat a objectes, en el qual a una classe se li subministren una sèrie d'objectes, en compte de ser la pròpia classe la que els crea. Es podria dir que aquest patró extreu la responsabilitat de la creació d'instàncies d'un component per delegar-la a una classe "injectora". Aquest comportament és el que permet treballar amb dades poc dependents i poc acoblades.

7.3.2. Lombok

El projecte *Lombok* és una llibreria de *Java* que permet als programadors reduir la quantitat de codi base i bàsic que requereix *Java*. En aquest projecte s'ha usat per evitar definir els mètodes coneguts com a *getters* i *setters*. Aquests mètodes són indispensables per treballar amb objectes i la seva

[12] Un *framework* és una eina que proporciona eines, mètodes, components i solucions. Ofereix *software* que permet crear aplicacions i usar funcionalitats, facilitant així el desenvolupament de productes.

feina és simplement retornar i assignar valors als atributs d'un objecte, respectivament.

Si afegim les etiquetes "@Getter" i "@Setter" sobre una classe, *Lombok* se n'ocuparà de proporcionar els mètodes esmentats sense necessitat d'escriure el codi pertinent. A més, aquesta llibreria proporciona també altres mètodes molt útils, com *toString()* per transformar objectes en seqüències de caràcters i *equals()* que permet comparar objectes per determinar si són iguals o no.

L'ús de *Lombok* és crucial en projectes on es treballa amb molts objectes i amb molts atributs, ja que a la llarga permet estalviar moltes línies de codi trivial.

CAPÍTOL 8

Anàlisi i disseny del sistema

En aquest apartat s'explicarà de manera conceptual *què* es vol aconseguir, mentre que al següent capítol s'explicarà *com*.

8.1. Anàlisi del sistema

8.1.1. Aplicació Android

Per seguir la tendència dels últims anys al món de les aplicacions mòbils, la intenció és que el "tema" principal de l'aplicació siguin colors foscos, ja que d'aquesta manera es busca no carregar la vista de l'usuari.

L'objectiu és que els processos per registrar-se, iniciar sessió i tancar-la siguin senzills i ràpids. També es vol que els processos per inserir dades, editar-les i eliminar-les siguin molt semblants entre ells, per tal de poder establir una estructura ben definida i unificada.

La idea és que l'aplicació tingui un menú principal lateral amb cinc pestanyes que englobin tots els apartats:

- Agenda. És la pestanya inicial. Aquesta ha de mostrar ordenades cronològicament les Activitats de l'usuari. Disposarà de dos botons, un que mostrarà/ocultarà les Activitats anteriors a la data actual i un

altre que desplegarà un menú contextual des del qual es puguin crear noves Activitats.

- Tasques. Aquesta pestanya ha de mostrar, també cronològicament ordenades, les Tasques de l'usuari. Hi haurà dos botons, un per mostrar/ocultar les Tasques que l'usuari hagi marcat com a "Realitzades" i un altre que obrirà un menú contextual per crear noves Tasques.
- Assignatures. Des d'aquesta pestanya l'usuari podrà veure totes les Assignatures que ha creat. Disposarà també de dos botons, un per mostrar/ocultar les Assignatures que l'usuari hagi marcat com a "ocultes" i un altre que mostrarà un menú contextual des del qual es podrà crear una nova Assignatura.
- Professors. Aquesta pestanya ha de mostrar tots els Professors que l'usuari hagi registrat. Hi haurà un botó per mostrar/ocultar els Professors que l'usuari hagi marcat com a "ocults" i un altre que desplegarà un menú contextual des del qual es podran crear nous Professors.
- Configuració. La pestanya de Configuració mostrarà un menú amb diferents apartats, des dels quals l'usuari pugui editar el seu perfil, crear un sistema de qualificació personalitzat, canviar l'idioma de l'aplicació i esborrar les seves dades.

8.1.2. Model de dades

Tenint en compte els requisits de l'aplicació i el que es vol oferir a l'usuari, calen les següents classes:

- User: Els Usuaris han de tenir un codi identificador, el nom de l'usuari, el seu correu electrònic i la contrasenya.
- Teacher: Dels Professors ens interessa saber el seu nom, cognoms, nombre de telèfon, correu electrònic, despatx i si l'usuari l'ha marcat com a "deshabilitat" o no. També té un codi identificador.
- Subject: Les Assignatures tenen un codi identificador, un títol, una descripció, un color, una nota final, el percentatge de la nota que l'usuari ha assolit, el percentatge que ha perdut i un atribut per saber si està "deshabilitada" o no.
- Task: Les Tasques tenen un codi identificador, un títol, una data, una anotació i un atribut que indica si ja ha estat o no "completada".
- Notification: De les Notificacions es vol conèixer quan han de ser enviades. Totes tenen també una *id*.
- Activity: Les Activitats tenen un codi identificador, un títol, el percentatge que representen, una data i una anotació.
- Group: Els Grups tenen un codi identificador, un títol, el percentatge que representen, una anotació i el nombre d'Activitats que agrupa.
- Mark: Les Qualificacions tenen un codi identificador, un valor textual, un valor numèric i el sistema de qualificació amb el qual s'avalua.
- PersonalizedMarkValue: De cada un dels valors del sistema de qualificació personalitzat, ens interessa conèixer el seu *id*, el valor textual amb el qual es representa i el valor numèric que té associat.

Els codis identificadors de cada entitat es creen automàticament i l'usuari no els pot veure ni modificar. A part, aquestes entitats poden tenir alguns atributs sense valor, com per exemple si no es coneix el nombre de telèfon d'un Professor o si no es vol donar una descripció a una Assignatura.

Les relacions entre entitats es pot veure a l'apartat 8.2.3.

8.2. Disseny del sistema

8.2.1. Logo i icona de l'aplicació

Per aquest projecte s'han dissenyat un logo de *MyStudentAgenda* que es mostra en algunes pantalles de l'aplicació:

Figura 8.2.1.1 - Logo de MyStudentAgenda en color blau clar

D'aquest logo n'existeixen tres versions, la que es pot veure a la figura 8.2.1.1, una d'igual però en color blau fosc i una altra en color groc. Aquests tres colors han sigut els escollits per ser els principals de l'app.

La icona amb la qual es representa *MyStudentAgenda* està dissenyada utilitzant aquests tres colors:

Figura 8.2.1.2 - Icona de l'aplicació MyStudentAgenda

8.2.2. Interfícies de l'aplicació

A part dels colors foscos amb els quals es juga a totes les pantalles (menys a la de càrrega, la d'iniciar sessió i la de registrar-se), s'ha volgut definir un color representatiu per a cada pestanya, excepte per a la de Configuració, i les seves respectives pantalles. A més, com que a les Assignatures tenen un color assignat, s'utilitza aquest per pintar algunes vistes.

Algunes de les pantalles de l'aplicació són:

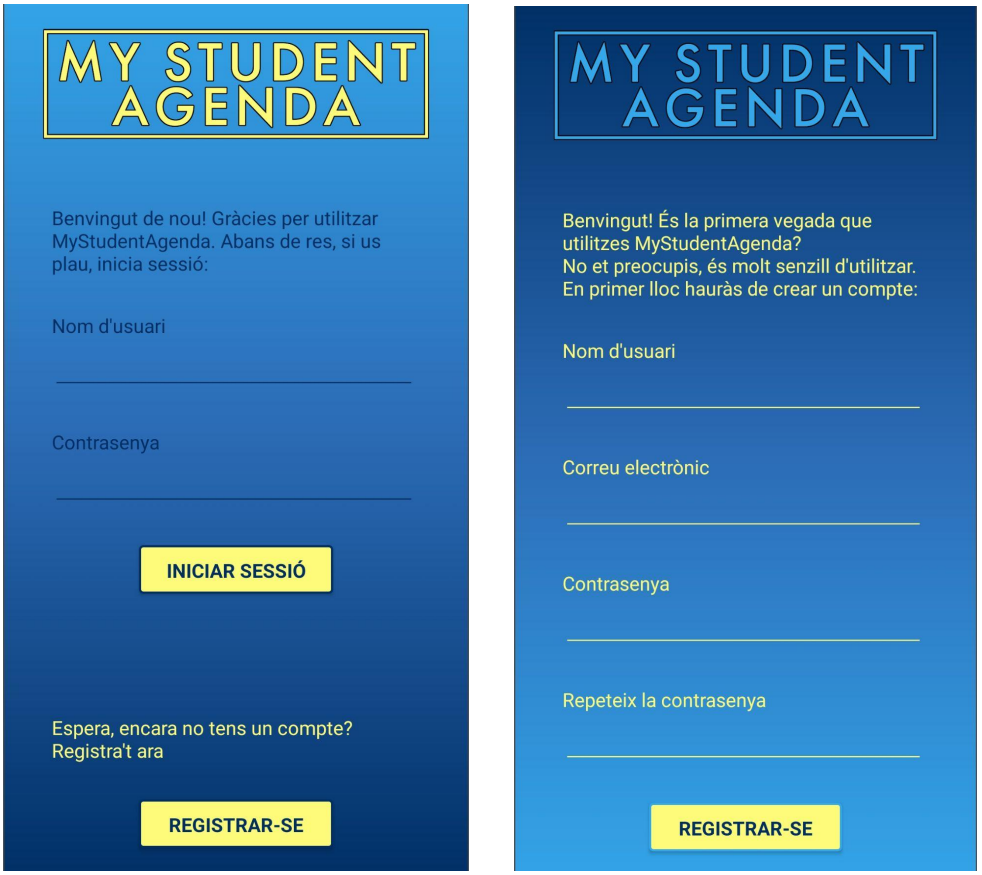


Figura 8.2.2.1 - Pantalles per iniciar sessió (esquerra) i registrar-se (dreta)

Les pantalles de *login* i *register* són molt semblants: es juga amb els mateixos colors i es tracta de procediments molt simplificats. Tot i que a la figura 8.2.2.1 no es vegi per falta d'espai, a la pantalla per registrar-se també hi ha un botó per anar a la pantalla d'iniciar sessió.

Figura 8.2.2.2 - Pantalla inicial de l'aplicació, la que mostra les Activitats (esquerra), i pantalla de Tasques (dreta)

Quan es navega a les pestanyes "Agenda" i "Tasques" es veuen les Activitats i les Tasques, respectivament, de l'usuari. Les dues mostren la informació corresponent en forma de calendari, ometent els dies en els quals no hi ha esdeveniments registrats. A la imatge de l'esquerra de la figura 8.2.2.2 s'estan mostrant només les Activitats la data de les quals és posterior a la del moment de capturar la imatge. En canvi, a la imatge de la dreta està activada l'opció de mostrar totes les Tasques, sense diferenciar entre les que ja estan completades i les que no.

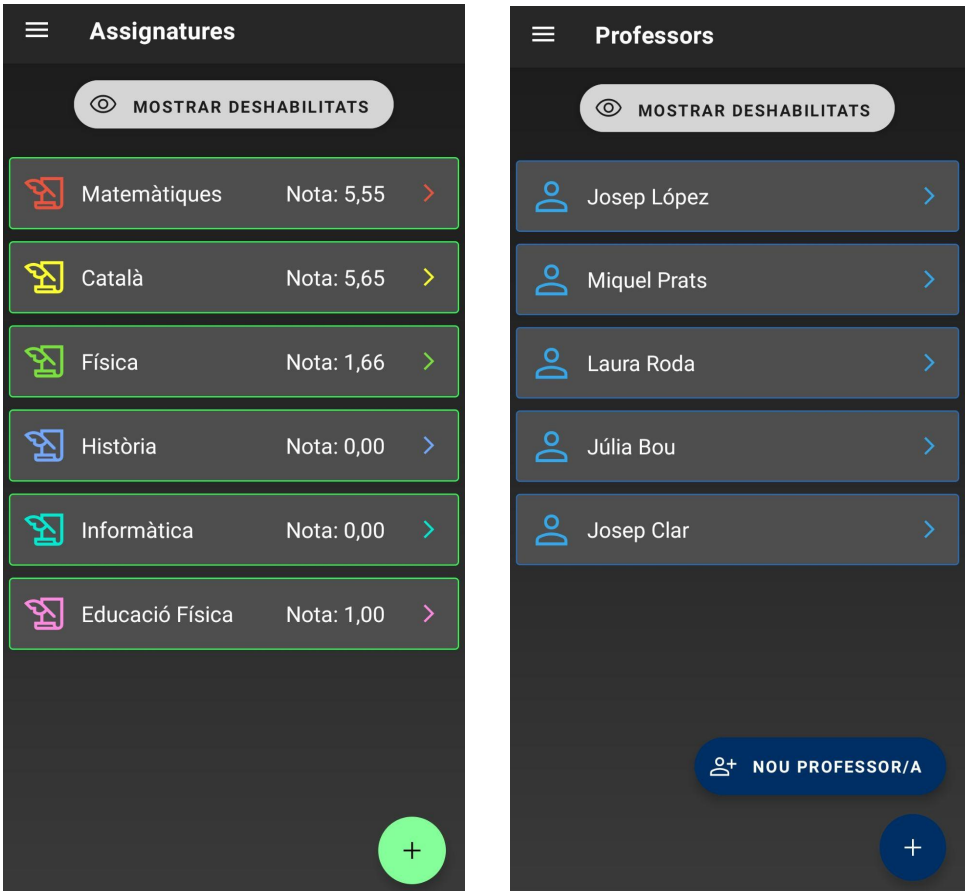


Figura 8.2.2.3 - Pantalla d'Assignatures (esquerra) i pantalla de Professors (dreta)

Quan es navega a les pestanyes "Assignatures" i "Professors" es veuen les Assignatures i els Professors, respectivament, de l'usuari. Aquestes pantalles mostren en forma de llistat les dades que pertoquen a cada una en funció de si l'usuari vol veure-les o no.

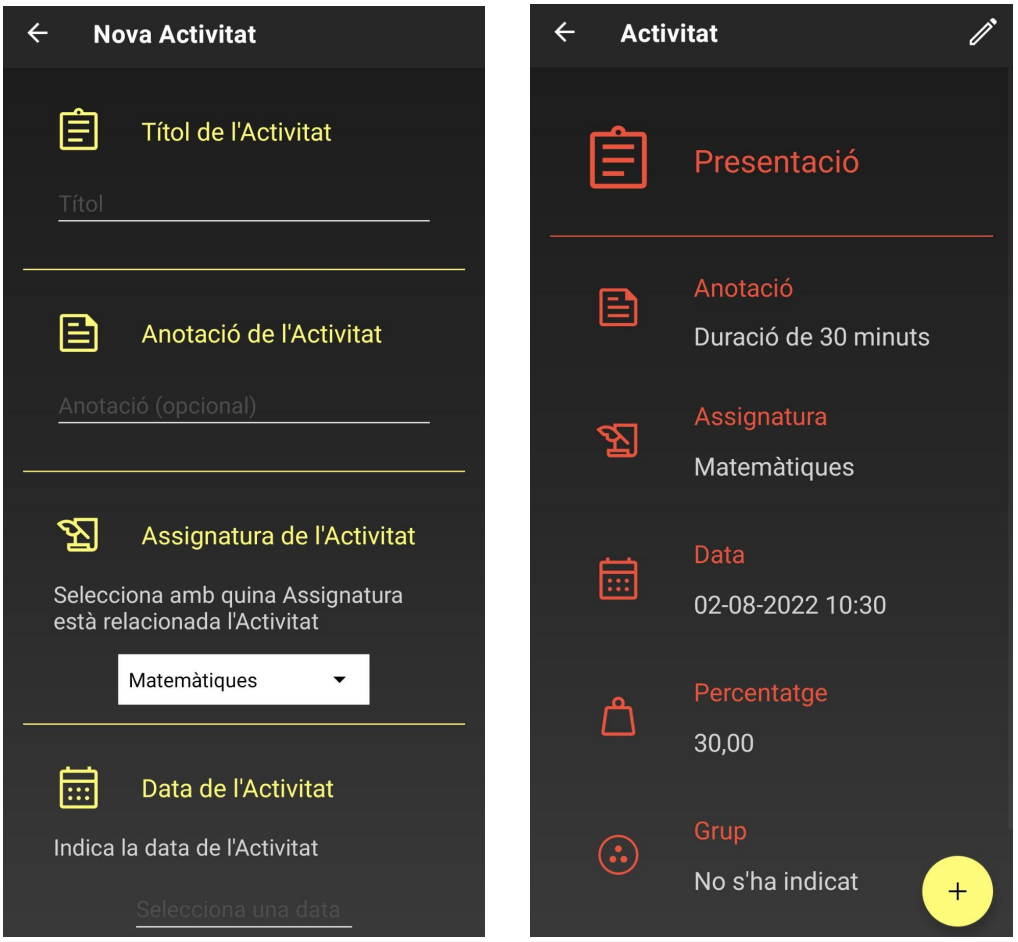


Figura 8.2.2.4 - Pantalles per crear una nova Activitat (esquerra) i per consultar la seva informació (dreta)

Totes les pantalles per inserir dades i veure informació segueixen el mateix estil que es pot veure a la figura 8.2.2.4. Són pantalles amb *scroll* i, per tant, no es pot apreciar tot el seu contingut, però sí que serveix per fer-se'n una idea.

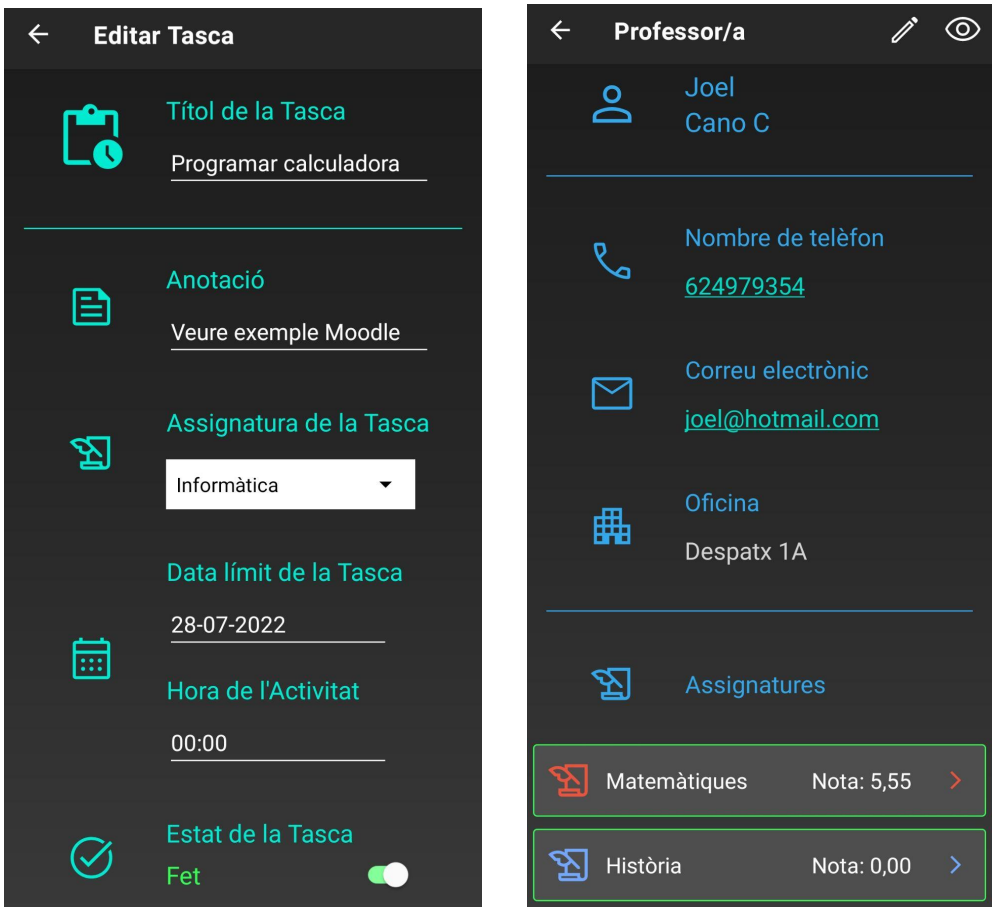


Figura 8.2.2.5 - Pantalles per editar una Tasca (esquerra) i per veure la informació d'un professor

Les pantalles per editar informació també comparteixen un mateix estil. A la part inferior d'aquestes hi ha un botó per guardar els canvis i un altre per eliminar les dades completament.

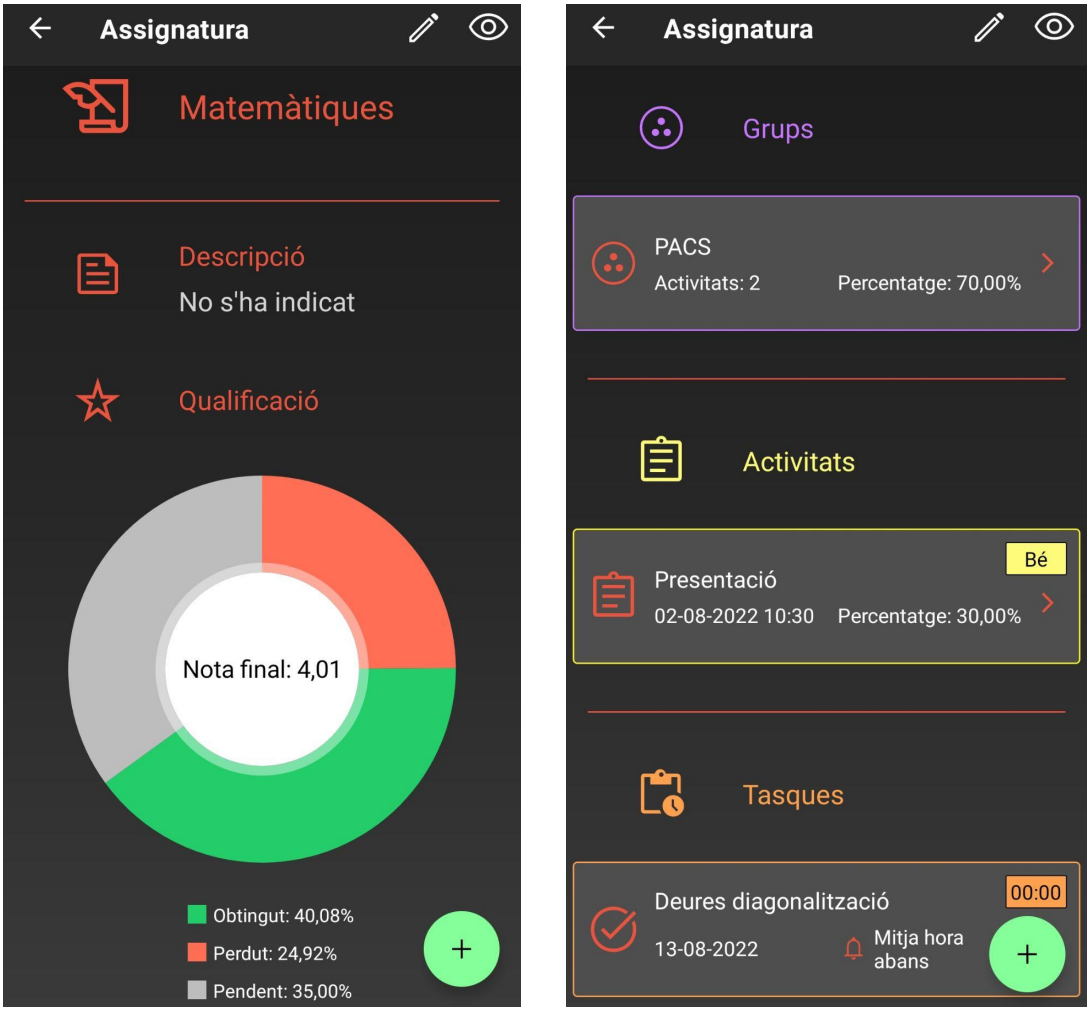


Figura 8.2.2.6 - Pantalla d'informació d'una Assignatura

La pantalla d'informació d'una Assignatura és on es centralitza la gran quantitat de dades que es gestiona a l'aplicació. Des d'aquesta es poden veure la nota actual de l'assignatura, el seu nom, descripció, professors que l'imparteixen, les seves Activitats i Grups i les seves Tasques. Aquesta informació és també navegable i editable.

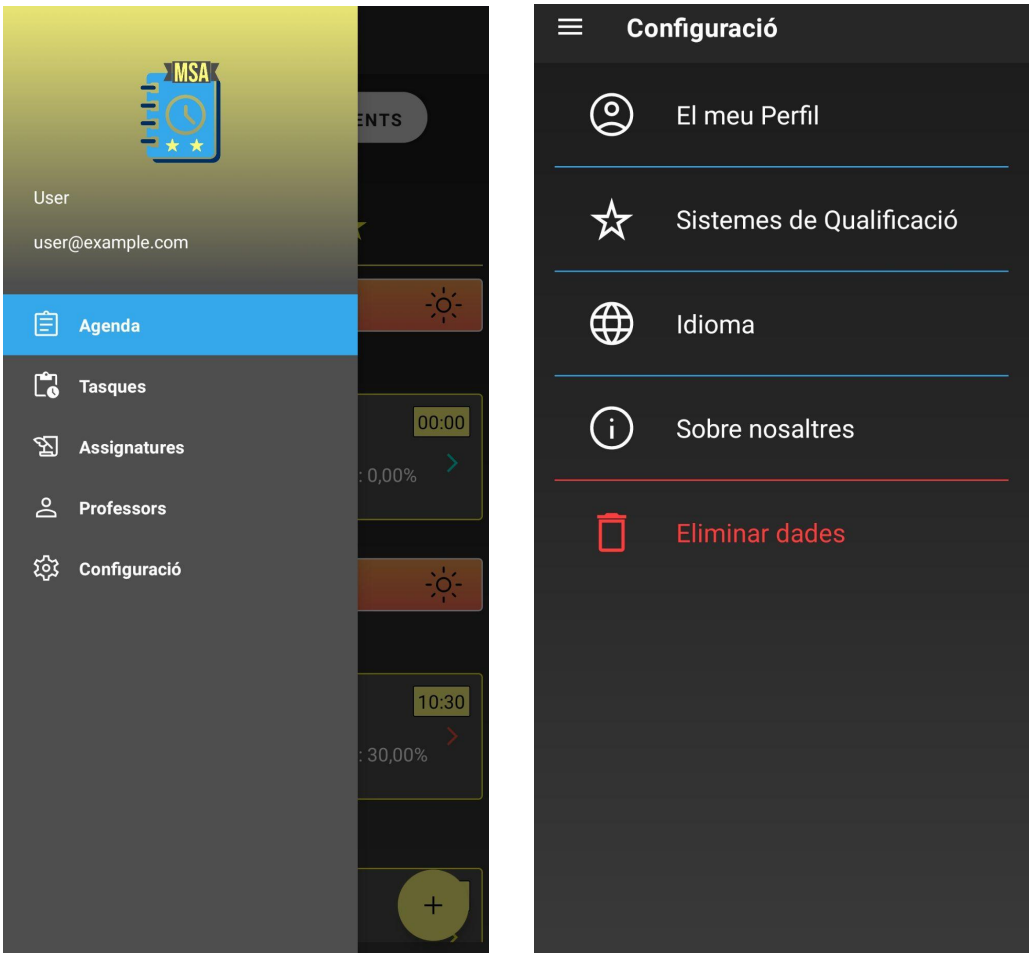


Figura 8.2.2.7 - Menú lateral (esquerra) i menú de Configuració (dreta)

El menú lateral es pot desplegar mitjançant el botó representat amb tres línies horitzontals col·locat a la part superior esquerra de la majoria de pantalles de l'aplicació.

8.2.3. Model de Base de Dades

Considerant les dades que es volen guardar (descrites a l'apartat 8.1.2), s'ha definit el següent model:

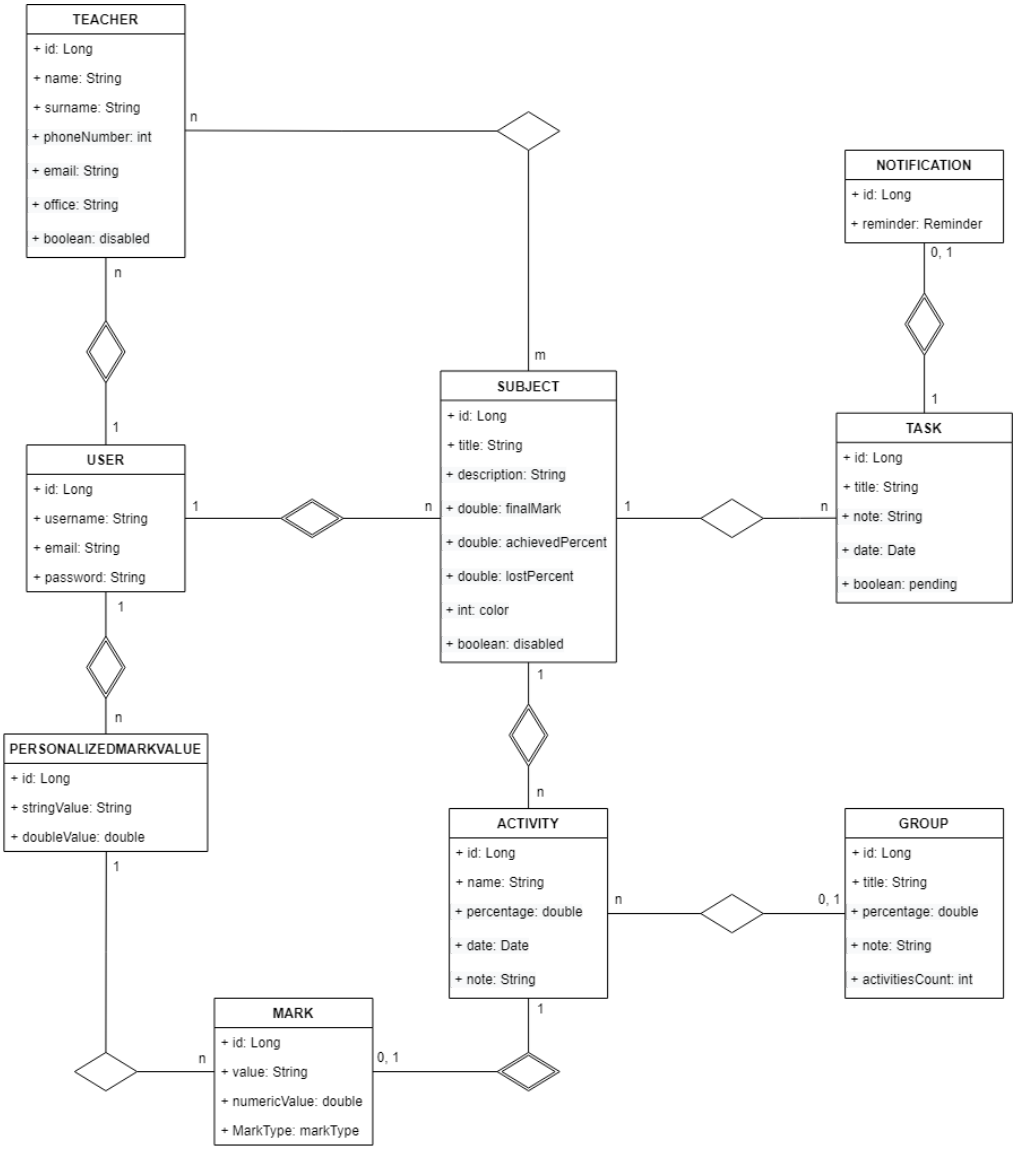


Figura 8.2.3.1 - Esquema de la base de dades de MyStudentAgenda

CAPÍTOL 9

Implementació i proves

Al tractar-se aquest d'un projecte que s'emmarca com a Projecte Final de Grau, s'ha desenvolupat per a que funcioni de manera "local". És a dir, el servidor i les dades que aquest conté, només existeixen a la màquina que l'executa, no es guarden a cap servidor d'Internet. És important entendre que això crea una dependència entre la màquina que fa córrer el servidor i el dispositiu que executa l'aplicació Android, ja que aquests han d'estar connectats en tot moment.

Tenint això en compte hi ha dues maneres possibles de fer servir l'aplicació:

- La primera seria mitjançant un emulador de dispositius, com el que porta integrat *Android Studio*, per tal de crear imatges (dispositius) que es connectin mitjançant el port 8080, que és el port TCP[13] destinat a ser el port de proves per a connexions HTTP.
- La segona és connectant un dispositiu físic amb la màquina a través d'un cable USB i alhora connectar-lo també al port 8080.

Aquesta dependència crea una limitació òbvia, doncs per poder accedir a les dades i usar l'aplicació cal estar connectat "físicament" al servidor. L'alliberació d'aquesta restricció es tractarà amb més detall a l'apartat 12.

[13] Protocol de control de transmissió. Es tracta d'un protocol d'Internet que crea connexions entre dispositius per a que aquests puguin enviar i rebre dades.

9.1. Implementació

Durant el procés d'implementació han sorgit alguns problemes i contratemps. En aquest apartat es mencionen els més destacables.

9.1.1. Idioma de l'aplicació i SharedPreferences

Una de les primeres dificultats que va aparèixer va ser el de guardar la selecció d'idioma de l'usuari. Per defecte, l'aplicació utilitza el llenguatge predeterminat del dispositiu que l'executa i, tot i que al moment de canviar l'idioma tot semblava funcionar correctament, aquesta configuració es reiniciava al tancar l'app. En aquest punt, després de realitzar una petita investigació, es va fer ús de les *SharedPreferences*.

L'objecte *SharedPreferences* apunta a un arxiu que conté parelles clau-valor i proporciona mètodes senzills per llegir-lo i escriure-hi. Aquestes dades es guarden a la memòria interna del dispositiu i no es veuen afectades pels diferents estats que pot prendre una *Activity*. Això permet guardar algunes preferències de l'usuari i que aquestes es mantinguin fins i tot quan es "destrueix" l'aplicació, d'aquesta manera no cal que l'usuari configuri certs aspectes cada vegada que accedeix a l'app.

Per solucionar aquesta problemàtica, doncs, es va definir una constant global "LANGUAGE_KEY" que es fa servir com a clau per al valor de la configuració de l'idioma. A partir d'aquesta clau, cada vegada que s'inicia l'aplicació es llegeix el valor de l'objecte al qual apunta i d'aquesta manera s'estableix l'idioma preferit de l'usuari. Des de la pantalla de selecció del

llenguatge es sobreescriu el valor de l'objecte quan hi ha algun canvi. Aquestes operacions es realitzen mitjançant els mètodes estàtics de la classe *LanguageHelper*.

Ja coneixent la potència d'aquesta eina, es va decidir utilitzar-la també per a guardar l'estat dels botons que hi ha a les pestanyes "Agenda", "Tasques", "Assignatures" i "Professors", des dels quals els usuaris poden escollir quina informació volen que es mostri a la pantalla.

A l'apartat 14 s'explica amb més profunditat com ha sigut la implementació d'aquestes configuracions mitjançant les *SharedPreferences*.

9.1.2. Tasques asíncrones

En el desenvolupament d'aplicacions és molt important que l'usuari entengui què està fent l'app en tot moment. Si és necessari executar operacions complexes o que es poden demorar uns segons, els usuaris n'han d'estar assabentats per evitar problemes i funcionaments inesperats.

En el cas de *MyStudentAgenda*, quan es realitzen operacions d'escriptura/lectura d'informació al servidor es mostra una finestra amb un cercle de càrrega, de manera que no es poden portar a terme altres moviments fins que el que s'està executant acaba.

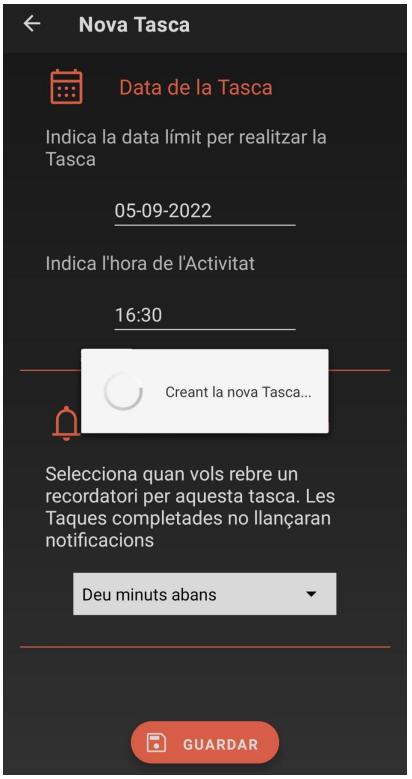


Figura 9.1.2.1 -Cercle de càrrega que es mostra quan s'està creant una Tasca

Per aconseguir aquest comportament s'ha optat per definir una classe abstracta *MSAAsyncTask* en comptes de fer servir la popular *AsyncTask* d'Android, ja que aquesta darrera es troba *deprecated*[14] a partir de la versió *R*.

Per la implementació d'aquesta s'utilitza un objecte *ExecutorService*, a partir del qual es defineixen tres mètodes que sempre s'executen en l'ordre següent:

[14] Que una classe o un mètode estiguin obsolets vol dir que ja no tenen suport oficial, és a dir, ja rebran actualitzacions ni es solucionaran possibles problemes. Tot i que es poden seguir utilitzant, és recomanable evitar aquest tipus de situacions.

- onPreExecute(): Aquest mètode es crida just abans que s'executi *doInBackground()* i generalment es fa servir per mostrar el cercle de càrrega que s'ha comentat anteriorment.
- doInBackground(): En aquest mètode es realitzen operacions *en background*, és a dir, en segon pla.
- onPostExecute(): Aquest mètode es crida just quan el mètode *doInBackground()* acaba la seva execució. És aquí on s'amaga el cercle de càrrega i es pot mostrar un missatge indicant que l'operació ha sigut exitosa.

9.1.3. Enums

Per definir quin tipus de recordatori té cada *Notification* o sobre quin sistema de qualificació s'ha ponderat cada *Mark*, s'han definit els tipus *Reminder* i *MarkType* (es poden veure més en detall a l'apartat 14).

Els *Enums* són un tipus de dada especial que permeten a una variable comportar-se com una constant. S'ha optat per utilitzar aquest tipus en comptes de simples constants perquè als *Enums* s'hi poden afegir mètodes. Aquests han sigut de gran utilitat, ja que, juntament amb les classes *ReminderHelper* i *MarkTypeHelper*, s'han pogut obtenir els valors textuals en funció de l'idioma de l'app per cada *Enum* per mostrar-los.

9.1.4. Notificacions

Per tal d'establir, editar i eliminar les notificacions que defineixen els usuaris, s'ha creat una classe *NotificationService* que hereta de la classe *BroadcastReceiver* i reserva un canal de notificacions per l'aplicació. Un cop determinat aquest canal, s'utilitza una instància d'un objecte *AlarmManager* per establir notificacions amb un títol, un cos i una *id*. És a través d'aquesta *id* que es pot fer la gestió per modificar o eliminar notificacions prèviament configurades.

9.1.5. Operacions atòmiques

Per evitar que al servidor hi hagi dades inconsistents i evitar possibles errors, els mètodes que escriuen a la base de dades estan etiquetats amb *@Transactional*. Aquesta anotació indica que una operació és atòmica, és a dir, que o bé s'executen totes les línies de codi correctament o bé no es persisteix res. D'aquesta manera podem evitar casos on, per exemple, si mentre s'està assignant un Professor a una Assignatura es vincula en aquesta direcció, però abans que es vinculi en l'altre s'apaga el dispositiu, tindríem que un Professor està assignat a una Assignatura però a aquesta no li consta que li hagin assignat aquest Professor.

9.2. Proves

Durant tot el procés d'implementació de l'app, ja amb el servidor funcional, s'ha treballat amb un *sample set* (joc de proves) generat automàticament.

Aquestes dades s'insereixen a la base de dades cada vegada que el servidor es reinicia, d'aquesta manera no es treballa amb un cas buit des de l'aplicació i es poden fer proves més realistes.

Més enllà dels càlculs que es realitzen a les Assignatures quan es qualifiquen Activitats (d'això es fa una demostració a l'apartat 10.1), s'ha invertit molt de temps en comprovar que les Notificacions es poden crear, editar i eliminar correctament. Com que aquestes dues últimes operacions són difícilment demostrables per escrit, en aquest apartat es demostrarà com és el procés de configurar i rebre una notificació per a una Tasca.

Primerament, crearem una tasca "Demostració" amb data límit el 31/08/2022 a les 18:19:

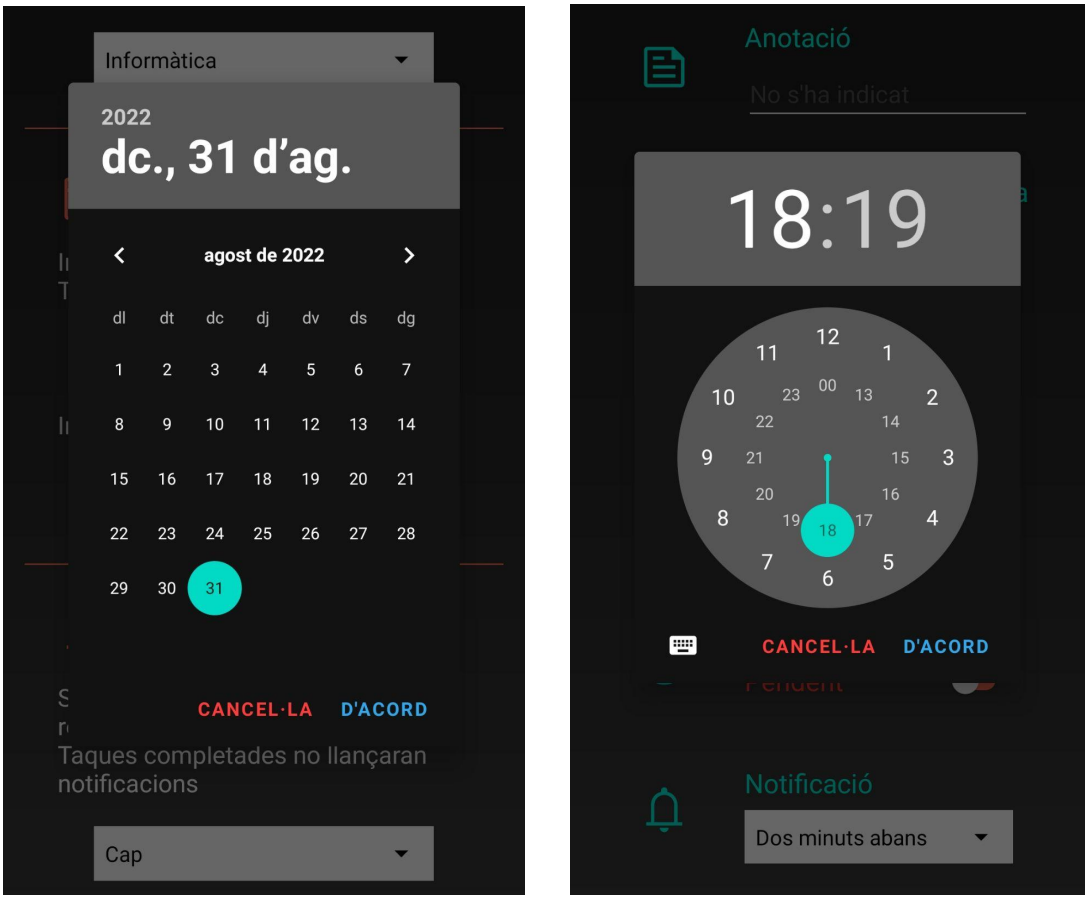


Figura 9.1.1 - Diàleg per establir la data límit d'una Tasca (esquerra) i diàleg per establir l'hora límit d'una Tasca (dreta)

A continuació editarem la Tasca i afegirem una Notificació perquè aquesta s'enviï dos minuts abans d'arribar a la data indicada prèviament:

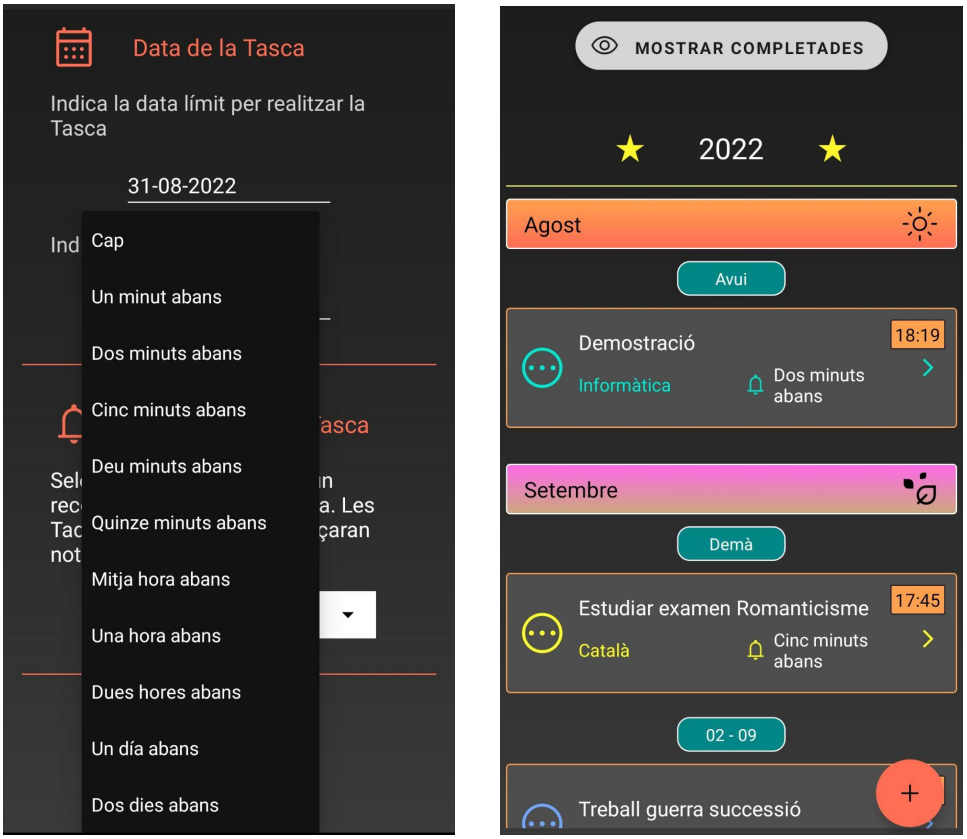


Figura 9.1.2 - Diàleg per escollir quan es vol rebre la Notificació per la Tasca (esquerra) i vista de la Tasca un cop ha estat creada (dreta)

Arribat el moment, és a dir, el dia 27/08/2022 a les 18:17, arriba aquesta notificació al dispositiu:

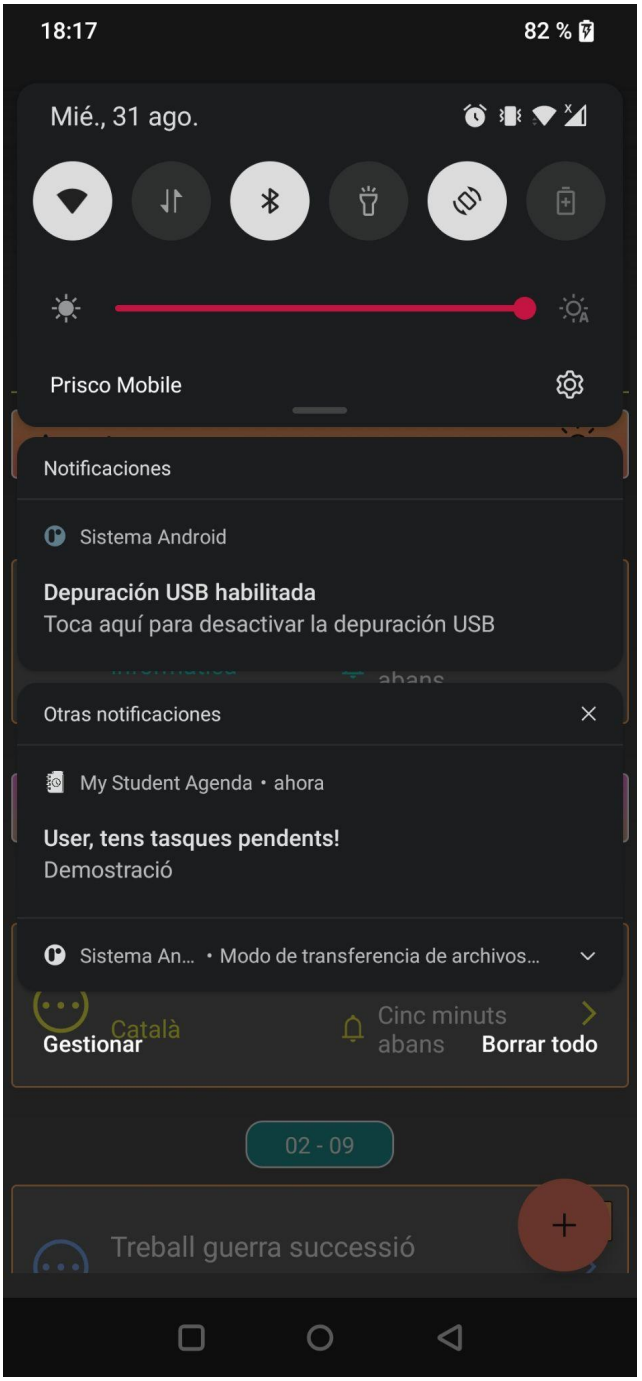


Figura 9.1.3 - Vista de la Notificació rebuda

CAPÍTOL 10

Implantació i resultats

10.1. Implantació

El procés de desenvolupament ha sigut bastant lineal i la planificació descrita a l'apartat 6 ha permès una distribució equitativa de la feina al llarg dels mesos.

10.1.1. Servidor

Per implementar el servidor primer es va definir un *package*[15] "domain" on es troben la classe *Global* (que crea les dades d'exemple), l'arxiu *Constants.java* (on es defineixen les constants del sistema), els *Enums* comentats anteriorment a l'apartat 9.1.3 i una classe *MSALogger* que mostra missatges a la terminal del servidor en funció de les operacions que es realitzen a la base de dades.

Seguidament, es va definir el paquet "entity" que conté totes les classes de les entitats de l'aplicació. Posteriorment, es va crear el *package* "repository" amb els repositoris de cada entitat.

A continuació es van implementar les classes *Service* per a cada entitat un paquet "service". Per comprovar que els mètodes funcionessin correctament es van fer algunes proves sobre la base de dades. Per conèixer

[15] Paquets de classes. D'aquesta manera un programa pot utilitzar classes amb el mateix nom sempre que aquestes estiguin en diferents *packages*.

en tot moment l'estat de la informació al servidor, es va fer servir el programa *HeidiSQL*.

Una vegada acabats els serveis es van definir els controladors. En un *package* "controller" es va definir un *Controller* per cada entitat. Per testejar l'API de *MyStudentAgenda* es van definir les crides en un fitxer *JSON* i es van realitzar proves de funcionament amb el programa *Postman*.

10.1.2. Aplicació

Quan el servidor ja estava preparat per funcionar i respondre les peticions necessàries, es va començar el procés d'implementació de l'aplicació.

El primer sprint es va dedicar íntegrament en configurar alguns paràmetres de l'aplicació i establir un sistema de gestió de sessions per poder iniciar sessió, registrar-se i tancar sessió. A més es va fer ús de la llibreria *PersistentCookieJar* (apartat 7.2.3) per evitar haver d'iniciar sessió cada vegada que es volia fer servir l'aplicació.

10.1.2.1. Pestanya Configuració

La pestanya "Configuració" va ser la primera que es va implementar (excepte l'apartat de "Sistemes de Qualificació", que es va afegir pràcticament al final). Quan es selecciona aquesta pestanya es mostra un menú amb diferents etiquetes que porten a diferents pantalles:

- El meu perfil: Els usuaris poden modificar el seu nom, correu i contrasenya. Per poder editar les seves dades sempre han d'introduir la seva contrasenya actual.
- Sistemes de Qualificació: Els usuaris veuen quin valor numèric representa cada ponderació dels sistemes de qualificació integrats a *MyStudentAgenda*. També poden definir la seva pròpia classificació, podent inserir fins a 15 valors.
- Idioma: Els usuaris poden canviar l'idioma de l'aplicació. Les llengües disponibles són el català, el castellà i l'anglès.
- Sobre nosaltres: S'explica quin projecte que hi ha al darrere de l'aplicació i es facilita una adreça de contacte perquè els usuaris puguin reportar errors, sol·licitar funcionalitats, etc.
- Eliminar dades: Hi ha tres botons. Un elimina tots els Professors, l'altre elimina totes les Assignatures (i, per tant, totes les Tasques, Activitats, Grups, Notificacions,...) i l'últim elimina l'usuari i totes les seves dades. Aquesta pantalla s'ha implementat per complir amb una normativa (que de moment només és per iOS, però tard o d'hora arribarà a Android) que dicta que totes les aplicacions han d'oferir la possibilitat d'eliminar les dades de l'usuari des de la mateixa aplicació.

10.1.2.1. Pestanya Professors

Un cop els usuaris podien accedir a les configuracions, es va implementar la pestanya de "Professors" i totes les pantalles corresponents. Per poder mostrar el llistat de *Teachers* es va crear una classe *TeacherViewAdapter* que

hereta de *RecyclerView.Adapter<RecyclerView.ViewHolder>*. La pantalla principal de la pestanya utilitza una instància d'aquest *adapter*[16] per mostrar els Professors en forma de llista. Aquesta vista s'infla (es carrega) dinàmicament cada vegada que es visita la pantalla, ja que d'aquesta manera s'actualitzen les dades en cas que s'hagin modificat.

Es van dissenyar i implementar també les modals per crear nous Professors, editar-los i eliminar-los. Al ser les entitats amb menys pes a l'aplicació, la dificultat d'aquesta part no va ser gaire alta.

10.1.2.2. Pestanya Assignatures

Quan ja es podien crear, editar i eliminar *Teachers*, va començar un *sprint* sencer (dues setmanes) dedicat únicament a implementar totes les pantalles necessàries per gestionar les Assignatures.

Es va crear la classe *SubjectViewAdapter*, semblant a l'*adapter* vist anteriorment però aquesta vegada ajustat a les *Subjects*. La pantalla principal de la pestanya és molt semblant a la dels Professors i s'usa la mateixa tecnologia.

La pantalla d'informació de l'Assignatura, en canvi, inclou moltes vistes i implementar-la va ser un procés molt llarg. Primer es va utilitzar la llibreria *MPAndroidChart* (vegeu apartat 7.2.5) per mostrar una gràfica amb les notes de l'Assignatura, i després, a mesura que s'avançava en les altres pestanyes, es van crear tres adaptadors nous (*ActivityViewAdapter*, *GroupViewAdapter* i

[16] Un *adapter* és un mecanisme d'Android que actúa com a pont entre les dades de l'aplicació i les vistes contingudes en un *RecyclerView*, *ListView*, etc.

TaskViewAdapter) i es va reaprofitar el *TeacherViewAdapter* per mostrar les Activitats, els Grups, les Tasques i els Professors assignats a l'Assignatura, respectivament.

Tot plegat, són cinc vistes dinàmiques que s'han de carregar quan s'entra a veure una Assignatura i que han d'estar subjectes a rebre actualitzacions en funció de les accions de l'usuari. Per exemple, si un usuari elimina una Activitat amb certa ponderació i que es trobava dins d'un Grup, s'han d'actualitzar les vistes del Grup, les seves Activitats i la qualificació global de l'Assignatura.

10.1.2.3. Pestanya Agenda

Per mostrar les Activitats en forma de calendari es va definir el ja mencionat *ActivityViewAdapter*. A diferència dels adaptadors definits fins al moment, aquest va ser més delicat perquè es van haver de definir diferents tipus per a permetre diferents vistes (per afegir separadors pels anys, mesos i dies).

A les pantalles per crear i editar Activitats es llancen *dialogs* per escollir la data i l'hora de cada una. El selector d'Assignatures no mostra les deshabilitades i si s'intenta establir un percentatge que suposaria que la suma d'*Activities* d'una *Subject* superés el 100%, la inserció no es podrà realitzar.

L'edició de les Qualificacions de les Assignatures es tractarà a l'apartat 10.1.2.7.

10.1.2.4. Pestanya Tasques

Semblant a la implementació de l'apartat anterior: es defineix l'adaptador *TaskViewAdapter* i es permet la creació de diferents tipus.

També a la pantalla de creació/edició es mostren els diàlegs per definir data/hora i el selector d'Assignatura no mostra les deshabilitades. A diferència de les Activitats, però, les Tasques ni poden pertànyer a un Grup ni tenen una Qualificació, tot i que poden tenir Notificacions associades.

El comportament de les Tasques amb les Notificacions és el següent:

- Una Tasca pot tenir, o no, una Notificació associada.
- Si una Tasca no té una Notificació, se li pot assignar una per rebre un recordatori uns minuts abans.
- Si una Tasca té una Notificació programada però la Tasca es marca com a "feta", el recordatori es silencia i no es rep.
- Si una Tasca té una Notificació programada però la Tasca passa de l'estat "fet" a "pendent", es reactiva la Notificació.
- Si una Tasca té una Notificació però s'edita o bé la data límit de la Tasca o bé el recordatori de la Notificació, s'edita el recordatori que s'havia programat.
- Si una Tasca té una Notificació però s'eliminen una o altre, el recordatori es destrueix.

10.1.2.5. Notificacions

Durant el procés d'implementació de les *Notifications* es va dur a terme un petit procés de recerca per esbrinar quin sistema era l'òptim per programar recordatoris.

Com s'explica a l'apartat 9.1.4, es va implementar una classe *NotificationService* que reserva un canal de notificacions al dispositiu i amb l'ajuda dels objectes *NotificationManager* i *AlarmManager* s'obté un control total sobre cada recordatori que es programa a través del seu *notificationId*.

Les notificacions mostren la icona de l'aplicació (sense el cercle del fons), un títol per defecte i un contingut que indica a quina Tasca pertany la Notificació.

10.1.2.6. Sistema de Qualificació Personalitzat

Durant l'*sprint* S9 es van implementar totes les pantalles i modals relacionades amb les Qualificacions. Durant la primera part d'aquest, es va desenvolupar un sistema que permet als usuaris definir un Sistema de Qualificació propi amb fins a 15 valors diferents. Aquests valors estan formats per una part "alfabètica" i una numèrica:

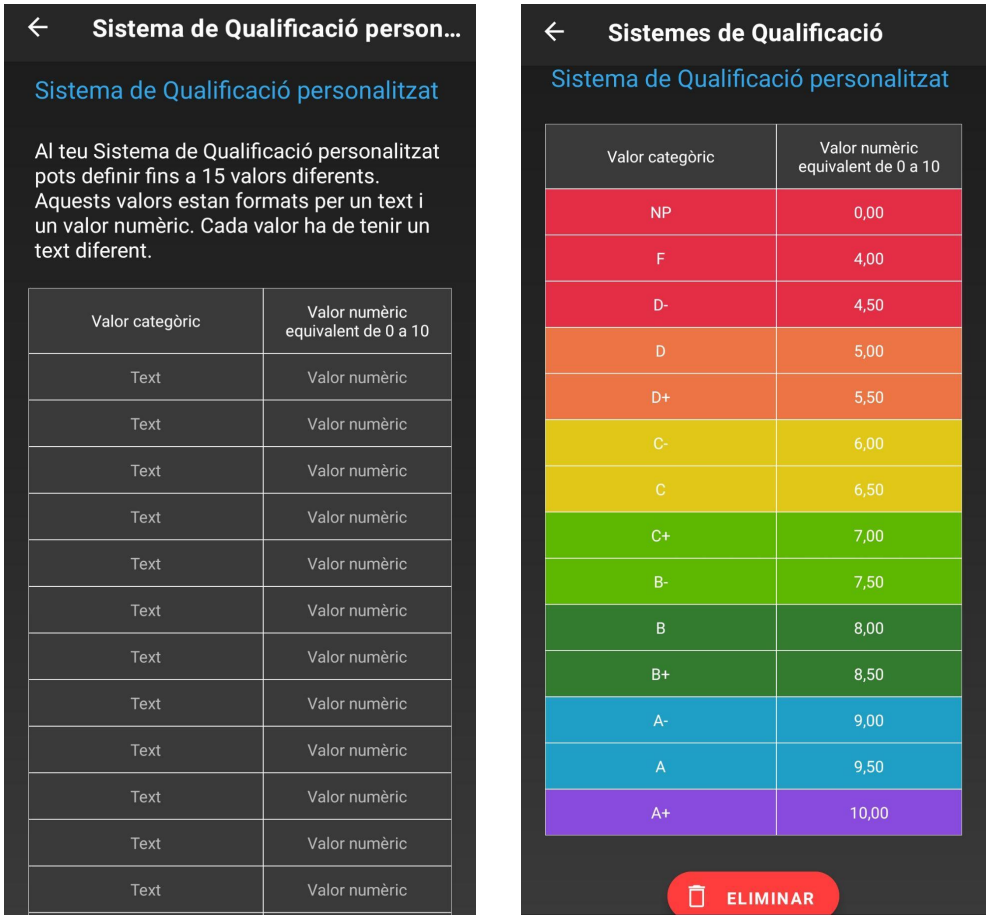


Figura 10.1.2.6.1 - Pantalla des de la qual els usuaris poden definir el seu sistema de qualificació personalitzat (esquerra) i vista del sistema definit (dreta)

Aquest sistema no es pot editar un cop definit, si es vol fer alguna modificació s'ha d'eliminar (les Qualificacions que l'usin com a sistema, perdran el seu valor) i tornar a declarar.

10.1.2.7. Qualificacions

La segona part de l'*sprint* S9 es va dedicar a dissenyar la modal del diàleg per assignar *Qualifications* a les *Activities*.

Si des del menú de vista d'una Activitat s'obre el menú contextual del botó flotant que hi ha a la part inferior dreta, apareix un botó per crear o editar la Qualificació de l'Activitat. Si es selecciona un sistema de qualificació numèric, es mostra un *EditText* el qual obre el teclat numèric. En canvi, si es selecciona un sistema de qualificació "categòric", els possibles valors d'aquest es despleguen mitjançant un *Spinner*.

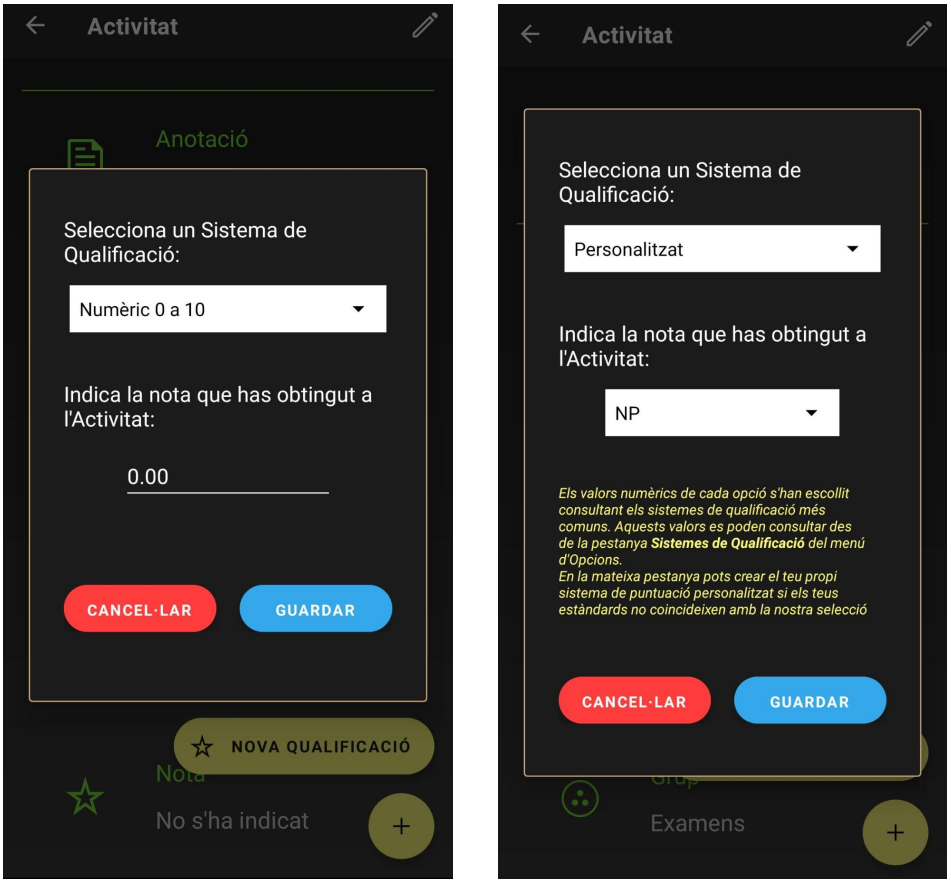


Figura 10.1.2.7.1 -Modal de creació d'una Qualificació numèrica (esquerra) i la mateixa modal al seleccionar un sistema categòric (dreta)

10.1.2.8. Quality Assurance

Quan es va considerar l'aplicació com a completada, es va iniciar un procés de depuració i correcció d'errors. Més enllà d'un error que causava que el comptador d'Activitats dels Grups no s'actualitzés quan s'eliminava una *Activity*, es van fer algunes correccions en certs *layouts* per corregir algunes vistes i alguna falta d'ortografia.

10.2. Resultats

Per tal de demostrar que l'aplicació funciona correctament, es realitzarà una *demo* on es posarà a prova l'aspecte clau i característic de l'aplicació. Es crearà una Assignatura, se li assignaran unes Activitats i aquestes es qualificaran per comprovar que els càlculs es duguin a terme correctament.

L'Assignatura tindrà el títol "Química", deixarem la descripció buida i el seu color serà un rosa claret. A l'Assignatura se li assignaran un parell de Professors creats prèviament.

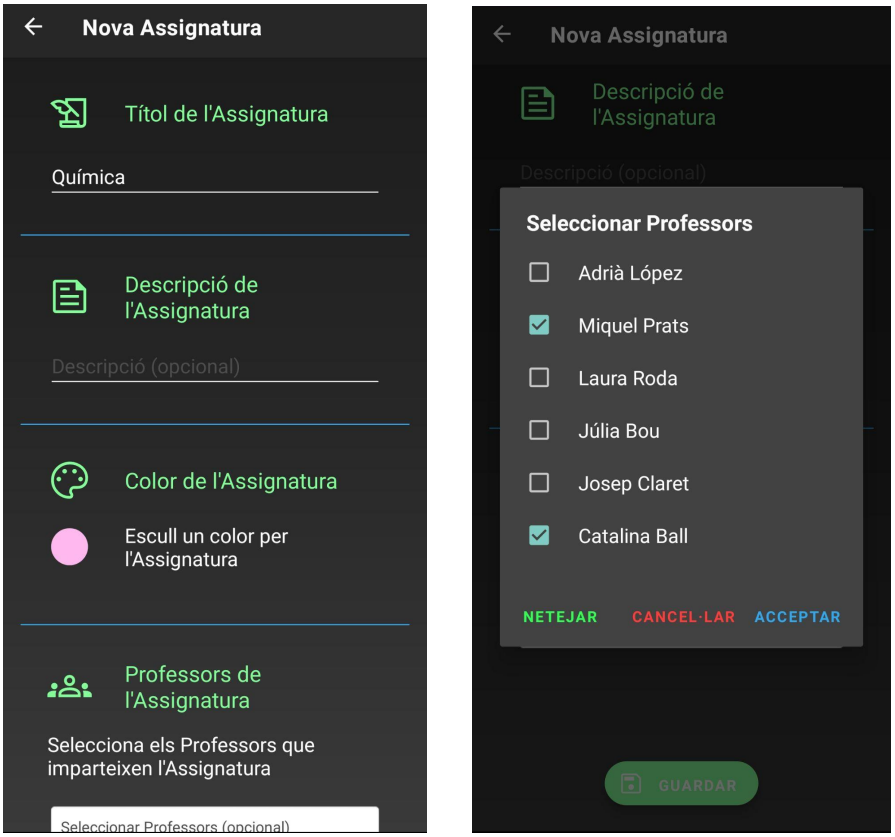


Figura 10.2.1 - Creació de l'Assignatura (esquerra) i selecció de Professors (dreta)

Una vegada creada l'Assignatura crearem un grup "Pràctiques" amb descripció "Pràctica 1 i Pràctica 2". Per obrir el menú de creació de Grups simplement hem de fer *click* sobre el botó flotant que es troba sempre a la part inferior dreta de la pantalla d'una Assignatura.

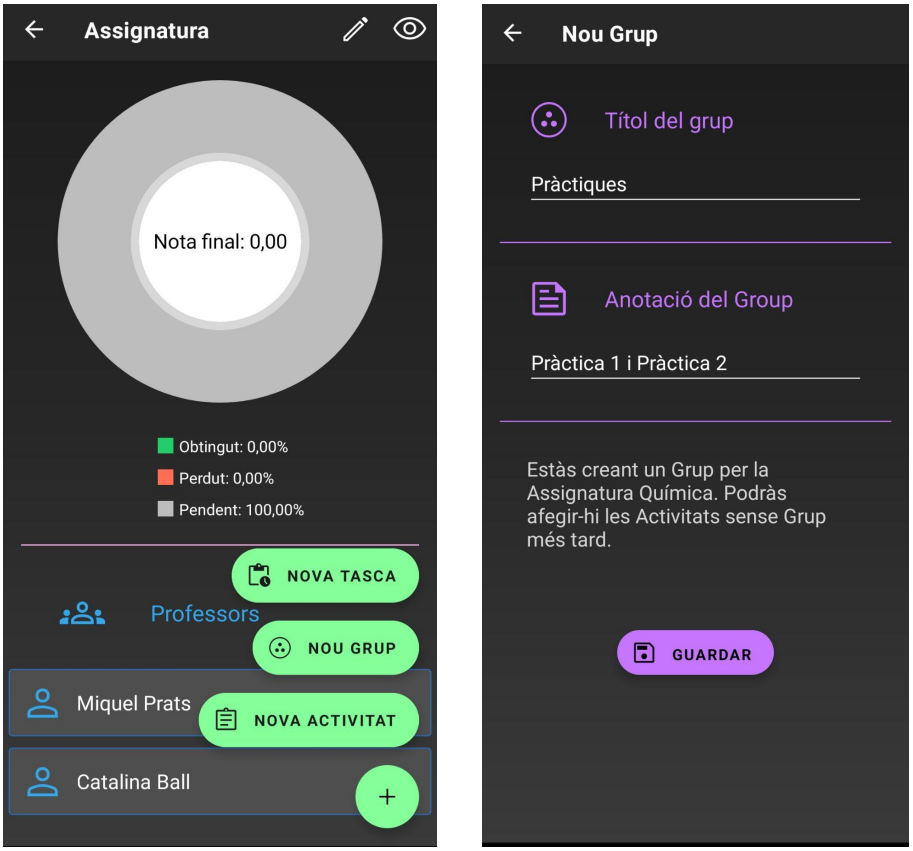


Figura 10.2.2 - Menú contextual a la pantalla d'una Assignatura (esquerra) i creació del Grup (dreta)

A continuació crearem tres Activitats:

- Pràctica 1: Amb data 20/09/2022 i un pes del 20%.
- Pràctica 2: Amb data 14/10/2022 i un pes del 30%.

- Examen final: Amb data 29/10/2022 i un pes del 50%.

Aquestes Activitats es poden crear bé des de la pestanya d'Activitats o bé des de la pantalla de l'Assignatura a la qual els assignarem.

Afegim les Activitats "Pràctica 1" i "Pràctica 2" al Grup "Pràctiques". Per fer-ho entrarem a la pantalla del Grup i desplegarem el menú contextual per veure un botó amb el text "Afegir Activitat". Aquest obre una vista des d'on es poden escollir quines Activitats es vol afegir al Grup.

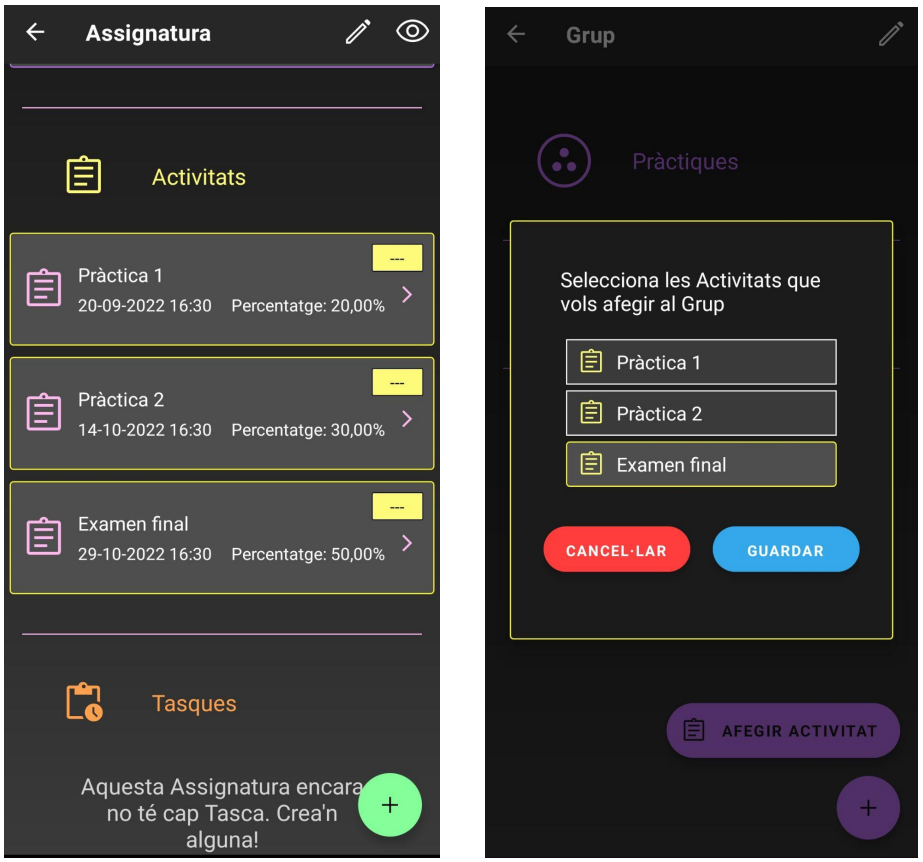


Figura 10.2.3 - Vista de les Activitats creades (esquerra) i menú per escollir les Activitats que es volen afegir al Grup (dreta)

Per comprovar que la nota de l'Assignatura es calcula correctament, assignarem les següents Qualificacions:

- Pràctica 1: Usant el sistema numèric 0-100, una ponderació de 43.
- Pràctica 2: Usant el sistema categòric F-S, una qualificació de "C" (que equival a un 7/10).
- Examen final: Usant el sistema numèric 0-20, una nota de 17,5.

Els càlculs són els següents:

NF = (Nota P1 * 0.2) + (Nota P2 * 0.3) + (Nota EF * 0.5)

NF = (43/100 * 0.2) + (7/10 * 0.3) + (17.5/20 * 0.5) = 0.7335 ≃ 0.734

Podem considerar que la nota final de l'Assignatura és 7.34, que és exactament el que mostra el gràfic:

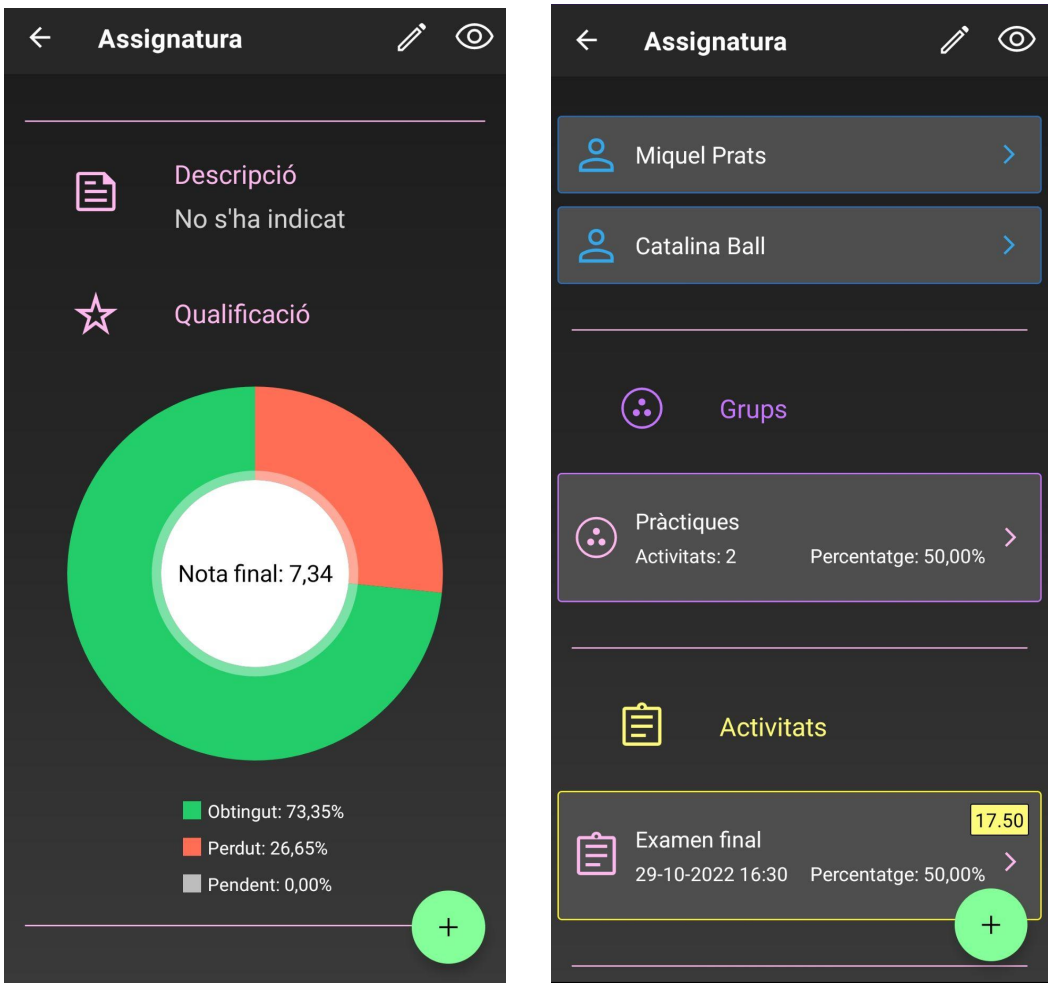


Figura 10.2.4 - Gràfic que indica la nota final de l'Assignatura (esquerra) i vista dels llistats de Professors, Grups i Activitats d'aquesta (dreta)

Com es pot veure, el procés per portar un seguiment de les notes és molt simple i ràpid. No hi ha informació redundant ni pantalles pesades, ja que des del principi la idea era crear una aplicació intuïtiva i fàcil d'utilitzar.

CAPÍTOL 11

Conclusions

Al llarg d'aquest projecte s'han pogut aplicar i ampliar els coneixements de programació Android i del *framework* Spring. Però el més enriquidor ha sigut desenvolupar un projecte des de zero i acabar obtenint un producte completament satisfactori amb les premisses inicials. Sense dubte, aplicar la metodologia SCRUM ha sigut molt útil a l'hora d'organitzar la feina i poder invertir el temps correctament.

El resultat final és una aplicació de la qual l'autor n'està molt orgullós que funcioni tan bé i que compleixi amb tots els requisits que es van marcar a l'inici. Es pot concloure que s'han assolit amb èxit tots els objectius del projecte.

L'única cosa que sembla que no ha acabat de quedar del tot bé és l'estètica dels menús. Es volia obtenir una aplicació amb una entonació fosca per seguir amb la línia actual de jugar amb aquest tipus de lluminositat, però al llarg del procés semblava que potser hagués sigut millor optar per uns colors més vius i clars, i delegar els colors foscos a menús molt concrets.

CAPÍTOL 12

Treball futur

En un futur la intenció és publicar l'aplicació a la *Play Store* i veure el *feedback* dels usuaris. Per poder-la llançar primer s'haurà d'eliminar la dependència que existeix entre el servidor i el dispositiu que executa l'aplicació, ja que actualment aquesta només funciona si els dos estan connectats per cable o corren sobre la mateixa màquina. La solució és pagar un servei de servidor d'Internet perquè faci córrer el meu servidor i els dispositius s'hi puguin connectar lliurement.

Abans, però, m'agradaria ajustar l'estètica de l'aplicació, afegint a les opcions un selector per decidir si es vol fer servir l'aplicació en "Tema clar" o "Tema fosc".

Un canvi molt gran en el qual vaig pensar mentre desenvolupava l'aplicació i que suposaria una reconstrucció quasi total d'aquesta, és la possibilitat de permetre usuaris de tipus "Professor" i de tipus "Alumne". D'aquesta manera els professors podrien tenir control sobre les Qualificacions de les Activitats dels seus alumnes i podrien assignar un delegat/da per Assignatura que pogués crear Tasques comunes per a tots els alumnes d'una mateixa Assignatura. És un concepte molt verd i que requereix un estudi i anàlisi complet del servidor i l'aplicació.

CAPÍTOL 13

Bibliografia

---

[1] «Pàgina web per desenvolupadors Android» [En línia]. Disponible: https://developer.android.com/guide.

[2] «Pàgina web de tutorials Spring» [En línia]. Disponible: https://www.baeldung.com/.

[3] «Pàgina web d'articles relacionats amb Spring» [En línia]. Disponible: https://www.tutorialspoint.com/spring/spring_overview.htm.

[4] «Repositori de Github de la llibreria PersistentCookieJar» [En línia]. Disponible: https://github.com/franmontiel/PersistentCookieJar.

[5] «Repositori de Github de la llibreria ButterKnife» [En línia]. Disponible: https://github.com/JakeWharton/butterknife.

[6] «Repositori de Github de la llibreria QuadFlask ColorPicker» [En línia]. Disponible:https://github.com/QuadFlask/colorpicker.

[7] «Repositori de Github de la llibreria MPAndroidChart» [En línia]. Disponible: https://github.com/PhilJay/MPAndroidChart.

[8] «Pàgina oficial del projecte Lombok» [En línia]. Disponible: https://projectlombok.org/features/Data.

[9] «Article sobre els sistemes operatius als dispositius mòbils» [En línia]. Disponible: https://en.wikipedia.org/wiki/Mobile_operating_system.

[10] «Pàgina web oficial de la comunitat StackOverflow» [En línia]. Disponible: https://stackoverflow.com/.

[11] «Pàgina web oficial de la llibreria MaterialDesign» [En línia]. Disponible: https://material.io/resources.

CAPÍTOL 14

Annexos

Annex 1. SharedPreferences

Quan s'inicia l'aplicació s'invoca el mètode *loadSelectedLanguage()* que carrega l'idioma que l'usuari ha escollit (si encara no n'ha escollit cap, s'utilitza l'idioma per defecte del dispositiu):

```
public static void loadSelectedLanguage(Activity activity) {
    SharedPreferences preferences = activity.
            getApplicationContext().
            getSharedPreferences(Global.LANGUAGE_KEY,  i: 0);
    String locale = preferences.getString(Global.LANGUAGE_KEY,  s1: "");
    LanguageHelper.changeCurrentLanguage(activity.getResources(), locale);
}

public static void changeCurrentLanguage(Resources res, String loc) {
    Locale locale = new Locale(loc);
    Locale.setDefault(locale);
    Configuration config = new Configuration();
    config.setLocale(locale);
    res.updateConfiguration(config, res.getDisplayMetrics());
}
```

Figura 14.1 - Mètodes per carregar l'idioma de preferència de l'usuari

Per canviar aquesta configuració, des de la pantalla d'elecció d'idioma es sobreescriu el valor a l'objecte *SharedPreferences*:

```
String lang = tvLanguageSelector.getText().toString();
String locale = LanguageHelper.getLocaleFromLanguage(lang);

SharedPreferences preferences = activity.
        getApplicationContext().
        getSharedPreferences(Global.LANGUAGE_KEY, i: 0);
SharedPreferences.Editor editor = preferences.edit();
editor.putString(Global.LANGUAGE_KEY, locale);
editor.apply();
editor.commit();
LanguageHelper.changeCurrentLanguage(activity.getBaseContext().getResources(), locale);
```

Figura Annex 14.2 - Codi per modificar l'idioma de l'aplicació

Annex 2. Enums

Els possibles valors que poden prendre els objectes de tipus *MarkType* són:

```
public enum MarkType {
    NUMERIC_0_TO_5,
    NUMERIC_0_TO_10,
    NUMERIC_0_TO_20,
    NUMERIC_0_TO_50,
    NUMERIC_0_TO_100,
    CATEGORICAL_F_TO_S,
    CATEGORICAL_FAIL_TO_DISTINCTION,
    PERSONALIZED
}
```

Figura Annex 14.3 - Enum MarkType

Els *Reminder* poden ser del tipus:

```
public enum Reminder {
    NONE,
    ONE_MINUTE_BEFORE,
    TWO_MINUTES_BEFORE,
    FIVE_MINUTES_BEFORE,
    TEN_MINUTES_BEFORE,
    FIFTEEN_MINUTES_BEFORE,
    HALF_AN_HOUR_BEFORE,
    ONE_HOUR_BEFORE,
    TWO_HOURS_BEFORE,
    ONE_DAY_BEFORE,
    TWO_DAYS_BEFORE
}
```

Figura Annex 14.3 - Enum Reminder