

UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques i Enginyeria Informàtica

DOCUMENTACIÓ

Grau en Enginyeria Informàtica

Curs 2022-2023 | Sisè Semestre

CompraSmart

Autor:

Mario VILAR (20392724)
David DÍEZ (20392643)
Andreu FIGUERAS (20392621)
María VALDERRAMA (20392562)

Professor:

Karim LEKADIR

28 de maig de 2023

PRESENTACIÓ DE L'APLICACIÓ

Una aplicació perquè mai no t'hagis de preocupar més pel preu de les teves necessitats als supermercats!



UNIVERSITAT DE
BARCELONA

Aquesta obra està subjecta a una llicència de Creative Commons
“Reconeixement-NoComercial-SenseObraDerivada 4.0 Internacional”.



Índex

| | |
|--|-------------|
| Codis font | V |
| Índex de figures | VII |
| Introducció | VIII |
| 1 Requeriments | I |
| 1.1 Requisits | I |
| 1.1.1 Funcionals | I |
| 1.1.2 No funcionals | I |
| 1.2 Diagrama de casos d'ús i diagrama de classes | 2 |
| 1.3 Mock-ups | 9 |
| 2 Documentació | 15 |
| 2.1 Descripció dels layouts | 15 |
| 2.2 Overview del model | 24 |
| 2.3 Capa de la vista | 27 |
| 2.4 ViewModel | 35 |
| 2.5 Capa de persistència i scraping de dades | 37 |
| 2.6 Testing | 38 |
| 2.6.1 Unit testing | 38 |
| 2.6.2 Integration testing | 40 |
| 2.6.3 Performance testing | 41 |
| 2.6.4 Usability testing | 41 |
| 2.6.5 Non-functional testing | 45 |
| 3 Conclusions | 46 |
| A Miscel·lània | 47 |
| A.1 Codis dels Layout | 47 |
| A.2 Codis del Model | 76 |

ÍNDEX

| | | |
|-----|---------------------------------------|------------|
| A.3 | Codis de la Vista | 94 |
| A.4 | Codis del ViewModel | 140 |
| A.5 | Codis del scraping de dades | 147 |
| | Bibliografia | 150 |
| | Índex terminològic | 151 |

Índex de codis

| | | |
|------|--|----|
| A.I | Codi del Layout corresponent al <i>Launcher</i> de l'aplicació. | 49 |
| A.2 | Codi del Layout corresponent al registre de l'usuari. | 52 |
| A.3 | Codi del Layout corresponent a l'inici de sessió (<i>ActivityIniciSessio</i>). | 55 |
| A.4 | Codi del Layout corresponent a les cards de la llista de la compra (<i>LlistaCompraCard</i>). | 57 |
| A.5 | Layout de l'activitat d'afegir un producte (<i>ActivityAfegirProducte</i>). | 58 |
| A.6 | Codi del Layout corresponent a veure els detalls d'un producte. | 61 |
| A.7 | Layout de l'activitat de visualitzar la llista de la compra. | 63 |
| A.8 | Layout del fragment del càlcul de la llista de supermercats. | 64 |
| A.9 | Layout del fragment d'escollar la llista quan s'ha fet el càlcul. | 66 |
| A.10 | Layout del fragment de veure els productes. | 67 |
| A.11 | Layout del diàleg d'eliminar una llista. | 68 |
| A.12 | Layout de l'estructura d'element d'una llista. | 70 |
| A.13 | Layout de l'estructura d'element d'una llista que podem seleccionar. | 71 |
| A.14 | Layout de la Card de la llista de productes. | 73 |
| A.15 | Layout de l'item d'algunes llistes. | 74 |
| A.16 | Layout de l'item d'algunes llistes. | 75 |
| A.17 | Layout de l'spinner. | 75 |
| A.18 | Codi de la classe corresponent a la gestió de les llistes de la compra des del model (<i>LlistesCompraRepository</i>). | 79 |
| A.19 | Configuració de la llista de la compra (<i>LlistesCompra</i>). | 80 |
| A.20 | Configuració de la classe <i>Usuari</i> | 81 |
| A.21 | Configuració de la classe <i>Producte</i> | 83 |
| A.22 | Configuració de la classe <i>InfoSuperProducte</i> | 83 |
| A.23 | Codi de la classe corresponent a <i>InfoProducteRepository</i> | 87 |
| A.24 | Configuració de la classe <i>ProductRepository</i> | 89 |
| A.25 | Configuració de la classe <i>UserRepository</i> | 91 |
| A.26 | Configuració de la classe <i>CalculadoraRepository</i> | 94 |
| A.27 | Configuració de la classe <i>MainActivity</i> | 95 |
| A.28 | Configuració de la classe <i>RegistreActivity</i> | 98 |

| | |
|---|-----|
| A.29 Configuració de la classe <i>IniciSessioActivitat</i> | 101 |
| A.30 Configuració de la classe <i>LlistaCompraActivity</i> | 107 |
| A.31 Configuració de la classe <i>VeureLlistaCompraActivity</i> | III |
| A.32 Configuració de la classe <i>VeureDetailsProducteActivity</i> | II4 |
| A.33 Configuració de la classe <i>ListAdapter</i> | II6 |
| A.34 Configuració de la classe <i>ListItem</i> | II7 |
| A.35 Configuració de la classe <i>LlistaCompraCardAdapter</i> | II9 |
| A.36 Configuració de la classe <i>LlistaProductesCardAdapter</i> | 122 |
| A.37 Configuració de la classe <i>LlistaAllProductsCardAdapter</i> | 126 |
| A.38 Configuració de la classe <i>LlistaSupermercatCardAdapter</i> | 128 |
| A.39 Configuració de la classe <i>ProductesAComprarCardAdapter</i> | 129 |
| A.40 Configuració de la classe <i>SpinnerItem</i> | 130 |
| A.41 Configuració de la classe <i>SpinnerAdapter</i> | 131 |
| A.42 Configuració de la classe <i>DialogEliminarLlista</i> | 133 |
| A.43 Configuració de la classe <i>FragmentEscollidirLlista</i> | 136 |
| A.44 Configuració de la classe <i>FragmentSupermercats</i> | 138 |
| A.45 Configuració de la classe <i>FragmentVeureProductes</i> | 140 |
| A.46 Configuració de la classe <i>LlistesCompraActivityViewModel</i> | 142 |
| A.47 Configuració de la classe <i>ProductesActivityViewModel</i> | 143 |
| A.48 Configuració de la classe <i>UsersActivityViewModel</i> | 144 |
| A.49 Configuració de la classe <i>InfoProducteActivityViewModel</i> | 146 |
| A.50 Configuració de la classe que emmagatzema les dades d'un producte en la base de dades. . . | 149 |

Índex de figures

| | | |
|------|---|----|
| 1.1 | Diagrama de casos d'ús | 2 |
| 1.2 | Diagrama de classes del Model. | 3 |
| 1.3 | Diagrama de la capa de Vista. | 5 |
| 1.4 | Diagrama de la capa de ViewModel. | 7 |
| 1.5 | Landing page | 9 |
| 1.6 | Registre | 10 |
| 1.7 | Inici de sessió | 10 |
| 1.8 | Llistes de la compra | II |
| 1.9 | Edició d'una llista de la compra | II |
| 1.10 | Càlcul de la millor opció | 12 |
| 1.11 | Llistes amb els càlculs a millor opció | 12 |
| 1.12 | Una d'aquestes llistes, amb els articles escollits per l'algorisme | 13 |
| 1.13 | Una altra llista de la compra | 13 |
| 1.14 | Informació general d'un producte | 14 |
| 2.1 | El Layout del Launcher de la nostra aplicació. | 15 |
| 2.2 | El Layout del Sign-Up de la nostra aplicació. | 16 |
| 2.3 | El Layout del Sign-In de la nostra aplicació. | 18 |
| 2.4 | El Layout de la visualització de la llista de productes. | 19 |
| 2.5 | El Layout de la visualització de la llista de la compra (<i>VeureLlistaCompra</i>). | 20 |
| 2.6 | Layout de l'activitat d'afegir un producte (<i>ActivityAfegirProducte</i>). | 20 |
| 2.7 | El Layout de la visualització de veure detalls d'un producte (<i>VeureDetallsProducte</i>). | 21 |
| 2.8 | El Layout del fragment del càlcul de la llista de supermercats (<i>FragmentSupermercats</i>). | 22 |
| 2.9 | El Layout del fragment d'escol·lar llista, (<i>FragmentEscol·larLlista</i>). | 23 |
| 2.10 | El Layout del fragment d'escol·lar llista, (<i>FragmentVeureProductes</i>). | 24 |
| A.1 | El Layout del diàleg d'eliminar una llista. | 69 |
| A.2 | El Layout de l'estructura d'element d'una llista. | 70 |
| A.3 | El Layout de la Card de la llista de productes. | 73 |

| | |
|-------------------------------------|----|
| A.4 El Layout de l'spinner. | 76 |
|-------------------------------------|----|

Introducció

Tots i totes hem utilitzat els *post-its* per apuntar que necessitàvem pa o hem anat al supermercat amb el mòbil en una mà i el bloc de notes obert. I, una vegada allà, especialment durant aquests últims mesos d'inflació desbocada, costa mantenir el tipus després que paguem i ens donin la factura. Nosaltres proposem que això no torni a passar mai.

La idea principal de l'aplicació és desenvolupar un sistema enfocat a la gestió de les llistes de la compra. A més, es podran comparar preus entre supermercats perquè puguem estalviar al màxim.

Volem crear una aplicació fàcil i intuïtiva que ens permeti fer les nostres pròpies llistes de la compra i comparar els preus dels productes en diferents supermercats. Amb l'aplicació, ja no hauríem de preocupar-nos per portar la llista de la compra en paper o per recordar-nos dels preus dels productes que necessites.

Per utilitzar l'aplicació, volem mantenir la simplicitat. Només caldria que introduíssim els productes que necessitem i la seva quantitat en la llista de la compra. A continuació, l'aplicació ens mostraria una llista de supermercats propers a la nostra ubicació i els preus dels productes de la llista seleccionada en cada un d'ells. D'aquesta manera, podríem comparar i decidir on comprar per obtenir el millor preu.

A més a més, l'aplicació també ens permetria guardar les nostres llistes de la compra per a futurs usos i afegir nous productes a la llista en qualsevol moment. També podríem afegir notes i detalls addicionals als productes, com ara la marca o el preu que volem pagar, per ajudar-te a prendre les millors decisions de compra.

Amb l'aplicació, podríem estalviar temps i diners, ja que podríem comparar els preus dels productes de la nostra llista en diferents supermercats sense haver de visitar-los un per un. A més a més, l'aplicació ens ajudaria a controlar els nostres costos de la compra i a mantenir el pressupost baix en aquests temps tan durs per a les nostres butxaques.

En resum, l'aplicació que volem crear és una eina útil i pràctica per a aquelles persones que volen estalviar temps i diners en la seva compra diària.

Requeriments

I.I REQUISITS

I.I.1 FUNCIONALS

1. El sistema ha de permetre als usuaris fer el log-in mitjançant una direcció de correu electrònic i una contrasenya.
2. El sistema ha de permetre a l'usuari enregistrar-se a la nostra aplicació mitjançant un nom, un cognom, una direcció de correu electrònic i una contrasenya.
3. El sistema ha de permetre a l'usuari visualitzar totes les seves llistes de la compra que hagi creat prèviament.
4. El sistema ha de permetre a l'usuari el crear una nova llista de la compra amb diferents productes.
5. El sistema ha de permetre a l'usuari modificar una llista de la compra ja existent.
6. El sistema permetrà a l'usuari eliminar una llista de la compra ja existent.
7. El sistema permetrà a l'usuari decidir a quins supermercats en concret vol anar per fer la compra.
8. El sistema permetrà a l'usuari calcular quins productes d'una llista de la compra determinada ha de comprar cada supermercat per tal d'estalviar el màxim possible.
9. El sistema permetrà visualitzar els preus dels productes perquè el mateix usuari comprovi quin supermercat és més econòmic.

I.I.2 NO FUNCIONALS

1. Fiabilitat: L'aplicació ha de ser fiable i funcionar sense errors per evitar situacions frustrants per als usuaris.
2. Personalització: L'aplicació ha de permetre als usuaris escollir entre el tema clar i fosc.
3. Escalabilitat: L'aplicació ha de ser escalable per a poder suportar un gran nombre de usuaris i una gran quantitat de dades.
4. Manteniment: L'aplicació ha de ser fàcil de mantenir i actualitzar per a garantir el seu bon funcionament al llarg del temps.

DIAGRAMA DE CASOS D'ús I DIAGRAMA DE CLASSES

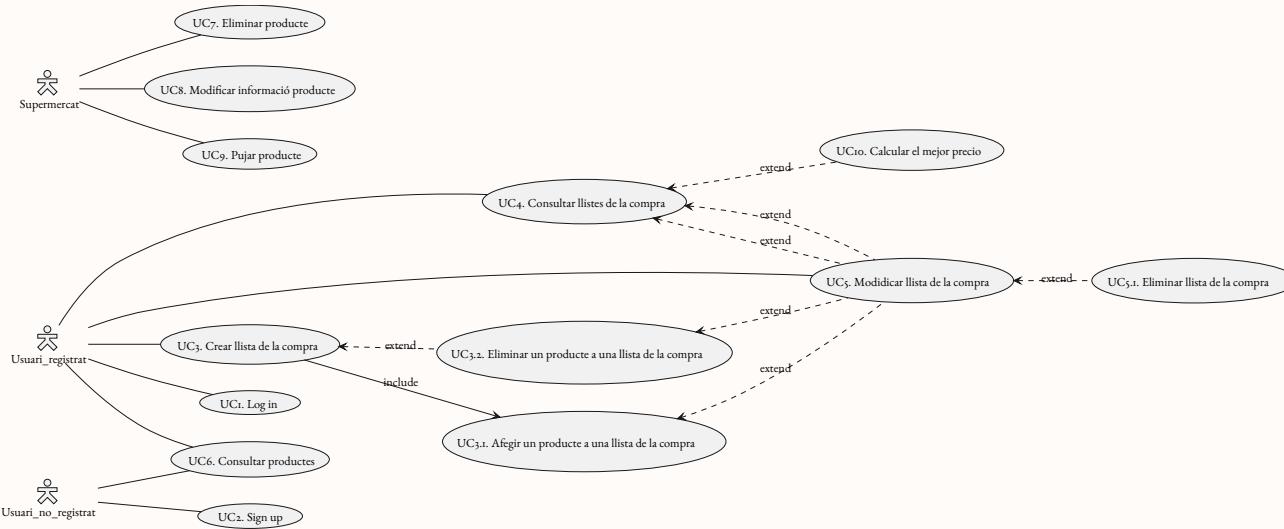


Figura 1.1: Diagrama de casos d'ús

Hem distingit entre tres *actors* en el nostre disseny: l'usuari registrat, el no registrat i el supermercat. El *supermercat* pot afegir els seus productes a la nostra aplicació, amb la seva informació corresponent, incloent-hi el preu. També pot informar-nos de si a la llarga modifica algun dels atributs dels ítems publicats, o si directament els elimina.

L'*usuari registrat* podrà fer *log-in* per accedir a una gran varietat de possibilitats. Pot consultar les llistes de la compra que hagi creat, o bé crear-ne una de nova. Tota llista haurà de tenir mínim un article i es podrà modificar en tot moment, així com eliminar-la del tot. Aquestes modificacions són, en efecte, afegir o eliminar-ne productes.

L'*usuari no registrat*, en canvi, solament podrà fer consultes esporàdiques de productes i, en cas que així ho vulgui, registrar-se. Una vegada s'hagi inscrit (*sign-up*), passarà a ser *usuari registrat*. L'objectiu és clar: promoure la inscripció a la nostra aplicació.

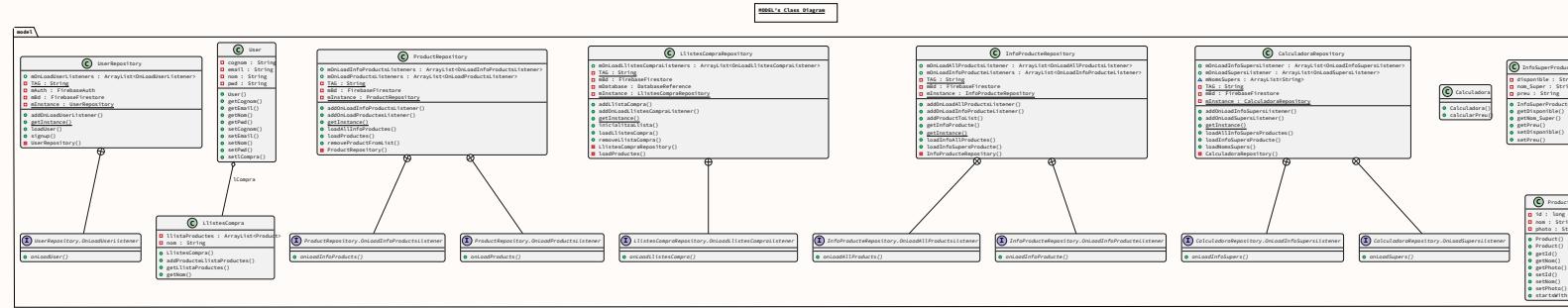


Figura 1.2: Diagrama de classes del Model.

El diagrama mostra les classes del nostre Model i les relacions que estableixen entre elles. Hi ha diverses classes i interfícies. Cada classe representa un component del model, i les interfícies descriuen els oients per rebre esdeveniments generats per algunes de les classes.

1. La classe **User** representa els usuaris de l'aplicació, i la classe **UserRepository** és responsable de carregar i desar informació de l'usuari. Té una llista d'objectes **OnLoadUserListener** i interactua amb la base de dades per carregar els usuaris i registrar-ne de nous.
2. La classe **LlistesCompra** representa una llista de la compra i **LlistesCompraRepository** és responsable de carregar i desar informació de les llistes de la compra.
 - Les interfícies **OnLoadLlistesCompralListener** defineixen el mètode abstracte `onLoadLlistesCompral()`, que serà implementat per altres classes per a rebre notificacions quan es carreguin les llistes de compra.
3. La classe **Producte** representa un producte i **ProducteRepository** és responsable de carregar i desar informació dels productes.
 - Té llistes d'objectes **OnLoadInfoProductsListener** i **OnLoadProductsListener** i interactua amb la base de dades per carregar tota la informació dels productes, carregar els productes i eliminar productes de la llista.
4. Aquesta última classe té dues interfícies. La classe **InfoSuperProducte** representa informació detallada d'un producte. La classe **InfoProducteRepository** és responsable de carregar informació tant de **Producte** com de **InfoSuperProducte**.
 - Aquesta classe, concretament, té dues interfícies, **OnLoadAllProductsListener** i **OnLoadInfoProducteListener**, per rebre esdeveniments quan es carreguen tots els productes o la informació detallada d'un producte.

- ⁴
- 5. **Calculadora:** Aquesta classe representa la calculadora de l'aplicació. Té un constructor Calculadora() i un mètode calcularPreu() que s'encarrega de realitzar els càlculs de preu necessaris.

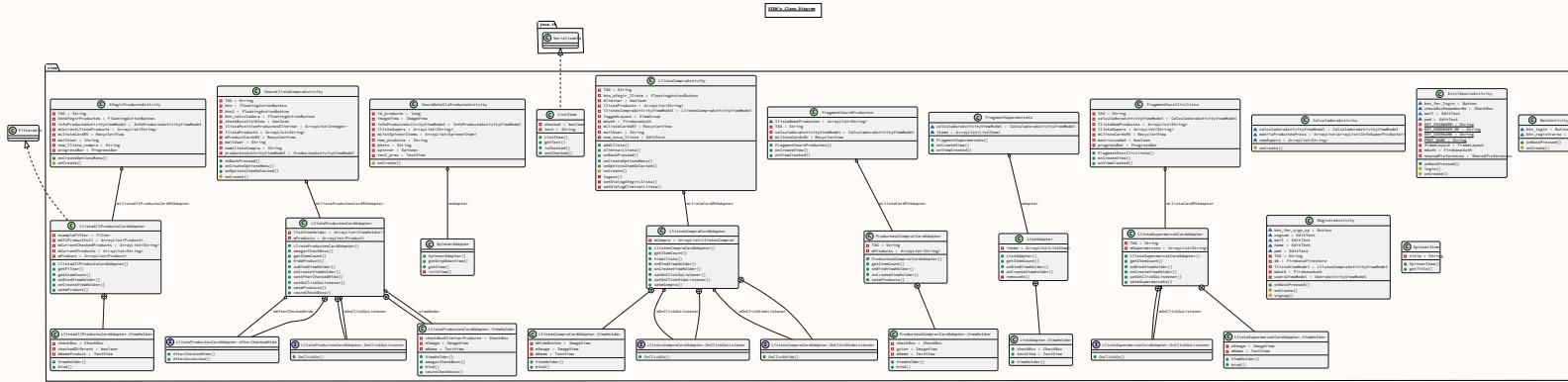


Figura 1.3: Diagrama de la capa de Vista.

Per no estendre'ns en detall, comentarem solament les classes més importants i les seves relacions.

1. **AfegirProducteActivity**: classe que representa l'activitat per afegir productes a una llista de la compra. Conté botons, adapters i una llista d'elements. *Té una relació d'agregació amb InfoProducteActivityViewModel, una classe que gestiona la informació dels productes.*
2. **CalculadoraActivity**: Aquesta classe representa una activitat de calculadora. Té una propietat **calculadoraActivityViewModel**, **matriuProductesPreu** i **nomSupers**, i té un mètode **onCreate()**.
3. **DialogEliminarLlista**: classe que representa un diàleg per eliminar una llista de la compra. Conté una instància de la classe **LlistaCompraActivity** i un enter que indica la posició de la llista a eliminar. *Té una relació d'agregació amb LlistaCompraActivity.*
4. **FragmentEscolhirLlista**: Aquesta classe representa un fragment per seleccionar una llista. Té diverses propietats com **TAG**, **calculadoraActivityViewModel**, **llistaNomProductes**, **llistaSupers**, **mLlistaCardsRV**, **matrixLoaded**, **progressBar**, i també té mètodes com **FragmentEscolhirLlista()**, **onCreateView()** i **onViewCreated()**.
5. **FragmentSupermercats**: Aquesta classe representa un fragment per mostrar els supermercats. Té una propietat **calculadoraActivityViewModel**, una llista d'items i té mètodes com **FragmentSupermercats()**, **onCreateView()** i **onViewCreated()**.
6. **FragmentVeureProductes**: Aquesta classe representa un fragment per veure els productes. Té diverses propietats com **llistaNomsProductes**, **TAG**, **calculadoraActivityViewModel**, **mLlistaCardsRV**, i té mètodes com **FragmentVeureProductes()**, **onCreateView()** i **onViewCreated()**.

- 6
- 7. IniciSessioActivity: classe que representa l'activitat d'inici de sessió de l'aplicació. Conté botons i camps d'edició de text per al correu electrònic i la contrasenya de l'usuari. *Té una relació d'agregació amb la classe FirebaseAuth.*
 - 8. LlistaCompraActivity: classe que representa l'activitat principal de l'aplicació, on es mostren les llistes de la compra existents. Conté botons, adapters i una llista d'elements. Té una relació d'agregació amb LlistesCompraActivityViewModel, una classe que gestiona la informació de les llistes de la compra.
 - 9. LlistaProductesCardAdapter: classe que representa un adaptador per a una llista de productes d'una llista de la compra. Conté una llista d'objectes Product i mètodes per establir un listener per a quan es fa clic en un element i per a amagar i mostrar les caixes de selecció.
 - 10. LlistaAllProductsCardAdapter: Aquesta classe representa un adaptador per a la llista de tots els productes. Té propietats com exampleFilter, mAllProductFull, mCurrentCheckedProducts, mCurrentProducts, mProduct, i té mètodes com LlistaAllProductsCardAdapter(), getFilter(), getItemCount(), onBindViewHolder(), onCreateViewHolder() i setmProduct().
 - II. AfterCheckedHide, OnClickGoListener, OnClickChangeCheck són interfícies.



Figura 1.4: Diagrama de la capa de ViewModel.

A continuació es detallen les classes, així com les seves funcions. No detallarem relacions entre elles perquè, simplement, no apareixen al diagrama (de manera que no existeixen físicament, en el codi).

1. **InfoProducteActivityViewModel:** Té com a atributs un TAG, un MutableLiveData que conté una llista d'objectes Product, un InfoProducteRepository i un MutableLiveData que conté una llista d'objectes InfoSuperProducte. Aquesta classe té diversos mètodes: addProducteALLlistaCompra() per afegir un producte a la llista de la compra, getAllProducts() per obtenir la llista completa de productes, getInfoSuperProducte() per obtenir la informació detallada d'un producte, loadAllProductsFromRepository() per carregar tots els productes del repositori, loadInfoProducteFromRepository() per carregar la informació detallada d'un producte del repositori, setLlistaAllProducts() per establir la llista completa de productes i setLlistesInfoProducte() per establir la llista d'informació detallada dels productes.
2. **InfoSuperProducte:** Té com a atributs un TAG, un MutableLiveData que conté una llista d'objectes LlistesCompra, un LlistesCompraRepository i un MutableLiveData que conté una llista d'objectes Product. Els mètodes són els següents: addLlista() per afegir una llista de la compra, getmCompra() per obtenir la llista completa de llistes de la compra, getmProducts() per obtenir la llista com-

pleta de productes, `inicialitzaLlista()` per inicialitzar una llista de la compra, `loadLlistesCompraFromRepository()` per carregar totes les llistes de la compra del repositori, `removeLlistaFromHome()` per eliminar una llista de la compra de l'activitat i `setLlistesCompra()` per establir la llista completa de llistes de la compra.

3. `ProductesActivityViewModel`: Té com a atributs un TAG, un `MutableLiveData` que conté una llista d'objectes `Product` i un `ProducteRepository`. Aquesta classe té diversos mètodes; `getmProducts()` per obtenir la llista completa de productes, `loadInfoProductsFromRepository()` per carregar la informació detallada d'un producte del repositori, `loadProductsFromRepository()` per carregar tots els productes del repositori, `removeProductFromList()` per eliminar un producte de la llista d'activitat i `setProducts()` per establir la llista completa de productes.
4. `UsersActivityViewModel`: Té com a atributs un TAG, un `MutableLiveData` que conté una llista d'objectes `User` i un `UserRepository`. Té com a atributs un TAG, un `MutableLiveData` que conté una llista d'objectes `Product`, un `InfoProducteRepository` i un `MutableLiveData` que conté una llista d'objectes `InfoSuperProducte`. Aquesta classe té diversos mètodes: `getmUsers()` per obtenir la llista completa d'usuaris, `setUsers()` per establir la llista completa d'usuaris i `signup()` per registrar un nou usuari.
5. La classe `CalculadoraActivityViewModel` és responsable de la lògica de negoci de la calculadora de preus. Té les següents propietats: `TAG` (etiqueta), `calculadora` (instància de la classe `Calculadora`), `mAllSupers` (dades observables per a una llista de supermercats), `mCalculadoraRepository` (repositori per a la calculadora), `mSupermercats` (dades observables per a una llista de supermercats) i `matrix` (dades observables per a una matriu de productes). Proporciona diversos mètodes com `calcularPreu()` per calcular el preu, `getMatrix()` per obtenir la matriu de productes, `getmAllSupers()` i `getmSupermercats()` per obtenir les llistes de supermercats, `loadInfoMatrix()` per carregar les dades de la matriu, `loadSupersFromRepository()` per carregar els supermercats des del repositori i altres mètodes per establir les dades.

I.3 MOCK-UPS

La secció de *mock-ups* de la nostra aplicació de llista de la compra és una part crucial del nostre procés de desenvolupament. A través de la creació de mock-ups, hem estat capaços de visualitzar les funcions i característiques de l'aplicació abans de la seva implementació. Això ens ha permès fer ajustos i refinaments abans de la implementació final, assegurant que l'aplicació sigui intuïtiva i facil d'utilitzar per als usuaris.

Després de realitzar una investigació exhaustiva sobre les necessitats dels usuaris i les tendències actuals en el disseny d'applications mòbils, hem creat una sèrie de mock-ups que representen les funcions i característiques de la nostra aplicació amb gran detall. Aquesta secció del nostre informe descriu els diferents mock-ups que hem dissenyat per a l'aplicació, incloent pantalles per a la pàgina d'inici de sessió, la pàgina principal de la llista de la compra, el formulari d'afegir elements a la llista i molts altres. Cada mock-up ha estat dissenyat per a proporcionar la màxima usabilitat i eficiència als usuaris, mentre que també es manté un disseny atractiu i coherent en tota l'aplicació.



Figura I.5: Landing page

En primer lloc, fixem-nos en la part superior de la barra. Tenim un botó d'informació per ajudar-nos en qualsevol moment, en cas que trobem que algun pas que vulguem fer no està indicat de manera clara o és molt poc intuïtiu.

Més avall tenim un títol, que ens incita a registrar-nos/signar a l'aplicació per poder gaudir dels seus avantatges. Després de la foto, trobem dos enllaços o botons: el primer, un formulari d'inici de sessió per a, valgui la redundància, iniciar sessió. També, trobem un botó que ens porta a la pàgina de registre, per als usuaris que encara no tenen un compte a l'aplicació. Aquest *mock-up* és senzill i elegant, amb una paleta de colors atractiva i un disseny que permet una fácil legibilitat. Els botons presents tenen una funcionalitat clara i el botó d'ajuda és útil per als usuaris que la necessitin.

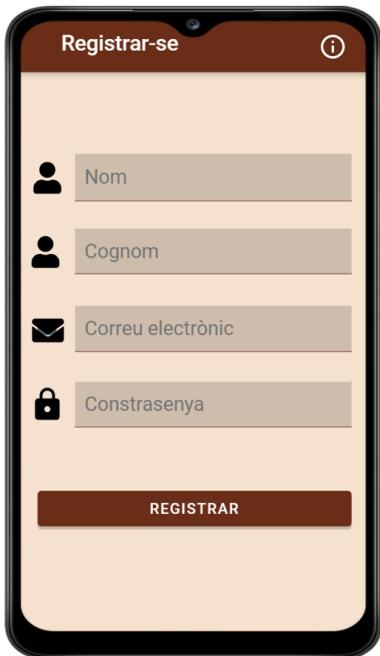


Figura 1.6: Registre

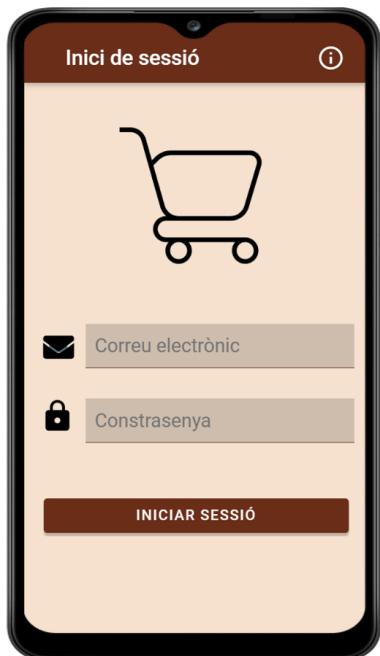


Figura 1.7: Inici de sessió

El mock-up de la pantalla de registre és una representació visual de com es veurà la pantalla d'inici de sessió en la nostra aplicació de llista de la compra. Aquesta pantalla serà la primera que els usuaris veuran quan iniciïn l'aplicació per primera vegada.

Es demanaran:

1. Nom i cognoms.
2. Correu (ha de ser un correu vàlid).
3. Contrassenya, que haurà de complir unes certes condicions.

Finalment, el botó de *Registrar-se* permetrà entrar aquestes dades a la nostra base de dades.

En general, el mock-up de la pantalla d'inici de sessió té un disseny atractiu i funcional que està dissenyat per a proporcionar als usuaris una experiència d'inici fàcil i sense complicacions. Aquesta pantalla serà la clau per a accedir a l'aplicació i, per tant, és important que estigui ben dissenyada i sigui fàcil d'utilitzar. En aquest sentit, els camps d'entrada estan clarament etiquetats i són fàcils de llegir.

Rutinàriament, caldrà introduir:

1. Correu.
2. Contrassenya.



Figura 1.8: Llistes de la compra

Per suposat, el mock-up de la pantalla de llistes de la compra és una representació visual de com es veurà la pantalla principal de l'aplicació de llista de la compra després que l'usuari hagi iniciat sessió. Aquesta pantalla és el centre de l'aplicació, ja que és aquí on els usuaris poden crear noves llistes de la compra i accedir a les que ja han creat.

En el mock-up, es pot veure que hi ha un botó d'afegir nova llista de la compra, així com una llista de les llistes de la compra existents. També hi ha un botó d'opcions, que permet als usuaris esborrar o editar la llista.

Aquesta pantalla és la clau per a utilitzar l'aplicació de manera efectiva, ja que els usuaris poden accedir a totes les seves llistes de la compra aquí i gestionar-les de manera eficient.

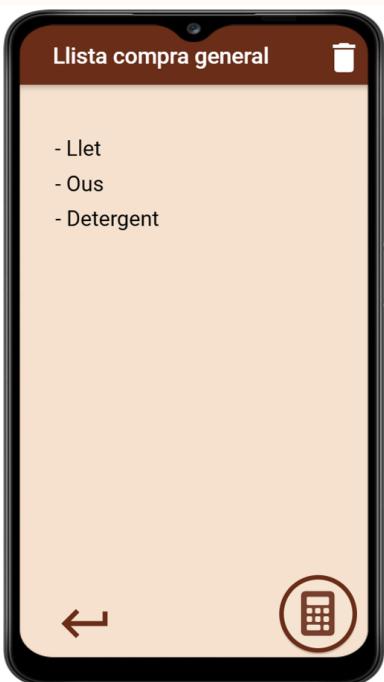


Figura 1.9: Edició d'una llista de la compra

El mock-up de la pantalla d'una llista de la compra específica és una representació visual de com es veurà la pantalla que es mostra quan un usuari selecciona una llista de la compra en particular. Aquesta pantalla és on els usuaris poden veure tots els productes que han afegit a la seva llista, així com editar o esborrar aquests productes.

En el mock-up, es pot veure que hi ha una llista de productes amb els seus noms i quantitats. També hi ha botons per a editar o esborrar cada producte. A més a més, hi ha un botó d'afegir nou producte per als usuaris que necessiten afegir un producte nou a la seva llista (o treure'n un). També hi ha un botó per eliminar la llista en qüestió.

Els usuaris poden veure el preu total de la seva llista de la compra en la part inferior de la pantalla, pitjant el botó corresponent, el que els ajuda a mantenir el control de les seves despeses i a planificar el seu pressupost.



Figura 1.10: Càlcul de la millor opció

El mock-up mostra diferents botons o pestanyes que permeten als usuaris seleccionar diferents supermercats per comparar els preus. Això podria incloure les botigues més populars de la zona o les cadenes que els usuaris han seleccionat prèviament com a preferides. Això permet als usuaris veure immediatament quins supermercats ofereixen millors preus i fer una decisió informada sobre on comprar.



Figura 1.11: Llistes amb els càlculs a millor opció

Totalment lligat a l'anterior, aquest mock-up inclou una secció de resultats que mostra als usuaris quines botigues ofereixen els millors preus per als diferents productes de la seva llista de la compra, després d'haver aplicat l'algorisme de cerca exhaustiva, que encara hauríem de detallar més endavant. De fet, dividim la llista que li hem entrat al principi en tantes llistes com supermercats haguem seleccionat, cadascuna amb els productes amb millors preus. Això ajuda als usuaris a determinar quins supermercats valen la pena visitar i quins poden ser evitats per estalviar temps i diners.

Clar està, si hi ha un supermercat molt car, per molt que l'haguem escollit, pot ser que cap dels articles el puguem comprar allà si la nostra premissa és estalviar al màxim.

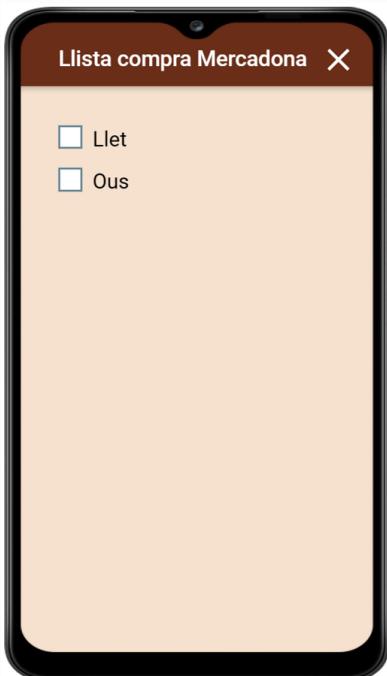


Figura I.12: Una d'aquestes llistes, amb els articles escollits per l'algorisme

No té molt misteri, és semblant al que exposàvem a la figura I.6. Recordem, aquesta llista és resultat d'haver aplicat l'algorisme i els articles que s'hi troben són els de preu més baix en els supermercats seleccionats.



Figura I.13: Una altra llista de la compra

Les observacions fets a I.6 i I.12 apliquen i ho expliquen perfectament.



Figura 1.14: Informació general d'un producte

Aquesta és la informació d'un producte. En general, intentarem mostrar una fotografia perquè l'usuari s'asseguri que és el que desitja. Posarem, al seu torn, el preu de tal producte en un supermercat concret. Idealment, crearem un *scroll-down* que ens permeti fer això.

Documentació

2.I

DESCRIPCIÓ DELS LAYOUTS

En aquesta secció, explorarem els layouts d'una aplicació.

Definició 2.1 (Layout). El *layout* d'una aplicació es refereix a la manera en què els diferents elements de la interfície d'usuari estan disposats i organitzats. És important tenir un bon layout, perquè els usuaris puguin navegar per l'aplicació de manera fàcil i intuïtiva.

Analitzarem els diferents tipus de Layout en els nostres mock-ups, com ara el layout lineal, el de graella i d'altres. També explorarem com utilitzar les diferents característiques de cada tipus de layout per crear una interfície d'usuari atractiva i funcional. Explorarem els layouts de manera natural, i no hi ha millor manera de començar que pel principi.



Els Layouts utilitzats són: *ConstraintLayout*, *FrameLayout*, *LinearLayout*, *TableLayout*. Altres recursos utilitzats per aquesta activitat són: *ImageView*, *EditText* i *Button*.

Figura 2.1: El Layout del Launcher de la nostra aplicació.

El codi que proporcionarem a continuació defineix la interfície d'usuari d'un *launcher* d'una aplicació d'Android. Els elements utilitzats, en més detall que abans, són els següents:

1. L'element primitiu és un `ConstraintLayout`, que és un tipus de disseny que permet definir les relacions de posicionament entre fills amb restriccions.
2. S'utilitzen els atributs d'espai de noms (namespace) `android`, `app` i `tools` per definir propietats de la interfície d'usuari. Per exemple, `android:layout_width` i `android:layout_height` són atributs de l'espai de noms `android` que defineixen l'amplada i l'alçada dels elements, mentre que `app:layout_constraintBottom_toBottomOf` i `app:layout_constraintEnd_toEndOf` són atributs de l'espai de noms `app` que defineixen les restriccions de posicionament dels elements.
3. S'utilitzen diferents tipus d'elements com ara `FrameLayout`, `LinearLayout`, `TextView`, `ImageView` i `TableLayout` per estructurar la interfície d'usuari. Cada element té atributs específics que defineixen les seves propietats com ara l'amplada, l'alçada, el color de fons, el contingut, etc.
4. S'utilitzen les referències d'identificadors (ID) com `@+id/nom_identificador` per identificar de manera única els elements de la interfície d'usuari. Aquests identificadors es poden utilitzar per accedir als elements des del codi de l'aplicació i afegir-los funcionalitats interactives. S'utilitzen també les etiquetes d'estil (style) com `@style/nom_estil` per aplicar estils predefinits als elements de la interfície d'usuari. Aquests estils es poden definir en un fitxer separat i reutilitzar en diferents elements de l'aplicació.

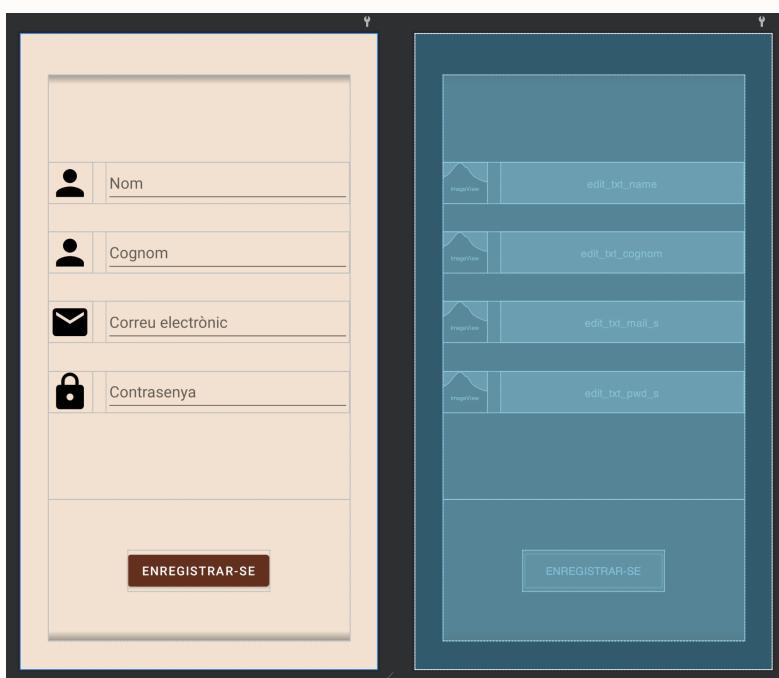


Figura 2.2: El Layout del Sign-Up de la nostra aplicació.

El codi és un arxiu de disseny d'una activitat dins d'una aplicació Android, escrit en XML. L'arxiu defineix la interfície d'usuari d'una pantalla de registre amb diversos camps d'entrada (`EditText`) i un botó de registre (`Button`) dins d'un `ConstraintLayout`.

Els Layouts utilitzats són: `ConstraintLayout`, `FrameLayout`, `LinearLayout`, `TableLayout`. Altres recursos utilitzats per aquesta activitat són: `ImageView`, `EditText` i `Button`.

1. L'arxiu comença amb la declaració de la versió de l'XML i l'encodatge utilitzat.
2. El ConstraintLayout és el contenidor principal que s'utilitza per definir la posició i relació dels elements dins de la interfície d'usuari.
3. Els atributs `xmlns:android`, `xmlns:app` i `xmlns:tools` són espais de noms utilitzats per definir els atributs específics d'Android, les biblioteques d'Android Support i les eines de desenvolupament d'Android Studio, respectivament.
4. Els atributs `android:layout_width` i `android:layout_height` defineixen l'amplada i l'alçada del ConstraintLayout com a `match_parent`, que vol dir que s'estendrà per ocupar tot l'espai disponible de la pantalla.
5. L'atribut `android:background` defineix el color de fons del ConstraintLayout utilitzant una referència a un color definit en un fitxer de recursos de l'aplicació.
6. L'atribut `tools:context` especifica el context de la vista, que és utilitzat per l'eina de desenvolupament d'Android Studio per mostrar la vista de previsualització de la interfície d'usuari en el mode de disseny.
7. Dins del ConstraintLayout, hi ha un FrameLayout que conté un LinearLayout. Aquest LinearLayout conté una taula (TableLayout) amb diverses files (TableRow). Cada fila de la taula conté un ImageView i un EditText. L'ImageView és una imatge d'ícone i l'EditText és un camp d'entrada de text per a l'usuari introduir les seves dades de registre, com el nom, cognom, correu electrònic i contrasenya.
8. Al final del LinearLayout hi ha un altre FrameLayout que conté un botó de registre (Button) amb un id `btn_fer_signup`. Aquest botó té un color de fons definit utilitzant una referència a un color definit en un fitxer de recursos i mostra el text `signup`.

Observació 2.2. Cal destacar que hi ha alguns atributs com `tools:ignore` i `tools:ignore=DuplicateSpeakableTextCheck` que són utilitzats per indicar a l'eina de desenvolupament d'Android Studio que ignore certs errors o advertències de l'anàlisi del codi XML durant la previsualització de la interfície d'usuari, però aquestes anotacions no tenen cap efecte en l'execució de l'aplicació en temps d'execució.

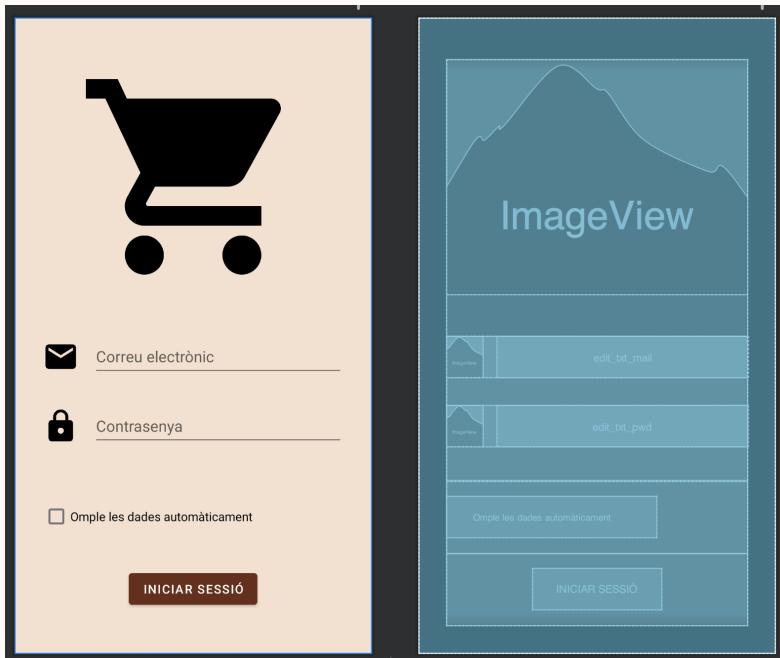


Figura 2.3: El Layout del Sign-In de la nostra aplicació.

Els Layouts utilitzats són: *ConstraintLayout*, *FrameLayout*, *LinearLayout*, *TableLayout*. Altres recursos utilitzats per aquesta activitat són: *ImageView*, *EditText* i *Button*.

El codi és un arxiu *XML* que defineix la interfície d’usuari d’una activitat d’inici de sessió en una aplicació d’Android. Aquí està una descripció dels elements i atributs principals utilitzats en el codi:

1. La vista principal és un *ConstraintLayout*, que és un layout de llibreria *AndroidX* que permet crear interfícies d’usuari amb restriccions entre els elements.
2. Dins del *ConstraintLayout*, hi ha dos *FrameLayout*. El primer conté una *ProgressBar* que té un identificador únic i s’oculta per defecte, i servirà per la transició entre el *login* i la següent activitat.
3. El segon *FrameLayout* que ocupa tot l’espai disponible i té marges de 32 dp a cada costat i 48 dp a dalt i a baix. Dins del *FrameLayout*, hi ha un *LinearLayout* que ocupa tot l’espai disponible verticalment i té orientació vertical.
4. Dins del *LinearLayout*, hi ha diversos elements d’interfície d’usuari, com ara un *ImageView*, que mostra una imatge d’un carro de la compra, i un *TableLayout*, que conté dues files de *TableRow*, cada una amb un *ImageView* i un *EditText* per al correu electrònic i la contrasenya de l’usuari.
5. Els *EditText* tenen atributs com *hint* per mostrar text de suggeriment quan estan buits i *inputType* per especificar el tipus d’entrada de text (adreça de correu electrònic i contrasenya en aquest cas).
6. Al final del *LinearLayout*, hi ha un altre *FrameLayout* que conté un *Button* amb un identificador únic *btn_fer_login* que té el text *login* com a etiqueta i un color de fons específic.
7. També tenim un *FrameLayout* amb un *Checkbox* per si volem recordar les nostres dades.

El codi XML, A.14, proporciona un disseny a Android que utilitza el ConstraintLayout de la biblioteca d'AndroidX. Aquest Layout conté un TextView, un ImageView i un Button.

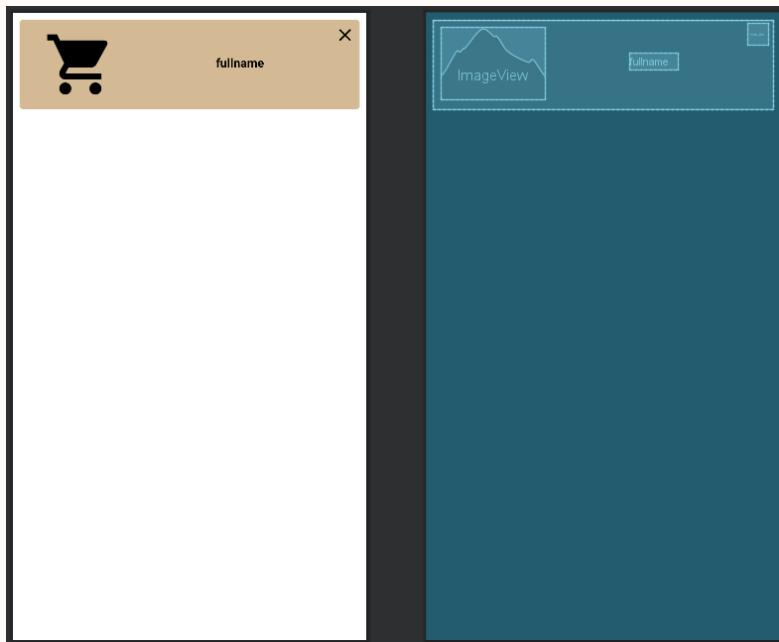


Figura 2.4: El Layout de la visualització de la llista de productes.

El TextView mostra el títol de cada llista de la compra, el ImageView serveix per mostrar el dibuix d'un carrito i, el Button, en forma de creu, es per poder eliminar la llista quan sigui necessari. Si fem click a la carta podem accedir a informació més detallada sobre la llista de la compra. Quan cliquem el botó d'eliminar llista salta un *pop up* de confirmació. No creiem necessari explicar el Layout d'aquest *pop up* atés a la seva simplicitat.

El codi XML de A.7 defineix una targeta de Material Design utilitzant la classe MaterialCardView de la biblioteca de disseny de Google. La vista MaterialCardView envolta un LinearLayout que conté un CheckBox i un TextView. (*Pàgina següent.*)

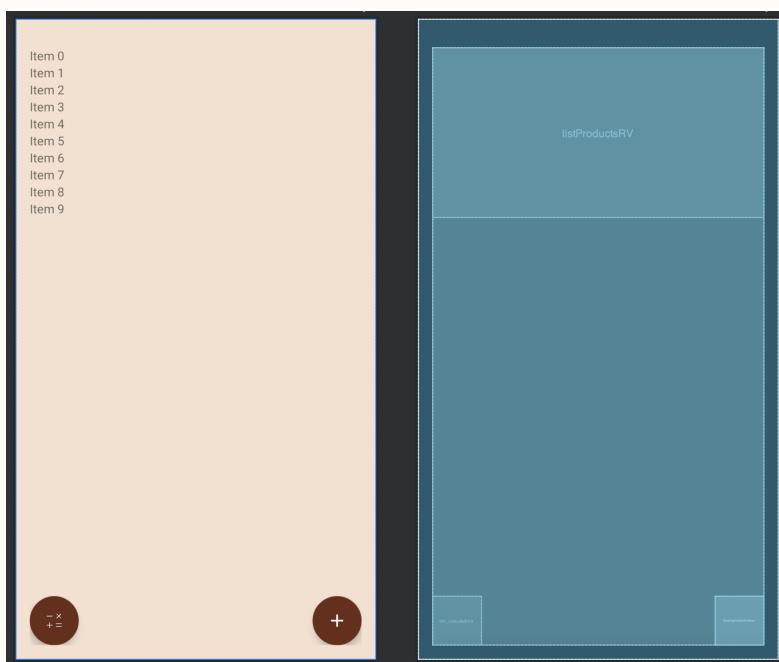


Figura 2.5: El Layout de la visualització de la llista de la compra (*VeureLlistaCompra*).

En el *ConstraintLayout*, hi ha un *FrameLayout* que ocupa tota la pantalla. Dins del *FrameLayout*, hi ha un *LinearLayout* vertical, que ocupa tota la pantalla. El *LinearLayout* conté un *TextView* que mostra un títol, seguit d'un *ScrollView*. El *ScrollView* conté un *RelativeLayout*, que a la seva vegada té un *RecyclerView*. El *RecyclerView* mostra una llista d'elements que es pot desplaçar verticalment. Tenim dos botons d'accio flotants (*FloatingActionButton*), un per afegir més productes i l'altre per fer el càlcul de la millor opció.

El codi XML, A.5, proporciona una representació visual d'una pantalla d'aplicació per veure les llistes de les compres que tenim planejades utilitzant el *ConstraintLayout* i altres components.

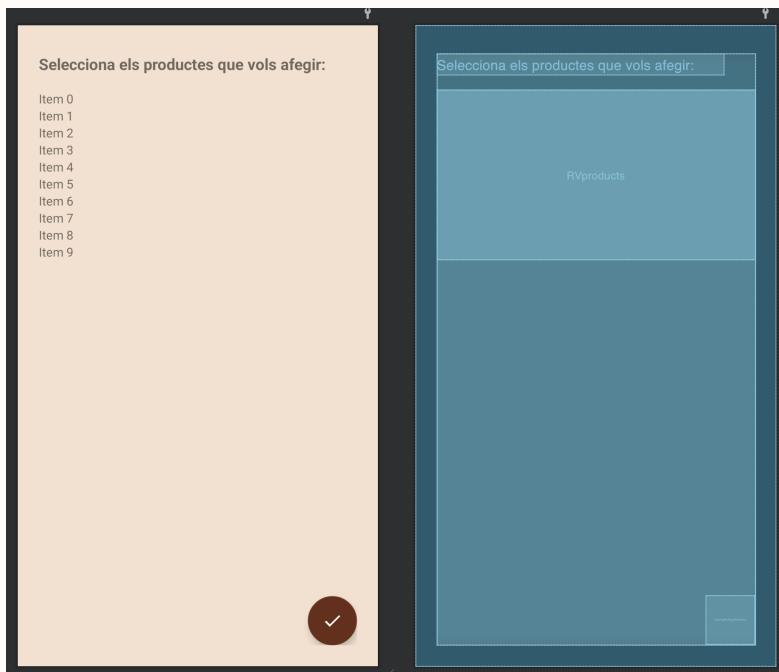


Figura 2.6: Layout de l'activitat d'afegir un producte (*ActivityAfegirProducte*).

Es defineix una vista *ConstraintLayout* que conté un *FrameLayout*, que a la vegada conté un *LinearLayout* amb un *TextView* i un *RecyclerView* dins d'un *ScrollView*. També hi ha un *FloatingActionButton* a la part inferior dreta. S'han definit algunes propietats de disseny, com la mida i la posició dels diferents elements, el color de fons i el contingut de l'element *FloatingActionButton*.

1. La pantalla sembla tenir un títol ("Selecciona els productes que vols afegir:") en negreta i un *RecyclerView* (RVproducts) que conté una llista de productes. El *RecyclerView* es mostra dins d'un *ScrollView* per permetre als usuaris fer scroll en la llista de productes.
2. En efecte, hi ha un FloatingActionButton que es mostra a la part inferior de la pantalla, i que té una icona d'un tic. Aquest botó té un fons marró i un contorn blanc, i és clicable. El botó té una icona blanca de marca de verificació (`baseline_check_24`) o el símbol de suma en un fons marró.

El codi XML A.6 defineix la vista de la pàgina VeureDetailsProducte de l'aplicació Android. Utilitza un ConstraintLayout com a contenidor principal i dins d'aquest hi ha un FrameLayout que a la vegada conté un LinearLayout i un FloatingActionButton. El LinearLayout és on es mostra la llista de productes i hi ha un ScrollView per a la navegació. El FloatingActionButton es mostra a la part inferior de la pantalla i té una imatge d'un check.



Figura 2.7: El Layout de la visualització de veure detalls d'un producte (*VeureDetailsProducte*).

La vista es defineix utilitzant un ConstraintLayout que omple tota la pantalla. Dins del ConstraintLayout hi ha un FrameLayout que ocupa tota la pantalla i conté un LinearLayout que es fa responsable de mostrar el contingut de la pàgina. El LinearLayout té tres elements: una imatge (ImageView) per a la imatge del producte, un text (TextView) per al títol i un TableLayout per a mostrar els supermercats on es pot comprar el producte (amb un Spinner per al nom del supermercat i un text amb el nom del producte).

El fitxer XML, A.8, segueix una estructura jeràrquica amb els següents elements principals:

1. **ConstraintLayout**: És el contenidor principal que ocupa tot l'espai disponible de la pantalla. Defineix les regles de posició i alineació dels seus elements interns utilitzant restriccions.
2. **FrameLayout**: És un contenidor que ocupa tot l'espai disponible dins del ConstraintLayout. Els seus elements interns es superposen un sobre l'altre.
3. **LinearLayout**: Conté diversos elements interns disposats de forma vertical. Té marges laterals de 32dp i ocupa tot l'espai disponible dins del FrameLayout.
4. **ImageView**: Mostra una imatge del carro de la compra amb una amplada mínima de 350dp i una alçada mínima de 230dp. Té un pes relatiu de 2.5 dins del LinearLayout.
5. **TextView**: Mostra un text centrat amb estils de negreta i cursiva. Té un pes relatiu de 1 dins del LinearLayout.
6. **ScrollView**: Permet desplaçar el contingut si és més gran que l'espai disponible. Conté un RelativeLayout que actua com a contenidor per al RecyclerView.
7. **RecyclerView**: Mostra una llista de supermercats. Ocupa tot l'espai disponible dins del ScrollView.
8. **Button**: Un botó amb el text "Endavant" que ocupa tot l'ample disponible dins del LinearLayout.

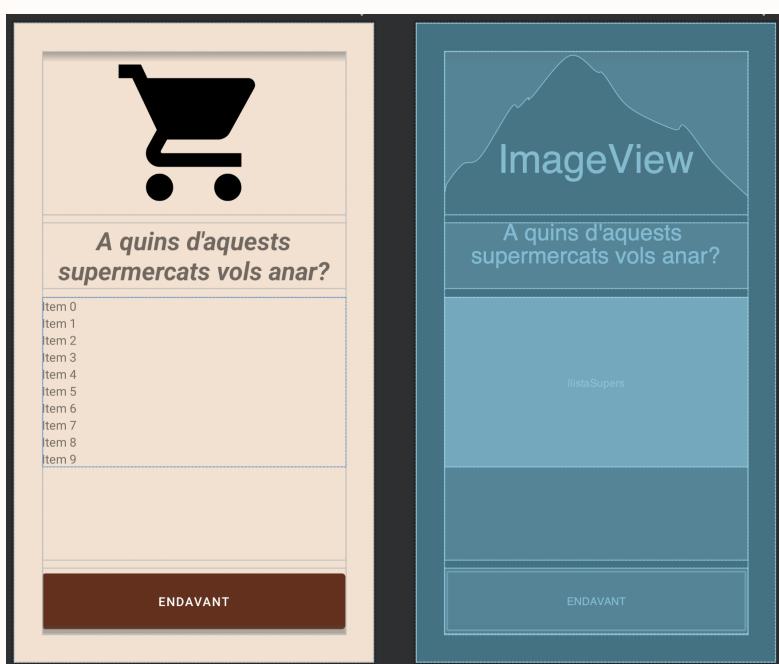


Figura 2.8: El Layout del fragment del càcul de la llista de supermercats (*FragmentSupermercats*).

La vista es defineix utilitzant un ConstraintLayout que omple tota la pantalla. Dins del ConstraintLayout, hi ha un FrameLayout que ocupa tota la pantalla. Dins del FrameLayout, es troba un LinearLayout que és responsable de mostrar el contingut de la pàgina. Aquest LinearLayout té orientació vertical. També s'aplica un marge de 32dp per donar espai als elements. Dins del LinearLayout, hi ha tres elements: Un ImageView, un TextView amb l'identificador, un ScrollView amb l'ample i l'alçada ajustables basats en el pes relatiu.

El codi XML, A.9, defineix una interfície d'usuari amb una imatge, un text, una llista desplaçable i una barra de progrés, tot dins d'un ConstraintLayout. El LinearLayout s'utilitza com a contenidor per als elements i el FrameLayout es fa servir per a la barra de progrés.



Figura 2.9: El Layout del fragment d'escol·lir llista, (*FragmentEscollirLlista*).

El ConstraintLayout és el contenidor principal, que indiquen que ocupa tot l'espai disponible a la pantalla. Dins del ConstraintLayout, hi ha un FrameLayout que també ocupa tota la pantalla. A continuació, s'estableixen les restriccions amb propietats que fixen el FrameLayout als límits de la pantalla. Dins del FrameLayout, hi ha un LinearLayout que ocupa tot l'espai disponible. L'orientació del LinearLayout està establerta com a vertical. Dins del LinearLayout, hi ha diversos elements: Un ImageView, un TextView i un ScrollView.

Dins del RelativeLayout, hi ha un RecyclerView. Fora del LinearLayout, hi ha un FrameLayout que ocupa tota la pantalla. A dins, hi ha un ProgressBar.

Ara l'últim Layout que comentarem. La resta són **molt simples**: per exemple, el layout d'un element d'una llista. Tots els que queden són bastant similars entre ells i, per no repetir-nos més tampoc, es podrà trobar la resta de Layouts utilitzats a [Cidis dels Layout](#).

El codi XML, A.10, defineix una interfície d'usuari amb un ScrollView que conté un RecyclerView, tot dins d'un ConstraintLayout. Aquesta estructura permet mostrar una llista de productes per comprar, amb la capacitat de desplaçar-se verticalment si la llista és llarga.



Figura 2.10: El Layout del fragment d'escol·lir llista, (*FragmentVeureProductes*).

El ConstraintLayout és el contenidor principal, que indiquen que ocupa tot l'espai disponible a la pantalla. Dins del ConstraintLayout, hi ha un FrameLayout que també ocupa tota la pantalla. A continuació, s'estableixen les restriccions amb propietats que fixen el FrameLayout als límits de la pantalla. Dins del FrameLayout, hi ha un LinearLayout que ocupa tot l'espai disponible. L'orientació del LinearLayout està establerta com a vertical. Dins del LinearLayout, hi ha un ScrollView que ocupa tot l'espai disponible. Dins del ScrollView, hi ha un RecyclerView.

2.2

OVERVIEW DEL MODEL

En aquesta secció, ens centrarem en el model d'una aplicació.

Definició 2.3 (Model). El *model* d'una aplicació és la part de l'aplicació que es relaciona amb les dades i la lògica de negoci. Això inclou la definició de les estructures de dades, les relacions entre elles i les operacions que s'hi poden realitzar a sobre.

Explorarem diferents estratègies per dissenyar el model de l'aplicació, les seves classes, les seves relacions i, en definitiva, el que hem exposat a 2.3. També abordarem com integrar la capa de persistència i la capa de vista amb el model per crear una aplicació eficient i funcional. A més, parlarem sobre com implementar la seguretat en el model per protegir les dades de l'usuari i com gestionar les excepcions i els errors per assegurar que l'aplicació sigui robusta i segura.

En general, comprendre el model d'una aplicació és crucial per a la creació d'aplicacions eficients i funcionals. Per tant, en aquesta secció, explorarem les diferents opcions disponibles i aprendrem com seleccionar la millor estratègia per a les necessitats específiques de l'aplicació.

Les classes `User`, `LlistesCompra`, `Product` són força triviales, en el sentit que solament estan formades per un conjunt d'atributs amb els seus respectius *setters* i *getters*. La més interessant, doncs, és la `LlistesCompraRepository` que comentarem tot seguit: La classe `LlistesCompraRepository` és una classe de repositori que utilitzava Firebase Firestore per gestionar les operacions de creació, lectura, actualització i eliminació de llistes de compra i productes associats per un usuari específic. Utilitzava una interfície per definir els mètodes que es poden utilitzar per interactuar amb Firebase, com ara la cerca, afegir, actualitzar i eliminar dades. Les principals funcionalitats són:

1. Cerca una llista de compra per nom i adreça de correu i l'elimina de Firebase Firestore.
2. Afegir una nova llista de compra per un usuari específic a Firebase Firestore.
3. Afegir un producte a una llista de compra específica d'un usuari a Firebase Firestore.
4. Eliminar un producte d'una llista de compra específica d'un usuari a Firebase Firestore.
5. Marcar o desmarcar un producte com a comprat en una llista de compra específica d'un usuari a Firebase Firestore.
6. Suport per afegir i eliminar listeners per rebre notificacions quan es carreguen les dades des de Firebase.

Comentarem una altra classe molt important, perquè ens serveix de connexió amb la nostra base de dades, és a dir, la nostra capa de persistència. La resta de classes amb el sufix *Repo* seran anàlogues i ja no n'explicarem més en detall, tot i que es podran trobar llistades a [Codi del model](#).

Dit això, parlem de `InfoProductRepository`. Aquesta classe fa servir *Firebase* per a connectar-se a una base de dades NoSQL en la núvol per a carregar informació sobre productes i supermercats.

1. En particular, aquesta classe defineix dos listeners que són cridats quan la informació dels productes o dels supermercats s'ha carregat completament. Aquests listeners permeten als desenvolupadors de l'aplicació respondre a aquests esdeveniments i actualitzar l'aplicació amb la informació carregada.
2. A més, la classe `InfoProductRepository` defineix diversos mètodes que interactuen amb la base de dades Firebase per a obtenir la informació necessària. Els mètodes com `getInfoProduct()` i `loadInfoAllProducts()` realitzen consultes a la base de dades i carreguen la informació resultant en estructures de dades locals que es poden accedir posteriorment.

Comentarem una altra classe molt important, la qual implementa una calculadora que calcula els preus més econòmics de la compra, utilitzant una matriu de dades.

Aquesta classe **Calculadora** forma part del paquet del model i té com a finalitat proporcionar la funcionalitat d'una calculadora per calcular els preus més econòmics per a una compra. A continuació es detalla el contingut de la classe:

1. S'importen les classes MutableLiveData i Observer de androidx.lifecycle, així com les classes ArrayList i HashMap de java.util. També s'importa la classe CalculadoraViewModel.
2. El constructor de la classe està buit.
3. El mètode calcularPreu utilitza diferents variables per a l'execució del càlcul. Aquestes variables inclouen dades de tipus HashMap<String, ArrayList<String>>, price de tipus double, minPrice de tipus double, supermercat de tipus String i minSuper de tipus String.
 - El mètode itera a través de la llista de productes i, per a cada producte, obté la informació associada als supermercats des de la matriu matrix. Llavors, compara els preus del producte en cada supermercat, tenint en compte els supermercats especificats en el paràmetre supermercats i si el producte està disponible en el supermercat.
 - A mesura que itera, el mètode actualitza les variables minPrice i minSuper amb el preu més baix i el supermercat associat més econòmics per a cada producte.
 - Finalment, el mètode construeix el resultat en el HashMap dades. Si el supermercat ja existeix com a clau en dades, afegeix el producte a la llista de productes associats a aquest supermercat. Si el supermercat no existeix, crea una nova llista amb el producte i l'afegeix al HashMap amb el supermercat com a clau.
 - Al final, el mètode retorna el HashMap dades que conté la informació dels supermercats més econòmics per a cada producte.

CAPA DE LA VISTA 2.3

En aquesta secció, explorarem la capa de vista d'una aplicació.

Definició 2.4 (Vista). La *capa de vista* és la part de l'aplicació que es relaciona amb la interfície d'usuari. Això inclou tots els components de la interfície, com ara els botons, camps de text, imatges i altres components.

Analitzarem com dissenyar una interfície d'usuari atractiva i fàcil d'usar, utilitzant diferents elements de disseny com ara colors, fonts i disposició de la informació. També explorarem diferents estratègies per a la navegació i la interacció amb l'usuari per assegurar que l'aplicació sigui intuïtiva i fàcil d'utilitzar. A més, parlarem sobre com implementar la interacció amb l'usuari, com ara la validació de dades i la gestió d'errors, per assegurar que l'aplicació sigui robusta i segura.

La vista de la nostra aplicació es fonamenta en un conjunt de diferents *Activity*.

Definició 2.5 (Activity). És una de les components fonamentals de l'arquitectura d'aplicacions Android basada en el patró de disseny Model-Vista-Controlador (MVC) o Model-Vista-ViewModel (MVVM). Un Activity en Android Studio defineix la interacció de l'usuari amb l'aplicació a través de la creació d'una interfície d'usuari que pot contenir elements com ara botons, camps de text, imatges i altres elements interactius. L'Activity és responsable de gestionar els cicles de vida de l'aplicació, com ara la creació, pausa, represa i destrucció, i també gestiona les interaccions d'usuari, com ara les entrades de l'usuari i les respostes de l'aplicació.

A continuació, una classe d'activitat (Activity) anomenada `MainActivity` que hereta de la classe `AppCompatActivity` del *framework* d'Android. Aquí es resumeix el seu contingut, que es basa en el mètode `onCreate`. Aquest mètode és cridat quan l'activitat és creada. Sobreescritint-lo, es defineix el comportament de l'activitat quan es crea.

1. Es configura la vista de l'activitat mitjançant el mètode `setContentView` per establir el contingut de la interfície d'usuari a partir del fitxer de disseny `activity_main` definit en el directori de recursos de l'aplicació.
2. Es vinculen les variables declarades amb els botons de la interfície d'usuari utilitzant el mètode `findViewById` `By Id` i passant-li els identificadors dels botons definits en el fitxer de disseny.
3. Es configura un `OnClickListener` per a cada botó utilitzant el mètode `setOnClickListener` per definir les accions que s'han de realitzar quan els botons siguin clicats.
4. Quan es fa clic al botó de login, es crea un intent per iniciar l'activitat `IniciSessioActivity` utilitzant la classe `Intent`, i s'inicia l'activitat mitjançant el mètode `startActivity`.

5. Quan es fa clic al botó de registre, es crea un intent per iniciar l'activitat `RegistreActivity` i s'inicia l'activitat mitjançant el mètode `startActivity`.

Passem a un altre codi, IniciSessioActivity. El següent codi serà el de l'inici de sessió. És una classe d'activitat (Activity) anomenada `IniciSessioActivity` que hereta de la classe `AppCompatActivity` del framework d'Android. Aquí es resumeix el seu contingut:

1. Mètode `onCreate`: Aquest mètode és cridat quan l'activitat és creada. Sobreescrivint-lo, es defineix el comportament de l'activitat quan es crea.

- Es configura la vista de l'activitat mitjançant el mètode `setContentView` per establir el contingut de la interfície d'usuari a partir del fitxer de disseny `activity_inici_sessio` definit en el directori de recursos de l'aplicació.
- Es vinculen les variables declarades amb els elements de la interfície d'usuari utilitzant el mètode `findViewById` i passant-li els identificadors corresponents definits en el fitxer de disseny.
- Es configura el títol de l'activitat a través del mètode `setTitle` de la classe `ActionBar` per establir el títol de la barra d'accions de l'activitat com Iniciar sessió.
- Es configura un `OnClickListener` per al botó de fer login utilitzant el mètode `setOnClickListener` per definir les accions que s'han de realitzar quan es fa clic en el botó.
- Quan es fa clic al botó de fer login, es criden els mètodes `getText` dels camps d'edició d'email i contrasenya per obtenir les dades introduïdes per l'usuari.
- Es crida el mètode `login` amb les dades d'usuari obtingudes per gestionar el procés d'autenticació amb Firebase.

2. Mètode `login`: Aquest mètode realitza l'autenticació de l'usuari amb Firebase utilitzant l'adreça de correu electrònic i la contrasenya proporcionades.

- Es crida el mètode `signInWithEmailAndPassword` de l'objecte `mAuth` de la classe `FirebaseAuth`, passant-li l'adreça de correu electrònic i la contrasenya obtingudes.
- Es defineix un `OnCompleteListener` per gestionar el resultat de l'autenticació.
- Si l'autenticació és exitosa, es crea un intent per iniciar l'activitat `LlistaCompraActivity` i es passa l'adreça de correu electrònic com a dada adicional mitjançant l'ús del mètode `putExtra` de la classe `Intent`. A continuació, s'inicia l'activitat mitjançant el mètode `startActivity` amb l'intent creat.
- Si l'autenticació falla, es mostra un missatge d'error a través d'un `Toast` utilitzant el mètode `makeText` de la classe `Toast` per mostrar un missatge breu a l'usuari amb la indicació que el login ha fallat.

- Una petita observació, tenim un `FrameLayout` amb una `ProgressBar` que es farà visible quan haguem picat el `Button` o bé si hem seleccionat anteriorment que recordi les nostres credencials. De fet, en aquest últim cas, visibilitzem aquest `FrameLayout` directament i passem a la pantalla de `VeureLlistaCompraActivity`.

Ara parlarem de la classe `RegistreActivity`. Aquesta implementa una activitat d'Android Studio que permet als usuaris registrar-se amb un nou compte utilitzant Firebase com a sistema d'autenticació i Firestore com a base de dades NoSQL per emmagatzemar les dades dels usuaris registrats. Resumidament, l'activitat principal és `RegistreActivity`, que té quatre camps d'entrada d'informació (nom, cognom, correu electrònic i contrasenya) i un botó per realitzar el registre. Quan l'usuari fa clic al botó de registre, es crida a la funció `signup()` que crea un nou usuari utilitzant Firebase Authentication i, si té èxit, crida a la funció `signup_db()` per desar les dades de l'usuari (nom, cognom i contrasenya) a Firebase Firestore. Si el registre falla, es mostra un missatge d'error en un `Toast`.

1. La classe `RegistreActivity` hereta de `AppCompatActivity`, que és una classe base per a les activitats d'Android que proporciona compatibilitat amb versions anteriors de l'API d'Android.
2. Es declaren els elements d'interfície d'usuari necessaris, com ara botons (`btn_fer_sign_up`), camps d'edició (`name`, `cognom`, `mail`, `pwd`) i instàncies de les classes `FirebaseAuth` (`mAuth`) i `FirebaseFirestore` (`db`) per interactuar amb Firebase Auth i Firestore.
3. En el mètode `onCreate`, s'estableix el contingut de l'activitat utilitzant el mètode `setContentView` amb l'arxiu de disseny `activity_registre.xml`.
4. S'estableix el títol de l'activitat a través del mètode `getSupportActionBar().setTitle`.
5. Es configura un `OnClickListener` per al botó `btn_fer_sign_up` que es dispara quan l'usuari fa clic en aquest botó. Quan es fa clic, es crida al mètode `signup` amb les dades d'usuari introduïdes als camps d'edició com a paràmetres.
6. El mètode `signup` fa servir la instància de `FirebaseAuth` (`mAuth`) per crear un nou compte d'usuari utilitzant l'adreça de correu electrònic i la contrasenya introduïdes pels usuaris amb el mètode `createUserWithEmailAndPassword`. Es configura un `OnCompleteListener` per gestionar el resultat de la tasca d'autenticació. Si l'autenticació és exitosa (`task.isSuccessful()`), s'invoca el mètode `signup_db` per afegir les dades d'usuari a la base de dades Firestore mitjançant el mètode `signup_db`, i es mostra un missatge breu amb un `Toast` que indica que el registre ha tingut èxit. En cas contrari, es mostra un missatge amb un `Toast` que indica que el registre ha fallat.

7. El mètode `signup_db` crea un nou document a la col·lecció `users` de Firestore amb l'adreça de correu electrònic de l'usuari com a identificador de document i guarda les dades d'usuari com a parells clau-valor en aquest document mitjançant el mètode `set`. Les dades d'usuari són emmagatzemades en un objecte `Map` que conté les claus nom, cognom i contrasenya.

Ara doncs, ens queden les classes `LlistaCompraActivity` i `VeureLlistaCompraActivity`, i una última `VeureDetailsProducteActivity`. **La resta de classes les trobarem a A.** La classe `LlistaCompraActivity` és una classe en Java que representa una activitat d'una aplicació Android per gestionar llistes de la compra. A continuació es comenten els aspectes més rellevants del codi:

1. La classe estén de `AppCompatActivity`, que és una classe base d'Android per a activitats que utilitzen la compatibilitat amb versions anteriors del sistema operatiu.
2. La classe té diverses variables d'instància, com ara `TAG` que s'utilitza per etiquetar `logs` i `llistesCompraActivityViewModel` que és una instància del `ViewModel` utilitzat a l'activitat.
3. En el mètode `onCreate()` es realitza la inicialització de l'activitat. S'estableix el disseny de l'activitat mitjançant `setContentView()` que infla l'arxiu de disseny `activity_llista_compra.xml`.
4. S'obtenen els extres passats a l'activitat a través de l'intent i s'assignen a les variables corresponents, com ara `mailUser`, amb `getIntent().getExtras()`, inicialitzar variables i referències, configurar adaptadors per a la llista de productes i establir observadors per a actualitzacions de dades.
5. Es crea una instància del `ViewModel` `LlistesCompraActivityViewModel` mitjançant `ViewModelProvider` i s'assigna a la variable `llistesCompraActivityViewModel`. S'obté una referència al `RecyclerView` mitjançant `findViewById()` i es configura el seu `LayoutManager` i `Adapter`. L'`Adapter` utilitzat és `LlistesCompraCardAdapter` i s'assigna a `mLlistaCardRVAdapter`. S'implementen `listeners` a l'`Adapter` per gestionar esdeveniments de clic als elements de la llista. Per exemple, quan es fa clic en un element, es crea un intent per iniciar una nova activitat `VeureLlistaCompraActivity` i se li pasen dades addicionals a través de `putExtra()`. Es registra un observador al `LiveData` `mCompra` del `ViewModel`, per rebre notificacions de canvis a la llista de compres. També, al botó d'afegir una llista, `btn_afegir_llista`. Finalment, es crida al mètode `loadLlistesCompraFromRepository()` del `ViewModel` per carregar les llistes de compres des del repositori i poblar la llista del `RecyclerView`.
6. La classe té mètodes per a afegir i eliminar llistes de compra, així com per a gestionar les interaccions de l'usuari, com ara tocar una llista de compra per a veure els detalls o tocar un botó per a afegir una nova llista. També inclou mètodes per a mostrar diàlegs d'alerta per a confirmar l'eliminació o afegir una nova llista.

7. El mètode `onOptionsItemSelected(MenuItem item)` és cridat quan es selecciona una opció del menú. En aquest cas, si s'ha seleccionat l'opció amb l'identificador `R.id.action_deletelist`, s'alterna la visibilitat dels elements de la interfície d'usuari (botons, caixes de selecció) i s'actualitza la variable `checkboxesVisible` en conseqüència.
8. El mètode `onBackPressed()` és cridat quan es prem el botó de retrocés del dispositiu. En aquest cas, es crea un intent per a redirigir l'usuari a l'activitat principal (`MainActivity`). Això es fa per a proporcionar una transició suau quan es prem el botó de retrocés i evitar tornar a l'activitat anterior.
9. Hi ha implementacions de mètodes com `onCreateOptionsMenu` i `onOptionsItemSelected` per a gestionar les opcions del menú.

Comentem `VeureLlistaCompraActivity`. El següent codi representa una activitat d'una aplicació Android per veure una llista de la compra.

1. La classe té diverses variables d'instància, com ara `nomLlistaCompra`, `mailUser` i `llistaProducts`, que són utilitzades per emmagatzemar les dades passades a l'activitat a través dels extres de l'intent.
2. En el mètode `onCreate()` es realitza la inicialització de l'activitat. S'estableix el disseny de l'activitat mitjançant `setContentView()` que infla l'arxiu de disseny `activity_veure_llista_compra.xml`.
3. S'obtenen els extres passats a l'activitat a través de l'intent mitjançant `getExtras()` i s'assignen als corresponents membres de la classe, com ara `nomLlistaCompra`, `mailUser` i `llistaProducts`.
4. Es configura el títol de l'`ActionBar` de l'activitat amb `getSupportActionBar().setTitle()` utilitzant el valor de `nomLlistaCompra`.
5. Es crea una instància de `ProductesActivityViewModel` utilitzant `ViewModelProvider`. Es configura el `RecyclerView` i s'estableix l'adaptador `mLlistaProductesCardRVAdapter`. Es configuren diversos listeners per als botons i l'adaptador. Es defineix un `Observer` per observar els canvis en la llista de productes i es carreguen els productes des del repositori utilitzant el `ViewModel`. S'implementen els mètodes `onCreateOptionsMenu()` i `onOptionsItemSelected()` per gestionar les opcions del menú.
6. El mètode `onBackPressed()` es crida quan l'usuari prem el botó de retrocés del dispositiu. En aquest cas, s'inicia una nova activitat (`LlistaCompraActivity`) mitjançant un intent i es finalitza l'activitat actual.

Ens queda, finalment, una única classe més per veure, `VeureDetallsProducteActivity`. Aquesta classe té molta relació amb una secció que veurem més endavant, *Scraping de dades*. Aquest codi és una activitat de l'aplicació que mostra els detalls d'un producte. Es fa servir l'arquitectura Model-View-ViewModel (MVVM) per gestionar la lògica de la vista i la persistència de dades. En concret, s'utilitzen les llibreries `AndroidX LiveData` i `ViewModel` per gestionar la vida útil de les dades i actualitzar la vista quan canviïn.

1. En primer lloc, es creen les variables necessàries per als elements gràfics que es mostraran a la pantalla. Es defineixen també les variables necessàries per a la gestió de la informació de la base de dades, com ara el ViewModel i el Repository.
2. A continuació, al mètode `onCreate`, s'inicialitzen els elements gràfics, s'obtenen els paràmetres extras que s'han passat a l'activitat i s'obté el nom del producte. Es crea una instància del ViewModel i s'associa amb l'activitat.
3. Es descodifica la imatge del producte a partir del codi Base64 i es mostra en el ImageView corresponent.
4. S'afegeix un observador que escolta els canvis en la llista de supermercats on es pot comprar el producte. Quan es rep una llista vàlida, es recorre la llista i s'afegeixen els noms dels supermercats a l'ArrayList `llistaSupers` i s'afegeixen les instàncies de SpinnerItem a l'ArrayList `mListSpinnerItems`¹. S'associa l'adaptador SpinnerAdapter a l'Spinner `spinner`, el qual mostra els noms dels supermercats. En concret, els elements que es mostren en el Spinner s'obtenen de la llista d'objectes `InfoSuperProducte`. Per cada element d'aquesta llista, s'afegeix un nou element a la llista `mListSpinnerItems` amb el nom del supermercat seleccionat.
5. Quan se selecciona un supermercat en l'spinner, es recorre la llista de `InfoSuperProducte` per obtenir el preu corresponent i es mostra en el TextView `text_preu`.
6. Finalment, s'observa la llista d'objectes `InfoSuperProducte` a través del ViewModel `infoProducteActivityViewModel`. Quan aquesta llista és actualitzada, s'executa la funció `onChanged` que omple les llistes `llistaSupers` i `mListSpinnerItems` i actualitza l'adapter del Spinner.

Definició 2.6 (Fragment). Els *Fragment* permeten dividir la interfície d'usuari en components més petits i modulares, el que facilita el desenvolupament i la gestió de la interfície d'usuari en aplicacions més grans. Cada Fragment té el seu propi disseny XML, controladors d'esdeveniments i lògica associada.

Els Fragments són gestionats per una activitat pare i poden ser afegits o reemplaçats en un contingut de fragments dins de l'activitat. Això permet crear interfícies d'usuari flexibles i adaptables, ja que es poden combinar múltiples fragments en una sola activitat.

Comentarem quatre últimes classes, l'activitat que s'encarrega de la calculadora i els tres fragments que hem creat per a aquesta última entrega.

Aquesta classe `CalculadoraActivity` és responsable de mostrar la interfície d'usuari de la calculadora i gestionar els fragments relacionats amb la selecció dels supermercats per al càlcul de preus. A continuació es detalla el contingut de la classe:

¹ La variable `llistaSupers` és inicialitzada com una llista buida d'Strings, i s'omplirà posteriorment amb noms de supermercats.

1. La classe declara diverses variables de classe, com matriuProductesPreus de tipus `ArrayList<ArrayList<InfoSuperProducte>>`, `nomSupers` de tipus `ArrayList<String>` i calculadoraActivityViewModel de tipus `CalculadoraActivityViewModel`.
2. La classe sobreescrivia el mètode `onCreate` per a l'inicialització de l'activitat. En aquest mètode, es realitza el següent:
 - S'inicialitza l'activitat amb el mètode `setContentView` que defineix la interfície d'usuari a partir del fitxer de disseny `activity_calculadora.xml` que es troba a la carpeta de recursos.
 - Es recupera el Bundle d'arguments passat a l'activitat mitjançant `getIntent().getExtras()`. Es crea un nou bundle i es guarda en ell l'`ArrayList` de noms de productes (`nomsProductes`) extret del Bundle d'arguments. Aquest bundle es passarà com a arguments a un fragment.
 - Es crea una instància del fragment `FragmentEscol·lirLlista` i s'estableixen els arguments mitjançant `setArguments(bundle)`. Es gestionen els fragments utilitzant `FragmentManager` i `FragmentTransaction`. Es realitza una transacció per reemplaçar el contingut del contenidor pel fragment creat anteriorment.

Aquest fragment `FragmentEscol·lirLlista` forma part de la funcionalitat de la calculadora de preus i té com a objectiu mostrar una llista de supers disponibles per a la compra i permetre als usuaris seleccionar un supermercat per veure els productes corresponents i el seu preu.

1. Es realitzen diverses importacions de classes, incloent models com `InfoSuperProducte` i `LlistesCompra`, i una classe de visualització de dades, `CalculadoraActivityViewModel`.
2. La classe declara diverses variables de classe, com `calculadoraActivityViewModel` de tipus `CalculadoraActivityViewModel`, `mLlistaCardsRV` de tipus `RecyclerView`, `mLlistaCardRVAdapter` de tipus `LlistaSupermercatCardAdapter`, i altres variables com `llistaSupers`, `llistaNomProductes` i `matrixLoaded`.
3. Sobreescrivia els mètodes `onCreateView` i `onViewCreated` per a la inicialització del fragment i la seva vista. Aquests mètodes inflen i estableixen la vista del fragment utilitzant el fitxer de disseny `fragment_escol·lir_llista.xml`.
4. En el mètode `onViewCreated`, es recupera el Bundle d'arguments passat al fragment i s'obtenen les llistes de noms de supermercats i de productes.
5. Es crea una instància del `CalculadoraActivityViewModel` i s'inicialitzen els components de la interfície d'usuari, com el `RecyclerView` i l'adaptador `LlistaSupermercatCardAdapter`. Aquest adaptador es configura amb les llistes de supers i s'estableix un listener per al clic en un supermercat.

6. S'implementa un Observer per a la càrrega de la matriu de dades, que es notifica a través del CalculadoraActivityViewModel. Quan la matriu de dades està plena, es canvia el valor de matrixLoaded a true.
7. Quan es fa clic en un supermercat, es verifica si la matriu de dades ja s'ha carregat. Si no, es mostra una barra de progrés i s'estableix un retard de 1 segon abans de cridar de nou aquest mètode. Un cop la matriu de dades està carregada, es crea un Bundle amb les dades de productes del supermercat seleccionat i s'inicia una transacció per mostrar el fragment FragmentVeureProductes amb aquestes dades.

El codi FragmentSupermercats crea un fragment que permet als usuaris seleccionar supermercats i productes, i després transmet aquestes seleccions a un altre fragment per a mostrar una llista d'opcions seleccionades.

1. La classe estén la classe Fragment i té diversos importacions necessàries per a la seva funcionalitat.
2. Té una instància de CalculadoraActivityViewModel, una classe de ViewModel utilitzada per a la comunicació entre fragments i l'activitat. Declara una instància de ListAdapter per utilitzar en un RecyclerView. Sobreescriu els mètodes onCreateView i onViewCreated per a la inicialització i configuració del fragment.
3. En onCreateView, infla el layout del fragment des de R.layout.fragment_calcul_llista_supers, A.8. En onViewCreated, configura el ViewModel, estableix el títol de l'activitat, obté els arguments passats al fragment, inicialitza una llista d'elements i configura un OnClickListener per al botó button_supers.
4. En el OnClickListener, recorre els elements de la llista i afegeix els elements seleccionats a una nova llista. Crea un Bundle amb les llistes seleccionades i els productes, i crea una instància d'un altre fragment anomenat FragmentEscoldirLlista.
5. Configura el Bundle com a arguments per al fragment creat i realitza una transacció de fragments per a mostrar el nou fragment.
6. Defineix un Observer per a observar una llista de supermercats. Quan es produueixen canvis en aquesta llista, es crea una nova llista d'elements a mostrar en un RecyclerView. S'estableix l'Observer per a observar els canvis en la llista de supermercats i es carreguen els supermercats a través del ViewModel.

FragmentVeureProductes crea un fragment que mostra una llista de productes a comprar. Utilitza un RecyclerView per a mostrar els productes utilitzant l'adaptador ProductesAComprarCardAdapter, del qual no parlem però podem trobar a A.39. També configura el ViewModel i obté els noms dels productes a través dels arguments passats al fragment.

1. Té una instància de CalculadoraActivityViewModel. Recordem que aquesta és una classe de ViewModel utilitzada per a la comunicació entre fragments i l'activitat.
2. Declara una instància de RecyclerView i un adaptador ProductesAComprarCardAdapter per utilitzar en el RecyclerView. Sobreescriu els mètodes onCreateViewHolder i onViewCreated per a la inicialització i configuració del fragment.
3. En onCreateView, infla el layout del fragment des de R.layout.fragment_veure_productes, A.10. En onViewCreated, configura el ViewModel, estableix el títol de l'activitat, obté els arguments passats al fragment i inicialitza una llista de noms de productes.
4. Configura el RecyclerView amb un LayoutManager i l'adaptador ProductesAComprarCardAdapter. Crea una nova instància del mateix fragment FragmentVeureProductes, però no es realitza cap acció amb aquesta nova instància.

2.4 VIEWMODEL

El ViewModel emmagatzema dades relacionades amb la UI de manera que persisteixen durant els canvis de configuració de l'aplicació, com quan l'usuari rota el dispositiu o canvia de configuració d'idioma. Això permet que les dades es mantinguin actualitzades i disponibles per a la UI, independentment de l'estat de l'activitat o fragment.

Definició 2.7 (ViewModel). El *ViewModel* és útil per a la separació de responsabilitats entre la lògica de negoci i la UI. Permet mantenir una lògica de negoci més robusta i escalable en una classe separada que pot ser reutilitzada en diferents activitats o fragments. El ViewModel també és conscient del cicle de vida de l'activitat o fragment associat, i es destrueix automàticament quan aquests components són destruïts, evitant així fuites de memòria.

Hi ha cinc classes presents en aquest paquet:

1. CalculadoraActivityViewModel,
2. InfoProducteActivityViewModel,
3. LlistesCompraActivityViewModel,
4. ProductesActivityViewModel,
5. UsersActivityViewModel.

Comentarem ara la classe `LlistesCompraActivityViewModel`, la qual ens permet gestionar les dades de les llistes de compra en una aplicació mitjançant l'ús de `MutableLiveData` per a mantenir les dades observades i un repositori per a gestionar l'accés a les dades. Aquesta classe té algunes característiques i funcionalitats que es podrien destacar:

1. El ViewModel té una referència a una instància de l'aplicació (`Application`) passada com a paràmetre al constructor. Això permet accedir als recursos i context de l'aplicació des del ViewModel.
2. El ViewModel té una propietat observada (`mCompra`) que és una instància de `MutableLiveData` que conté una llista d'objectes de `LlistesCompra`. `MutableLiveData` és una classe que proporciona dades observables que es poden actualitzar i observar des de components d'`UI` com ara activitats o fragments.
3. El ViewModel utilitza un `LlistesCompraRepository` per a gestionar l'accés a les dades de les llistes de compra. Aquest repositori sembla ser una classe personalitzada que implementa la lògica de negoci per a les operacions de lectura i escriptura de les llistes de compra.
4. El ViewModel té mètodes per a carregar les llistes de compra des del repositori (`loadLlistesCompraFromRepository`), obtenir els productes d'una llista de compra específica (`getmProducts`) i eliminar una llista de compra del repositori i de la llista observada (`removeLlistaFromHome`).
5. El ViewModel també implementa un listener (`OnLoadLlistesCompraListener`) per a rebre les actualitzacions de les llistes de compra des del repositori i actualitzar la llista observada en conseqüència.

Comentarem una última classe, ja que la resta són força anàlogues. Al final, aquestes classes no són sinó enllaços entre el model i la vista.

`CalculadoraActivityViewModel` defineix un ViewModel per a l'activitat de la calculadora. El ViewModel gestiona les dades observables i interactua amb el repositori per a carregar les dades necessàries per a la funcionalitat de la calculadora. També proporciona mètodes per a realitzar càlculs relacionats amb la calculadora.

1. Declara diverses instàncies de `MutableLiveData` per emmagatzemar dades observables. Té una instància de `CalculadoraRepository` per a interactuar amb la capa de dades de la calculadora. Declara una instància de `Calculadora` per a realitzar càlculs relacionats amb la calculadora.
2. El constructor de la classe inicialitza les instàncies i configura els observadors per a actualitzar les dades observables quan es carreguen els supermercats i la informació dels supermercats.
3. La classe té diversos mètodes per a obtenir i actualitzar les dades observables i realitzar càlculs relacionats amb la calculadora. Els mètodes `loadSupersFromRepository` i `loadInfoMatrix` interactuen amb el `CalculadoraRepository` per a carregar les dades dels supermercats i la informació dels supermercats.

2.5

CAPA DE PERSISTÈNCIA I SCRAPING DE DADES

En aquesta secció, explorarem la capa de persistència d'una aplicació.

Definició 2.8 (Persistència). La *capa de persistència* és responsable de gestionar la persistència de les dades de l'aplicació, és a dir, com les dades es guarden i es recuperen de la memòria permanent, com ara discs durs, bases de dades o altres sistemes d'emmagatzematge.

Analitzarem diferents opcions per gestionar la persistència de les dades, incloent bases de dades no relacionals com *Firebase*, fitxers de text i altres opcions. També explorarem com utilitzar les diferents característiques de cada opció per crear una capa de persistència eficient i fiable per a l'aplicació. A més, parlarem sobre com implementar la seguretat i la privacitat a la capa de persistència de l'usuari.

En un servei destinat a la cerca exhaustiva entre centenars de preus i productes cal accés a moltíssimes dades. Entre elles, les fotografies de tals productes, per poder llistar-los a la nostra aplicació de la manera més visual possible. En aquest sentit, necessitem una correspondència un a un (cada ítem té la seva pròpia foto, i cada foto està vinculada a un sol ítem).

En primer lloc, per tal que la nostra aplicació funcioni correctament, és necessari que tinguem tota la informació de cada producte a la nostra base de dades del *Firebase*. D'aquesta manera, considerem que per a cada producte necessitem emmagatzemar la seva disponibilitat i preu de cada supermercat, el seu nom i el seu identificador i, per acabar una fotografia que el representi per poder mostrar-la en l'aplicació.

D'una banda, ens centrarem primerament en explicar com s'ha aconseguit la informació dels productes relatives a la disponibilitat, el preu i el nom. En primer lloc, destacar que no hem utilitzat una gran quantitat de productes perquè considerem que no es necessari per comprovar el correcte funcionament de l'aplicació. En aquest sentit, la nostra base de dades constarà d'un total de 77 productes per fer aquest testeig.

Pel que fa a com s'ha emmagatzemat aquesta informació en el *Firebase*, hem fet servir un altre projecte del *Android Studio*, l'hem connectat a la nostra base de dades i, mitjançant un arxiu de text (.txt), on es troba tota la informació necessaria de cada producte separada per comes, s'ha pujat a la base de dades. Per veure amb més detall el codi que s'ha utilitzat, es pot consultar en l'apartat *Codis del scraping de dades*.

D'altra banda, explicarem com s'ha gestionat tota la part de les fotografies de cada producte.

Nosaltres, doncs, hem decidit declarar un nou atribut dins el *Firebase* que contingui una referència a la imatge en qüestió. Cal destacar, però, que ho farem per certes raons d'una forma més ortodoxa: les usarem en for-

mat *Base64*. Obtindrem les dades que necessitem amb una *query* a la base de dades, en particular, la imatge, per tractar-la dins l'entorn de la nostra aplicació i projectar-la. Es pot veure a la classe `VeureDetallsProducteActivity`, el codi A.32.

Es pot trobar el codi a l'arxiu `python.pdf`.

2.6

TESTING

Potser el títol d'aquesta secció no és el més precís, però resumeix bastant el que ens demanen és l'avaluació de l'aplicació amb els usuaris en diversos aspectes. El *testing* és una part del desenvolupament de *software* i té com a objectiu maximitzar la qualitat i el rendiment. És important de planejar els procediments de testeig quan el desenvolupament en sí comença. **Com se'ns ha demanat a classe, no farem cap d'aquests tipus de testeig, simplement deixarem indicat com els faríem.**

2.6.1

UNIT TESTING

Definició 2.9 (Unit testing). El *unit testing* (provees unitàries) és una pràctica de prova en el desenvolupament de programari que implica verificar la funcionalitat de components individuals, anomenats *units*, de manera aïllada. L'objectiu principal del unit testing és assegurar-se que les unitats de codi funcionen com s'espera i detectar errors o comportaments inesperats. Al final, es vol detectar errors i comportaments inesperats abans d'integrar-les en el conjunt global del sistema.

Un bon conjunt de provees unitàries cobreix diferents casos i situacions que podrien produir errors, incloent-hi casos de prova límit o inesperats. Les provees unitàries s'executen amb freqüència durant el cicle de desenvolupament del programari per identificar ràpidament qualsevol problema i permetre als desenvolupadors realitzar correccions.

En definitiva, hauríem de picar codi per testejar cadascuna de les nostres funcionalitats: registrar-se, fer login, afegir un producte a la llista... Adonem-nos que, en la majoria d'ocasions, això coincidirà amb les classes que hem declarat en la nostra aplicació. Com hem dit, no ho farem, així que ho donarem a mitges tintes:

I. Preparació.

- *Identificar les unitats de codi a provar:* Com ja hem dit podríem prendre com a referència d'unitat de codi les classes o mòduls que gestionen la nostra aplicació: la creació, la modificació o l'eliminació d'elements de la llista de la compra, la gestió de categories, llistes o altres opcions addicionals.

- *Establir les condicions de prova:* Determinar els casos de prova, les dades d'entrada i les expectatives de sortida per a cada unitat de codi.

2. Creació de casos de prova.

- *Crear casos de prova per a cada funcionalitat:* Per exemple, en el nostre cas, provar la creació d'un nou element a la llista, la modificació d'un element existent, l'eliminació d'un element o la gestió de categories.
- *Incloure casos de prova amb valors límit o inusuals:* Provar situacions límit. Algunes d'aquestes, en el nostre context, llistes de la compra buides, llistes amb un únic element, afegir un producte inexistent, calcular la millor opció amb productes que no es troben als supermercats que hem seleccionat. Hem d'assegurar que el codi gestioni correctament aquestes situacions especials.
- *Verificar casos de prova per a situacions d'error:* Provar el comportament de l'aplicació quan es produueixen errors, com intents de creació o modificació amb dades incorrectes o incompletes. Hem d'advertir els usuaris amb errors coherents, amb missatges *Toast* per exemple, que puguin ser compresos amb facilitat (no li podem indicar, per exemple, que s'està produint una violació de segment, ja que s'estaria perdent el canal de comunicació: s'ha de mantenir el registre estàndard i no canviar a l'especialitzat).

3. Implementació de les proves.

- *Escriure codi de prova per a cada cas de prova:* Implementar els tests que utilitzin les funcionalitats de les unitats de codi i comprovin que les sortides coincideixin amb les expectatives. *Ja hem dit que això no ho farem.*
- *Gestionar la configuració i les dependències:* Preparar l'entorn de prova, incloent la configuració de bases de dades temporals o altres dependències necessàries. *No ho farem.*

4. Execució de les proves.

5. Anàlisi i resolució d'errors.

- *Depurar i corregir els errors:* En cas d'errors, rastrejar-los, comprendre'ls i corregir-los en el codi de la unitat afectada.
- *Reexecutar les proves:* Després de realitzar les correccions, tornar a executar les proves per assegurar-se que els errors han estat solucionats i les unitats de codi funcionen correctament.

2.6.2 INTEGRATION TESTING

Definició 2.10 (Integration testing). Les proves d'integració (*integration testing*) és el següent pas natural després de l'*unit testing*: es defineixen com un tipus de proves on els mòduls de programari s'integren de manera lògica i es posen a prova com a grup. Un projecte de programari típic consta de diversos mòduls de programari, escrits per diferents programadors. L'objectiu d'aquest nivell de proves és exposar defectes en la interacció entre aquests mòduls de programari quan estan integrats. Això ajuda a detectar i solucionar problemes d'integració abans que el sistema sigui desplegat en un entorn de producció, millorant la fiabilitat i la qualitat del programari.

Un possible esbòs és el següent.

1. Identificar els components a provar. Identificar els mòduls, serveis o components que formen part de l'aplicació de la llista de la compra, com ara la interfície d'usuari, la capa de lògica de negoci, la capa de persistència de dades i altres integracions amb serveis externs.

2. Establir els escenaris d'integració. Definir els escenaris d'integració clau que reflecteixin les funcionalitats importants de l'aplicació, que coincideixen en gran part amb les unitats de codi que comentàvem a l'apartat anterior, *unit testing*. També, determinar les dades d'entrada necessàries per a cada escenari d'integració.

3. Crear casos de prova d'integració.

- Desenvolupar casos de prova que reflecteixin els escenaris d'integració definits prèviament.
- Assegurar-se que els casos de prova cobreixin diferents combinacions de funcionalitats i interaccions entre els components.

4. Implementar les proves d'integració.

- *Configurar l'entorn de prova.*
- *Escriure codi de prova:* Implementar els casos de prova que provin l'interacció entre els components i verifiquin els resultats esperats.
- *Establir els criteris de validació:* Definir els criteris per a considerar una prova d'integració com a èxit o fallida.

5. Executar les proves d'integració.

6. Anàlisi i resolució d'errors.

- *Depurar i corregir els errors:* Rastrejar, comprendre i corregir els errors identificats durant les proves d'integració.

- *Reexecutar les proves:* Després de realitzar les correccions, tornar a executar les proves d'integració per assegurar-se que els components funcionen de manera satisfactòria.

7. **Validació final.** Verificar la coherència de les dades: Comprovar que les dades són coherents i consistents en tot el sistema, des de la interfície d'usuari fins a la capa de persistència de dades.

2.6.3

PERFORMANCE TESTING

És bastant anàleg als anteriors, així que resumirem.

Definició 2.11 (Performance testing). El *performance testing* (provees de rendiment) és una pràctica de prova en el desenvolupament de programari que té com a objectiu avaluar i mesurar el rendiment i l'eficiència d'un sistema o aplicació sota condicions específiques de càrrega. Aquest tipus de provees es realitza per determinar com respon el sistema en termes de temps de resposta, capacitat de processament, escalabilitat i altres factors relacionats amb el rendiment. Les provees de rendiment poden incloure l'avaluació de factors com el temps de resposta del sistema, la capacitat de processament, la velocitat de transferència de dades i la tolerància a la càrrega i l'escalabilitat. Aquestes provees són essencials per identificar i resoldre problemes de rendiment, com ara punts de fallada, embussos de rendiment o ineficiències en l'execució del codi. *Performance testing* requerirà de dades mètriques de rendiment, indicadors i criteris.

1. **Definir els objectius de rendiment.** Establir els criteris de rendiment desitjats per a l'aplicació, com ara temps de resposta màxim per a les operacions crítiques i nombre màxim d'usuaris simultanis que l'aplicació ha de suportar.
2. **Identificar els escenaris de prova de rendiment.** Determinar els escenaris d'ús crítics de l'aplicació (ja ho hem fet), definir els volums de dades i càrregues típiques per a cada escenari.
3. **Configurar l'entorn de prova.**
4. **Definir els casos de prova de rendiment.** Desenvolupar casos de prova que representin els escenaris i establir-ne els paràmetres de prova.
5. **Implementar les provees de rendiment, executar les provees de rendiment.**
6. **Analitzar els resultats i optimitzar el rendiment.**

2.6.4

USABILITY TESTING

És el tipus de testeig on més se'ns ha insistit i on més ens estendrem.

Definició 2.12 (Usability testing). El *usability testing* (provees d'usabilitat) és una pràctica de prova en el desenvolupament de programari que es realitza per avaluar la facilitat d'ús, l'eficàcia i l'experiència de l'usuari d'un

sistema o aplicació. L'objectiu principal d'aquest tipus de proves és obtenir informació sobre com els usuaris interactuen amb el sistema i identificar problemes d'usabilitat per millorar-ne el disseny i l'experiència global de l'usuari. Durant les proves d'usabilitat, es recluten participants representatius dels usuaris potencials del sistema. Aquests participants realitzen tasques específiques o escenaris d'ús en el sistema mentre els observadors registren les seves interaccions, els seus comentaris i les seves reaccions. Aquest procés permet identificar problemes comuns, dificultats d'ús, confusions o mancances en el disseny de la interfície.

Les proves d'usabilitat són importants per garantir que un sistema o aplicació sigui fàcil d'utilitzar, satisfaci les necessitats dels usuaris i proporcioni una experiència positiva. Aquestes proves ajuden a detectar problemes d'interfície, disseny confús o complex, manca de claredat en les instruccions i altres factors que podrien afectar l'ús efectiu del sistema.

1. Establir objectius d'usabilitat. Definir els objectius d'usabilitat específics que volem aconseguir amb *CompraSmart*, com ara facilitat d'ús, eficiència, satisfacció de l'usuari, accessibilitat, entre altres.

2. Identificar els escenaris d'ús clau. *Ja ho hem fet abans.*

3. Seleccionar participants dels tests.

- Identificar una *mostra representativa* dels usuaris potencials de l'aplicació. És molt important que sigui molt àmplia.
- Assegurar una diversitat de participants que reflecteixi les característiques dels usuaris reals, com ara edat, nivell d'experiència tecnològica i objectius d'ús.

4. Preparar les tasques de prova.

- Definir les tasques específiques que els participants han de realitzar durant els tests d'usabilitat.
- Establir una seqüència lògica de tasques que cobreixi els escenaris d'ús clau identificats prèvia-ment.

5. Establir criteris d'avaluació. Establir criteris objectius per avaluar l'usabilitat de l'aplicació, com ara temps d'execució de les tasques, errors comesos, eficàcia i satisfacció de l'usuari.

6. Realitzar les sessions de test.

- Conduir sessions individuals amb cada participant, presentant les tasques i observant les seves interaccions amb l'aplicació.
- Encoratjar els participants a pensar en veu alta, expressar les seves expectatives i compartir els seus reptes i opinions durant la prova.

7. Recopilar dades i observacions. Recopilar observacions qualitatives sobre les dificultats trobades pels participants, els comentaris sobre la interfície, els suggeriments d' millora i altres aspectes destacats.

8. Analitzar els resultats.

- Revisar les dades recopilades i les observacions per identificar patrons i tendències.
- Identificar problemes d'usabilitat recurrents i priorititzar-los segons la seva importància i impacte en l'experiència de l'usuari.

9. Proposar millores i realitzar iteracions.

- Basant-te en els resultats dels tests d'usabilitat, proposar millores específiques per abordar els problemes identificats.
- Realitzar iteracions de disseny, implementació i tests per validar les millores i assegurar que s'aborden adequadament els problemes d'usabilitat.

També hem pensat que, als usuaris seleccionats amb els criteris que hem exposat, se'ls podria fer una enquesta. Les preguntes que hem pensat són les següents.

Pregunta 1: Quina opció et sembla més clara per afegir un element a la llista de la compra?

1. Un botó "Afegir" al costat de cada element de la llista.
2. Un botó "Afegir" a la part superior o inferior de la llista.
3. Un gest com arrossegar i deixar anar l'element a la llista.
4. Altres (especifica):

Pregunta 2: Quin mètode de cerca et sembla més eficient per trobar un element específic a la llista de la compra?

1. Cerca per paraules clau o termes relacionats.
2. Cerca per categories o etiquetes.
3. Cerca per filtres com ara preu, data o ubicació.
4. Altres (especifica):

Pregunta 3: Com preferies organitzar la llista de la compra?

1. Per categories (ex: làctics, fruites, productes de neteja).
2. En una única llista sense categories.
3. Personalitzar les categories segons les meves necessitats.
4. Altres (especifica):

Pregunta 4: Quina opció de sincronització amb altres dispositius o usuaris et sembla més útil?

1. Sincronització automàtica en temps real amb altres dispositius.
2. Possibilitat de compartir la llista amb altres usuaris.
3. Exportar/importar la llista en diferents formats (ex: CSV, PDF).
4. Altres (especifica):

Pregunta 5: Quina opció et sembla més clara per marcar un element com a comprat a la llista?

1. Una casella de selecció al costat de cada element.
2. Un botó "Comprat" al costat de cada element.
3. Canviar l'estil o color de l'element una vegada marcat com a comprat.
4. Altres (especifica):

Pregunta 6: Quina opció et sembla més útil per rebre notificacions o recordatoris relacionats amb la llista de la compra?

1. Notificacions push a l'aplicació.
2. Notificacions per correu electrònic.
3. Notificacions de missatges de text (SMS).
4. Altres (especifica):

Pregunta 7: Si hagues sis de puntuar de l'1 al 10, quin és el teu grau d'enteniment del nostre software?

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10

Pregunta 8: Si hagues sis de puntuar de l'1 al 10, satisfacció amb l'interfície visual.

1. 1
2. 2
3. 3
4. 4
5. 5
6. 6
7. 7
8. 8
9. 9
10. 10

Pregunta 9: Nivell d'intenció d'ús del nostre sistema.

1. Baix
2. Normal
3. Alt

2.6.5

NON-FUNCTIONAL TESTING

Definició 2.13 (Non-functional testing). Les proves no funcionals (*non-functional testing*) són una categoria de proves en el desenvolupament de programari que es centren en els aspectes no relacionats amb la funcionalitat directa d'un sistema o aplicació. Aquestes proves es realitzen per avaluar característiques com la compatibilitat, la portabilitat, la interoperabilitat, la seguretat, la fiabilitat, l'escalabilitat i altres factors importants que afecten l'experiència general d'ús i l'èxit d'un sistema.

- 1. Prova de compatibilitat.**
- 2. Prova de portabilitat.**
- 3. Prova de interoperabilitat.**
- 4. Prova de seguretat.**
- 5. Prova de fiabilitat.**
- 6. Prova d'escalabilitat.**

III

Conclusions

El desenvolupament de la nostra aplicació avança correctament i a bon ritme, ja que tenim la majoria dels casos d'ús, que volíem incorporar, implementats. Tot i això, encara queda el més important i també queda millorar parts del model, perquè l'estructura del codi encara no queda molt clara.

La funcionalitat que queda per implementar és la de calcular la millor ruta per fer la teva compra. Aquesta funcionalitat implicarà afegir un botó al *layout* de l'*activity VeureLlistaCompraActivity*. Aquest botó encara hem de decidir com interactuarà amb l'usuari, ja que tenim diferents opcions. La idea principal és implementar la funcionalitat mitjançant un, o varis, *fragment*, pel fet que aquesta noció se'ns va introduir fa poc temps i no l'hem pogut utilitzar. Aquest cas d'ús és el que considerem més complicat d'implementar, perquè consistirà en diferents pestanyes i el càlcul a fer no serà senzill, ja que dependrà de moltes variables.

Pel que fa a l'estructura del codi, hem seguit les indicacions donades a les classes de laboratori, però creiem que encara es pot millorar i no descartem un cop tinguem l'aplicació acabada, refactoritzar el codi allà on segui necessari.

També, cal comentar que el codi entregat pensem que representa més d'un 50% del codi total, però hem preferit organitzar-nos bé i tenir prou temps per a implementar la funcionalitat restant i corregir els errors que puguem trobar en aquesta entrega.

Si tenim en compte la part més estètica de la nostra aplicació, considerem que encara es podria millorar una mica més, sobretot el *layout* relacionat amb veure els detalls de cada producte. De la mateixa manera, de cara a la següent entrega voldríem personalitzar encara més les *cards* de les llistes de la compra per tal d'aconseguir que siguin més petites i amb les cantonades arrodonides.

Finalment, a escala personal, aquesta entrega ens ha servit per veure com s'ha d'organitzar un equip de treball, encara que ens hagim trobat davant d'un projecte senzill i petit. També hem pogut treballar el fet d'haver d'escriure codi conjuntament, tot i que cadascú fes la seva part, ja que després seria utilitzat per la resta de l'equip. Amb això, l'atmosfera de treball ha estat molt bona i no hi ha hagut cap problema, cosa que ha fet que treballar fos més fàcil i gratificant.

Miscel·lània

*Per sol·licitud del professor, s'han traslladat **tots els codis** rellevants a aquest apartat per facilitar la llegibilitat del document.*

A.1

CODIS DELS LAYOUT

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:background="@color/fondo"
9      tools:context=".view.MainActivity">
10
11
12    <FrameLayout
13        android:layout_width="match_parent"
14        android:layout_height="match_parent"
15        android:layout_marginStart="32dp"
16        android:layout_marginTop="48dp"
17        android:layout_marginEnd="32dp"
18        android:layout_marginBottom="32dp"
19        app:layout_constraintBottom_toBottomOf="parent"
20        app:layout_constraintEnd_toEndOf="parent"
21        app:layout_constraintStart_toStartOf="parent"
22        app:layout_constraintTop_toTopOf="parent">
23
24
25    <LinearLayout
26        android:id="@+id/image_user"
27        android:layout_width="match_parent"
28        android:layout_height="match_parent"
29        android:orientation="vertical"
30        tools:ignore="UselessParent">
31
32    <TextView
33        android:id="@+id/titulo"
34        android:layout_width="match_parent"
```

```
33     android:layout_height="wrap_content"
34     android:layout_weight="0"
35     android:fontFamily="cursive"
36     android:text="@string/tituloApp"
37     android:textAlignment="center"
38     android:textColor="@color/black"
39     android:textSize="40sp"
40     android:textStyle="bold" />
41
42 <ImageView
43     android:id="@+id/imageView"
44     android:layout_width="wrap_content"
45     android:layout_height="400dp"
46     android:contentDescription="@string/icono_lista_compra"
47     app:srcCompat="@drawable/app_logo1"
48     android:layout_marginBottom="30dp"
49     tools:ignore="ImageContrastCheck" />
50
51
52 <TableLayout
53     android:layout_width="match_parent"
54     android:layout_height="wrap_content"
55     android:layout_weight="3"
56     android:stretchColumns="*">
57
58 <TableRow
59     android:layout_width="wrap_content"
60     android:layout_height="match_parent"
61     tools:ignore="UselessParent">
62
63     <FrameLayout
64         android:layout_width="match_parent"
65         android:layout_height="match_parent">
66
67         <Button
68             android:id="@+id/btn_login"
69             android:layout_width="wrap_content"
70             android:layout_height="wrap_content"
71             android:layout_gravity="center"
72             android:backgroundTint="@color/marron"
73             android:text="@string/login" />
74     </FrameLayout>
75
76     <FrameLayout
77         android:layout_width="match_parent"
78         android:layout_height="match_parent">
```

```

79
80         <Button
81             android:id="@+id/btn_registrarse"
82             android:layout_width="wrap_content"
83             android:layout_height="wrap_content"
84             android:layout_gravity="center"
85             android:backgroundTint="@color/marron"
86             android:text="@string/signup" />
87         </FrameLayout>
88     </TableRow>
89
90     </TableLayout>
91 </LinearLayout>
92 </FrameLayout>
93
94 </androidx.constraintlayout.widget.ConstraintLayout>
```

Listing A.I: Codi del Layout corresponent al *Launcher* de l'aplicació.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:background="@color/fondo"
9      tools:context=".view.RegistreActivity">
10
11     <FrameLayout
12         android:layout_width="match_parent"
13         android:layout_height="match_parent"
14         android:layout_marginStart="32dp"
15         android:layout_marginTop="48dp"
16         android:layout_marginEnd="32dp"
17         android:layout_marginBottom="32dp">
18
19         <LinearLayout
20             android:layout_width="match_parent"
21             android:layout_height="match_parent"
22             android:orientation="vertical"
23             tools:ignore="UselessParent">
24
25             <TableLayout
26                 android:layout_width="match_parent"
```

```
26     android:layout_height="0dp"
27     android:layout_weight="3"
28     android:gravity="center"
29     android:stretchColumns="1">
30
31     <TableRow
32         android:layout_width="match_parent"
33         android:layout_height="match_parent"
34         android:layout_marginBottom="32dp">
35
36
37         <ImageView
38             android:id="@+id/imageView5"
39             android:layout_width="44dp"
40             android:layout_height="match_parent"
41             android:layout_marginEnd="16dp"
42             app:srcCompat="@drawable/baseline_person_24"
43             android:contentDescription="@string/icono_persona" />
44
45         <EditText
46             android:id="@+id/edit_txt_name"
47             android:layout_width="wrap_content"
48             android:layout_height="48dp"
49             android:autofillHints=""
50             android:ems="10"
51             android:hint="@string/nom"
52             android:inputType="textEmailAddress"
53             tools:ignore="VisualLintTextFieldSize" />
54     </TableRow>
55
56     <TableRow
57         android:layout_width="match_parent"
58         android:layout_height="match_parent"
59         android:layout_marginBottom="32dp">
60
61
62         <ImageView
63             android:id="@+id/imageView6"
64             android:layout_width="44dp"
65             android:layout_height="match_parent"
66             android:layout_marginEnd="16dp"
67             android:contentDescription="@string/icono_persona"
68             app:srcCompat="@drawable/baseline_person_24" />
69
70         <EditText
71             android:id="@+id/edit_txt_cognom"
```

```
72         android:layout_width="wrap_content"
73         android:layout_height="48dp"
74         android:autofillHints=""
75         android:ems="10"
76         android:hint="@string/cognom"
77         android:inputType="textEmailAddress"
78         tools:ignore="VisualLintTextFieldSize" />
79     </TableRow>
80
81     <TableRow
82         android:layout_width="match_parent"
83         android:layout_height="match_parent"
84         android:layout_marginBottom="32dp">
85
86         <ImageView
87             android:id="@+id/image_mail"
88             android:layout_width="51dp"
89             android:layout_height="match_parent"
90             android:layout_marginEnd="16dp"
91             android:contentDescription="@string/correu_electronic"
92             app:srcCompat="@drawable/baseline_mail_24" />
93
94         <EditText
95             android:id="@+id/edit_txt_mail_s"
96             android:layout_width="wrap_content"
97             android:layout_height="48dp"
98             android:autofillHints=""
99             android:ems="10"
100            android:hint="@string/correu_electronic"
101            android:inputType="textEmailAddress"
102            tools:ignore="VisualLintTextFieldSize,DuplicateSpeakableTextCheck"
103            />
104     </TableRow>
105
106     <TableRow
107         android:layout_width="match_parent"
108         android:layout_height="match_parent">
109
110         <ImageView
111             android:id="@+id/image_contrasena"
112             android:layout_width="34dp"
113             android:layout_height="match_parent"
114             android:layout_marginEnd="16dp"
115             android:contentDescription="@string/contrasenya"
116             app:srcCompat="@drawable/baseline_lock_24" />
```

```

117     <EditText
118         android:id="@+id/edit_txt_pwd_s"
119         android:layout_width="wrap_content"
120         android:layout_height="48dp"
121         android:autofillHints=""
122         android:ems="10"
123         android:hint="@string/contrasenya"
124         android:inputType="textPassword"
125         tools:ignore="VisualLintTextFieldSize,DuplicateSpeakableTextCheck"
126         ↵      />
127
128     </TableRow>
129
130
131     <FrameLayout
132         android:layout_width="match_parent"
133         android:layout_height="0dp"
134         android:layout_weight="1">
135
136         <Button
137             android:id="@+id/btn_fer_signup"
138             android:layout_width="wrap_content"
139             android:layout_height="wrap_content"
140             android:layout_gravity="center"
141             android:backgroundTint="@color/marron"
142             android:text="@string/signup"
143             tools:ignore="DuplicateSpeakableTextCheck" />
144     </FrameLayout>
145
146     </LinearLayout>
147
148 </FrameLayout>
149
</androidx.constraintlayout.widget.ConstraintLayout>
```

Listing A.2: Codi del Layout corresponent al registre de l'usuari.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:background="@color/fondo"
```

```
8     android:id="@+id/constraint_layout_inici_sessio"
9     tools:context=".view.IniciSessioActivity">
10
11    <FrameLayout
12        android:layout_width="match_parent"
13        android:layout_height="match_parent"
14        android:layout_marginStart="32dp"
15        android:layout_marginTop="48dp"
16        android:layout_marginEnd="32dp"
17        android:layout_marginBottom="32dp">
18
19        <LinearLayout
20            android:layout_width="match_parent"
21            android:layout_height="match_parent"
22            android:orientation="vertical"
23            tools:ignore="UselessParent">
24
25            <ImageView
26                android:id="@+id/imageView"
27                android:layout_width="match_parent"
28                android:layout_height="20dp"
29                android:layout_marginBottom="48sp"
30                android:layout_weight="3"
31                android:contentDescription="@string/icono_carro_compra"
32                app:srcCompat="@drawable/baseline_shopping_cart_24" />
33
34            <TableLayout
35                android:layout_width="match_parent"
36                android:layout_height="0dp"
37                android:layout_weight="2"
38                android:stretchColumns="1">
39
40                <TableRow
41                    android:layout_width="match_parent"
42                    android:layout_height="match_parent"
43                    android:layout_marginBottom="32dp">
44
45                    <ImageView
46                        android:id="@+id/imageView2"
47                        android:layout_width="34dp"
48                        android:layout_height="match_parent"
49                        android:layout_marginEnd="16dp"
50                        android:contentDescription="@string/correu_electronic"
51                        app:srcCompat="@drawable/baseline_mail_24" />
52
53                    <EditText
```

```
54         android:id="@+id/edit_txt_mail"
55         android:layout_width="wrap_content"
56         android:layout_height="48dp"
57         android:autofillHints=""
58         android:ems="10"
59         android:hint="@string/correu_electronic"
60         android:inputType="textEmailAddress"
61         tools:ignore="VisualLintTextFieldSize,DuplicateSpeakableTextCheck"
62         ↵      />
63
64     <TableRow
65         android:layout_width="match_parent"
66         android:layout_height="match_parent">
67
68         <ImageView
69             android:id="@+id/imageView3"
70             android:layout_width="42dp"
71             android:layout_height="match_parent"
72             android:layout_marginEnd="16dp"
73             android:contentDescription="@string/contrasenya"
74             app:srcCompat="@drawable/baseline_lock_24" />
75
76         <EditText
77             android:id="@+id/edit_txt_pwd"
78             android:layout_width="wrap_content"
79             android:layout_height="48dp"
80             android:autofillHints=""
81             android:ems="10"
82             android:hint="@string/contrasenya"
83             android:inputType="textPassword"
84             tools:ignore="VisualLintTextFieldSize,DuplicateSpeakableTextCheck"
85             ↵      />
86
87     </TableRow>
88
89     <TableLayout>
90         android:layout_width="match_parent"
91         android:layout_height="0dp"
92         android:layout_weight="1">
93
94         <CheckBox
95             android:id="@+id/check_remember_me"
96             android:layout_width="wrap_content"
97             android:layout_height="wrap_content"
98             android:layout_gravity="center_vertical"
```

```

98         android:backgroundTint="@color/marron"
99         android:text="@string/recordam"
100        tools:ignore="DuplicateSpeakableTextCheck" />
101    </FrameLayout>
102    <FrameLayout
103        android:layout_width="match_parent"
104        android:layout_height="0dp"
105        android:layout_weight="1">
106
107        <Button
108            android:id="@+id/btn_fer_loggin"
109            android:layout_width="wrap_content"
110            android:layout_height="wrap_content"
111            android:layout_gravity="center"
112            android:backgroundTint="@color/marron"
113            android:text="@string/login"
114            tools:ignore="DuplicateSpeakableTextCheck" />
115        </FrameLayout>
116    </LinearLayout>
117
118    </FrameLayout>
119
120    <FrameLayout
121        android:layout_width="match_parent"
122        android:layout_height="match_parent"
123        android:background="@color/fondo"
124        android:visibility="gone"
125        android:id="@+id/frame_layout_inici_sessio">
126        <ProgressBar
127            android:id="@+id/progress_bar"
128            android:layout_width="wrap_content"
129            android:layout_height="wrap_content"
130            android:layout_gravity="center"
131            android:visibility="visible" />
132    </FrameLayout>
133
134</androidx.constraintlayout.widget.ConstraintLayout>
```

Listing A.3: Codi del Layout corresponent a l'inici de sessió (*ActivityIniciSessio*).

```

1  <com.google.android.material.card.MaterialCardView
2      xmlns:tools="http://schemas.android.com/tools"
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      android:id="@+id/card"
```

```
5     android:layout_height="wrap_content"
6     android:layout_margin="8dp"
7     android:layout_width="match_parent"
8     android:radius="1000dp">
9
10    <androidx.constraintlayout.widget.ConstraintLayout
11        android:layout_width="match_parent"
12        android:layout_height="match_parent"
13        android:background="@color/marrongris"
14    >
15
16    <TextView
17        android:id="@+id/fullname"
18        android:layout_width="wrap_content"
19        android:layout_height="wrap_content"
20        android:layout_marginStart="15dp"
21        android:layout_marginTop="10dp"
22        android:text="fullname"
23        android:textColor="#000000"
24        android:textStyle="bold"
25        app:layout_constraintEnd_toStartOf="@+id/hide_btn"
26        app:layout_constraintStart_toEndOf="@+id/avatar"
27        app:layout_constraintTop_toBottomOf="@+id/hide_btn" />
28
29    <ImageView
30        android:id="@+id/avatar"
31        android:layout_width="121dp"
32        android:layout_height="84dp"
33        android:layout_marginStart="10dp"
34        android:layout_marginTop="10dp"
35        android:layout_marginBottom="10dp"
36        app:layout_constraintBottom_toBottomOf="parent"
37        app:layout_constraintStart_toStartOf="parent"
38        app:layout_constraintTop_toTopOf="parent"
39        app:layout_constraintVertical_bias="1.0"
40        app:srcCompat="@drawable/baseline_shopping_cart_24" />
41
42    <ImageButton
43        android:id="@+id/hide_btn"
44        android:layout_width="24dp"
45        android:layout_height="25dp"
46        android:layout_marginTop="5dp"
47        android:layout_marginEnd="5dp"
48        android:background="@color/marrongris"
49        app:layout_constraintEnd_toEndOf="parent"
50        app:layout_constraintTop_toTopOf="parent"
```

```
51     app:srcCompat="@drawable/baseline_close_24" />
52   </androidx.constraintlayout.widget.ConstraintLayout>
53
54 </com.google.android.material.card.MaterialCardView>
```

Listing A.4: Codi del Layout corresponent a les cards de la llista de la compra (*LlistaCompraCard*).

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3    xmlns:android="http://schemas.android.com/apk/res/android"
4    xmlns:app="http://schemas.android.com/apk/res-auto"
5    xmlns:tools="http://schemas.android.com/tools"
6    android:layout_width="match_parent"
7    android:layout_height="match_parent"
8    android:background="@color/fondo"
9    tools:context=".view.AfegirProducteActivity">
10
11  <FrameLayout
12    android:layout_width="match_parent"
13    android:layout_height="match_parent"
14    android:layout_marginStart="24dp"
15    android:layout_marginTop="32dp"
16    android:layout_marginEnd="24dp"
17    android:layout_marginBottom="24dp"
18    app:layout_constraintBottom_toBottomOf="parent"
19    app:layout_constraintEnd_toEndOf="parent"
20    app:layout_constraintStart_toStartOf="parent"
21    app:layout_constraintTop_toTopOf="parent">
22
23  <LinearLayout
24    android:layout_width="match_parent"
25    android:layout_height="match_parent"
26    android:orientation="vertical">
27
28    <TextView
29      android:id="@+id/titulo_afegir_producte"
30      android:layout_width="wrap_content"
31      android:layout_height="wrap_content"
32      android:layout_marginBottom="18dp"
33      android:text="Selecciona els productes que vols afegir:"
34      android:textSize="18sp"
35      android:textStyle="bold" />
36
37  <ScrollView
38    android:layout_width="match_parent"
```

```
38         android:layout_height="match_parent">
39
40     <RelativeLayout
41         android:layout_width="wrap_content"
42         android:layout_height="wrap_content">
43
44
45         <androidx.recyclerview.widget.RecyclerView
46             android:id="@+id/RVproducts"
47             android:layout_width="match_parent"
48             android:layout_height="match_parent" />
49     </RelativeLayout>
50 </ScrollView>
51
52 </LinearLayout>
53
54 <com.google.android.material.floatingactionbutton.FloatingActionButton
55     android:id="@+id/floatingBtnAfegirProducte"
56     android:layout_width="match_parent"
57     android:layout_height="wrap_content"
58     android:layout_gravity="bottom|end"
59     android:clickable="true"
60     android:contentDescription="accept afegir producte"
61     app:backgroundTint="@color/marron"
62     app:tint="@color/white"
63     app:srcCompat="@drawable/baseline_check_24" />
64
65 </FrameLayout>
66
67 <FrameLayout
68     android:layout_width="match_parent"
69     android:layout_height="match_parent"
70     android:background="@color/fondo"
71     android:visibility="visible"
72     android:id="@+id/frame_layout_afegir_producte">
73     <ProgressBar
74         android:id="@+id/progress_bar_afegir_producte"
75         android:layout_width="wrap_content"
76         android:layout_height="wrap_content"
77         android:layout_gravity="center"
78         android:visibility="visible" />
79 </FrameLayout>
80
81 </androidx.constraintlayout.widget.ConstraintLayout>
```

Listing A.5: Layout de l'activitat d'afegir un producte (*ActivityAfegirProducte*).

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3   xmlns:android="http://schemas.android.com/apk/res/android"
4   xmlns:app="http://schemas.android.com/apk/res-auto"
5   xmlns:tools="http://schemas.android.com/tools"
6   android:layout_width="match_parent"
7   android:layout_height="match_parent"
8   android:background="@color/fondo"
9   tools:context=".view.VeureDetailsProducteActivity">
10
11 <FrameLayout
12   android:layout_width="match_parent"
13   android:layout_height="match_parent"
14   app:layout_constraintBottom_toBottomOf="parent"
15   app:layout_constraintEnd_toEndOf="parent"
16   app:layout_constraintStart_toStartOf="parent"
17   app:layout_constraintTop_toTopOf="parent">
18
19 <LinearLayout
20   android:layout_width="match_parent"
21   android:layout_height="match_parent"
22   android:layout_margin="32dp"
23   android:orientation="vertical">
24
25 <ImageView
26   android:id="@+id/imageView"
27   android:layout_width="wrap_content"
28   android:layout_height="wrap_content"
29   android:layout_marginTop="20dp"
30   android:layout_marginBottom="10dp"
31   android:minWidth="350dp"
32   android:minHeight="230dp"
33   android:scaleType="fitXY"
34   android:background="@drawable/item_information_frame"
35   android:layout_gravity="center" />
36
37 <TextView
38   android:id="@+id/preview"
39   android:layout_width="match_parent"
40   android:layout_height="wrap_content"
41   android:layout_marginTop="10dp"
42   android:layout_marginBottom="10dp"
43   android:text="Informació de producte"
44   android:textAlignment="center"
45   android:textSize="29sp"
46   android:textStyle="bold|italic" />
```

```
46
47     <TableLayout
48         android:layout_width="match_parent"
49         android:layout_height="wrap_content" >
50
51     <TableRow>
52         <TextView
53             android:layout_width="wrap_content"
54             android:layout_height="wrap_content"
55             android:layout_gravity="center_vertical"
56             android:text="Supermercats"
57             android:textSize="16sp"
58             android:textStyle="italic"/>
59
60         <Spinner
61             android:id="@+id/spinner_producte"
62             android:layout_gravity="center_vertical"
63             android:layout_width="match_parent"
64             android:layout_height="wrap_content"
65             android:layout_marginStart="20dp"
66             android:layout_marginTop="8dp"
67             android:layout_marginEnd="8dp"
68             android:layout_marginBottom="8dp"
69             android:minHeight="48dp"
70             app:layout_constraintEnd_toEndOf="parent"
71             app:layout_constraintStart_toEndOf="@+id/textViewSupermercats"
72             app:layout_constraintTop_toTopOf="parent" />
73     </TableRow>
74
75     <TableRow>
76         <TextView
77             android:layout_width="wrap_content"
78             android:layout_height="wrap_content"
79             android:layout_gravity="center_vertical"
80             android:text="Preu del producte"
81             android:textSize="16sp"
82             android:textStyle="italic"/>
83
84         <TextView
85             android:id="@+id/ItemViewText"
86             android:layout_width="wrap_content"
87             android:layout_height="wrap_content"
88             android:layout_marginStart="30dp"
89             android:layout_marginTop="8dp"
90             android:layout_marginEnd="8dp"
91             android:layout_marginBottom="8dp"
```

```

92         android:textAlignment="inherit"
93         app:layout_constraintEnd_toEndOf="parent"
94         app:layout_constraintStart_toEndOf="@+id/textViewPreuProducte"
95         app:layout_constraintTop_toTopOf="parent" />
96     </TableRow>
97     </TableLayout>
98
99     </LinearLayout>
100    </FrameLayout>
101
102</androidx.constraintlayout.widget.ConstraintLayout>
```

Listing A.6: Codi del Layout corresponent a veure els detalls d'un producte.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
6      android:layout_width="match_parent"
7      android:layout_height="match_parent"
8      android:background="@color/fondo"
9      tools:context=".view.VeureLlistaCompraActivity">
10
11  <FrameLayout
12      android:layout_width="match_parent"
13      android:layout_height="match_parent"
14      android:layout_marginStart="16dp"
15      android:layout_marginTop="32dp"
16      android:layout_marginEnd="16dp"
17      android:layout_marginBottom="16dp"
18      app:layout_constraintBottom_toBottomOf="parent"
19      app:layout_constraintEnd_toEndOf="parent"
20      app:layout_constraintStart_toStartOf="parent"
21      app:layout_constraintTop_toTopOf="parent">
22
23      <LinearLayout
24          android:layout_width="match_parent"
25          android:layout_height="match_parent"
26          android:orientation="vertical">
27
28          <ScrollView
29              android:id="@+id/scroll_veure_llista"
30              android:layout_width="match_parent"
31              android:layout_height="match_parent">
```

```
31
32         <androidx.recyclerview.widget.RecyclerView
33             android:id="@+id/listProductsRV"
34             android:layout_width="match_parent"
35             android:layout_height="wrap_content" />
36     </ScrollView>
37 </LinearLayout>
38
39     <com.google.android.material.floatingactionbutton.FloatingActionButton
40         android:id="@+id/floatingActionButton"
41         android:layout_width="wrap_content"
42         android:layout_height="wrap_content"
43         android:layout_gravity="bottom|right"
44         android:clickable="true"
45         android:contentDescription="afegir producte"
46         android:visibility="visible"
47         app:backgroundTint="@color/marron"
48         app:srcCompat="@drawable/baseline_add_24"
49         app:tint="@color/white" />
50
51     <com.google.android.material.floatingactionbutton.FloatingActionButton
52         android:id="@+id	btn_calculadora"
53         android:layout_width="wrap_content"
54         android:layout_height="wrap_content"
55         android:layout_gravity="bottom|left"
56         android:clickable="true"
57         android:contentDescription="calculadora"
58         android:visibility="visible"
59         app:backgroundTint="@color/marron"
60         app:srcCompat="@drawable/calculator"
61         app:tint="@color/white" />
62
63     <com.google.android.material.floatingactionbutton.FloatingActionButton
64         android:id="@+id/floatingActionButton2"
65         android:layout_width="wrap_content"
66         android:layout_height="wrap_content"
67         android:layout_gravity="bottom|right"
68         android:clickable="true"
69         android:contentDescription="acceptar eliminar producte"
70         android:visibility="invisible"
71         app:backgroundTint="@color/marron"
72         app:srcCompat="@drawable/baseline_check_24"
73         app:tint="@color/white" />
74
75 </FrameLayout>
```

```
76 </androidx.constraintlayout.widget.ConstraintLayout>
```

Listing A.7: Layout de l'activitat de visualitzar la llista de la compra.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3   xmlns:android="http://schemas.android.com/apk/res/android"
4   xmlns:app="http://schemas.android.com/apk/res-auto"
5   xmlns:tools="http://schemas.android.com/tools"
6   android:layout_width="match_parent"
7   android:layout_height="match_parent"
8   android:background="@color/fondo"
9   tools:context=".view.CalculadoraActivity">
10
11 <FrameLayout
12   android:layout_width="match_parent"
13   android:layout_height="match_parent"
14   app:layout_constraintBottom_toBottomOf="parent"
15   app:layout_constraintEnd_toEndOf="parent"
16   app:layout_constraintStart_toStartOf="parent"
17   app:layout_constraintTop_toTopOf="parent">
18
19   <LinearLayout
20     android:layout_width="match_parent"
21     android:layout_height="match_parent"
22     android:layout_margin="32dp"
23     android:orientation="vertical">
24
25     <ImageView
26       android:id="@+id/imageViewCalculListaSupers"
27       android:layout_width="wrap_content"
28       android:layout_height="0dp"
29       android:layout_weight="2.5"
30       android:layout_marginTop="0dp"
31       android:layout_marginBottom="0dp"
32       android:minWidth="350dp"
33       android:minHeight="230dp"
34       android:scaleType="fitCenter"
35       android:src="@drawable/baseline_shopping_cart_24"
36       android:layout_gravity="center" />
37
38     <TextView
39       android:id="@+id/preview"
40       android:layout_width="match_parent"
41       android:layout_height="0dp"
```

```

41         android:layout_weight="1"
42         android:layout_marginTop="10dp"
43         android:layout_marginBottom="10dp"
44         android:text="A quins d'aquests supermercats vols anar?"
45         android:textAlignment="center"
46         android:textSize="29sp"
47         android:textStyle="bold|italic" />
48
49     <ScrollView
50         android:layout_width="match_parent"
51         android:layout_height="0dp"
52         android:layout_marginTop="0dp"
53         android:layout_weight="4"
54         android:layout_marginBottom="0dp">
55         <RelativeLayout
56             android:layout_width="wrap_content"
57             android:layout_height="wrap_content">
58
59             <androidx.recyclerview.widget.RecyclerView
60                 android:id="@+id/llistaSupers"
61                 android:layout_width="match_parent"
62                 android:layout_height="match_parent" />
63         </RelativeLayout>
64     </ScrollView>
65     <Button
66         android:id="@+id/button_supers"
67         android:layout_width="match_parent"
68         android:layout_marginTop="10dp"
69         android:layout_weight="1"
70         android:layout_height="0dp"
71         android:text="Endavant" >
72     </Button>
73
74     </LinearLayout>
75 </FrameLayout>
76
77 </androidx.constraintlayout.widget.ConstraintLayout>
```

Listing A.8: Layout del fragment del càlcul de la llista de supermercats.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      xmlns:app="http://schemas.android.com/apk/res-auto"
5      xmlns:tools="http://schemas.android.com/tools"
```

```
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     android:background="@color/fondo">
9
10    <LinearLayout
11        android:layout_width="match_parent"
12        android:layout_height="match_parent"
13        android:layout_margin="32dp"
14        android:orientation="vertical">
15
16        <ImageView
17            android:id="@+id/icona_carro"
18            android:layout_width="wrap_content"
19            android:layout_height="0dp"
20            android:layout_weight="2.5"
21            android:layout_marginTop="0dp"
22            android:layout_marginBottom="5dp"
23            android:layout_gravity="center"
24            android:minWidth="350dp"
25            android:minHeight="230dp"
26            android:scaleType="fitCenter"
27            app:srcCompat="@drawable/baseline_shopping_cart_24" />
28
29        <TextView
30            android:id="@+id/prova"
31            android:layout_width="wrap_content"
32            android:layout_height="0dp"
33            android:layout_weight="1"
34            android:layout_marginTop="5dp"
35            android:layout_marginBottom="5dp"
36            android:text="Aquestes són les llistes de la compra per cada supermercat"
37            android:textAlignment="center"
38            android:textSize="24sp"
39            android:textStyle="bold|italic">
40
41    </TextView>
42
43    <ScrollView
44        android:id="@+id/scrollView_fragment"
45        android:layout_width="match_parent"
46        android:layout_height="0dp"
47        android:layout_weight="5"
48        android:layout_marginTop="5dp"
49        android:background="@color/fondo">
50
51        <RelativeLayout
```

```

52         android:layout_width="match_parent"
53         android:layout_height="match_parent">
54
55     <androidx.recyclerview.widget.RecyclerView
56         android:id="@+id/llistaCardRV_fragment"
57         android:layout_width="match_parent"
58         android:layout_height="wrap_content" />
59
60     </RelativeLayout>
61
62     </ScrollView>
63 </LinearLayout>
64
65 <FrameLayout
66     android:layout_width="match_parent"
67     android:layout_height="match_parent" >
68
69     <ProgressBar
70         android:id="@+id/progressBarFragmentEscollarLlista"
71         style="?android:attr/progressBarStyle"
72         android:layout_width="match_parent"
73         android:layout_height="wrap_content"
74         android:layout_gravity="center_vertical"
75         android:focusableInTouchMode="false"
76         android:focusedByDefault="false"
77         android:forceHasOverlappingRendering="false"
78         tools:visibility="gone" />
79     </FrameLayout>
80
81 </androidx.constraintlayout.widget.ConstraintLayout>

```

Listing A.9: Layout del fragment d'escollar la llista quan s'ha fet el càcul.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent"
8     android:background="@color/fondo"
9     tools:context=".view.VeureLlistaCompraActivity">
10
11     <FrameLayout
12         android:layout_width="match_parent"

```

```

12     android:layout_height="match_parent"
13     android:layout_marginStart="16dp"
14     android:layout_marginTop="32dp"
15     android:layout_marginEnd="16dp"
16     android:layout_marginBottom="16dp"
17     app:layout_constraintBottom_toBottomOf="parent"
18     app:layout_constraintEnd_toEndOf="parent"
19     app:layout_constraintStart_toStartOf="parent"
20     app:layout_constraintTop_toTopOf="parent">
21
22     <LinearLayout
23         android:layout_width="match_parent"
24         android:layout_height="match_parent"
25         android:orientation="vertical">
26
27         <ScrollView
28             android:id="@+id/scroll_veure_llista"
29             android:layout_width="match_parent"
30             android:layout_height="match_parent">
31
32             <androidx.recyclerview.widget.RecyclerView
33                 android:id="@+id/listProductsToBuyRV_fragment"
34                 android:layout_width="match_parent"
35                 android:layout_height="wrap_content" />
36         </ScrollView>
37     </LinearLayout>
38
39     </FrameLayout>
40 </androidx.constraintlayout.widget.ConstraintLayout>

```

Listing A.10: Layout del fragment de veure els productes.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     xmlns:tools="http://schemas.android.com/tools"
6     android:layout_width="match_parent"
7     android:layout_height="match_parent">
8
8     <TextView
9         android:id="@+id/text_dialog_eliminar"
10        android:layout_width="wrap_content"
11        android:layout_height="wrap_content"
12        android:layout_marginTop="32dp"

```

```
13     android:text="Segur que vols eliminar aquesta llista?"  
14     android:textSize="20sp"  
15     app:layout_constraintEnd_toEndOf="parent"  
16     app:layout_constraintStart_toStartOf="parent"  
17     app:layout_constraintTop_toTopOf="parent" />  
18  
19     <!--<Button  
20         android:id="@+id/button_si_eliminar"  
21         android:layout_width="wrap_content"  
22         android:layout_height="wrap_content"  
23         android:layout_marginStart="76dp"  
24         android:text="Sí"  
25         app:layout_constraintStart_toStartOf="parent"  
26         app:layout_constraintTop_toBottomOf="@+id/text_dialog_eliminar" />  
27  
28     <Button  
29         android:id="@+id/button_no_eliminar"  
30         android:layout_width="wrap_content"  
31         android:layout_height="wrap_content"  
32         android:text="No"  
33         app:layout_constraintEnd_toEndOf="parent"  
34         app:layout_constraintStart_toEndOf="@+id/button_si_eliminar"  
35         app:layout_constraintTop_toBottomOf="@+id/text_dialog_eliminar" />-->  
36 </androidx.constraintlayout.widget.ConstraintLayout>
```

Listing A.11: Layout del diàleg d'eliminar una llista.

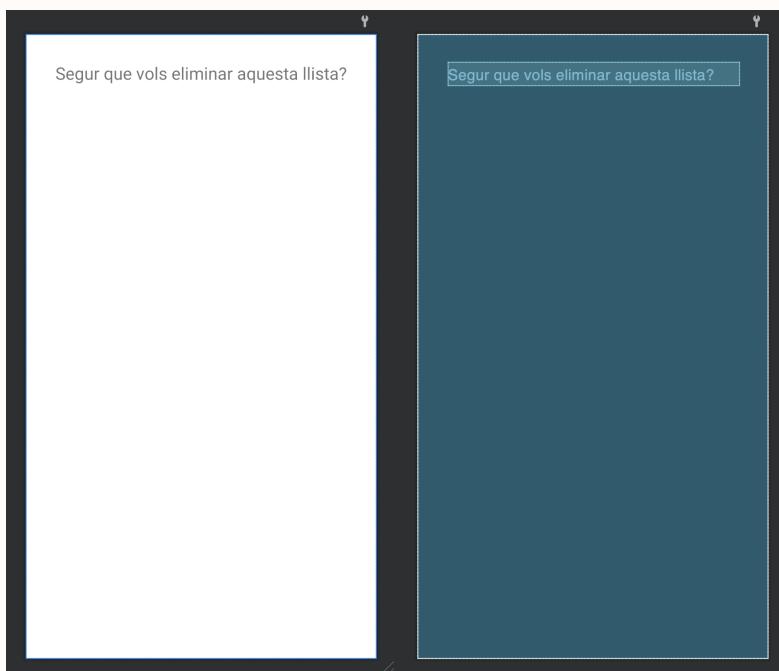


Figura A.1: El Layout del diàleg d'eliminar una llista.

Una vista senzilla per a un diàleg d'alerta que pregunta a l'usuari si vol eliminar una llista determinada. La vista principal del diàleg és un ConstraintLayout, que és un layout que s'utilitza per definir posicions relatives dels diferents elements que hi ha a la pantalla. Dins d'aquest ConstraintLayout, hi ha un TextView amb un missatge que pregunta a l'usuari si està segur que vol eliminar la llista. Aquest TextView està centrat verticalment i horitzontalment dins del ConstraintLayout.

Tenim uns quants fitxers XML molt similars, item.xml, item_list.xml, item_list_check.xml, llista_productes_card.xml i llista_produpte_cada_supermercat_card.xml. Solament n'explicarem un i llistarem la resta.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout
3   xmlns:android="http://schemas.android.com/apk/res/android"
4   xmlns:app="http://schemas.android.com/apk/res-auto"
5   xmlns:tools="http://schemas.android.com/tools"
6   android:layout_width="match_parent"
7   android:layout_height="wrap_content"
8   android:orientation="horizontal">
9
9   <ImageView
10    android:id="@+id/imageView4"
11    android:layout_width="wrap_content"
12    android:layout_height="wrap_content"
13    android:layout_alignParentStart="true"
14    android:layout_alignParentTop="true"
15    app:layout_constraintBottom_toBottomOf="parent"
16    app:layout_constraintStart_toStartOf="parent"
17    app:layout_constraintTop_toTopOf="parent"
18    app:srcCompat="@drawable/baseline_remove_24" />
19

```

```

20 <CheckBox
21     android:id="@+id/checkBoxEliminar"
22     android:layout_width="48dp"
23     android:layout_height="48dp"
24     app:layout_constraintBottom_toBottomOf="parent"
25     app:layout_constraintEnd_toStartOf="@+id/textViewcheckBox"
26     app:layout_constraintStart_toStartOf="parent"
27     app:layout_constraintTop_toTopOf="parent" />
28
29 <TextView
30     android:id="@+id/textViewcheckBox"
31     android:layout_width="wrap_content"
32     android:layout_height="wrap_content"
33     android:layout_centerVertical="true"
34     android:layout_toEndOf="@+id/imageView4"
35     android:layout_weight="10"
36     android:text="Just a text"
37     android:textSize="18sp"
38     app:layout_constraintBottom_toBottomOf="parent"
39     app:layout_constraintStart_toEndOf="@+id/imageView4"
40     app:layout_constraintTop_toTopOf="parent" />
41
42 </androidx.constraintlayout.widget.ConstraintLayout>

```

Listing A.12: Layout de l'estructura d'element d'una llista.

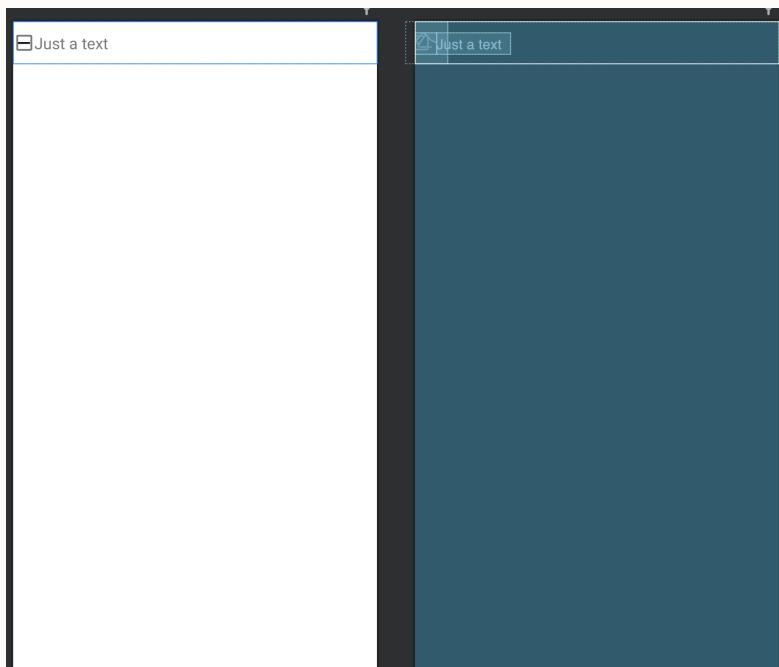


Figura A.2: El Layout de l'estructura d'element d'una llista.

Aquest codi defineix una vista personalitzada que consta d'una casella de selecció i un text al costat. La vista també té un botó d'imatge a l'esquerra per eliminar la vista completa. La vista està continguda dins d'un ConstraintLayout, que està configurat per alinear el botó d'imatge a l'esquerra i centrar verticalment el contingut restant.

```

1 <com.google.android.material.card.MaterialCardView
2   ↳ xmlns:tools="http://schemas.android.com/tools"
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:id="@+id/cardchecked"
6     android:layout_height="wrap_content"
7     android:layout_margin="0dp"
8     android:layout_width="match_parent"
9     android:outlineProvider="none"
10    android:radius="1000dp">
11
12
13  <LinearLayout
14    android:layout_width="match_parent"
15    android:layout_height="match_parent"
16    android:background="@color/fondo"
17    android:orientation="horizontal">
18
19    <CheckBox
20      android:id="@+id/checkBox"
21      android:layout_width="wrap_content"
22      android:layout_height="wrap_content"
23      android:layout_gravity="center_vertical"
24      android:layout_marginEnd="16dp"/>
25
26    <TextView
27      android:id="@+id/textViewcheckBox"
28      android:layout_width="0dp"
29      android:layout_height="wrap_content"
30      android:layout_gravity="center"
31      android:layout_weight="1"
32      android:text="Just a text"
33      android:textSize="18sp" />
34  </LinearLayout>
</com.google.android.material.card.MaterialCardView>

```

Listing A.13: Layout de l'estrucció d'element d'una llista que podem seleccioanar.

```

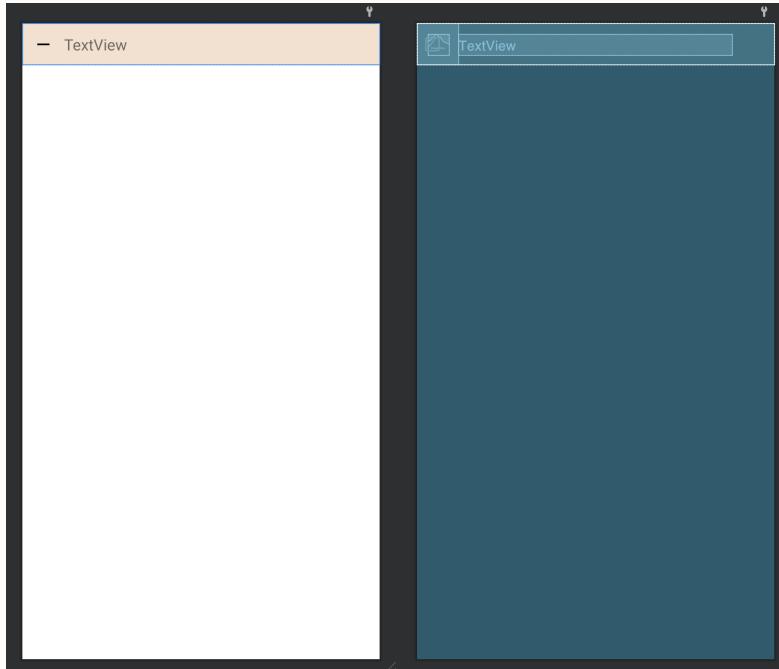
1 <com.google.android.material.card.MaterialCardView
2   ↳ xmlns:tools="http://schemas.android.com/tools"
3     xmlns:android="http://schemas.android.com/apk/res/android"
4     xmlns:app="http://schemas.android.com/apk/res-auto"
5     android:id="@+id/cardProducte"
6     android:layout_height="wrap_content"
       android:layout_width="match_parent"

```

```
7     android:layout_margin="0dp"
8     android:layout_marginRight="10dp"
9     android:outlineProvider="none"
10    android:radius="1000dp">
11
12   <androidx.constraintlayout.widget.ConstraintLayout
13     android:layout_width="match_parent"
14     android:layout_height="match_parent"
15     android:background="@color/fondo">
16
17   <ImageView
18     android:id="@+id/guion"
19     android:layout_width="wrap_content"
20     android:layout_height="wrap_content"
21     android:layout_gravity="center_vertical"
22     android:layout_marginStart="12dp"
23     android:visibility="visible"
24     app:layout_constraintBottom_toBottomOf="parent"
25     app:layout_constraintRight_toLeftOf="@+id/nom_producte"
26     app:layout_constraintStart_toStartOf="parent"
27     app:layout_constraintTop_toTopOf="parent"
28     app:srcCompat="@drawable/baseline_remove_24" />
29
30   <CheckBox
31     android:id="@+id/checkBoxProducteEliminar"
32     android:layout_width="wrap_content"
33     android:layout_height="wrap_content"
34     android:layout_weight="0"
35     android:visibility="invisible"
36     app:layout_constraintBottom_toBottomOf="parent"
37     app:layout_constraintStart_toStartOf="parent"
38     app:layout_constraintTop_toTopOf="parent"
39     app:layout_constraintVertical_bias="0.0" />
40
41   <TextView
42     android:id="@+id/nom_producte"
43     android:layout_width="315dp"
44     android:layout_height="wrap_content"
45     android:text="TextView"
46     android:textSize="18dp"
47     app:layout_constraintBottom_toBottomOf="parent"
48     app:layout_constraintStart_toEndOf="@+id/checkBoxProducteEliminar"
49     app:layout_constraintTop_toTopOf="parent" />
50   </androidx.constraintlayout.widget.ConstraintLayout>
51
```

```
52 </com.google.android.material.card.MaterialCardView>
```

Listing A.14: Layout de la Card de la llista de productes.



Aquest codi defineix una vista de Material CardView per a una llista de productes en una aplicació d'Android. La vista inclou una imatge (*guió*) a l'esquerra, una casella de selecció (checkboxProducteEliminar) a l'esquerra, i un nom de producte (nom_producte) a la dreta. El *guió* és una imatge que indica que l'usuari pot eliminar el producte i apareix a l'esquerra de la vista.

Figura A.3: El Layout de la Card de la llista de productes.

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:app="http://schemas.android.com/apk/res-auto"
4   xmlns:tools="http://schemas.android.com/tools"
5   android:layout_width="match_parent"
6   android:layout_height="wrap_content"
7   android:orientation="horizontal">
8
9   <CheckBox
10    android:id="@+id/checkBoxItem"
11    android:layout_width="wrap_content"
12    android:layout_height="wrap_content" />
13
14   <TextView
15    android:id="@+id/textViewItem"
16    android:layout_width="wrap_content"
17    android:layout_height="wrap_content"
18    android:textSize="16sp"
19    android:hint="List item" />
```

```
20 </LinearLayout>  
21
```

Listing A.15: Layout de l'item d'algunes llistes.

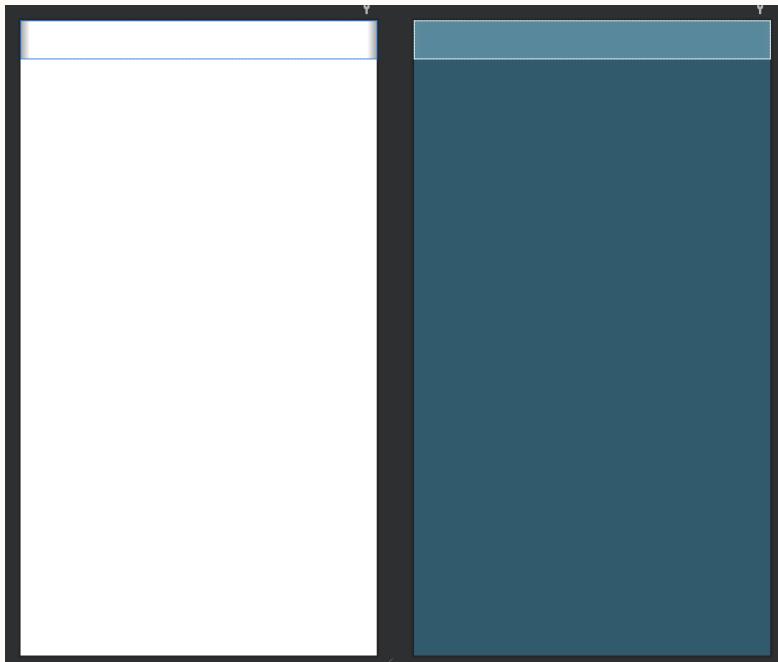
```
1 <com.google.android.material.card.MaterialCardView  
2   ↳ xmlns:android="http://schemas.android.com/apk/res/android"  
3     xmlns:app="http://schemas.android.com/apk/res-auto"  
4     xmlns:tools="http://schemas.android.com/tools"  
5     android:id="@+id/cardProducte"  
6     android:layout_width="match_parent"  
7     android:layout_height="wrap_content"  
8     android:layout_margin="0dp"  
9     android:layout_marginRight="10dp"  
10    android:outlineProvider="none"  
11    android:radius="1000dp"  
12    tools:visibility="visible">  
13  
13  <androidx.constraintlayout.widget.ConstraintLayout  
14    android:layout_width="match_parent"  
15    android:layout_height="match_parent"  
16    android:background="@color/fondo">  
17  
18  <CheckBox  
19    android:id="@+id/checkBox_producte_fragment"  
20    android:layout_width="wrap_content"  
21    android:layout_height="wrap_content"  
22    android:layout_weight="0"  
23    android:visibility="visible"  
24    app:layout_constraintBottom_toBottomOf="parent"  
25    app:layout_constraintStart_toStartOf="parent"  
26    app:layout_constraintTop_toTopOf="parent"  
27    app:layout_constraintVertical_bias="0.0" />  
28  
29  <ImageView  
30    android:id="@+id/guion33"  
31    android:layout_width="wrap_content"  
32    android:layout_height="wrap_content"  
33    android:visibility="invisible"  
34    app:layout_constraintBottom_toBottomOf="parent"  
35    app:layout_constraintEnd_toEndOf="@+id/checkBox_producte_fragment"  
36    app:layout_constraintStart_toStartOf="parent"  
37    app:layout_constraintTop_toTopOf="@+id/checkBox_producte_fragment"  
38    app:srcCompat="@drawable/baseline_remove_24" />  
39
```

```
40 <TextView  
41     android:id="@+id/nom_producte_fragment"  
42     android:layout_width="315dp"  
43     android:layout_height="wrap_content"  
44     android:text="TextView"  
45     android:textSize="18dp"  
46     app:layout_constraintBottom_toBottomOf="parent"  
47     app:layout_constraintStart_toEndOf="@+id/checkBox_producte_fragment"  
48     app:layout_constraintTop_toTopOf="parent" />  
49 </androidx.constraintlayout.widget.ConstraintLayout>  
50  
51 </com.google.android.material.card.MaterialCardView>
```

Listing A.16: Layout de l'item d'algunes llistes.

```
1 <?xml version="1.0" encoding="utf-8"?>  
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
3     android:layout_width="match_parent"  
4     android:layout_height="wrap_content"  
5     android:orientation="horizontal">  
6  
7     <TextView  
8         android:id="@+id/textViewSpinner"  
9         android:layout_width="fill_parent"  
10        android:layout_height="wrap_content"  
11        android:layout_gravity="center"  
12        android:padding="10dp"  
13        android:textColor="#000"  
14        android:textSize="16sp" />  
15 </LinearLayout>
```

Listing A.17: Layout de l'spinner.



No té res a explicar.

Figura A.4: El Layout de l'spinner.

A.2

CODIS DEL MODEL

```
1 package edu.ub.pis3.model;
2
3 import android.util.Log;
4 import android.util.Pair;
5
6 import androidx.annotation.NonNull;
7
8 import com.google.android.gms.tasks.OnCompleteListener;
9 import com.google.android.gms.tasks.Task;
10 import com.google.firebaseio.database.DatabaseReference;
11 import com.google.firebaseio.firebaseio.FirebaseDatabase;
12 import com.google.firebaseio.firestore.DocumentReference;
13 import com.google.firebaseio.firestore.DocumentSnapshot;
14 import com.google.firebaseio.firebaseio.FieldValue;
15 import com.google.firebaseio.firebaseio.FirebaseFirestore;
16 import com.google.firebaseio.firebaseio.QueryDocumentSnapshot;
17 import com.google.firebaseio.firebaseio.QuerySnapshot;
18
19 import java.util.*;
20
21 public class LlistesCompraRepository {
```

```
23     private static final String TAG = "LlistesCompraRepository";
24     private static final LlistesCompraRepository mInstance = new LlistesCompraRepository();
25     private FirebaseFirestore mBd;
26     public ArrayList<OnLoadLlistesCompraListener> mOnLoadLlistesCompraListeners;
27
28     private DatabaseReference mDatabase;
29
30     public interface OnLoadLlistesCompraListener {
31         void onLoadLlistesCompra(ArrayList<LlistesCompra> llistesCompra);
32     }
33     private LlistesCompraRepository(){
34         mBd = FirebaseFirestore.getInstance();
35         mOnLoadLlistesCompraListeners = new ArrayList<>();
36         mDatabase = FirebaseDatabase.getInstance().getReference();
37     }
38
39     public static LlistesCompraRepository getInstance(){return mInstance;}
40
41     public void addOnLoadLlistesCompraListener(OnLoadLlistesCompraListener listener){
42         mOnLoadLlistesCompraListeners.add(listener);
43     }
44
45     public void loadLlistesCompra(ArrayList<LlistesCompra> llistesCompras, String mailUser){
46         llistesCompras.clear();
47         DocumentReference docRef = mBd.collection("users").document(mailUser);
48         mBd.collection("users").document(mailUser).get().addOnCompleteListener(new
49             → OnCompleteListener<DocumentSnapshot>() {
50                 @Override
51                 public void onComplete(@NonNull Task<DocumentSnapshot> task) {
52                     if(task.isSuccessful()){
53
54                         DocumentSnapshot document = task.getResult();
55                         ArrayList<String> llista = (ArrayList<String>) document.get("Llistes
56                         → compra");
57
58                         for(String str : llista){
59                             LlistesCompra llistaCompra = new LlistesCompra(str);
60                             loadProductes(llistaCompra, mailUser);
61                             llistesCompras.add(llistaCompra);
62                         }
63                         for (OnLoadLlistesCompraListener l: mOnLoadLlistesCompraListeners) {
64                             l.onLoadLlistesCompra(llistesCompras);
65                         }
66                     }else {
67                         Log.d(TAG, "Error getting documents: ", task.getException());
68                     }
69                 }
70             });
71     }
72 }
```

```

67         }
68     });
69 }
70
71 private void loadProductes(LlistesCompra llistaCompra, String mailUser){
72
73     mBd.collection("users").document(mailUser).get().addOnCompleteListener(new
74         OnCompleteListener<DocumentSnapshot>(){
75             @Override
76             public void onComplete(@NonNull Task<DocumentSnapshot> task) {
77                 if (task.isSuccessful()) {
78
79                     DocumentSnapshot document = task.getResult();
80                     ArrayList<Long> llistaIds = (ArrayList<Long>)
81                         document.get(llistaCompra.getNom());
82                     for (Long i : llistaIds) {
83                         //llistaCompra.addProducteLlistaProductes(new Product(i, (String)
84                         mBd.collection("products").document(String.valueOf(i)).get().getResult().get
85                         ("name")));
86
87                         mBd.collection("products2").document(String.valueOf(i)).get().addOnComplete
88                         OnCompleteListener<DocumentSnapshot>() {
89                             @Override
90                             public void onComplete(@NonNull Task<DocumentSnapshot> task) {
91                                 if (task.isSuccessful()){
92                                     DocumentSnapshot document = task.getResult();
93                                     llistaCompra.addProducteLlistaProductes(new Product(i,
94                                         (String) document.get("name")));
95                                 }
96                             }
97                         );
98
99                     }
100
101
102                     docRef.update("Llistes_compra", FieldValue.arrayRemove(nomLlista));
103                     docRef.update(updates).addOnCompleteListener(new OnCompleteListener<Void>() {
104                         @Override
105                         public void onComplete(@NonNull Task<Void> task) {

```

```

106
107     }
108 });
109 }
110
111 public void addLlistaCompra(LlistesCompra llista, String mailUser,
112     → ArrayList<LlistesCompra> llistes){
113
114     DocumentReference docRef = mBd.collection("users").document(mailUser);
115     docRef.update("Llistes compra", FieldValue.arrayUnion(llista.getNom()));
116
117     docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
118         @Override
119         public void onComplete(@NonNull Task<DocumentSnapshot> task) {
120             if (task.isSuccessful()) {
121                 DocumentSnapshot document = task.getResult();
122                 Map<String, Object> map = document.getData();
123                 List<Long> llistaNova = new ArrayList<>();
124                 map.put(llista.getNom(), llistaNova);
125                 docRef.set(map);
126             }
127         }
128     });
129
130     public void inicialitzaLlista(String mailUser){
131         DocumentReference docRef = mBd.collection("users").document(mailUser);
132
133         docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
134             @Override
135             public void onComplete(@NonNull Task<DocumentSnapshot> task) {
136                 if (task.isSuccessful()) {
137                     DocumentSnapshot document = task.getResult();
138                     Map<String, Object> map = document.getData();
139                     List<String> llistaNova = new ArrayList<>();
140                     map.put("Llistes compra", llistaNova);
141                     docRef.set(map);
142                 }
143             }
144         });
145     }

```

Listing A.18: Codi de la classe corresponent a la gestió de les llistes de la compra des del model (*LlistesCompraRepository*).

```
1 package edu.ub.pis3.model;
2
3 import java.util.ArrayList;
4
5 public class LlistesCompra {
6
7     private String nom;
8     private ArrayList<Product> llistaProductes;
9
10    public LlistesCompra(String nom){
11        this.nom = nom;
12        this.llistaProductes = new ArrayList<>();
13    }
14
15    public String getNom() {
16        return nom;
17    }
18
19    public ArrayList<Product> getLlistaProductes(){
20        return this.llistaProductes;
21    }
22
23    public void addProducteLlistaProductes(Product p){
24        this.llistaProductes.add(p);
25    }
26}
```

Listing A.19: Configuració de la llista de la compra (*LlistesCompra*).

```
1 package edu.ub.pis3.model;
2
3 public class User {
4
5     private LlistesCompra lCompra;
6     private String nom;
7     private String pwd;
8     private String email;
9
10    private String cognom;
11
12    public User(String nom, String pwd, String email, String cognom , LlistesCompra lCompra)
13        {
14            this.nom = nom;
15            this.pwd = pwd;
16            this.email = email;
```

```
16     this.cognom = cognom;
17     this.lCompra = lCompra;
18 }
19
20     public String getPwd() {
21         return pwd;
22     }
23
24     public String getNom() {
25         return nom;
26     }
27
28     public String getEmail() {
29         return email;
30     }
31
32 //public LlistesCompra getlCompra() {
33 //    return lCompra;
34 //}
35
36     public void setNom(String nom) {
37         this.nom = nom;
38     }
39
40     public void setPwd(String pwd) {
41         this.pwd = pwd;
42     }
43
44     public void setEmail(String email) {
45         this.email = email;
46     }
47
48     public void setlCompra(LlistesCompra lCompra) {
49         this.lCompra = lCompra;
50     }
51
52     public String getCognom() {
53         return cognom;
54     }
55
56     public void setCognom(String cognom) {
57         this.cognom = cognom;
58     }
59 }
```

Listing A.20: Configuració de la classe *Usuari*.

```
1 package edu.ub.pis3.model;
2
3 public class Product {
4     private long id;
5     private String nom;
6     private String photo;
7
8     public Product(long _id){
9         this.id = _id;
10    }
11
12    public Product(long _id, String _nom){
13        this.id = _id;
14        this.nom = _nom;
15    }
16
17    public String getNom(){
18        return this.nom;
19    }
20
21    public long getId() {
22        return id;
23    }
24    public void setNom(String _nom){
25        this.nom = _nom;
26    }
27    public void setId(Long _id){
28        this.id = _id;
29    }
30
31    public String getPhoto(){
32        return this.photo;
33    }
34
35    public void setPhoto(String photo) {
36        this.photo = photo;
37    }
38
39    public boolean startsWith(String text){
40        String str = this.nom.substring(0,text.length());
41        if (str.toLowerCase().equals(text)){
42            return true;
43        }else{
44            return false;
45        }
46    }
}
```

47}

Listing A.21: Configuració de la classe *Producte*.

```

1 package edu.ub.pis3.model;
2
3 public class InfoSuperProducte {
4     private String nom_Super;
5     private String disponible;
6     private String preu;
7
8     public InfoSuperProducte(String _nom_super){
9         this.nom_Super = _nom_super;
10    }
11
12    public String getNom_Super(){
13        return this.nom_Super;
14    }
15
16    public String getDisponible(){
17        return this.disponible;
18    }
19
20    public String getPreu(){
21        return this.preu;
22    }
23
24    public void setDisponible(String _disponible){
25        this.disponible = _disponible;
26    }
27
28    public void setPreu(String preu) {
29        this.preu = preu;
30    }
31
32}

```

Listing A.22: Configuració de la classe *InfoSuperProducte*.

```

1 package edu.ub.pis3.model;
2
3 import android.util.Log;
4 import android.util.Pair;
5

```

```
6 import androidx.annotation.NonNull;
7
8 import com.google.android.gms.tasks.OnCompleteListener;
9 import com.google.android.gms.tasks.Task;
10 import com.google.firebaseio.database.FirebaseDatabase;
11 import com.google.firebaseio.firestore.DocumentReference;
12 import com.google.firebaseio.firestore.DocumentSnapshot;
13 import com.google.firebaseio.firebaseio.FieldValue;
14 import com.google.firebaseio.firebaseio.FirebaseFirestore;
15 import com.google.firebaseio.firebaseio.QueryDocumentSnapshot;
16 import com.google.firebaseio.firebaseio.QuerySnapshot;
17
18 import java.util.ArrayList;
19
20 public class InfoProducteRepository {
21     public interface OnLoadInfoProducteListener {
22         void onLoadInfoProducte(ArrayList<InfoSuperProducte> llistesInfoProducts);
23     }
24
25     public interface OnLoadAllProductsListener {
26         void onLoadAllProducts(ArrayList<Product> llistaProducts);
27     }
28     private static final String TAG = "InfoProducteRepository";
29     private static final InfoProducteRepository mInstance = new InfoProducteRepository();
30     private FirebaseFirestore mBd;
31     public ArrayList<OnLoadInfoProducteListener> mOnLoadInfoProducteListeners;
32
33     public ArrayList<OnLoadAllProductsListener> mOnLoadAllProductsListener;
34     private InfoProducteRepository(){
35         mBd = FirebaseFirestore.getInstance();
36         mOnLoadInfoProducteListeners = new ArrayList<>();
37         mOnLoadAllProductsListener = new ArrayList<>();
38     }
39
40     public static InfoProducteRepository getInstance(){return mInstance;}
41     public void addOnLoadInfoProducteListener(OnLoadInfoProducteListener listener){
42         mOnLoadInfoProducteListeners.add(listener);
43     }
44     public void addOnLoadAllProductsListener(OnLoadAllProductsListener listener){
45         mOnLoadAllProductsListener.add(listener);
46     }
47     public void getInfoProducte(ArrayList<InfoSuperProducte> llistaInfoSuperProducts, Long
48     → id_producte){
49         llistaInfoSuperProducts.clear();
50
51         mBd.collection("products2").document("Supers").get().addOnCompleteListener(new
52         → OnCompleteListener<DocumentSnapshot>() {
```

```

51     @Override
52     public void onComplete(@NonNull Task<DocumentSnapshot> task) {
53         if(task.isSuccessful()) {
54             DocumentSnapshot document = task.getResult();
55             ArrayList<String> supers = (ArrayList<String>) document.get("Supers");
56
57             int aux = 1;
58             for (String s : supers){
59                 InfoSuperProducte infoSuperProducte = new InfoSuperProducte(s);
60                 if(!s.equals("Escull un supermercat")){
61                     loadInfoSupersProducte(infoSuperProducte, id_producte,
62                         supers.size(), aux, llistaInfoSuperProductes);
63                 }
64                 llistaInfoSuperProductes.add(infoSuperProducte);
65                 aux++;
66             }
67             /*for (OnLoadInfoProducteListener l : mOnLoadInfoProducteListeners){
68                 l.onLoadInfoProducte(llistaInfoSuperProductes);
69             }*/
70             else{
71                 Log.d(TAG, "Error getting documents: ", task.getException());
72             }
73         });
74     }
75
76     public void loadInfoSupersProducte(InfoSuperProducte infoSuperProducte, Long
77         id_producte, int numeroSupers, int aux, ArrayList<InfoSuperProducte>
78         llistaInfoSuperProductes){
79
80         mBd.collection("products2").document(Long.toString(id_producte)).get().addOnCompleteListener(
81             OnCompleteListener<DocumentSnapshot>() {
82                 @Override
83                 public void onComplete(@NonNull Task<DocumentSnapshot> task) {
84                     if(task.isSuccessful()){
85                         DocumentSnapshot document= task.getResult();
86                         ArrayList<String> info1product = (ArrayList<String>)
87                             document.get(infoSuperProducte.getNom_Super());
88                         for(String str : info1product){
89                             if(str.equals("0") || str.equals("1")){
90                                 infoSuperProducte.setDisponible(str);
91                             }else{
92                                 infoSuperProducte.setPreu(str);
93                             }
94                         //setNomProducte(infoSuperProducte, id_producte);
95                     }
96                 }
97             }
98         );
99     }

```

```

91         if(aux == numeroSupers) {
92             for (OnLoadInfoProducteListener l : mOnLoadInfoProducteListeners) {
93                 l.onLoadInfoProducte(llistaInfoSuperProductes);
94             }
95         }
96     }
97 }
98 );
99 }
100
101 /*private void setNomProducte(InfoSuperProducte infoSuperProducte, Long id_producte){
102
103     mBd.collection("products").document(Long.toString(id_producte)).get().addOnCompleteListener(new
104     OnCompleteListener<DocumentSnapshot>() {
105
106         @Override
107         public void onComplete(@NonNull Task<DocumentSnapshot> task) {
108
109             if(task.isSuccessful()){
110                 DocumentSnapshot document = task.getResult();
111                 String nom = document.getString("Nom producte");
112                 infoSuperProducte.setNom_producte(nom);
113             }
114         }
115     });
116 }
117 */
118
119 public void loadInfoAllProductes(ArrayList<Product> llistaProductes, ArrayList<Product>
120     llista2){
121
122     llistaProductes.clear();
123
124     mBd.collection("products2").get().addOnCompleteListener(new
125     OnCompleteListener<QuerySnapshot>() {
126
127         @Override
128         public void onComplete(@NonNull Task<QuerySnapshot> task) {
129
130             if(task.isSuccessful()){
131                 for(QueryDocumentSnapshot document : task.getResult()){
132                     if(!document.getId().equals("Supers")){
133                         Product p = new Product(Long.parseLong(document.getId()),
134                         String.valueOf(document.get("name")));
135                         llistaProductes.add(p);
136                         llista2.add(p);
137                     }
138                 }
139
140                 for(OnLoadAllProductsListener l : mOnLoadAllProductsListener){
141                     l.onLoadAllProducts(llistaProductes);
142                 }
143             }
144         }
145     });
146 }
```

```

132     });
133
134 }
135 public void addProductToList(String nomLlista, String mailUser, Long numProduct){
136     DocumentReference docRef = mBd.collection("users").document(mailUser);
137     docRef.update(nomLlista, FieldValue.arrayUnion(numProduct));
138 }
139
140 }
```

Listing A.23: Codi de la classe corresponent a *InfoProducteRepository*.

```

1 package edu.ub.pis3.model;
2
3 import androidx.annotation.NonNull;
4
5 import com.google.android.gms.tasks.OnCompleteListener;
6 import com.google.android.gms.tasks.Task;
7 import com.google.firebaseio.DocumentReference;
8 import com.google.firebaseio.DocumentSnapshot;
9 import com.google.firebaseio.FieldValue;
10 import com.google.firebaseio.FirebaseFirestore;
11
12 import org.checkerframework.checker.units.qual.A;
13
14 import java.util.ArrayList;
15
16 public class ProductRepository {
17     private static final String TAG = "ProductRepository";
18     private static final ProductRepository mInstance = new ProductRepository();
19     private FirebaseFirestore mBd;
20     public ArrayList<ProductRepository.OnLoadProductsListener> mOnLoadProductsListeners;
21     public ArrayList<ProductRepository.OnLoadInfoProductsListener>
22         → mOnLoadInfoProductsListeners;
23
24     public interface OnLoadProductsListener {
25         void onLoadProducts(ArrayList<Product> products);
26     }
27
28     public interface OnLoadInfoProductsListener {
29         void onLoadInfoProducts(ArrayList<Product> products);
30     }
31
32     private ProductRepository(){
33         mBd = FirebaseFirestore.getInstance();
```

```
33     mOnLoadProductsListeners = new ArrayList<>();
34     mOnLoadInfoProductsListeners = new ArrayList<>();
35 }
36
37     public static ProductRepository getInstance(){
38         return mInstance;
39     }
40     public void addOnLoadProductsListener(OnLoadProductsListener listener){
41         mOnLoadProductsListeners.add(listener);
42     }
43     public void addOnLoadInfoProductsListener(OnLoadInfoProductsListener listener){
44         mOnLoadInfoProductsListeners.add(listener);
45     }
46
47     public void loadProducts(ArrayList<Product> products, String nomLlistaCompra, String
48     → mailUser){
49         products.clear();
50         mBd.collection("users").document(mailUser).get().addOnCompleteListener(new
51         → OnCompleteListener<DocumentSnapshot>() {
52             @Override
53             public void onComplete(@NonNull Task<DocumentSnapshot> task) {
54                 if(task.isSuccessful()){
55                     DocumentSnapshot document = task.getResult();
56                     ArrayList<Long> idsProductes = (ArrayList<Long>)
57                     → document.get(nomLlistaCompra);
58                     for (Long p: idsProductes){
59                         Product product = new Product(p);
60                         //loadAllInfoProducts(product, p);
61                         products.add(product);
62                     }
63                 }
64             }
65         });
66     }
67
68     public void loadAllInfoProducts(ArrayList<Product> llistaProductes){
69         for(Product p : llistaProductes){
70
71             → mBd.collection("products2").document(Long.toString(p.getId())).get().addOnCompleteListener(
72             → OnCompleteListener<DocumentSnapshot>() {
73                 @Override
74                 public void onComplete(@NonNull Task<DocumentSnapshot> task) {
75                     if(task.isSuccessful()){


```

```

74         DocumentSnapshot document = task.getResult();
75         p.setNom(document.getString("name"));
76         p.setPhoto(document.getString("photo"));
77
78         for(OnLoadInfoProductsListener l : mOnLoadInfoProductsListeners){
79             l.onLoadInfoProducts(llistaProductes);
80         }
81     }
82 }
83 );
84 }
85
86 /*mBd.collection("productos").document(Long.toString(id_product)).get().addOnCompleteListener(
87 OnCompleteListener<DocumentSnapshot>() {
88     @Override
89     public void onComplete(@NonNull Task<DocumentSnapshot> task) {
90         if(task.isSuccessful()){
91             DocumentSnapshot document = task.getResult();
92             product.setNom(document.getString("name"));
93             product.setPhoto(document.getString("photo"));
94         }
95     }
96 }
97 */
98
99     public void removeProductFromList(String nomLlista, String mailUser, Long numProduct){
100        DocumentReference docRef = mBd.collection("users").document(mailUser);
101        docRef.update(nomLlista, FieldValue.arrayRemove(numProduct));
102    }
103 }
```

Listing A.24: Configuració de la classe *ProductRepository*.

```

1 package edu.ub.pis3.model;
2
3 import android.util.Log;
4 import android.widget.Toast;
5
6 import androidx.annotation.NonNull;
7
8 import com.google.android.gms.tasks.OnCompleteListener;
9 import com.google.android.gms.tasks.Task;
10 import com.google.firebase.auth.AuthResult;
11 import com.google.firebase.auth.FirebaseAuth;
12 import com.google.firebaseio.DocumentReference;
```

```
13 import com.google.firebaseio.DocumentSnapshot;
14 import com.google.firebaseio.FirebaseFirestore;
15
16 import java.util.*;
17 import java.util.concurrent.Executor;
18
19 public class UserRepository {
20     private static final String TAG = "UserRepository";
21     private static final UserRepository mInstance = new UserRepository();
22     private FirebaseFirestore mBd;
23     private FirebaseAuth mAuth;
24
25     public interface OnLoadUserListener {
26         void onLoadUser(User user);
27     }
28
29     public ArrayList<OnLoadUserListener> mOnLoadUserListeners = new ArrayList<>();
30
31     private UserRepository() {
32         mBd = FirebaseFirestore.getInstance();
33         mAuth = mAuth = FirebaseAuth.getInstance();
34     }
35
36     public static UserRepository getInstance() {
37         return mInstance;
38     }
39
40     public void addOnLoadUserListener(OnLoadUserListener listener) {
41         mOnLoadUserListeners.add(listener);
42     }
43
44     public void loadUser(String mailUser, User user){
45         DocumentReference docRef = mBd.collection("users").document(mailUser);
46
47         docRef.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
48             @Override
49             public void onComplete(@NonNull Task<DocumentSnapshot> task) {
50                 if(task.isSuccessful()) {
51                     DocumentSnapshot document = task.getResult();
52                     String nom = document.getString("nom");
53                     String cognom = document.getString("cognom");
54                     String contrassenya = document.getString("contrasenya");
55
56                     user.setEmail(mailUser);
57                     user.setNom(nom);
58                     user.setPwd(contrassenya);
59                 }
60             }
61         });
62     }
63 }
```

```

59         user.setCognom(cognom);
60     }
61   }
62 );
63 }
64
65 public void signup(String nom, String cognom, String mail, String pwd){
66   Map<String, Object> user = new HashMap<>();
67   user.put("nom", nom);
68   user.put("cognom", cognom);
69   user.put("contrasenya", pwd);
70   List<String> nomLlistes = new ArrayList<>();
71   user.put("Llistes compra", nomLlistes);
72
73   DocumentReference docRef = mBd.collection("users").document(mail);
74   docRef.set(user);
75 }
76
77 }
```

Listing A.25: Configuració de la classe *UserRepository*.

```

1 package edu.ub.pis3.model;
2
3 import android.util.Log;
4
5 import androidx.annotation.NonNull;
6
7 import com.google.android.gms.tasks.OnCompleteListener;
8 import com.google.android.gms.tasks.Task;
9 import com.google.firebaseio.DocumentReference;
10 import com.google.firebaseio.DocumentSnapshot;
11 import com.google.firebaseio.firebaseio.FirebaseFirestore;
12 import com.google.firebaseio.QueryDocumentSnapshot;
13 import com.google.firebaseio.QuerySnapshot;
14
15 import java.util.ArrayList;
16 import java.util.HashMap;
17
18 public class CalculadoraRepository {
19   private static final String TAG = "CalculadoraRepository";
20   private static final CalculadoraRepository mInstance = new CalculadoraRepository();
21   private FirebaseFirestore mBd;
22   public ArrayList<OnLoadSupersListener> mOnLoadSupersListener;
23 }
```

```
24 public ArrayList<OnLoadInfoSupersListener> mOnLoadInfoSupersListener;
25
26 ArrayList<String> mNomsSupers;
27
28 public interface OnLoadSupersListener {
29     void onLoadSupers(ArrayList<String> llistaSupers);
30 }
31
32 public interface OnLoadInfoSupersListener {
33     void onLoadInfoSupers(HashMap<String, ArrayList<InfoSuperProducte>> matrix);
34 }
35
36 private CalculadoraRepository(){
37     mBd = FirebaseFirestore.getInstance();
38     mOnLoadSupersListener = new ArrayList<>();
39     mOnLoadInfoSupersListener = new ArrayList<>();
40     mNomsSupers = new ArrayList<>();
41 }
42 public static CalculadoraRepository getInstance(){
43     return mInstance;
44 }
45
46 public void addOnLoadSupersListener(OnLoadSupersListener listener){
47     mOnLoadSupersListener.add(listener);
48 }
49
50 public void addOnLoadInfoSupersListener(OnLoadInfoSupersListener listener){
51     if(mOnLoadInfoSupersListener.size()==0){
52         mOnLoadInfoSupersListener.add(listener);
53     }
54 }
55
56 public void loadNomsSupers(ArrayList<String> nomssupers){
57     nomssupers.clear();
58     mNomsSupers.clear();
59     mBd.collection("products2").document("Supers").get().addOnCompleteListener(new
60         OnCompleteListener<DocumentSnapshot>() {
61             @Override
62             public void onComplete(@NonNull Task<DocumentSnapshot> task) {
63                 if(task.isSuccessful()){
64                     DocumentSnapshot document = task.getResult();
65                     ArrayList<String> llista = (ArrayList<String>) document.get("Supers");
66                     for(String str : llista){
67                         nomssupers.add(str);
68                         mNomsSupers.add(str);
69                     }
70                 }
71             }
72         });
73 }
74
75 public void addOnLoadInfoSupersListener(OnLoadInfoSupersListener listener){
76     mOnLoadInfoSupersListener.add(listener);
77 }
```

```

69         for(OnLoadSupersListener l : mOnLoadSupersListener){
70             l.onLoadSupers(nomsSupers);
71         }
72     }else{
73         Log.d(TAG, "Error getting documents: ", task.getException());
74     }
75 }
76 });
77 }
78
79 public void loadAllInfoSupersProductes(HashMap<String, ArrayList<InfoSuperProducte>>
80     matrix){
81     //matrixx.clear();
82     mBd.collection("products2").get().addOnCompleteListener(new
83     OnCompleteListener<QuerySnapshot>() {
84         @Override
85         public void onComplete(@NonNull Task<QuerySnapshot> task) {
86             if(task.isSuccessful()){
87                 int aux = 1;
88                 for(QueryDocumentSnapshot document : task.getResult()){
89                     String id = document.getId();
90                     if(!id.equals("Supers")){
91                         //ArrayList<String> nomSupers = new ArrayList<>();
92                         //loadNomsSupers(nomSupers);
93                         int aux2 = 1;
94                         ArrayList<InfoSuperProducte> llista = new ArrayList<>();
95                         for(String s : mNomsSupers){
96                             InfoSuperProducte infoSuperProducte = new
97                             InfoSuperProducte(s);
98                             if(!s.equals("Escull un supermercat")){
99                                 loadInfoSupersProducte(infoSuperProducte,
100                                     Long.parseLong(id), aux, aux2, mNomsSupers.size(),
101                                     matrix);
102                                 llista.add(infoSuperProducte);
103                             }
104                             aux2++;
105                         }
106
107                         matrix.put(String.valueOf(document.get("name")), llista);
108                     }
109                     aux++;
110                 }
111             }
112         }
113     });
114 }
115 });

```

```

110    }
111
112
113    public void loadInfoSupersProducte(InfoSuperProducte infoSuperProducte, Long
114        → id_producte, int aux, int aux2, int numSupers, HashMap<String,
115        → ArrayList<InfoSuperProducte>> matrix){
116
117        → mBd.collection("products2").document(Long.toString(id_producte)).get().addOnCompleteListener(
118            → OnCompleteListener<DocumentSnapshot>() {
119                @Override
120                public void onComplete(@NonNull Task<DocumentSnapshot> task) {
121                    if(task.isSuccessful()){
122                        DocumentSnapshot document= task.getResult();
123                        ArrayList<String> info1product = (ArrayList<String>)
124                            → document.get(infoSuperProducte.getNom_Super());
125                        for(String str : info1product){
126                            if(str.equals("0") || str.equals("1")){
127                                infoSuperProducte.setDisponible(str);
128                            }else{
129                                infoSuperProducte.setPreu(str);
130                            }
131                        }
132
133                    }
134                }
135            });
136        }
137    }

```

Listing A.26: Configuració de la classe *CalculadoraRepository*.

A.3

CODIS DE LA VISTA

```

1 package edu.ub.pis3.view;
2
3 import androidx.appcompat.app.AppCompatActivity;
4

```

```
5 import android.content.Intent;
6 import android.os.Bundle;
7 import android.view.View;
8 import android.widget.Button;
9
10 import edu.ub.pis3.R;
11
12 public class MainActivity extends AppCompatActivity {
13
14     Button btn_login;
15     Button btn_registrarse;
16
17     @Override
18     protected void onCreate(Bundle savedInstanceState) {
19         super.onCreate(savedInstanceState);
20         setContentView(R.layout.activity_main);
21
22         btn_login = (Button) findViewById(R.id.btn_login);
23         btn_registrarse = (Button) findViewById(R.id.btn_registrarse);
24
25         getSupportActionBar().hide();
26
27         btn_login.setOnClickListener(new View.OnClickListener() {
28             @Override
29             public void onClick(View view) {
30                 Intent intent1 = new Intent(MainActivity.this, IniciSessioActivity.class);
31                 startActivity(intent1);
32             }
33         });
34
35         btn_registrarse.setOnClickListener(new View.OnClickListener() {
36             @Override
37             public void onClick(View view) {
38                 Intent intent2 = new Intent(MainActivity.this, RegistreActivity.class);
39                 startActivity(intent2);
40             }
41         });
42     }
43
44     @Override
45     public void onBackPressed() {
46         Intent intent1 = new Intent(MainActivity.this, MainActivity.class);
47         startActivity(intent1);
48     }
49 }
```

Listing A.27: Configuració de la classe *MainActivity*.

```
1 package edu.ub.pis3.view;
2
3 import androidx.annotation.NonNull;
4 import androidx.appcompat.app.AppCompatActivity;
5 import androidx.lifecycle.ViewModelProvider;
6
7 import android.content.Intent;
8 import android.os.Bundle;
9 import android.util.Log;
10 import android.view.View;
11 import android.widget.Button;
12 import android.widget.EditText;
13 import android.widget.Toast;
14
15 import com.google.android.gms.tasks.OnCompleteListener;
16 import com.google.android.gms.tasks.OnFailureListener;
17 import com.google.android.gms.tasks.OnSuccessListener;
18 import com.google.android.gms.tasks.Task;
19 import com.google.firebase.auth.AuthResult;
20 import com.google.firebase.auth.FirebaseAuth;
21 import com.google.firebaseio.firebaseio.FirebaseFirestore;
22
23 import java.util.ArrayList;
24 import java.util.HashMap;
25 import java.util.List;
26 import java.util.Map;
27
28 import edu.ub.pis3.R;
29 import edu.ub.pis3.viewmodel.LlistesCompraViewModel;
30 import edu.ub.pis3.viewmodel.UsersViewModel;
31
32 public class RegistreActivity extends AppCompatActivity {
33     private String TAG = "LlistaCompraActivity";
34     Button btn_fer_sign_up;
35     EditText name;
36     EditText cognom;
37     EditText mail;
38     EditText pwd;
39     private UsersViewModel usersViewModel;
40     private FirebaseAuth mAuth;
41     private FirebaseFirestore db;
42     private LlistesCompraViewModel llistaViewModel;
43
44     @Override
```



```
87         Toast.makeText(getApplicationContext(), "Sign up failed",
88             → Toast.LENGTH_SHORT).show();
89     }
90 }
91 }
92
93 @Override
94 public void onBackPressed() {
95     Intent intent1 = new Intent(RegistreActivity.this, MainActivity.class);
96     startActivity(intent1);
97 }
98 }
```

Listing A.28: Configuració de la classe *RegistreActivity*.

```
1 package edu.ub.pis3.view;
2
3 import androidx.annotation.NonNull;
4 import androidx.appcompat.app.AppCompatActivity;
5
6 import android.content.Intent;
7 import android.content.SharedPreferences;
8 import android.os.Bundle;
9 import android.os.Handler;
10 import android.view.View;
11 import android.widget.Button;
12 import android.widget.CheckBox;
13 import android.widget.EditText;
14 import android.widget.FrameLayout;
15 import android.widget.ProgressBar;
16 import android.widget.Toast;
17
18 import com.google.android.gms.tasks.OnCompleteListener;
19 import com.google.android.gms.tasks.Task;
20 import com.google.firebase.auth.AuthResult;
21 import com.google.firebase.auth.FirebaseAuth;
22
23 import edu.ub.pis3.R;
24
25 public class IniciSessioActivity extends AppCompatActivity {
26
27     Button btn_fer_login;
28     EditText mail;
29     EditText pwd;
```

```
30    private FirebaseAuth mAuth;
31    private FrameLayout frameLayout;
32
33    CheckBox checkBoxRememberMe;
34
35    private SharedPreferences sharedpreferences;
36    private static final String PREF_NAME = "login_prefs";
37    private static final String KEY_USERNAME = "username";
38    private static final String KEY_PASSWORD = "password";
39    private static final String KEY_REMEMBER_ME = "remember_me";
40
41    @Override
42    protected void onCreate(Bundle savedInstanceState) {
43        super.onCreate(savedInstanceState);
44        setContentView(R.layout.activity_inici_sessio);
45
46        btn_fer_login = (Button) findViewById(R.id.btn_fer_loggin);
47        mail = (EditText) findViewById(R.id.edit_txt_mail);
48        pwd = (EditText) findViewById(R.id.edit_txt_pwd);
49        checkBoxRememberMe = findViewById(R.id.check_remember_me);
50        mAuth = FirebaseAuth.getInstance();
51
52        frameLayout = findViewById(R.id.frame_layout_inici_sessio);
53
54        getSupportActionBar().setTitle("Iniciar sessió");
55        sharedpreferences = getSharedPreferences(PREF_NAME, MODE_PRIVATE);
56
57        boolean rememberMe = sharedpreferences.getBoolean(KEY_REMEMBER_ME, false);
58
59        if (rememberMe) {
60            frameLayout.setVisibility(View.VISIBLE);
61            String username = sharedpreferences.getString(KEY_USERNAME, "");
62            String password = sharedpreferences.getString(KEY_PASSWORD, "");
63            mail.setText(username);
64            pwd.setText(password);
65            checkBoxRememberMe.setChecked(true);
66            Handler handler = new Handler();
67            handler.postDelayed(new Runnable() {
68                @Override
69                public void run() {
70                    login(username, password);
71                }
72            }, 700);
73
74        }
75    }
```

```
76     btn_fer_login.setOnClickListener(new View.OnClickListener() {
77         @Override
78         public void onClick(View view) {
79             frameLayout.setVisibility(View.VISIBLE);
80             if(mail.getText().toString().equals("") ||
81                 ↳ pwd.getText().toString().equals("")) {
82                 Toast.makeText(getApplicationContext(), "No pots deixar camps en blanc",
83                 ↳ Toast.LENGTH_SHORT).show();
84             } else {
85                 String _mail = mail.getText().toString();
86                 String _pwd = pwd.getText().toString();
87                 login( _mail, _pwd);
88             }
89         }
90     });
91
92     protected void login(String _mail, String _pwd){
93         mAuth.signInWithEmailAndPassword(_mail, _pwd).addOnCompleteListener(this, new
94             ↳ OnCompleteListener<AuthResult>() {
95                 @Override
96                 public void onComplete(@NonNull Task<AuthResult> task) {
97                     if(task.isSuccessful()){
98                         if (checkBoxRememberMe.isChecked()) {
99                             sharedPreferences.edit()
100                             .putString(KEY_USERNAME, _mail)
101                             .putString(KEY_PASSWORD, _pwd)
102                             .putBoolean(KEY_REMEMBER_ME, true)
103                             .apply();
104                     } else {
105                         sharedPreferences.edit().clear().apply();
106                     }
107                     Intent intent1 = new Intent(IniciSessioActivity.this,
108                         ↳ LlistaCompraActivity.class);
109                     intent1.putExtra("user", _mail);
110                     startActivity(intent1);
111                     finish();
112                     Handler handler = new Handler();
113                     handler.postDelayed(new Runnable() {
114                         @Override
115                         public void run() {
116                             frameLayout.setVisibility(View.GONE);
117                         }
118                     }, 500);
119                 } else {
120                     Toast.makeText(getApplicationContext(), "Log in failed",
121                     ↳ Toast.LENGTH_SHORT).show();
122                 }
123             }
124         });
125     }
126 }
```

```
118         frameLayout.setVisibility(View.GONE);  
119     }  
120 }  
121 );  
122 }  
123 @Override  
124 public void onBackPressed() {  
125     Intent intent1 = new Intent(IniSessionActivity.this, MainActivity.class);  
126     startActivity(intent1);  
127     finish();  
128 }  
129 }
```

Listing A.29: Configuració de la classe *IniSessionActivitat*.

```
1 package edu.ub.pis3.view;  
2  
3 import android.content.Intent;  
4 import android.content.SharedPreferences;  
5 import android.net.Uri;  
6 import android.os.Bundle;  
7 import android.view.Menu;  
8 import android.view.MenuItem;  
9 import android.view.View;  
10 import android.view.ViewGroup;  
11 import android.widget.EditText;  
12 import android.widget.Button;  
13 import android.widget.Toast;  
14  
15 import androidx.annotation.NonNull;  
16 import androidx.appcompat.app.AppCompatActivity;  
17 import androidx.lifecycle.Observer;  
18 import androidx.lifecycle.ViewModelProvider;  
19 import androidx.recyclerview.widget.LinearLayoutManager;  
20 import androidx.recyclerview.widget.RecyclerView;  
21  
22 import androidx.fragment.app.DialogFragment;  
23 import android.app.AlertDialog;  
24 import android.app.Dialog;  
25 import android.content.DialogInterface;  
26  
27 import com.google.android.material.floatingactionbutton.FloatingActionButton;  
28 import com.google.firebase.auth.FirebaseAuth;  
29  
30 import java.io.Serializable;
```

```
31 import java.util.ArrayList;
32
33 import edu.ub.pis3.R;
34 import edu.ub.pis3.model.LlistesCompra;
35 import edu.ub.pis3.viewmodel.LlistesCompraActivityViewModel;
36
37 public class LlistaCompraActivity extends AppCompatActivity {
38
39     private final String TAG = "LlistesCompraActivity";
40     private LlistesCompraActivityViewModel llistesCompraActivityViewModel;
41     private RecyclerView mListaCardsRV;
42
43     private ViewGroup loggedLayout;
44     private LlistesCompraCardAdapter mListaCardRVAdapter;
45
46     private FirebaseAuth mAuth = FirebaseAuth.getInstance();
47
48     private String mailUser;
49     private ArrayList<String> llistaProducts;
50     private FloatingActionButton btn_afegir_llista;
51     private EditText nom_nova_llista;
52
53     private boolean eliminar = false;
54
55     @Override
56     protected void onCreate(Bundle savedInstanceState) {
57         super.onCreate(savedInstanceState);
58         setContentView(R.layout.activity_llista_compra);
59
60         AlertDialog adAfegirLlista = setDialogAfegirLlista();
61
62         Bundle extras = getIntent().getExtras();
63         mailUser = extras.getString("user");
64
65         btn_afegir_llista = (FloatingActionButton) findViewById(R.id.button_afegir_llista);
66
67         getSupportActionBar().setTitle("CompraSmart");
68
69         llistesCompraActivityViewModel = new ViewModelProvider(this)
70             .get(LlistesCompraActivityViewModel.class);
71
72         //loggedLayout = findViewById(R.id.loggedLayout);
73
74         mListaCardsRV = findViewById(R.id.listaCardRV);
75
76         LinearLayoutManager manager = new LinearLayoutManager(this,
```

```
77         LinearLayoutManager.VERTICAL, false);
78
79     mListaCardsRV.setLayoutManager(manager);
80     mListaCardRVAdapter = new LlistesCompraCardAdap-
81     ter(llistesCompraViewModel.getmCompra().getValue());
82
83     mListaCardRVAdapter.setOnClickListener(new
84     ↪ LlistesCompraCardAdapter.OnClickGoListener() {
85         @Override
86         public void OnClickGo(String nomLista) {
87             Intent intent1 = new Intent(LlistaCompraActivity.this,
88             ↪ VeureLlistaCompraActivity.class);
89             intent1.putExtra("nom llista compra", nomLista);
90             intent1.putExtra("user", mailUser);
91             llistaProducts = llistesCompraViewModel.getmProducts(nomLista);
92             intent1.putStringArrayListExtra("products", llistaProducts);
93             startActivityForResult(intent1);
94         }
95     });
96
97
98     mListaCardRVAdapter.setOnClickHideListener(new
99     ↪ LlistesCompraCardAdapter.OnClickHideListener(){
100        @Override
101        public void OnClickHide(int position){
102            AlertDialog adEliminarLista = setDialogEliminarLista(position);
103            adEliminarLista.show();
104        }
105    });
106
107    btn_afegir_llista.setOnClickListener(new View.OnClickListener() {
108        @Override
109        public void onClick(View view) {
110            adAfegirLista.show();
111        }
112    });
113
114    mListaCardsRV.setAdapter(mListaCardRVAdapter);
115
116    final Observer<ArrayList<LlistesCompra>> observerLlistesCompra = new
117    ↪ Observer<ArrayList<LlistesCompra>>() {
118        @Override
119        public void onChanged(ArrayList<LlistesCompra> users) {
120            mListaCardRVAdapter.notifyDataSetChanged();
121        }
122    };
123 }
```

```
118     llistesCompraActivityViewModel.getmCompra().observe(this, observerLlistesCompra);  
119  
120     // A partir d'aquí, en cas que es faci cap canvi a la llista d'usuaris, HomeActivity  
121     // ho sabrà  
122     llistesCompraActivityViewModel.loadLlistesCompraFromRepository(mailUser); //  
123     // Internament pobla els usuaris de la BBDD  
124  
125 }  
126  
127 public void eliminarLlista(int position){  
128     llistesCompraActivityViewModel.removeLlistaFromHome(position, mailUser);  
129     mLlistaCardRVAdapter.hideLlista(position);  
130 }  
131  
132 public void addLlista(String nameList){  
133     llistesCompraActivityViewModel.addLlista(nameList, mailUser);  
134 }  
135  
136 private AlertDialog setDialogAfegirLlista(){  
137     AlertDialog.Builder builder = new AlertDialog.Builder(this);  
138     builder.setTitle("Vols afegir una nova llista?");  
139     //builder.setTitle("Afegir nova llista");  
140     //builder.setMessage("Vols afegir una nova llista?");  
141     nom_nova_llista = new EditText(this);  
142     builder.setView(nom_nova_llista);  
143  
144     builder.setPositiveButton("Sí", new DialogInterface.OnClickListener() {  
145         @Override  
146         public void onClick(DialogInterface dialogInterface, int i) {  
147             // Guardem el nom que hem escrit  
148             String text = nom_nova_llista.getText().toString();  
149  
150             // Un cop ha quedat guardat a la variable l'esborrem pel pròxim cop  
151             nom_nova_llista.setText("");  
152  
153             // En cas que el nom escrit sigui null escrivim un nom random  
154             if(text.equals("")){  
155                 text = "Nova llista";  
156             }  
157  
158             boolean found = false;  
159  
160             // Mirem que els noms de les llistes no estigui repetits  
161             for(LlistesCompra llista:  
162                 llistesCompraActivityViewModel.getmCompra().getValue()) {  
163                 // Mirem si el nostre nom coincideix amb el d'alguna llista  
164                 if(text.equals(llista.getNom())) {
```

```

161                     found = true;
162                 }
163             }
164
165             // Si no hem trobat cap nombre que coincideixi
166             if(!found){
167                 addLlista(text);
168
169                 Intent intent1 = new Intent(LlistaCompraActivity.this,
170                     → AfegirProducteActivity.class);
171                 intent1.putExtra("nom llista compra", text);
172                 intent1.putExtra("user", mailUser);
173                 llistaProducts = llistesCompraActivityViewModel.getmProducts(text);
174                 intent1.putStringArrayListExtra("products", llistaProducts);
175                 startActivity(intent1);
176             }
177             // En cas d'haver trobat una coincidència salta un missatge
178             else{
179                 Toast.makeText(getApplicationContext(), "El nom de la llista no és
180                     → vàlid", Toast.LENGTH_SHORT).show();
181             }
182         }
183     builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
184         @Override
185         public void onClick(DialogInterface dialogInterface, int i) {
186             dialogInterface.dismiss();
187         }
188     });
189
190     AlertDialog ad = builder.create();
191     return ad;
192 }
193
194 private AlertDialog setDialogEliminarLlista(int position){
195     AlertDialog.Builder builder = new AlertDialog.Builder(this);
196     builder.setTitle("Vols eliminar aquesta llista?");
197     //builder.setTitle("Afegir nova llista");
198     //builder.setMessage("Vols afegir una nova llista?");
199     //nom_nova_llista = new EditText(this);
200     //builder.setView(nom_nova_llista);
201
202     builder.setPositiveButton("Sí", new DialogInterface.OnClickListener() {
203         @Override
204         public void onClick(DialogInterface dialogInterface, int i) {

```

```
205         //String text = nom_nova_llista.getText().toString();
206         eliminarLlista(position);
207     }
208 });
209 builder.setNegativeButton("No", new DialogInterface.OnClickListener() {
210     @Override
211     public void onClick(DialogInterface dialogInterface, int i) {
212         dialogInterface.dismiss();
213     }
214 });
215
216 AlertDialog ad = builder.create();
217 return ad;
218 }
219
220 @Override
221 public boolean onCreateOptionsMenu(Menu menu) {
222     getMenuInflater().inflate(R.menu.menu_logout, menu);
223     return true;
224 }
225
226 @Override
227 public boolean onOptionsItemSelected(@NonNull MenuItem item) {
228     if (item.getItemId() == R.id.menu_logout) {
229         logout();
230         return true;
231     }
232     return super.onOptionsItemSelected(item);
233 }
234
235 @Override
236 public void onBackPressed() {
237     Intent intent1 = new Intent(LlistaCompraActivity.this, MainActivity.class);
238     startActivity(intent1);
239     finish();
240 }
241
242 private void logout() {
243     SharedPreferences sharedpreferences = getSharedPreferences("login_prefs",
244         MODE_PRIVATE);
245     sharedpreferences.edit().clear().apply();
246     mAuth.signOut();
247     Intent intent = new Intent(LlistaCompraActivity.this, MainActivity.class);
248     startActivity(intent);
249     finish();
250 }
```

250

}

Listing A.30: Configuració de la classe *LlistaCompraActivity*.

```
1 package edu.ub.pis3.view;
2
3 import androidx.annotation.NonNull;
4 import androidx.appcompat.app.AppCompatActivity;
5
6 import android.content.Intent;
7 import android.os.Bundle;
8 import android.view.Menu;
9 import android.view.MenuItem;
10 import android.view.View;
11 import android.widget.CheckBox;
12 import android.widget.ImageView;
13 import android.widget.ListView;
14 import android.widget.Toast;
15
16 import com.google.android.material.floatingactionbutton.FloatingActionButton;
17
18 import androidx.lifecycle.Observer;
19 import androidx.lifecycle.ViewModelProvider;
20 import androidx.recyclerview.widget.LinearLayoutManager;
21 import androidx.recyclerview.widget.RecyclerView;
22
23 import org.checkerframework.checker.units.qual.A;
24
25 import java.util.ArrayList;
26
27 import edu.ub.pis3.R;
28 import edu.ub.pis3.model.LlistesCompra;
29 import edu.ub.pis3.model.Product;
30 import edu.ub.pis3.viewmodel.LlistesCompraViewModel;
31 import edu.ub.pis3.viewmodel.ProductesActivityViewModel;
32
33 public class VeureLlistaCompraActivity extends AppCompatActivity {
34
35     private final String TAG = "VeureLlistaCompraActivity";
36     private String nomLlistaCompra, mailUser;
37     private ArrayList<String> llistaProducts;
38     private ProductesActivityViewModel productesActivityViewModel;
39     private FloatingActionButton btn, btn2, btn_calculadora;
40     private RecyclerView mProductCardsRV;
41     private LlistaProductesCardAdapter mLlistaProductesCardRVAdapter;
```

```
42     private boolean checkBoxsVisibles;
43     private ArrayList<Integer> llistaPositionProductesEliminar;
44
45     @Override
46     protected void onCreate(Bundle savedInstanceState) {
47         super.onCreate(savedInstanceState);
48         setContentView(R.layout.activity_veure_llista_compra);
49
50         Bundle extras = getIntent().getExtras();
51         nomLlistaCompra = extras.getString("nom llista compra");
52         mailUser = extras.getString("user");
53         llistaProducts = extras.getStringArrayList("products");
54         btn = (FloatingActionButton) findViewById(R.id.floatingActionButton);
55         btn2 = (FloatingActionButton) findViewById(R.id.floatingActionButton2);
56         btn_calculadora = (FloatingActionButton) findViewById(R.id.btn_calculadora);
57         checkBoxsVisibles = false;
58         llistaPositionProductesEliminar = new ArrayList<Integer>();
59
60
61         getSupportActionBar().setTitle(nomLlistaCompra);
62
63         productesActivityViewModel = new ViewModelProvider(this)
64             .get(ProductesActivityViewModel.class);
65
66         mProductCardsRV = findViewById(R.id.listProductsRV);
67
68         LinearLayoutManager manager = new LinearLayoutManager(this,
69                         LinearLayoutManager.VERTICAL, false);
70
71         mProductCardsRV.setLayoutManager(manager);
72         mLlistaProductesCardRVAdapter = new LlistaProductesCardAdap-
73             ter(productesActivityViewModel.getmProducts().getValue());
74
75         mLlistaProductesCardRVAdapter.setOnClickListener(new
76             LlistaProductesCardAdapter.OnClickGoListener() {
77             @Override
78             public void OnClickGo(String nomProducte, Long id_producte, String photo) {
79                 Intent intent1 = new Intent(VeureLlistaCompraActivity.this,
80                     VeureDetallsProducteActivity.class);
81                 intent1.putExtra("nom producte", nomProducte);
82                 intent1.putExtra("id producte", id_producte);
83                 intent1.putExtra("photo", photo);
84                 startActivity(intent1);
85             }
86         });
87     }
88 }
```

```
85     mLlistaProductesCardRVAdapter.setAfterCheckedHide(new
86         ↳ LlistaProductesCardAdapter.AfterCheckedHide() {
87             @Override
88             public void AfterCheckedHide(int position) {
89                 //llistaPositionProductesEliminar.add(position);
90                 productesActivityViewModel.removeProductFromList(position, nomLlistaCompra,
91                     ↳ mailUser);
92                 mLlistaProductesCardRVAdapter.hideProduct(position);
93                 llistaProducts.remove(position);
94             }
95
96             @Override
97             public void AfterUnchecked(int position) {
98                 /*int counter = 0;
99                 for(Integer p : llistaPositionProductesEliminar){
100                     if(p == position){
101                         llistaPositionProductesEliminar.remove(counter);
102                         int counter2=0;
103                         for(Integer i: llistaPositionProductesEliminar){
104                             if(i>counter){
105                                 llistaPositionProductesEliminar.remove(counter2);
106                                 llistaPositionProductesEliminar.add(i-1);
107                             }
108                             counter2++;
109                         }
110                     }
111                     counter++;
112                 }*/
113             }
114
115             btn.setOnClickListener(new View.OnClickListener() {
116                 @Override
117                 public void onClick(View view) {
118                     Intent intent = new Intent(VeureLlistaCompraActivity.this,
119                         ↳ AfegirProducteActivity.class);
120                     intent.putExtra("user", mailUser);
121                     intent.putExtra("nom llista compra", nomLlistaCompra);
122                     intent.putStringArrayListExtra("products", llistaProducts);
123                     startActivity(intent);
124                 }
125             });
126
127             btn2.setOnClickListener(new View.OnClickListener() {
128                 @Override
129                 public void onClick(View view) {
```

```
128     /*for(Integer i: llistaPositionProductesEliminar){
129         productesActivityViewModel.removeProductFromList(i, nomLlistaCompra,
130         ↳ mailUser);
131         mLlistaProductesCardRVAdapter.hideProduct(i);
132         int counter=0;
133         for(Integer j: llistaPositionProductesEliminar){
134             if(j>i){
135                 llistaPositionProductesEliminar.remove(counter);
136                 llistaPositionProductesEliminar.add(j-1);
137             }
138             counter++;
139         }
140     }*/
141     //llistaPositionProductesEliminar.clear();
142     mLlistaProductesCardRVAdapter.amagarCheckBoxs();
143     checkBoxsVisibles = false;
144     btn.setVisibility(View.VISIBLE);
145     btn2.setVisibility(View.INVISIBLE);
146 }
147
148 btn_calculadora.setOnClickListener(new View.OnClickListener() {
149     @Override
150     public void onClick(View view) {
151         Intent intent = new Intent(VeureLlistaCompraActivity.this,
152         ↳ CalculadoraActivity.class);
153         intent.putExtra("user", mailUser);
154         intent.putExtra("nom llista compra", nomLlistaCompra);
155         intent.putStringArrayListExtra("products", llistaProducts);
156         startActivityForResult(intent);
157     }
158 });
159
160 mLlistaProductesCardRV.setAdapter(mLlistaProductesCardRVAdapter);
161
162 final Observer<ArrayList<Product>> observerProductes = new
163     ↳ Observer<ArrayList<Product>>() {
164     @Override
165     public void onChanged(ArrayList<Product> products) {
166         mLlistaProductesCardRVAdapter.notifyDataSetChanged();
167     }
168 };
169
170 productesActivityViewModel.getmProducts().observe(this, observerProductes);
171
productesActivityViewModel.loadProductsFromRepository(mailUser, nomLlistaCompra);
```

```

171
172    }
173
174    @Override
175    public boolean onCreateOptionsMenu(Menu menu) {
176        getMenuInflater().inflate(R.menu.menu_llistes, menu);
177        return super.onCreateOptionsMenu(menu);
178    }
179
180    @Override
181    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
182        switch (item.getItemId()){
183            case R.id.action_deletelist:
184                if(checkBoxsVisibles){
185                    mListaProductesCardRVAdapter.amagarCheckBoxs();
186                    checkBoxsVisibles = false;
187                    btn.setVisibility(View.VISIBLE);
188                    btn2.setVisibility(View.INVISIBLE);
189                }else{
190                    mListaProductesCardRVAdapter.veureCheckBoxs();
191                    checkBoxsVisibles = true;
192                    btn.setVisibility(View.INVISIBLE);
193                    btn2.setVisibility(View.VISIBLE);
194                }
195                break;
196        }
197        return super.onOptionsItemSelected(item);
198    }
199
200    // Sirve para determinar a qué página vamos a retroceder
201    @Override
202    public void onBackPressed() {
203        // aquí defines la pantalla a la que se dirige el botón de retroceso
204        Intent intent = new Intent(VeureLlistaCompraActivity.this,
205            LlistaCompraActivity.class);
206        intent.putExtra("user", mailUser);
207        startActivity(intent);
208        finish();
209    }
}

```

Listing A.31: Configuració de la classe *VeureLlistaCompraActivity*.

```
1 package edu.ub.pis3.view;
2
3 import androidx.appcompat.app.AppCompatActivity;
4 import androidx.lifecycle.Observer;
5 import androidx.lifecycle.ViewModelProvider;
6
7 import android.graphics.Bitmap;
8 import android.graphics.BitmapFactory;
9 import android.os.Bundle;
10 import android.util.Base64;
11 import android.util.Log;
12 import android.util.Pair;
13 import android.view.View;
14 import android.widget.AdapterView;
15 import android.widget.ArrayAdapter;
16 import android.widget.ImageView;
17 import android.widget.Spinner;
18 import android.widget.TextView;
19 import android.widget.Toast;
20
21 import com.google.firebase.firestore.DocumentReference;
22 import com.google.firebase.firestore.FirebaseFirestore;
23
24 import java.util.ArrayList;
25
26 import edu.ub.pis3.R;
27 import edu.ub.pis3.model.InfoProducteRepository;
28 import edu.ub.pis3.model.InfoSuperProducte;
29 import edu.ub.pis3.model.LlistesCompra;
30 import edu.ub.pis3.model.LlistesCompraRepository;
31 import edu.ub.pis3.viewmodel.InfoProducteActivityViewModel;
32 import edu.ub.pis3.viewmodel.LlistesCompraActivityViewModel;
33
34 public class VeureDetailsProducteActivity extends AppCompatActivity {
35     private long id_producte;
36     private String nom_producte, photo;
37
38     private ArrayList<String> llistaSupers;
39     private Spinner spinner;
40     private ArrayList<SpinnerItem> mListSpinnerItems;
41     private SpinnerAdapter mAdapter;
42
43     private TextView text_preu;
44     private ImageView imageView;
45     private InfoProducteActivityViewModel infoProducteActivityViewModel;
46
```

```
47
48     @Override
49     protected void onCreate(Bundle savedInstanceState) {
50         super.onCreate(savedInstanceState);
51         setContentView(R.layout.activity_veure_detalls_producte);
52         spinner = (Spinner) findViewById(R.id.spinner_producte);
53         text_preu = (TextView) findViewById(R.id.ItemViewText);
54         imageView = (ImageView) findViewById(R.id.imageView);
55         llistaSupers = new ArrayList<>();
56         mListSpinnerItems = new ArrayList<>();
57
58
59         Bundle extras = getIntent().getExtras();
60         id_producte = extras.getLong("id_producte");
61         nom_producte = extras.getString("nom_producte");
62         getSupportActionBar().setTitle(nom_producte);
63         infoProducteActivityViewModel = new ViewModelProvider(this)
64             .get(InfoProducteActivityViewModel.class);
65
66         /* ****
67          * fotografia del nostre producte
68          * **** */
69         photo = extras.getString("photo");
70         // descriptem la foto des de base64
71         byte[] decodedBytes = Base64.decode(photo, Base64.DEFAULT);
72         Bitmap decodedBitmap = BitmapFactory.decodeByteArray(decodedBytes, 0,
73             decodedBytes.length);
74         // la línia més important, on fixem la imatge que volem mostrar
75         imageView.setImageBitmap(decodedBitmap);
76
77         final Observer<ArrayList<InfoSuperProducte>> observerInfoSuperProducte = new
78             Observer<ArrayList<InfoSuperProducte>>() {
79             @Override
80             public void onChanged(ArrayList<InfoSuperProducte> infoSuperProducts) {
81                 if(infoSuperProducts!=null){
82                     for(InfoSuperProducte i : infoSuperProducts){
83                         // Aquí hay que añadir la condición del spinner
84                         // Sería algo del estilo:
85                         // if(i.getAvailable().equals("1"))
86                         if(i.getNom_Super().equals("Escull un supermercat")){
87                             llistaSupers.add(i.getNom_Super());
88                             mListSpinnerItems.add(new SpinnerItem(i.getNom_Super()));
89                         }else{ //Si de vdd es un super hay que mirar si esta disponible
90                             if(i.getDisponible().equals("1")){
91                                 llistaSupers.add(i.getNom_Super());
92                                 mListSpinnerItems.add(new SpinnerItem(i.getNom_Super()));
93                             }
94                         }
95                     }
96                 }
97             }
98         };
99         observerInfoSuperProducte.observe(this);
100    }
```

```

91         }
92     }
93     mAdapter = new SpinnerAdapter(getApplicationContext(),
94         mListSpinnerItems);
95     spinner.setAdapter(mAdapter);
96
97     spinner.setOnItemSelectedListener(new
98         AdapterView.OnItemSelectedListener() {
99             @Override
100             public void onItemSelected(AdapterView<?> adapterView, View view,
101                 int i, long l) {
102                 String nom_super_spinner = ((SpinnerItem)
103                     adapterView.getItemAtPosition(i)).getTitle();
104                 for(InfoSuperProducte j : infoSuperProducts){
105                     if ((j.getNom_Super()).equals(nom_super_spinner)){
106                         String preu = j.getPreu();
107                         if(preu != null) preu += "€";
108                         text_preu.setText(preu);
109                     }
110                 }
111             }
112             @Override
113             public void onNothingSelected(AdapterView<?> adapterView) {
114                 Toast.makeText(VeureDetailsProducteActivity.this, "No és vàlid",
115                     Toast.LENGTH_SHORT).show();
116             }
117         });
118     } else {
119         text_preu.setText("Escull un supermercat");
120     }
121 };
122 infoProducteActivityViewModel.getInfoSuperProducte().observe(this,
123     observerInfoSuperProducte);
124 //Hay que cargar la info del producto de la base de datos.
125 infoProducteActivityViewModel.loadInfoProducteFromRepository(id_producte);
126 }
127 }

```

Listing A.32: Configuració de la classe *VeureDetailsProducteActivity*.

```
1 package edu.ub.pis3.view;
2 import java.util.*;
3 import android.content.*;
4 import android.widget.*;
5 import android.view.*;
6
7 import androidx.annotation.NonNull;
8 import androidx.recyclerview.widget.RecyclerView;
9
10 import edu.ub.pis3.R;
11
12 public class ListAdapter extends RecyclerView.Adapter<ListAdapter.ViewHolder> {
13     private ArrayList<ListItem> items;
14
15     public ListAdapter(ArrayList<ListItem> items) {
16         this.items = items;
17     }
18
19     @NonNull
20     @Override
21     public ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
22         View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item, parent,
23             false);
24         return new ViewHolder(view);
25     }
26
27     @Override
28     public void onBindViewHolder(@NonNull ViewHolder holder, int position) {
29         ListItem item = items.get(position);
30
31         holder.checkBox.setChecked(item.isChecked());
32         holder.textView.setText(item.getText());
33
34         holder.checkBox.setOnCheckedChangeListener(new
35             CompoundButton.OnCheckedChangeListener() {
36                 @Override
37                 public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
38                     item.setChecked(isChecked);
39                 }
40             });
41
42     @Override
43     public int getItemCount() {
44         return items.size();
45     }
```

```
45
46     public class ViewHolder extends RecyclerView.ViewHolder {
47         public CheckBox checkBox;
48         public TextView textView;
49
50         public ViewHolder(@NonNull View itemView) {
51             super(itemView);
52             checkBox = itemView.findViewById(R.id.checkBoxItem);
53             textView = itemView.findViewById(R.id.textViewItem);
54         }
55     }
56
57     private void removeAt(int position) {
58         items.remove(position);
59         notifyItemRemoved(position);
60         notifyDataSetChanged(position, items.size());
61     }
62 }
```

Listing A.33: Configuració de la classe *ListAdapter*.

```
1 package edu.ub.pis3.view;
2 import android.os.Parcel;
3 import android.os.Parcelable;
4 import androidx.annotation.NonNull;
5 import java.io.Serializable;
6
7 public class ListItem implements Serializable {
8     private String text;
9     private boolean checked;
10
11     public ListItem(String text, boolean checked) {
12         this.text = text;
13         this.checked = checked;
14     }
15
16     public String getText() {
17         return text;
18     }
19
20     public boolean isChecked() {
21         return checked;
22     }
23
24     public void setChecked(boolean checked) {
```

```
25     this.checked = checked;
26 }
27 }
```

Listing A.34: Configuració de la classe *ListItem*.

```
1 package edu.ub.pis3.view;
2
3 import android.view.LayoutInflater;
4 import android.view.View;
5 import android.view.ViewGroup;
6 import android.widget.ImageView;
7 import android.widget.TextView;
8
9 import androidx.annotation.NonNull;
10 import androidx.recyclerview.widget.RecyclerView;
11
12 import java.util.ArrayList;
13
14 import edu.ub.pis3.R;
15 import edu.ub.pis3.model.LlistesCompra;
16
17
18 public class LlistesCompraCardAdapter extends
19     RecyclerView.Adapter<LlistesCompraCardAdapter.ViewHolder> {
20
21     public interface OnClickGoListener {
22         void OnClickGo(String nomLlista);
23     }
24     public interface OnClickHideListener {
25         void OnClickHide(int position);
26     }
27
28     private ArrayList<LlistesCompra> mCompra;
29     private OnClickHideListener mOnClickHideListener;
30
31
32     public LlistesCompraCardAdapter(ArrayList<LlistesCompra> llistaCompra){
33         mCompra = llistaCompra;
34     }
35     public void setOnClickHideListener(OnClickHideListener listener) {
36         this.mOnClickHideListener = listener;
37     }
38 }
```

```
39     public void setOnClickGoListener(OnClickGoListener listener) {
40         this.mOnClickGoListener = listener;
41     }
42
43     public void setmCompra(ArrayList<LlistesCompra> mCompra) {
44         this.mCompra = mCompra;
45         notifyDataSetChanged();
46     }
47
48     @Override
49     public int getItemCount() {
50         return mCompra.size();
51     }
52
53     public void hideLista(int position) {
54         notifyItemRemoved(position);
55     }
56
57     @NonNull
58     @Override
59     public LlistesCompraCardAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent,
60         int viewType){
61         LayoutInflater inflater = LayoutInflater.from(parent.getContext());
62         View view = inflater.inflate(R.layout.llista_compra_card, parent, false);
63         return new LlistesCompraCardAdapter.ViewHolder(view);
64     }
65
66     @Override
67     public void onBindViewHolder(@NonNull LlistesCompraCardAdapter.ViewHolder holder, int
68         position){
69         holder.bind(mCompra.get(position), this.mOnClickGoListener,
70             this.mOnClickHideListener);
71     }
72
73     public static class ViewHolder extends RecyclerView.ViewHolder{
74         private final ImageView mImage;
75         private final TextView mName;
76         private final ImageView mHideButton;
77
78         public ViewHolder(@NonNull View itemView){
79             super(itemView);
80             mImage = itemView.findViewById(R.id.avatar);
81             mName = itemView.findViewById(R.id.fullname);
82             mHideButton = itemView.findViewById(R.id.hide_btn);
83         }
84     }
85 }
```

```

82     public void bind(final LlistesCompra llistesCompra, OnClickGoListener listener,
83         ↳ OnClickHideListener listener2){
84         mName.setText(llistesCompra.getNom());
85         //Picasso.get().load(llistesCompra.getURL()).into(mImage);
86         itemView.findViewById(R.id.card).setOnClickListener(new View.OnClickListener() {
87             @Override
88             public void onClick(View view) {
89                 listener.OnClickGo(mName.getText().toString());
90             }
91         );
92         mHideButton.setOnClickListener(new View.OnClickListener() {
93             @Override
94             public void onClick(View view) {
95                 listener2.OnClickHide(getAdapterPosition());
96             }
97         );
98     }
99 }
```

Listing A.35: Configuració de la classe *LlistaCompraCardAdapter*.

```

1 package edu.ub.pis3.view;
2
3 import android.view.LayoutInflater;
4 import android.view.View;
5 import android.view.ViewGroup;
6 import android.widget.CheckBox;
7 import android.widget.CompoundButton;
8 import android.widget.ImageView;
9 import android.widget.CompoundButton.OnCheckedChangeListener;
10 import android.widget.TextView;
11
12 import androidx.annotation.NonNull;
13 import androidx.recyclerview.widget.RecyclerView;
14
15 import java.util.ArrayList;
16
17 import edu.ub.pis3.R;
18 import edu.ub.pis3.model.LlistesCompra;
19 import edu.ub.pis3.model.Product;
20 import edu.ub.pis3.view.LlistesCompraCardAdapter;
21
22 public class LlistaProductesCardAdapter extends
23     ↳ RecyclerView.Adapter<LlistaProductesCardAdapter.ViewHolder>{
```

```
23
24     public interface OnClickGoListener {
25         void OnClickGo(String nomProducte, Long id_producte, String photo);
26     }
27
28     public interface AfterCheckedHide {
29         void AfterCheckedHide(int position);
30         void AfterUnchecked(int position);
31     }
32     private ArrayList<Product> mProducts;
33     private LlistaProductesCardAdapter.OnClickGoListener mOnClickGoListener;
34     private AfterCheckedHide mAFTERCHECKEDHIDE;
35     private ViewHolder viewHoder;
36     private ArrayList<ViewHolder> listViewHolder;
37
38     public LlistaProductesCardAdapter(ArrayList<Product> llistaProducts){
39         mProducts = llistaProducts;
40         listViewHolder = new ArrayList<>();
41     }
42
43     public void setOnClickGoListener(LlistaProductesCardAdapter.OnClickGoListener listener)
44     {
45         this.mOnClickGoListener = listener;
46     }
47     public void setAfterCheckedHide(AfterCheckedHide listener){
48         this.mAfterCheckedHide = listener;
49     }
50
51     public void setmProducts(ArrayList<Product> mProducts) {
52         this.mProducts = mProducts;
53         notifyDataSetChanged();
54     }
55
56     public void veureCheckBoxs(){
57         for(ViewHolder v : listViewHolder){
58             v.veureCheckboxes();
59         }
60     }
61
62     public void amagarCheckBoxs(){
63         for(ViewHolder v : listViewHolder){
64             v.amagarCheckboxes();
65         }
66     }
67
68     public void hideProduct(int position) {
```

```
68     notifyItemRemoved(position);
69 }
70
71 @Override
72 public int getItemCount() {
73     return mProducts.size();
74 }
75
76 @NonNull
77 @Override
78 public LlistaProductesCardAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup
    parent, int viewType){
79     LayoutInflater inflater = LayoutInflater.from(parent.getContext());
80     View view = inflater.inflate(R.layout.llista_productes_card, parent, false);
81     viewHolder = new LlistaProductesCardAdapter.ViewHolder(view);
82     listViewHolder.add(viewHolder);
83     return viewHolder;
84 }
85
86 @Override
87 public void onBindViewHolder(@NonNull LlistaProductesCardAdapter.ViewHolder holder, int
    position){
88     holder.bind(mProducts.get(position), this.mOnClickGoListener,
    this.mAfterCheckedHide);
89 }
90
91 public static class ViewHolder extends RecyclerView.ViewHolder{
92     private final ImageView mImage;
93
94     private final CheckBox checkBoxEliminarProducte;
95     private final TextView mName;
96
97     public ViewHolder(@NonNull View itemView){
98         super(itemView);
99         mImage = itemView.findViewById(R.id.guion);
100        mName = itemView.findViewById(R.id.nom_producte);
101        checkBoxEliminarProducte = itemView.findViewById(R.id.checkBoxProducteEliminar);
102    }
103
104    public void bind(final Product product, OnClickGoListener listener, AfterCheckedHide
    listener2){
105        mName.setText(product.getNom());
106
107        itemView.findViewById(R.id.cardProducte).setOnClickListener(new
    View.OnClickListener() {
108            @Override
```

```

109         public void onClick(View view) {
110             listener.OnClickGo(product.getNom(), product.getId(),
111             → product.getPhoto());
112         }
113     });
114
115     checkBoxEliminarProducte.setOnCheckedChangeListener(new OnCheckedChangeListener
116     → (){
117         @Override
118         public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
119             // b is true only if check
120             listener2.AfterCheckedHide(getAdapterPosition());
121             /*
122             if(b) {
123                 listener2.AfterCheckedHide(getAdapterPosition());
124             }else{
125                 listener2.AfterUnchecked(getAdapterPosition());
126             }*/
127         }
128     });
129
130     public void veureCheckboxes(){
131         mImage.setVisibility(View.INVISIBLE);
132         checkBoxEliminarProducte.setVisibility(View.VISIBLE);
133     }
134
135     public void amagarCheckBoxes(){
136         mImage.setVisibility(View.VISIBLE);
137         checkBoxEliminarProducte.setVisibility(View.INVISIBLE);
138     }
139 }
```

Listing A.36: Configuració de la classe *LlistaProductesCardAdapter*.

```

1 package edu.ub.pis3.view;
2
3 import android.view.LayoutInflater;
4 import android.view.View;
5 import android.view.ViewGroup;
6 import android.widget.CheckBox;
7 import android.widget.CompoundButton;
8 import android.widget.Filter;
9 import android.widget.Filterable;
```

```
10 import android.widget.TextView;
11 import android.widget.CompoundButton.OnCheckedChangeListener;
12
13 import androidx.annotation.NonNull;
14 import androidx.recyclerview.widget.RecyclerView;
15
16 import org.checkerframework.checker.units.qual.A;
17
18 import java.util.ArrayList;
19 import java.util.List;
20
21 import edu.ub.pis3.R;
22 import edu.ub.pis3.model.Product;
23
24 public class LlistaAllProductsCardAdapter extends
25     RecyclerView.Adapter<LlistaAllProductsCardAdapter.ViewHolder> implements Filterable {
26
27     private ArrayList<Product> mProduct;
28     private ArrayList<Product> mAllProductFull;
29     private ArrayList<String> mCurrentProducts;
30     private ArrayList<String> mCurrentCheckedProducts;
31     //private boolean beenFiltered;
32
33     public LlistaAllProductsCardAdapter(ArrayList<Product> llistaProductes,
34         ArrayList<String> mCurrentProducts){
35
36         mProduct = new ArrayList<>();
37         mProduct.addAll(llistaProductes);
38         mAllProductFull = new ArrayList<>();
39         this.mCurrentProducts = mCurrentProducts;
40         mCurrentCheckedProducts = new ArrayList<>();
41     }
42
43
44
45     @NonNull
46     @Override
47     public LlistaAllProductsCardAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup
48         parent, int viewType) {
49
50         LayoutInflator inflater = LayoutInflator.from(parent.getContext());
51         View view = inflater.inflate(R.layout.item_list_check, parent, false);
52         return new LlistaAllProductsCardAdapter.ViewHolder(view);
53     }
54 }
```

```
53     @Override
54     public void onBindViewHolder(@NonNull LlistaAllProductsCardAdapter.ViewHolder holder,
55         int position) {
56         holder.bind(mProduct.get(position).getNom(), mCurrentProducts,
57             mCurrentCheckedProducts);
58     }
59
60     @Override
61     public int getItemCount() {
62         return mProduct.size();
63     }
64
65     @Override
66     public Filter getFilter() {
67         return exampleFilter;
68     }
69
70     private Filter exampleFilter = new Filter() {
71         @Override
72         protected FilterResults performFiltering(CharSequence charSequence) {
73             ArrayList<Product> filteredList = new ArrayList<>();
74             if (charSequence == null || charSequence.length() == 0){
75                 filteredList.addAll(mAllProductFull);
76             } else{
77                 String filterPattern = charSequence.toString().toLowerCase().trim();
78                 for (Product p : mAllProductFull){
79                     if(p.getNom().toLowerCase().contains(filterPattern)){
80                         filteredList.add(p);
81                     }
82                 }
83             }
84             FilterResults results = new FilterResults();
85             results.values = filteredList;
86             return results;
87         }
88
89         @Override
90         protected void publishResults(CharSequence charSequence, FilterResults
91             filterResults) {
92             mProduct.clear();
93             mProduct.addAll((List)filterResults.values);
94             notifyDataSetChanged();
95         };
96
97         public static class ViewHolder extends RecyclerView.ViewHolder{
```

```
96     private CheckBox checkBox;
97     private final TextView mNameProduct;
98
99     private boolean checkedDiferent;
100    public ViewHolder(@NonNull View itemView){
101        super(itemView);
102        checkBox = itemView.findViewById(R.id.checkBox);
103        mNameProduct = itemView.findViewById(R.id.textViewcheckBox);
104        checkedDiferent = false;
105    }
106
107    public void bind(String nomProducte, ArrayList<String> mCurrentLlistaProductes,
108                     ArrayList<String> mCurrentCheckedProducts){
109        mNameProduct.setText(nomProducte);
110        boolean pInList = mCurrentCheckedProducts.contains(nomProducte);
111        /* Lo que pasa es que el RV recicla tmbn los checkboxes como le da la gana.
112         * Entonces, a veces el checkbox de un producto pasa a ser de otro.
113         * Si el checkBox es diferente de si el producto esta en mi lista de checked, es
114         * decir,
115         * si por cosas internas del RV el check a cambiado pero en realidad no le ha
116         * dado al check.
117         * En este caso utilizo la variable boolean checkedDiferent para despues en el
118         * onCheckedChanged saber que estoy en ese caso y ignorarlo.
119         * Otra cosa: el ^ del if es una xor pq solo quiero que me entre al if si
120         * son diferentes.
121         */
122        if(pInList ^ checkBox.isChecked()){
123            checkedDiferent = true;
124            checkBox.setChecked(pInList);
125        } else {
126            checkedDiferent = false;
127        }
128        checkBox.setOnCheckedChangeListener(new OnCheckedChangeListener() {
129            @Override
130            public void onCheckedChanged(CompoundButton compoundButton, boolean b) {
131                // b is true only if checkbox is checked
132                if(!checkedDiferent){
133                    if(b){
134                        if(!mCurrentLlistaProductes.contains(nomProducte)){
135                            mCurrentLlistaProductes.add(nomProducte);
136                            mCurrentCheckedProducts.add(nomProducte);
137                        }
138                    } else {
139                        mCurrentLlistaProductes.remove(nomProducte);
140                        mCurrentCheckedProducts.remove(nomProducte);
141                    }
142                }
143            }
144        });
145    }
146}
```

```

139
140
141
142
143     itemView.setOnClickListener(new View.OnClickListener() {
144         @Override
145         public void onClick(View view) {
146             if(!checkBox.isChecked()){ //Si estaba sin check
147                 checkBox.setChecked(true); //pongo el check
148             } else {
149                 checkBox.setChecked(false);
150             }
151         }
152     });
153 }
154 }
155 }
```

Listing A.37: Configuració de la classe *LlistaAllProductsCardAdapter*.

```

1 package edu.ub.pis3.view;
2
3 import android.view.LayoutInflater;
4 import android.view.View;
5 import android.view.ViewGroup;
6 import android.widget.ImageView;
7 import android.widget.TextView;
8
9 import androidx.annotation.NonNull;
10 import androidx.recyclerview.widget.RecyclerView;
11
12 import java.util.ArrayList;
13
14 import edu.ub.pis3.R;
15 import edu.ub.pis3.model.LlistesCompra;
16
17 public class LlistaSupermercatCardAdapter extends
18     RecyclerView.Adapter<LlistaSupermercatCardAdapter.ViewHolder>{
19
20     private final String TAG = "LlistaSupermercatCardAdapter";
21     private ArrayList<String> mSupermercats;
22     private OnClickGoListener mOnClickGoListener;
23     public interface OnClickGoListener {
24         void OnClickGo(String nomSuper);
25     }
```

```
25
26     public void setOnClickGoListener(OnClickGoListener listener) {
27         this.mOnClickGoListener = listener;
28     }
29
30     public LlistaSupermercatCardAdapter(ArrayList<String> llistaSupermercats){
31         mSupermercats = llistaSupermercats;
32     }
33
34     public void setmSupermercats(ArrayList<String> mSupermercats) {
35         this.mSupermercats = mSupermercats;
36         notifyDataSetChanged();
37     }
38
39     @Override
40     public int getItemCount() {return mSupermercats.size();}
41
42     @NonNull
43     @Override
44     public LlistaSupermercatCardAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup
45         → parent, int viewType){
46         LayoutInflater inflater = LayoutInflater.from(parent.getContext());
47         View view = inflater.inflate(R.layout.llista_supermercat_card, parent, false);
48         return new LlistaSupermercatCardAdapter.ViewHolder(view);
49     }
50
51     @Override
52     public void onBindViewHolder(@NonNull LlistaSupermercatCardAdapter.ViewHolder holder,
53         → int position){
54         holder.bind(mSupermercats.get(position), this.mOnClickGoListener);
55     }
56
57     public static class ViewHolder extends RecyclerView.ViewHolder{
58         private final ImageView mImage;
59         private final TextView mName;
60
61         public ViewHolder(@NonNull View itemView){
62             super(itemView);
63             mImage = itemView.findViewById(R.id.avatar);
64             mName = itemView.findViewById(R.id.nom_supermercat);
65         }
66
67         public void bind(final String nomSupermercat, OnClickGoListener listener){
68             mName.setText(nomSupermercat);
69             //Picasso.get().load(llistesCompra.getURL()).into(mImage);
70             itemView.findViewById(R.id.card_supermercat).setOnClickListener(new
71                 → View.OnClickListener() {
```

```
69         @Override
70         public void onClick(View view) {
71             listener.OnClickGo(nomSupermercat);
72         }
73     });
74 }
75 }
76 }
```

Listing A.38: Configuració de la classe *LlistaSupermercatCardAdapter*.

```
1 package edu.ub.pis3.view;
2
3 import android.view.LayoutInflater;
4 import android.view.View;
5 import android.view.ViewGroup;
6 import android.widget.CheckBox;
7 import android.widget.ImageView;
8 import android.widget.TextView;
9
10 import androidx.annotation.NonNull;
11 import androidx.recyclerview.widget.RecyclerView;
12
13 import java.util.ArrayList;
14
15 import edu.ub.pis3.R;
16 import edu.ub.pis3.model.Product;
17
18 public class ProductesAComprarCardAdapter extends
19     RecyclerView.Adapter<ProductesAComprarCardAdapter.ViewHolder>{
20
21     private final String TAG = "ProductesAComprarCardAdapter";
22     private ArrayList<String> mProducts;
23
24     public ProductesAComprarCardAdapter(ArrayList<String> llistaProducts){
25         mProducts = llistaProducts;
26     }
27
28     public void setmProducts(ArrayList<String> mProducts) {
29         this.mProducts = mProducts;
30         notifyDataSetChanged();
31     }
32
33     @Override
34     public int getItemCount() {
```

```

34     return mProducts.size();
35 }
36
37 @NonNull
38 @Override
39 public ProductesAComprarCardAdapter.ViewHolder onCreateViewHolder(@NonNull ViewGroup
40     parent, int viewType){
41     LayoutInflater inflater = LayoutInflater.from(parent.getContext());
42     View view = inflater.inflate(R.layout.llista_producte_cada_supermercat_card, parent,
43     false);
44     return new ProductesAComprarCardAdapter.ViewHolder(view);
45 }
46
47 @Override
48 public void onBindViewHolder(@NonNull ProductesAComprarCardAdapter.ViewHolder holder,
49     int position){
50     holder.bind(mProducts.get(position));
51 }
52
53 public static class ViewHolder extends RecyclerView.ViewHolder{
54     private final CheckBox checkBox;
55     private final ImageView guion;
56     private final TextView mName;
57
58     public ViewHolder(@NonNull View itemView){
59         super(itemView);
60         mName = itemView.findViewById(R.id.nom_producte_fragment);
61         checkBox = itemView.findViewById(R.id.checkBox_producte_fragment);
62         guion = itemView.findViewById(R.id.guion33);
63     }
64
65     public void bind(final String nomProducte){
66         mName.setText(nomProducte);
67         if(mName.getText().toString().equals("No hi ha productes en aquesta llista")){
68             checkBox.setVisibility(View.INVISIBLE);
69             guion.setVisibility(View.VISIBLE);
70         }else{
71             checkBox.setVisibility(View.VISIBLE);
72             guion.setVisibility(View.INVISIBLE);
73         }
74     }
75 }
76 }

```

Listing A.39: Configuració de la classe *ProductesAComprarCardAdapter*.

```
1 package edu.ub.pis3.view;  
2  
3 public class SpinnerItem {  
4     private String title;  
5  
6     public SpinnerItem(String _title){  
7         this.title = _title;  
8     }  
9  
10    public String getTitle(){  
11        return this.title;  
12    }  
13}
```

Listing A.40: Configuració de la classe *SpinnerItem*.

```
1 package edu.ub.pis3.view;  
2  
3 import android.app.Activity;  
4 import android.content.Context;  
5 import android.support.annotation.NonNull;  
6 import android.support.annotation.Nullable;  
7 import android.util.Log;  
8 import android.view.LayoutInflater;  
9 import android.view.View;  
10 import android.view.ViewGroup;  
11 import android.widget.ArrayAdapter;  
12 import android.widget.BaseAdapter;  
13 import android.widget.TextView;  
14  
15 import java.util.ArrayList;  
16  
17 import edu.ub.pis3.R;  
18  
19 public class SpinnerAdapter extends ArrayAdapter<SpinnerItem> {  
20  
21     public SpinnerAdapter(Context context, ArrayList<SpinnerItem> items) {  
22         super(context, 0, items);  
23     }  
24  
25     //SEGUIR GIT  
26     //https://gist.github.com/codinginflow/8c3863e4b06311873162ef7f512228eb  
27  
28  
29}
```

```

30     @NonNull
31     @Override
32     public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent)
33     {
34         return initView(position, convertView, parent);
35     }
36
37     @Override
38     public View getDropDownView(int position, @Nullable View convertView, @NonNull ViewGroup
39     parent) {
40         return initView(position, convertView, parent);
41     }
42
43     private View initView(int position, View convertView, ViewGroup parent) {
44         if (convertView == null) {
45             convertView = LayoutInflater.from(getContext()).inflate(
46                 R.layout.spinner_item, parent, false
47             );
48         }
49
50         TextView textViewName = convertView.findViewById(R.id.textViewSpinner);
51
52         SpinnerItem currentItem = getItem(position);
53
54         if (currentItem != null) {
55             textViewName.setText(currentItem.getTitle());
56         }
57
58         return convertView;
59     }
60 }
```

Listing A.41: Configuració de la classe *SpinnerAdapter*.

```

1 package edu.ub.pis3.view;
2
3 import androidx.fragment.app.DialogFragment;
4 import android.app.AlertDialog;
5 import android.app.Dialog;
6 import android.content.DialogInterface;
7 import android.view.LayoutInflater;
8 import android.view.View;
9 import android.view.ViewGroup;
10 import android.widget.ImageView;
11 import android.widget.TextView;
```

```
12  
13 import androidx.annotation.NonNull;  
14 import android.os.Bundle;  
15 import androidx.recyclerview.widget.LinearLayoutManager;  
16  
17 import edu.ub.pis3.R;  
18 import edu.ub.pis3.model.LlistesCompra;  
19  
20 public class DialogEliminarLlista extends DialogFragment {  
21  
22     private int position;  
23     private String mailUser;  
24     private LlistaCompraActivity llista;  
25     public Dialog onCreateDialog(Bundle savedInstanceState) {  
26         AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
27         LayoutInflater inflater = requireActivity().getLayoutInflater();  
28  
29         builder.setView(inflater.inflate(R.layout.dialog_eliminar_llista, null))  
30             .setNegativeButton(R.string.button_no_eliminar, new  
31                 DialogInterface.OnClickListener() {  
32                     @Override  
33                     public void onClick(DialogInterface dialogInterface, int i) {  
34                         DialogEliminarLlista.this.getDialog().cancel();  
35                     }  
36                 })  
37             .setPositiveButton(R.string.button_si_eliminar, new  
38                 DialogInterface.OnClickListener() {  
39                     @Override  
40                     public void onClick(DialogInterface dialogInterface, int i) {  
41                         llista.eliminarLlista(getPosition());  
42                     }  
43                 });  
44  
45         return builder.create();  
46     }  
47  
48     public void setData(int position, String mailUser, LlistaCompraActivity llista){  
49         this.position = position;  
50         this.mailUser = mailUser;  
51         this.llista = llista;  
52     }  
53  
54     public int getPosition(){  
55         return this.position;  
56     }
```

55}

Listing A.42: Configuració de la classe *DialogEliminarLlista*.

```
1 package edu.ub.pis3.view;
2
3 import edu.ub.pis3.model.InfoSuperProducte;
4 import edu.ub.pis3.model.LlistesCompra;
5 import edu.ub.pis3.viewmodel.CalculadoraActivityViewModel;
6
7 import android.content.Intent;
8 import android.icu.text.IDNA;
9 import android.net.Uri;
10 import android.os.Handler;
11 import android.support.annotation.NonNull;
12 import android.view.LayoutInflater;
13 import android.view.View;
14 import android.view.ViewGroup;
15 import android.os.Bundle;
16 import android.widget.EditText;
17 import android.widget.ProgressBar;
18
19 import androidx.fragment.app.FragmentManager;
20 import androidx.fragment.app.FragmentTransaction;
21 import androidx.lifecycle.Observer;
22 import androidx.lifecycle.ViewModelProvider;
23 import androidx.recyclerview.widget.LinearLayoutManager;
24 import androidx.recyclerview.widget.RecyclerView;
25
26 import com.google.android.material.floatingactionbutton.FloatingActionButton;
27
28 import java.util.ArrayList;
29 import java.util.HashMap;
30
31 import edu.ub.pis3.R;
32 import edu.ub.pis3.viewmodel.LlistesCompraActivityViewModel;
33
34 public class FragmentEscolhirLlista extends androidx.fragment.app.Fragment{
35
36     private final String TAG = "FragmentEscolhirLlista";
37     private CalculadoraActivityViewModel calculadoraActivityViewModel;
38
39     private RecyclerView mListaCardsRV;
40
41     private LlistaSupermercatCardAdapter mListaCardRVAdapter;
```

```
42  
43     private ArrayList<String> llistaSupers;  
44     private ArrayList<String> llistaNomProductes;  
45     private boolean matrixLoaded;  
46  
47     private ProgressBar progressBar;  
48  
49     public FragmentEscollidirLlista(){  
50  
51     @Override  
52     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle  
→     savedInstanceState){  
53         return inflater.inflate(R.layout.fragment_escollidir_llista, container, false);  
54     }  
55  
56     @Override  
57     public void onViewCreated(@NonNull View view, @NonNull Bundle savedInstanceState){  
58         super.onViewCreated(view, savedInstanceState);  
59         //getSupportActionBar().setTitle("Llistes de la compra");  
60         getActivity().setTitle("Quina llista vols veure?");  
61         matrixLoaded = false;  
62         Bundle args = getArguments();  
63         ArrayList<ListItem> items = new ArrayList<>();  
64         llistaNomProductes = args.getStringArrayList("llistaNomProductes");  
65         items = (ArrayList<ListItem>) args.get("llistaSupermercats");  
66  
67         llistaSupers = new ArrayList<>();  
68         progressBar = (ProgressBar)  
→         view.findViewById(R.id.progressBarFragmentEscollidirLlista);  
69         progressBar.setVisibility(View.GONE);  
70  
71         for(ListItem item: items){  
72             llistaSupers.add(item.getText());  
73         }  
74  
75         calculadoraActivityViewModel = new ViewModelProvider(this)  
76             .get(CalculadoraActivityViewModel.class);  
77  
78         mListaCardsRV = view.findViewById(R.id.llistaCardRV_fragment);  
79  
80         mListaCardsRV.setLayoutManager(new LinearLayoutManager(view.getContext()));  
81         //mListaCardRVAdapter = new LlistaSupermercatCardAdap-  
→         ter(calculadoraActivityViewModel.getmSupermercats().getValue());  
82         mListaCardRVAdapter = new LlistaSupermercatCardAdapter(llistaSupers);  
83  
84         mListaCardRVAdapter.setOnClickListener(new  
→         LlistaSupermercatCardAdapter.OnClickListener() {
```

```
85     @Override
86     public void OnClickGo(String nomSuper) {
87         if(!matrixLoaded) {
88             progressBar.setVisibility(View.VISIBLE);
89             Handler handler = new Handler();
90             handler.postDelayed(new Runnable() {
91                 @Override
92                 public void run() {
93                     progressBar.setVisibility(View.GONE);
94                     matrixLoaded = true;
95                     OnClickGo(nomSuper);
96                 }
97             }, 1000);
98         }
99         if(matrixLoaded){
100             Bundle bundle = new Bundle();
101             HashMap<String, ArrayList<String>> map = new HashMap<>();
102             map = calculadoraActivityViewModel.calcularPreu(llistaSupers,
103             ↳ llistaNomProductes,
104             calculadoraActivityViewModel.getMatrix().getValue());
105
106             ArrayList<String> llistaProductesSuper = map.get(nomSuper);
107             bundle.putStringArrayList("products", llistaProductesSuper);
108
109             FragmentManager fragmentManager = getParentFragmentManager();
110             FragmentTransaction fragmentTransaction =
111             ↳ fragmentManager.beginTransaction();
112             FragmentVeureProductes fragment = new FragmentVeureProductes();
113             // Set the created Bundle as the arguments for the receiving fragment
114             fragmentTransaction.addToBackStack(null);
115             fragment.setArguments(bundle);
116             fragmentTransaction.replace(R.id.fragment_veure_productes_llista,
117             ↳ fragment);
118             fragmentTransaction.commit();
119         }
120     });
121
122     mLlistaCardsRV.setAdapter(mLlistaCardRVAdapter);
123
124     //calculadoraActivityViewModel.loadAllInfoSupersProductes();
125
126     final Observer<HashMap<String, ArrayList<InfoSuperProducte>>> observerMatrix = new
127     ↳ Observer<HashMap<String, ArrayList<InfoSuperProducte>>>() {
128         @Override
```

```

127     public void onChanged(HashMap<String, ArrayList<InfoSuperProducte>>
128         → stringArrayListHashMap) {
129         if(stringArrayListHashMap!=null && !stringArrayListHashMap.isEmpty()){
130             matrixLoaded = true;
131         }
132     };
133     calculadoraActivityViewModel.getMatrix().observe(getViewLifecycleOwner(),
134         → observerMatrix);
135     calculadoraActivityViewModel.loadInfoMatrix();
136     //ArrayList<String> llistaSupermercats = (ArrayList<String>)
137     //    → args.get("llistaSupermercats");*/
138     //calculadoraActivityViewModel.setmSupermercats(llistaSupermercats);
139
140     FragmentManager fragmentManager = getParentFragmentManager();
141     FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
}

```

Listing A.43: Configuració de la classe *FragmentEscolhirLlista*.

```

1 package edu.ub.pis3.view;
2
3 import android.content.Intent;
4 import android.os.Bundle;
5 import android.view.LayoutInflater;
6 import android.view.View;
7 import android.view.ViewGroup;
8 import android.widget.Button;
9 import androidx.annotation.NonNull;
10 import androidx.annotation.Nullable;
11 import androidx.fragment.app.Fragment;
12 import androidx.fragment.app.FragmentManager;
13 import androidx.fragment.app.FragmentTransaction;
14 import androidx.lifecycle.Observer;
15 import androidx.lifecycle.ViewModelProvider;
16 import androidx.recyclerview.widget.LinearLayoutManager;
17 import androidx.recyclerview.widget.RecyclerView;
18 import java.util.ArrayList;
19 import edu.ub.pis3.R;
20 import edu.ub.pis3.model.LlistesCompra;
21 import edu.ub.pis3.viewmodel.CalculadoraActivityViewModel;
22
23 public class FragmentSupermercats extends Fragment {
24

```

```
25 CalculadoraActivityViewModel calculadoraActivityViewModel;
26 ListAdapter adapter;
27 ArrayList<ListItem> items;
28
29 public FragmentSupermercats() {
30 }
31
32 @Override
33 public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
→ savedInstanceState) {
34     return inflater.inflate(R.layout.fragment_calcul_llista_supers, container, false);
35 }
36
37 @Override
38 public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
39     super.onViewCreated(view, savedInstanceState);
40     calculadoraActivityViewModel = new ViewModelProvider(this)
41         .get(CalculadoraActivityViewModel.class);
42
43     getActivity().setTitle("Escull supermercats");
44
45     Bundle args = getArguments();
46     ArrayList<String> llistaNomProductes = args.getStringArrayList("products");
47     items = new ArrayList<>();
48
49     final Button button = view.findViewById(R.id.button_supers);
50     button.setOnClickListener(new View.OnClickListener() {
51         public void onClick(View v) {
52             ArrayList<ListItem> checked_supers = new ArrayList<>();
53             for (ListItem item : items) {
54                 if (item.isChecked()) {
55                     checked_supers.add(item);
56                 }
57                 if (!item.isChecked() && checked_supers.contains(item)) {
58                     checked_supers.remove(item);
59                 }
60             }
61
62             Bundle bundle = new Bundle();
63             bundle.putSerializable("llistaSupermercats", checked_supers);
64             bundle.putStringArrayList("llistaNomProductes", llistaNomProductes);
65
66             FragmentEscolhirLlista fragment = new FragmentEscolhirLlista();
67
68             // Set the created Bundle as the arguments for the receiving fragment
69             fragment.setArguments(bundle);
```

```

70         fragmentManager = getParentFragmentManager();
71         FragmentTransaction fragmentTransaction =
72             → fragmentManager.beginTransaction();
73             fragmentTransaction.addToBackStack(null);
74
75             fragmentTransaction.replace(R.id.fragment_escollir_llista, fragment);
76             fragmentTransaction.commit();
77         });
78
79     final Observer<ArrayList<String>> observerSupers = new Observer<ArrayList<String>>()
80     → {
81         @Override
82         public void onChanged(ArrayList<String> strings) {
83             if(strings != null & strings.size()!=0){
84                 ArrayList<String> supers = strings;
85                 supers.remove("Escull un supermercat");
86
87                 for(String supermercat : supers) {
88                     items.add(new ListItem(supermercat,false));
89                 }
90
91                 RecyclerView recyclerView = view.findViewById(R.id.llistaSupers);
92                 adapter = new ListAdapter(items);
93                 recyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
94                 recyclerView.setAdapter(adapter);
95             }
96         };
97         calculadoraActivityViewModel.getAllSupers().observe(getViewLifecycleOwner(),
98             → observerSupers);
99         calculadoraActivityViewModel.loadSupersFromRepository();
100    }
}

```

Listing A.44: Configuració de la classe *FragmentSupermercats*.

```

1 package edu.ub.pis3.view;
2
3 import android.support.annotation.NonNull;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7 import android.os.Bundle;
8

```

```
9 import androidx.fragment.app.FragmentManager;
10 import androidx.fragment.app.FragmentTransaction;
11 import androidx.lifecycle.ViewModelProvider;
12 import androidx.recyclerview.widget.LinearLayoutManager;
13 import androidx.recyclerview.widget.RecyclerView;
14
15 import java.util.ArrayList;
16
17 import edu.ub.pis3.R;
18 import edu.ub.pis3.viewmodel.CalculadoraActivityViewModel;
19
20 public class FragmentVeureProductes extends androidx.fragment.app.Fragment {
21
22     private final String TAG = "FragmentVeureProductes";
23
24     private CalculadoraActivityViewModel calculadoraActivityViewModel;
25     private RecyclerView mListaCardsRV;
26     private ProductesAComprarCardAdapter mListaCardRVAdapter;
27     ArrayList<String> llistaNomsProductes;
28     public FragmentVeureProductes(){}
29
30     @Override
31     public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
32     → savedInstanceState){
33         return inflater.inflate(R.layout.fragment_veure_productes, container, false);
34     }
35
36     @Override
37     public void onViewCreated(@NonNull View view, @NonNull Bundle savedInstanceState){
38         super.onViewCreated(view, savedInstanceState);
39         getActivity().setTitle("Llista de la compra");
40
41         Bundle args = getArguments();
42         calculadoraActivityViewModel = new ViewModelProvider(this)
43             .get(CalculadoraActivityViewModel.class);
44
45         mListaCardsRV = view.findViewById(R.id.listProductsToBuyRV_fragment);
46
47         mListaCardsRV.setLayoutManager(new LinearLayoutManager(view.getContext()));
48         llistaNomsProductes = args.getStringArrayList("products");
49
50         if(llistaNomsProductes==null) {
51             llistaNomsProductes = new ArrayList<>();
52             llistaNomsProductes.add("No hi ha productes en aquesta llista");
53         }
54         mListaCardRVAdapter = new ProductesAComprarCardAdapter(llistaNomsProductes);
```

```

54     mLlistaCardsRV.setAdapter(mLlistaCardRVAdapter);
55
56
57     FragmentManager fragmentManager = getParentFragmentManager();
58     FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
59     FragmentVeureProductes fragment = new FragmentVeureProductes();
60     fragmentTransaction.addToBackStack(null);
61 }
62 }
```

Listing A.45: Configuració de la classe *FragmentVeureProductes*.

A.4 CODIS DEL VIEWMODEL

```

1 package edu.ub.pis3.viewmodel;
2
3 import android.app.Application;
4
5 import androidx.lifecycle.AndroidViewModel;
6 import androidx.lifecycle.MutableLiveData;
7
8 import java.util.ArrayList;
9
10 import edu.ub.pis3.model.LlistesCompra;
11 import edu.ub.pis3.model.LlistesCompraRepository;
12 import edu.ub.pis3.model.Product;
13
14 public class LlistesCompraActivityViewModel extends AndroidViewModel {
15
16     private final String TAG = "LlistesCompraActivityViewModel";
17     private final MutableLiveData<ArrayList<LlistesCompra>> mCompra;
18     private LlistesCompraRepository mLlistesCompraRepository;
19
20     public LlistesCompraActivityViewModel(Application application){
21         super(application);
22         mCompra = new MutableLiveData<>(new ArrayList<>());
23         mLlistesCompraRepository = LlistesCompraRepository.getInstance();
24
25         mLlistesCompraRepository.addOnLoadLlistesCompraListener(new
26             → LlistesCompraRepository.OnLoadLlistesCompraListener(){
27                 @Override
28                 public void onLoadLlistesCompra(ArrayList<LlistesCompra> llistes) {
```

```
28         LlistesCompraViewModel.this.setLlistesCompra(llistes);
29     }
30 }
31 }
32
33 public MutableLiveData<ArrayList<LlistesCompra>> getmCompra() {
34     return mCompra;
35 }
36
37 public void setLlistesCompra(ArrayList<LlistesCompra> llista){
38     mCompra.setValue(llista);
39 }
40
41 public void loadLlistesCompraFromRepository(String mailUser){
42     mLlistesCompraRepository.loadLlistesCompra(mCompra.getValue(), mailUser);
43 }
44
45 public ArrayList<String> getmProducts(String llistaCompra){
46     ArrayList<String> llistaProductes = new ArrayList<>();
47     for (LlistesCompra s : mCompra.getValue()){
48         if(s.getNom().equals(llistaCompra)){
49             for(Product p : s.getLlistaProductes()){
50                 llistaProductes.add(p.getNom());
51             }
52         }
53     }
54     return llistaProductes;
55 }
56 public void removeLlistaFromHome(int position, String mailUser) {
57     LlistesCompra llista;
58     llista = mCompra.getValue().get(position);
59     mLlistesCompraRepository.removeLlistaCompra(llista.getNom(), mailUser);
60     mCompra.getValue().remove(position);
61 }
62 public void addLlista(String nameList, String mailUser){
63     LlistesCompra llista = new LlistesCompra(nameList);
64     ArrayList<LlistesCompra> llistes = mCompra.getValue();
65     llistes.add(llista);
66     setLlistesCompra(llistes);
67
68     mLlistesCompraRepository.addLlistaCompra(llista, mailUser, llistes);
69 }
70 public void inicialitzaLlista(String mailUser){
71     mLlistesCompraRepository.inicialitzaLlista(mailUser);
72 }
```

74

}

Listing A.46: Configuració de la classe *LlistesCompraActivityViewModel*.

```
1 package edu.ub.pis3.viewmodel;
2
3 import android.app.Application;
4
5 import androidx.lifecycle.AndroidViewModel;
6 import androidx.lifecycle.MutableLiveData;
7
8 import java.util.ArrayList;
9
10 import edu.ub.pis3.model.LlistesCompra;
11 import edu.ub.pis3.model.LlistesCompraRepository;
12 import edu.ub.pis3.model.Product;
13 import edu.ub.pis3.model.ProductRepository;
14
15 public class ProductesActivityViewModel extends AndroidViewModel {
16     private final String TAG = "ProductesActivityViewModel";
17     private final MutableLiveData<ArrayList<Product>> mProducts;
18     private ProductRepository mProductRepository;
19
20     public ProductesActivityViewModel(Application application){
21         super(application);
22         mProducts = new MutableLiveData<>(new ArrayList<>());
23         mProductRepository = ProductRepository.getInstance();
24
25         mProductRepository.addOnLoadProductesListener(new
26             → ProductRepository.OnLoadProductsListener() {
27                 @Override
28                 public void onLoadProducts(ArrayList<Product> products) {
29                     ProductesActivityViewModel.this.setProducts(products);
30                     loadInfoProductsFromRepository();
31                 }
32             });
33
34         mProductRepository.addOnLoadInfoProductsListener(new
35             → ProductRepository.OnLoadInfoProductsListener() {
36                 @Override
37                 public void onLoadInfoProducts(ArrayList<Product> products) {
38                     ProductesActivityViewModel.this.setProducts(products);
39                 }
39     }
```

```

40
41     public MutableLiveData<ArrayList<Product>> getmProducts(){
42         return mProducts;
43     }
44
45     public void setProducts(ArrayList<Product> products){
46         mProducts.setValue(products);
47     }
48
49     public void loadProductsFromRepository(String mailUser, String nomLlistaCompra){
50         mProductRepository.loadProducts(mProducts.getValue(), nomLlistaCompra, mailUser);
51     }
52
53     public void loadInfoProductsFromRepository(){
54         mProductRepository.loadAllInfoProducts(mProducts.getValue());
55     }
56
57     public void removeProductFromList(int position, String nomLlista, String mailUser){
58         Product product = mProducts.getValue().get(position);
59         mProductRepository.removeProductFromList(nomLlista, mailUser, product.getId());
60         mProducts.getValue().remove(position);
61     }
62 }
```

Listing A.47: Configuració de la classe *ProductsActivityViewModel*.

```

1 package edu.ub.pis3.viewmodel;
2
3 import android.app.Application;
4
5 import androidx.lifecycle.AndroidViewModel;
6 import androidx.lifecycle.MutableLiveData;
7
8
9 import java.util.*;
10
11 import edu.ub.pis3.model.User;
12 import edu.ub.pis3.model.UserRepository;
13
14
15 public class UsersActivityViewModel extends AndroidViewModel{
16
17     private final String TAG = "UsersActivityViewModel";
18     private final MutableLiveData<ArrayList<User>> mUsers;
19     private UserRepository mUsersRepository;
20
21     public UsersActivityViewModel(Application application){
```

```

22     super(application);
23     mUsers = new MutableLiveData<>(new ArrayList<>());
24     mUsersRepository = UserRepository.getInstance();
25
26     /*mUsersRepository.addOnLoadUsersListener(new UserRepository.OnLoadUsersListener(){
27         @Override
28         public void onLoadUsersCompra(ArrayList<User> users) {
29             UsersActivityViewModel.this.setUsers(users);
30         }
31     });*/
32 }
33
34     public MutableLiveData<ArrayList<User>> getmUsers() {
35         return mUsers;
36     }
37
38     public void setUsers(ArrayList<User> llista){
39         mUsers.setValue(llista);
40     }
41
42     public void signup(String nom, String cognom, String mail, String pwd){
43         mUsersRepository.signup(nom, cognom, mail, pwd);
44     }
45 }
```

Listing A.48: Configuració de la classe *UsersActivityViewModel*.

```

1 package edu.ub.pis3.viewmodel;
2
3 import android.app.Application;
4
5 import androidx.lifecycle.AndroidViewModel;
6 import androidx.lifecycle.MutableLiveData;
7
8 import java.util.ArrayList;
9
10 import edu.ub.pis3.model.InfoProducteRepository;
11 import edu.ub.pis3.model.InfoSuperProducte;
12 import edu.ub.pis3.model.Product;
13
14 public class InfoProducteActivityViewModel extends AndroidViewModel {
15     private final String TAG = "InfoProducteActivityViewModel";
16     private final MutableLiveData<ArrayList<InfoSuperProducte>> mProduct;
17     private final MutableLiveData<ArrayList<Product>> mAllProducts;
18     private ArrayList<Product> mAllProductsNoMutable;
```

```
19  private InfoProducteRepository mInfoProducteRepository;  
20  
21  public InfoProducteActivityViewModel(Application application) {  
22      super(application);  
23      this.mProduct = new MutableLiveData<>(new ArrayList<>());  
24      this.mAllProducts = new MutableLiveData<>(new ArrayList<>());  
25      this.mAllProductsNoMutable = new ArrayList<>();  
26      mInfoProducteRepository = InfoProducteRepository.getInstance();  
27      mInfoProducteRepository.addOnLoadInfoProducteListener(new  
28          InfoProducteRepository.OnLoadInfoProducteListener() {  
29              @Override  
30              public void onLoadInfoProducte(ArrayList<InfoSuperProducte>  
31                  llistesInfoProductes) {  
32                  InfoProducteActivityViewMo-  
33                      del.this.setLlistesInfoProducte(llistesInfoProductes);  
34              }  
35          });  
36  
37      mInfoProducteRepository.addOnLoadAllProductsListener(new  
38          InfoProducteRepository.OnLoadAllProductsListener() {  
39              @Override  
40              public void onLoadAllProducts(ArrayList<Product> llistaProducts) {  
41                  InfoProducteActivityViewModel.this.setLlistaAllProducts(llistaProducts);  
42              }  
43          });  
44      }  
45  
46      public void setLlistesInfoProducte(ArrayList<InfoSuperProducte> llistesInfoProducte){  
47          mProduct.setValue(llistesInfoProducte);  
48      }  
49  
50      public void setLlistaAllProducts(ArrayList<Product> llistaAllProducts){  
51          mAllProducts.setValue(llistaAllProducts);  
52      }  
53  
54      public ArrayList<Product> getmAllProductsNoMutable() {  
55          return mAllProductsNoMutable;  
56      }  
57  
58      public void setmAllProductsNoMutable(ArrayList<Product> mAllProductsNoMutable) {  
59          this.mAllProductsNoMutable = mAllProductsNoMutable;  
60      }  
61  
62      public MutableLiveData<ArrayList<InfoSuperProducte>> getInfoSuperProducte(){  
63          return mProduct;  
64      }
```

```

61
62     public MutableLiveData<ArrayList<Product>> getAllProducts(){
63         return mAllProducts;
64     }
65
66     public void loadInfoProducteFromRepository(Long id_producte){
67         mInfoProducteRepository.getInfoProducte(mProduct.getValue(), id_producte);
68     }
69
70     /*public InfoSuperProducte getOneInfoSuperProducte(String nom_super){
71         for(InfoSuperProducte info : mProduct.getValue()){
72             if(info.getNom_Super().equals(nom_super)){
73                 return info;
74             }
75         }
76         return null;
77     }
78
79     /*public ArrayList<String> getSupers(){
80         ArrayList<String> llistaSupers = new ArrayList<>();
81
82         for(InfoSuperProducte info : mProduct.getValue()){
83             llistaSupers.add(info.getNom_Super());
84         }
85         return llistaSupers;
86     }
87
88     /*public String getNomProducte(Long id){
89         return mProduct.getValue().get(0).getNom_producte();
90     }*/
91
92     public void loadAllProductsFromRepository(){
93         mInfoProducteRepository.loadInfoAllProductes(mAllProducts.getValue(),
94             mAllProductsNoMutable);
95     }
96
97     public void addProducteAllistaCompra(String nomLlista, String mailUser, Long
98         numProduct){
99         mInfoProducteRepository.addProductToList(nomLlista, mailUser, numProduct);
100    }
101}

```

Listing A.49: Configuració de la classe *InfoProducteActivityViewModel*.

CODIS DEL SCRAPING DE DADES

Aquest codi correspondria a ProductsFirebase.java, que és el primer pas de l'scraping: primer vam recollir les dades *escrites* dels productes, després ens vam encarregar de com buscar-ne les fotos.

```
1 package edu.ub.crearproductos6;
2
3 import androidx.appcompat.app.AppCompatActivity;
4
5 import android.os.Bundle;
6 import android.util.Log;
7 import android.view.View;
8 import android.widget.Button;
9 import android.widget.TextView;
10
11 import com.google.firebase.firestore.FirebaseFirestore;
12
13 import java.io.BufferedReader;
14 import java.io.IOException;
15 import java.io.InputStream;
16 import java.io.InputStreamReader;
17 import java.util.Arrays;
18 import java.util.HashMap;
19 import java.util.List;
20 import java.util.Map;
21
22 public class ProductsFirebase extends AppCompatActivity {
23     Button read;
24     TextView textView;
25     BufferedReader reader;
26
27     String text;
28     private FirebaseFirestore db;
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_main);
34
35         read = (Button) findViewById(R.id.button);
36         textView = (TextView) findViewById(R.id.textView);
37         db = FirebaseFirestore.getInstance();
38
39         read.setOnClickListener(new View.OnClickListener() {
```

```
40    @Override
41    public void onClick(View view) {
42        String text = "";
43        try {
44            InputStream is = getAssets().open("file.txt.txt");
45            reader = new BufferedReader(new InputStreamReader(is));
46            String line = reader.readLine();
47            while(line != null){
48                Log.d("StackOverflow", line);
49                String[] fila = line.split(",");
50                Map<String, Object> product = new HashMap<>();
51                product.put("Nom producte", fila[1]);
52                List<String> mercadona = Arrays.asList(fila[3], fila[4]);
53                product.put("Mercadona", mercadona);
54                List<String> condis = Arrays.asList(fila[5], fila[6]);
55                product.put("Condís", condis);
56                List<String> lidel = Arrays.asList(fila[7], fila[8]);
57                product.put("Lidel", lidel);
58                List<String> aldi = Arrays.asList(fila[9], fila[10]);
59                product.put("Aldi", aldi);
60                List<String> bonpreu = Arrays.asList(fila[11], fila[12]);
61                product.put("Bonpreu", bonpreu);
62                List<String> alcampo = Arrays.asList(fila[13], fila[14]);
63                product.put("Alcampo", alcampo);
64                List<String> sorli = Arrays.asList(fila[15], fila[16]);
65                product.put("Sorli", sorli);
66                List<String> amatller = Arrays.asList(fila[17], fila[18]);
67                product.put("Amatller Origen", amatller);
68                List<String> veritas = Arrays.asList(fila[19], fila[20]);
69                product.put("Veritas", veritas);
70                List<String> caprabo = Arrays.asList(fila[21], fila[22]);
71                product.put("Caprabo", caprabo);
72                List<String> carrefur = Arrays.asList(fila[23], fila[24]);
73                product.put("Carrefur", carrefur);
74                List<String> spar = Arrays.asList(fila[25], fila[26]);
75                product.put("Spar", spar);
76                List<String> consum = Arrays.asList(fila[27], fila[28]);
77                product.put("Consum", consum);
78                List<String> opencor = Arrays.asList(fila[29], fila[30]);
79                product.put("Opencor", opencor);
80                db.collection("products").document(fila[0]).set(product);
81                for (int i=0; i< fila.length; i++){
82                    Log.d("María", fila[i]);
83                }
84                line = reader.readLine();
85            }
        }
```

```
86         }catch (IOException ex){  
87             ex.printStackTrace();  
88         }  
89         textView.setText(text);  
90     }  
91 }  
92  
93 }
```

Listing A.50: Configuració de la classe que emmagatzema les dades d'un producte en la base de dades.

Bibliografia

- [Lewo7] John LEWIS. *Java software solutions : foundations of program design.* eng. 5th ed. Boston: Pearson Addison-Wesley, 2007. ISBN: 9780321409492.

For almost a decade, Java Software Solutions has been the worldwide best-selling textbook for introduction to programming using the Java language. This text is renowned for providing a solid foundation in programming techniques that leads to well-designed object-oriented software. The authors' emphasis on building solid problem solving and solid design skills is bolstered by their integration of a multitude of small and large realistic programming examples.

- [Muri5] Paul MURPHY. *Android Programming: Pushing the Limits.* English. 4th edition. Wiley, 2015. ISBN: 978-1119971414.

An advanced guide to Android programming that dives into topics such as advanced UI techniques, performance optimization, leveraging hardware features, and integrating with web services. Suitable for experienced Android developers looking to enhance their skills and build more sophisticated apps.

- [Lari6] Craig LARMAN. *Applying UML and patterns : an introduction to object-oriented analysis and design and iterative development.* eng. Third edition. Chennai: Pearson, 2016. ISBN: 9789332553941.

- [GGI7] Dawn GRIFFITH i David GRIFFITH. *Head First Android Development.* English. 2nd edition. O'Reilly Media, 2017. ISBN: 978-1491974056.

A beginner-friendly book that introduces Android app development using Java. It covers the essentials of building Android apps, including UI design, data storage, networking, and more. The book employs a hands-on approach with visual aids, examples, and exercises to reinforce learning.

- [PSMi7] Bill PHILLIPS, Chris STEWART i Kristin MARSICANO. *Android Programming: The Big Nerd Ranch Guide.* English. 4th. Big Nerd Ranch, 2017. ISBN: 978-0134706054.

A practical guide for learning Android development using Java. It covers the essentials of Android app development, including UI design, data persistence, networking, and more. The book follows a hands-on approach with plenty of code examples and exercises.

- [Guei8] Carmen GUERRERO. *Android App Development For Dummies.* English. 3rd edition. For Dummies, 2018. ISBN: 978-1119453856.

A beginner's guide to Android app development using Java. It covers the basics of Android app components, UI design, handling user input, data storage, and more. Suitable for those with little or no programming experience looking to get started with Android development.

Índex terminològic

| A | | | | R | |
|-----------------------------|----------------|------------------------|------------|---------------------|----------------|
| ActionBar | 31 | inici de sessió | 10 | RecyclerView | 20, 21 |
| Activity | 27 | integration testing | 40 | registre | 10 |
| Android Studio | 37 | | | repositori | 36 |
| AndroidX | 18 | L | | | |
| API | 29 | launcher | 15 | | |
| B | | layout | 15 | scroll-down | 14 |
| Base64 | 38 | LinearLayout | 15, 16, 18 | ScrollView | 21 |
| Button | 15, 16, 18 | LiveData | 30 | sistema | 1 |
| C | | log-in | 1 | Spinner | 21 |
| capa de persistència | 37 | login | 28 | supermercat | 2 |
| capa de vista | 27 | logs | 30 | | |
| Codis del scraping de dades | 37 | M | | T | |
| ConstraintLayout | 15, 16, 18, 20 | Manteniment | 1 | TableLayout | 15, 16, 18, 21 |
| E | | Map | 30 | TextView | 21 |
| EditText | 15, 16, 18 | mock-ups | 9 | Toast | 28 |
| Escalabilitat | 1 | model | 24 | | |
| F | | N | | U | |
| Fiabilitat | 1 | non-functional testing | 45 | UI | 36 |
| FirBase | 25, 37 | | | unit testing | 38 |
| Firebase | 37 | O | | usability testing | 41 |
| Fragment | 32 | onCreate | 28 | usuari no registrat | 2 |
| FrameLayout | 15, 16, 18, 20 | P | | usuari registrat | 2 |
| framework | 27 | performance testing | 41 | V | |
| | | Personalització | 1 | ViewModel | 35 |
| | | ProgressBar | 18 | | |
| I | | Q | | X | |
| ImageView | 15, 16, 18 | query | 38 | XML | 18 |