TFC Informàtica de sistemes
D. apli. dispositius mòbils-Android

IngredientApp

11/06/2014

Memòria - IngredientApp

TFC – Informàtica de sistemes

Autor: Alejandro Sandín Estevan

Consultors: Joan Vicent Orenga Serisuelo | Jordi Almirall López

A mi mujer Mónica que es mi fuerza de voluntad, mi perseverancia, cree en mí más que yo mismo. Y le debo, a lo menos, el resto de fines de semana de mi vida.

A mis más grandes proyectos, mis hijas Ariadna y mi recién nacida Valeria.

A mi familia que sabe lo que me ha costado.

ÍNDEX PRINCIPAL

ÍNDEX DE FIGURES

1 | INTRODUCCIÓ - MOTIVACIÓ

Cada vegada més les al·lèrgies e intoleràncies alimentaries son un fet més comú a la societat moderna, La OMS indica que fins un 40[1] per cent de la població mundial pateix aquestes alteracions. Paraules i expressions com celíacs, intoleràncies alimentaria, sensibilitat a la lactosa etc. ... son molt comuns a la nostre vida diària. A la televisió no deixen de repetir que augmenten les al·lèrgies any rere any a la població infantil.

Regularment es retiren del mercat a uns països sí, a d'altres no, ingredients d'aliments, components de pintures, envasos, sabons. Altres vegades son solament recomanacions, que finalment es transformen en prohibicions.

També hi ha un moviment molt creixent de demanda d'aliments i productes "bio". La pròpia paraula "bio" ha sigut font de legislació tant nacional com internacional. Hi han cadenes de supermercats dedicats només a questa línia d'aliments, i es estrany de no veure a grans distribuïdors (Carrefour, Alcampo, Dia) parts dedicades a aquests productes.

2 | OBJECTIUS – INGREDIENTAPP

Aquest document descriu una aplicació per dispositius Android. L'aplicació IngredientApp es un intent de auto controlar el que comprem, fem servir, o mengem. Es tracta d'una aplicació a la que s'hi defineixen unes preferències sobre els ingredients dels productes. I fent servir un smartphone sense haver de mirar la llista d'ingredients , l'aplicatiu li advertirà si el producte compleix els seus criteris.

L'aplicació a desenvolupar té la finalitat de crear una base de dades , accessible i compartida , de productes a nivell d'ingredients . Els productes seran, en principi, alimentaris de consum habitual .Encara que també es pot estendre a sabons , aigües embotellades i altres productes . Tot el que pugui tenir una llista d'ingredients . Segons la legislació actual [2]tot producte de consum ha de portar-la.

A nivell personal l'objectiu es introduir-me en la programació dispositius mòbils Android. Es uns oportunitat d'actualitzar coneixements i ampliar les oportunitats professionals. A la vegada posar en pràctica tots els coneixements adquirits durant els anys d'universitat.

2.1 | Aplicacions semblants | IntolerApp[3]

Aquesta és una aplicació molt semblant a la que volem desenvolupar. Es centra en tres intoleràncies/ al·lèrgies alimentaries, molt comuns: Ous, Gluten i Lactosa. És una aplicació que es paga i al 2012 va rebre un premi a la millor aplicació sanitària.

L'interfície gràfica conte amb una simplicitat molt ben buscada. Un menú amb poques opcions i molt directes. Consta de:

- Buscar un producte apuntant la càmera amb el lector de codis de barres.

[1] http://www.eltiempo.com/archivo/documento/CMS-13044280
[2] http://www.aepnaa.org/podemos-ayudar/normativa-etiquetado-25
[3] IntolerApp https://play.google.com/store/apps/details?id=com.neosono.camera.barcode&hl=es

- Recerca a la BD directament per fabricants i veure els seus productes.
- Una fitxa de producte clara i concisa que indica clarament si es un producte apte o no.
- Una pantalla de links d'interès
- Una pantalla d'informació de caràcter legal.
- Menú de l'aplicació disponible des de qualsevol punt de l'aplicació Recerca manual de productes
- Una Fitxa de producte clara i directa

FIGURA 1 PANTALLES INTOLERAPP

2.3 | Quin es el nostre valor afegit respecte a IntolerApp i altres semblants?

La base de dades de la nostre aplicació s'alimenta automàticament dels usuaris. Cada usuari pot tenir unes preferències lliures i no fixades com en el cas de IntolerApp i altres semblants. Encara que l'aplicació pot ser molt útil per gent amb al·lèrgies o intoleràncies, no esta limitada a aquest us.

Cada vegada que un usuari comparteix els productes que ha introduït. La resta d'usuaris tenen aquesta informació virtualment de forma instantània. I cadascú pot fer un us diferent.

Aquest son els punts on crec que l'aplicació pot ser més útil. Quan més gent la fa servir més útil es per tothom.

3 | FUNCIONALITATS PRINCIPALS

Per cobrir les necessitat de l'aplicació es van identificar aquestes funcionalitats/ usos principals:

3.1 | Aplicar les preferències personals

Una de les peces claus es que l'usuari pugui definir uns filtres o preferències. Aquestes son els patrons de recerca que s'aplicaran als ingredients del productes que trobem.

S'han de poder afegir, eliminar i modificar al llarga del temps.

3.2 | Buscar producte

És la funcionalitat principal. Es tracta de buscar un producte que volem usar/ comprar i necessitem informació immediata del mateix. Aquesta funcionalitat ha d'ajudar-se d'un lector de codis de barres per afavorir la recerca.

3.2.1 | Mostrar informació del producte

Una vegada trobem un producte s'ha de mostrar informació clara i concisa del mateix. I sobretot si el producte s'ajusta a les preferències de l'usuari.

3.2.2 | Incorporar nous productes.

Si un producte no existeix a la base de dades, s'ha de poder introduir a la base de dades. Aquests productes després es compartiran amb la resta d'usuaris al servidor central.

Per aconseguir que l'usuari no tingui "mandra" de compartir nous productes, aquesta interacció amb el dispositiu ha de ser àgil. Per afavorir aquesta introducció de dades (codi del producte, descripció e ingredients), s'ha de dotar a l'usuari d'eines d'introducció ràpida. Serien el reconeixement de text fent servir la càmera del dispositiu i el reconeixement de veu.

3.3 | Actualitzacions

Un punt crític es que la base de dades del dispositiu sigui actualitzable. Aquesta ha de créixer i prendre informació de tots els usuaris de l'aplicació i fer-la útil a la resta.

Una base de dades amb tots els productes del mercat es totalment inviable d'implementar. A més que cada usuari pot fer servir l'aplicació per un tipus de producte diferent: aliments, sabons, medicaments, o qualsevol altre producte de consum que porti una llista d'ingredients.

Així la facilitat d'actualització de la base de dades de l'aplicació marcarà la versatilitat de la mateixa.

4 | PLANIFICACIÓ – PLA DE TREBALL

Partint de les cites del pla docent i l'entrega de les PAC's de l'avaluació continuada es parteix el treball en diferents entregues. Seguir el pla de treball es part principal per assolir el desenvolupament del projecte.

Títol	Inici / Enunciat	Lliurament
PAC1	27/02/2014	12/03/2014
PAC2	13/03/2014	16/04/2014
PAC3	17/04/2014	21/05/2014
Entrega final	22/05/2014	11/06/2014
Debat virtual	23/06/2014	27/06/2014

FIGURA 2 PLANIFICACIÓ PAC'S

4.1 | Calendari de fites

Després d'avaluar la feina a fer es reparteixen entre les diferents PAC's seguint el diagrama establert que s'adjunta.

PAC 1	**21 días**	**mié 12/02/14**	**mié 12/03/14**
Planificació inicial	7 días	mié 12/02/14	jue 20/02/14
Preparar l'entorn de treball	4 días	vie 21/02/14	mié 26/02/14
Rercerca de Llibreries CCR	2 días	jue 27/02/14	mar 04/03/14
Recerca de llibreries OCR	2 días	mié 05/03/14	jue 06/03/14
Recerca de SQL local i central	2 días	vie 07/03/14	lun 10/03/14
Provar llibreries	2 días	mar 11/03/14	mié 12/03/14
PAC 2	**26 días**	**jue 13/03/14**	**mié 16/04/14**
Disseny ER de la BD local i central	2 días	jue 13/03/14	vie 14/03/14
Arrencada de servidor central	6 días	sáb 15/03/14	vie 21/03/14
Implementar i Poblar la BD, imprimir etiquetes	2 días	vie 21/03/14	lun 24/03/14
Recopilar productes reals	2 días	vie 21/03/14	lun 24/03/14
Especificar serveis web	5 días	mar 25/03/14	lun 31/03/14
Disseny de pantallas del client local Android	3 días	mar 01/04/14	jue 03/04/14
Especificar servei de sincronització	6 días	vie 04/04/14	vie 11/04/14
PAC 3	**25 días**	**jue 17/04/14**	**mié 21/05/14**
Implementar lector OCR	8 días	jue 17/04/14	dom 27/04/14
Implementar serveis web	6 días	lun 28/04/14	dom 04/05/14
Implementar serveis centrals	6 días	lun 05/05/14	dom 11/05/14
Implementar disseny client i consultes locals	6 días	lun 12/05/14	dom 18/05/14
Implementar sincronitzador BD	3 días	lun 19/05/14	mié 21/05/14
Lliurament Final	15 días	jue 24/04/14	mié 14/05/14
Debat Virtual	5 días	lun 23/06/14	vie 27/06/14

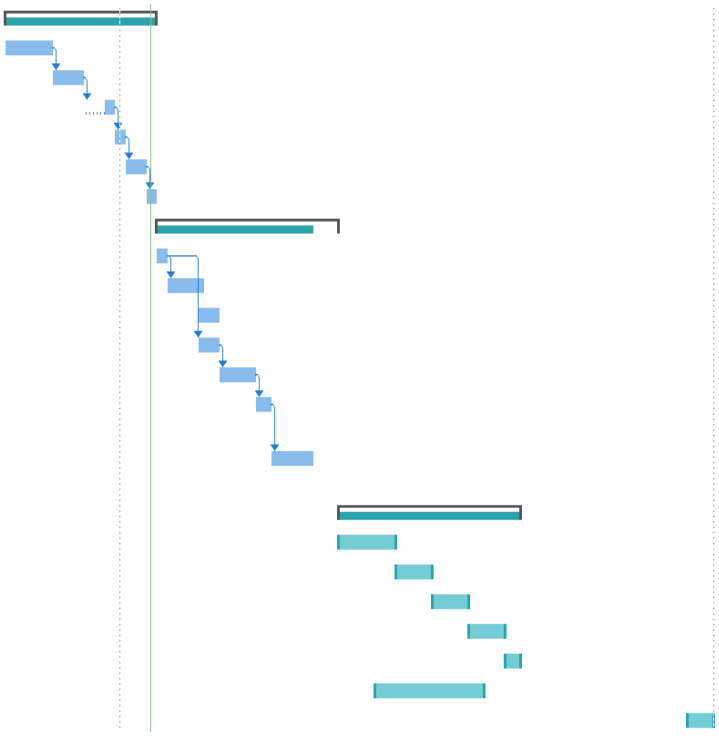


FIGURA 3 CALENDARI FITES

5 | ANÀLISI FUNCIONAL

Es van identificar diferents perfils que farien servir l'aplicació:

- Al·lèrgic que compra: Persona amb algun tipus d'al·lèrgia alimentaria i que normalment s'encarrega de fer la seva compra i/o la compra familiar.
- Jove de compra ràpida: Menja normalment fora de casa o compra aliments de forma ràpida. Després del gimnàs, de camí a la universitat etc. Amb algun tipus d'al·lèrgia alimentaria
- Singles: Els hi agrada veure que mengen, no els hi agraden els greixos trans, ni els conservants poc coneguts, volen saber que compren.

Tot i que les motivacions i comportaments a l'hora de fer servir l'aplicatiu son diferents en tots tres perfils. Els objectius son els mateixos. Les funcionalitats comunes a tots ells que es van identificar en forma de fluxos d'interacció.

5.1 | Els fluxos d'interacció

Les diferents funcionalitats e interacció queden recollides en quatre principals fluxos d'interacció entre l'usuari de l'aplicatiu i el dispositiu. També entre el dispositiu i el servidor central al núvol.

5.1.1 Preferències personals

Aquest flux fa referència a les diferents pantalles de personalització que l'usuari final ha de complimentar. A partir d'aquestes preferències s'orquestra tota l'aplicació i filtratge de productes.

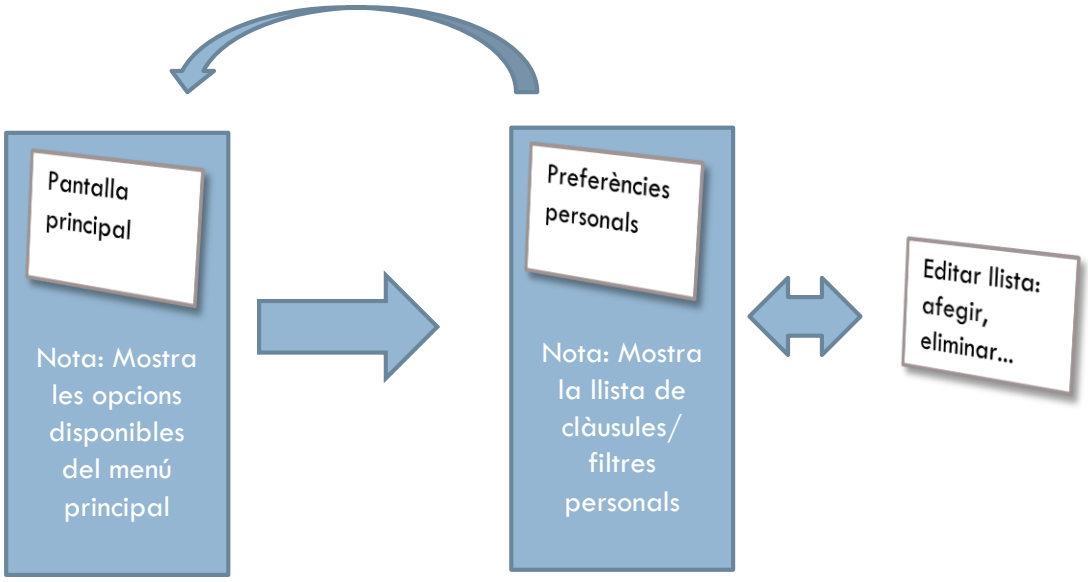


FIGURA 4 PREFERÈNCIES PERSONALS

5.1.2 Recerca de productes

Aquest els flux corresponent a la recerca de productes.

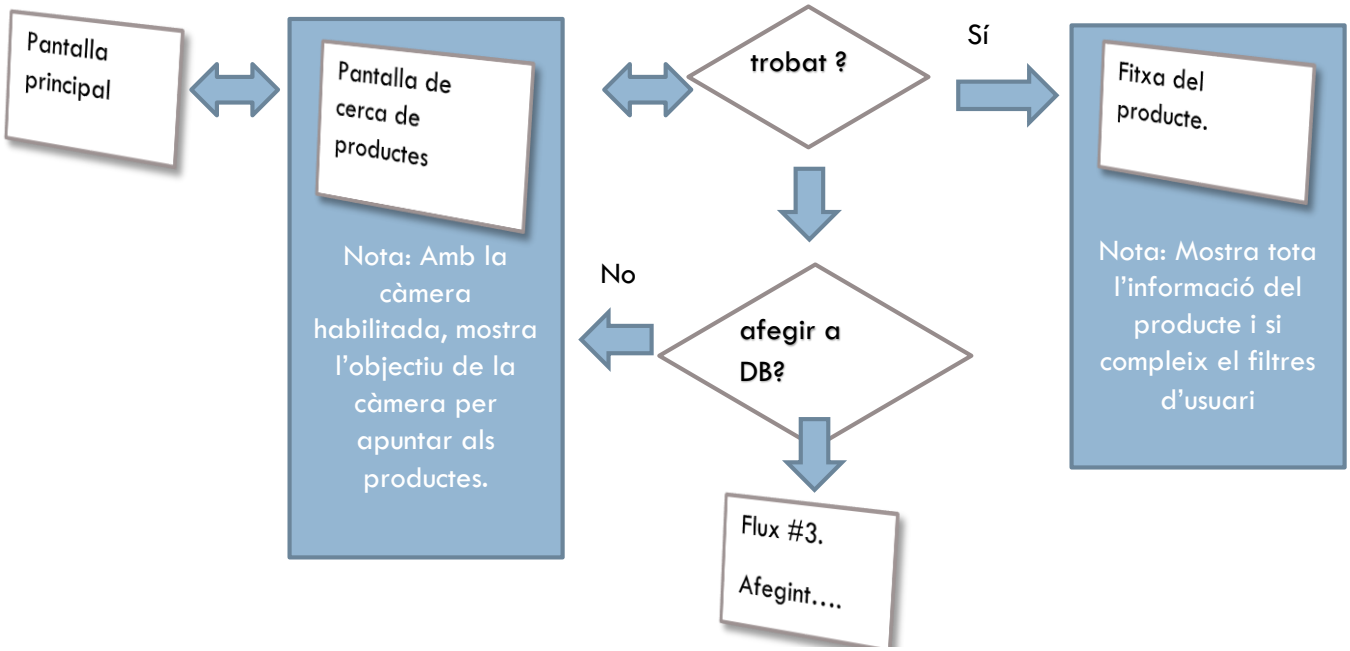


FIGURA 5 RECERCA DE PRODUCTES

5.1.3 Afegir producte a la base de dades local del dispositiu

Aquest flux correspon a quan l'usuari no troba un producte a la BD. I decideix introduir-lo ell mateix, i així compartir les dades del producte amb la resta d'usuaris.

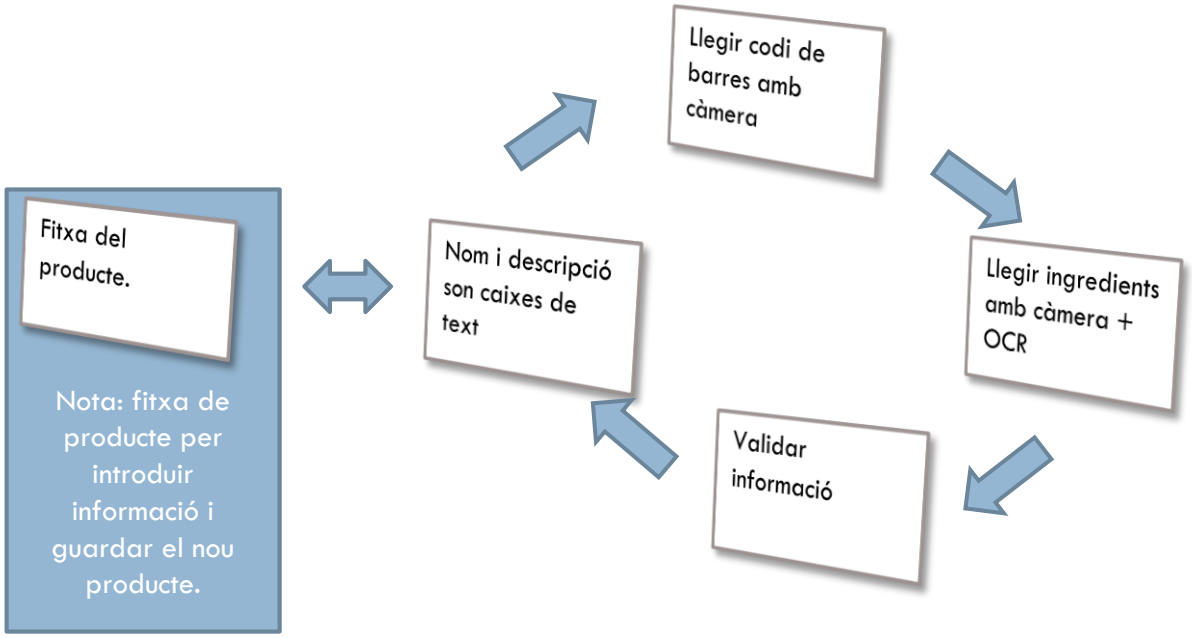


FIGURA 6 AFEGINT PRODUCTES

5.1.4 Sincronitzar les bases de dades del dispositiu mòbil i el servidor central

Aquest flux correspon al moment que l'usuari decideix sincronitzar la seva DB i actualitza així amb la DB global. Aquest moment que pot trigar és o menys depenent del temps que faci que no s'actualitza.

El productes llegits local ments s'afegeixen a la DB global. I els que li falten s'afegeixen a la BD local de l'usuari.

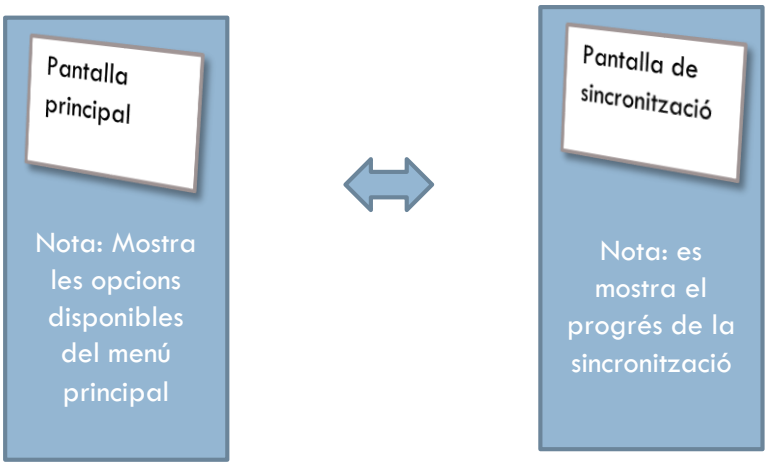


FIGURA 7 SINCRONITZANT BASE DE DADES

6 | DISSENY CONCEPTUAL [DE L'ANÀLISI AL DISSENY]

En una primera fases es va realitzar un primer esborrany de les pantalles de l'aplicatiu. Per plasmar una idea conceptual, res millor que llapis i paper. Per ajudar a fer un disseny més acurat es van fer servir plantilles [4]amb les formes principals del dispositiu.

Després de fer un parell d'iteracions sobre el disseny, i passant del disseny a ma alçada pur. Es van definir aquestes pantalles o activities del l'aplicatiu.

6.1 | Pantalla principal

FIGURA 8 PANTALLA PRINCIPAL

La pantalla inicial mostra text dalt de tot que indica en tot moment on som dins dels menús de l'aplicació. En aquest cas som al menú principal.

I un menú amb tres opcions:

1. Buscar un producte. Que obre la càmera per llegir el codi de barres.
2. Preferències personals. Que ens porta a una pantalla simple d'edició bàsica d'una llista. Editar, Eliminar, Afegir.
3. Sincronitzar la BD. Ens fa anar a la pantalla de sincronització que mostra un resum bàsic de l'estat i un botó per sincronitzar

Per navegar es fan servir els botons estàndards del dispositiu.

A baix del menú es reserva un espai per possibles alertes, d'actualitzacions pendents de la BD, actualitzacions de l'aplicació, publicitat. Es més una declaració d'intencions que una necessitat de l'aplicació.

6.2 | Pantalla de recerca de codis

La pantalla de recerca obre la càmera. Té una línia al centre per ajudar a apuntar bé cap al codi de barres que volem llegir.

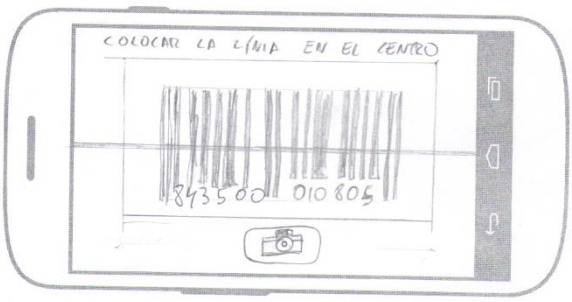


FIGURA 9 RECONEIXEMENT CODIS DE BARRES

Tan bon punt es detecta el codi de barres ens porta a la pantalla de resultats, o ens a d'indicar que no s'ha trobat el producte.

Tot i així si no es possible l'autodetecció es manté un botó manual per forçar l'autodetecció en forma d'icona de càmera.

La pantalla es mostra en format apaïsat per afavorir la detecció del codi de barres.

[4] Ui Stencils Scketch Pads http://www.uistencils.com

6.3 | Pantalla de resultat negatiu de la recerca

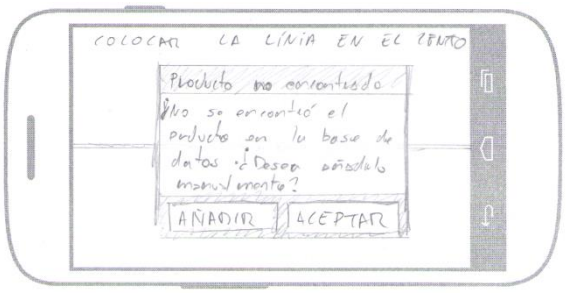


FIGURA 10 PRODUCTE NO TROBAT

Cas que els producte no es trobi a BD es mostra una alerta indicant que no s'ha trobat. A la vegada s'ofereix la possibilitat d'afegir el producte a la BD

6.4 | Pantalla de nou producte

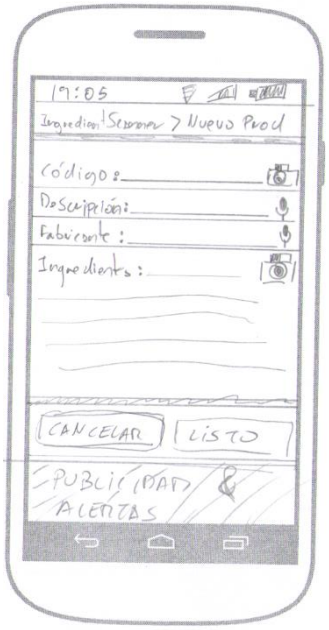


FIGURA 11 NOU PRODUCTE

Si no s'ha trobat el producte i l'usuari decideix afegir el producte s'arriba a aquesta pantalla.

El text superior indica on som: Nou producte.

Pot cancel·lar o introduir totes les dades. Que son:

Codi de barres amb la mateixa pantalla que la de recerca si prem el botó de càmera de la dreta. O manualment.

Descripció i fabricant. S'introdueix manualment o amb el reconeixedor de text Standard.

Ingredients: S'introdueix o bé manualment o amb la pantalla de reconeixement de text OCR.

Si tot es correcte es guarda la fitxa a la BD local.

6.5 | Pantalles del reconeixement de text OCR

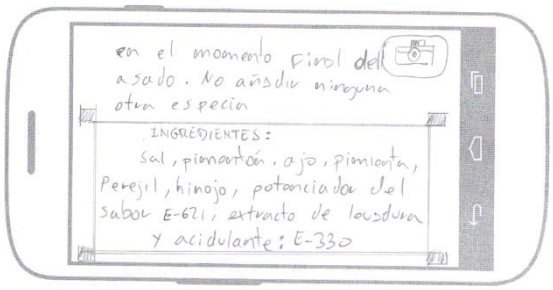
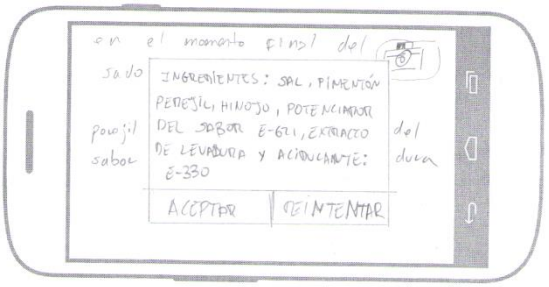


FIGURA 12 RECONEIXEMENTS OCR

Aquesta pantalla en horitzontal marca en un quadre el text que inclou els ingredients. Llavors l'usuari li dona al botó de la càmera i es reconeixen els ingredients.



Es mostra la resposta del reconeixement del text dels ingredients per acceptar, o per tornar a intentar.

Si és correcte s'introdueix el text al camp d'ingredients i es torna a la pantalla de fitxa de nou producte.

Aquesta part es va eliminar de l'aplicació en una iteració posterior ja que moltes vegades fallava el reconeixement del text. De vegades degut a una mala impressió del text, d'altres a la forma de l'embolcall, o el suport del mateix (plàstics transparents, tous, o molt brillants).
Es van introduir reconeixements de veu com a forma alternativa d'entrar dades ràpidament.

6.6 | Pantalla de preferències personals

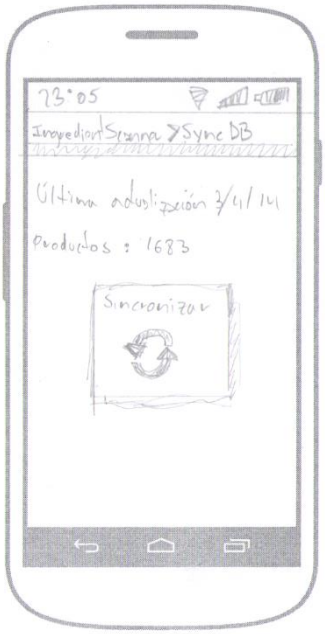
Es mostra el text dalt de tot indicant on som dins l'aplicació.

Una llista amb les icones d'eliminar i editar a l'esquerra de cada element de la llista.

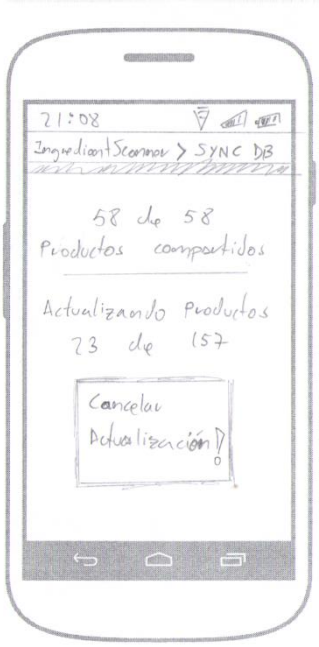
A sota de la pantalla un quadre de text per afegir noves preferències.

FIGURA 13 PREFERÈNCIES PERSONALS

6.7 | Pantalles de sincronització de la base de dades.



Es mostra un breu resum de l'estat actual i un botó central per iniciar la sincronització



Aquesta pantalla mostra el progrés i inclou un botó per cancel·lar l'acció a petició de

FIGURA 14 SINCRNITZANT DISPOSITIU

7 | DISSENY TÈCNIC

L'aplicatiu segueix un patró Client-Servidor. On els dispositius mòbils Android son els clients y el servidor esta ubicat al núvol.

A la figura es mostra una representació del patró i el disseny global de l'aplicació.

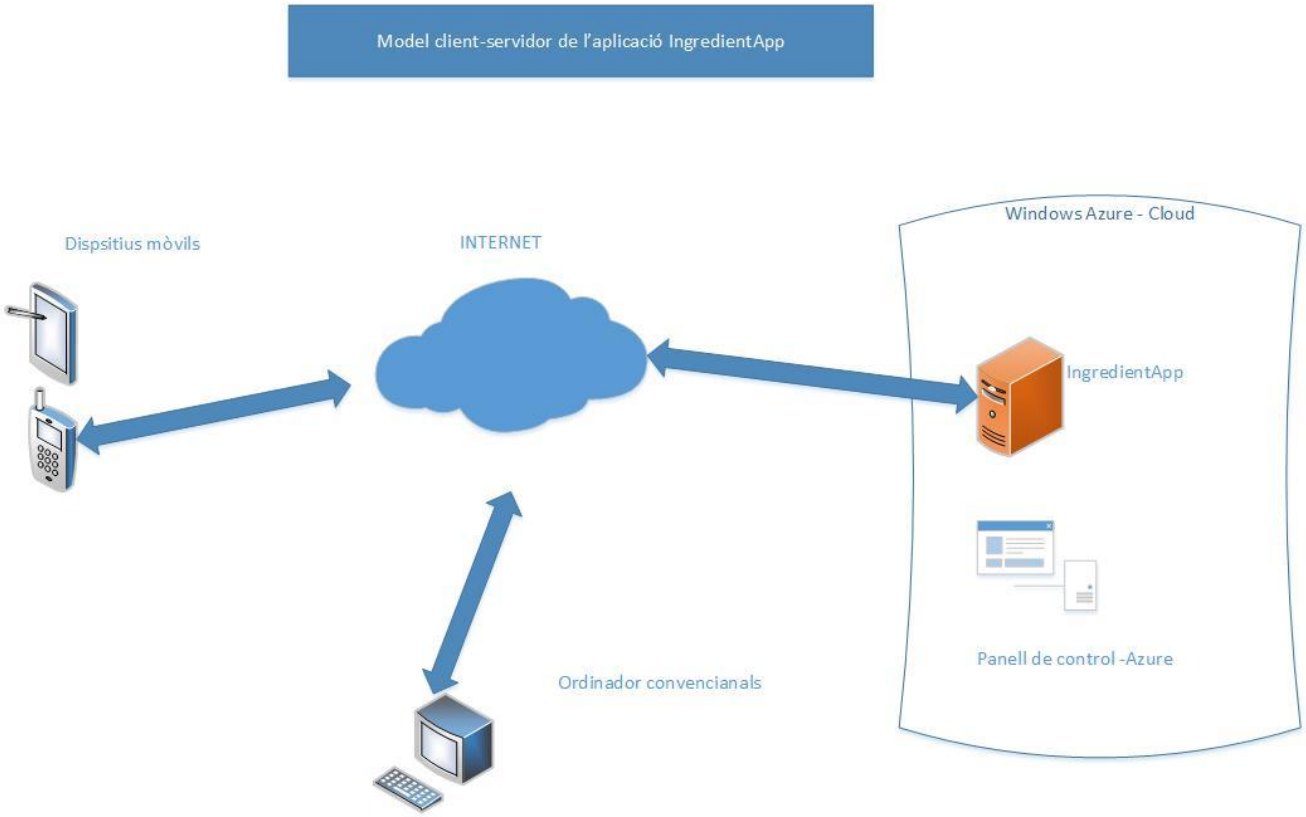


FIGURA 15 MODEL CLIENT-SERVIDOR | INGREDIENTAPP

Hi ha dues capes ben diferenciades els dispositius mòbils i els servidor. El dispositius son tal com dèiem, i és l'àrea de treball, els dispositius Android.

El servidor escollit en aquests cas es Windows Azure. Als pròxims punts es detallen.

7.1 | Servidor al núvol – Windows Azure

L'enfocament clàssic a l'hora de fer una aplicació client servidor amb presència a Internet, es muntar un servidor de pre-producció a un recurs local. En aquest servidor s'instal·len els serveis necessaris per fer córrer l'aplicatiu i depurar-lo.

Una vegada l'aplicatiu estar a un estat madur (Release) es busca un proveïdor d'internet, es contracten els serveis (magatzem de dades, servidors d'aplicacions HTML, permisos i quotes de disc), i finalment es publica.

Es va escollir en aquest cas Windows Azure principalment perquè aquestes diferents fases es poden fer automàticament. Conte amb una interfície de control ràpida i sobretot àgil. Des de el moment que es

comença a fer servir el servei s'ocupa de tot i no s'ha d'instal·lar cap servei. Tot es fa des de l'esmentada plana web de control.

FIGURA 16 PANELL DE CONTROL WINDOWS AZURE

Per la part del servidor s'ha implementat una web i un Api REST. El llenguatge utilitzat es C#. Una avantatge de fer servir el C# es que l'aplicatiu es pot publicar directament al núvol de Windows Azure directament des del Visual Studio[5]. Que conte amb una base de dades SQL Server-Azure que es manté des del mateix panell de control.

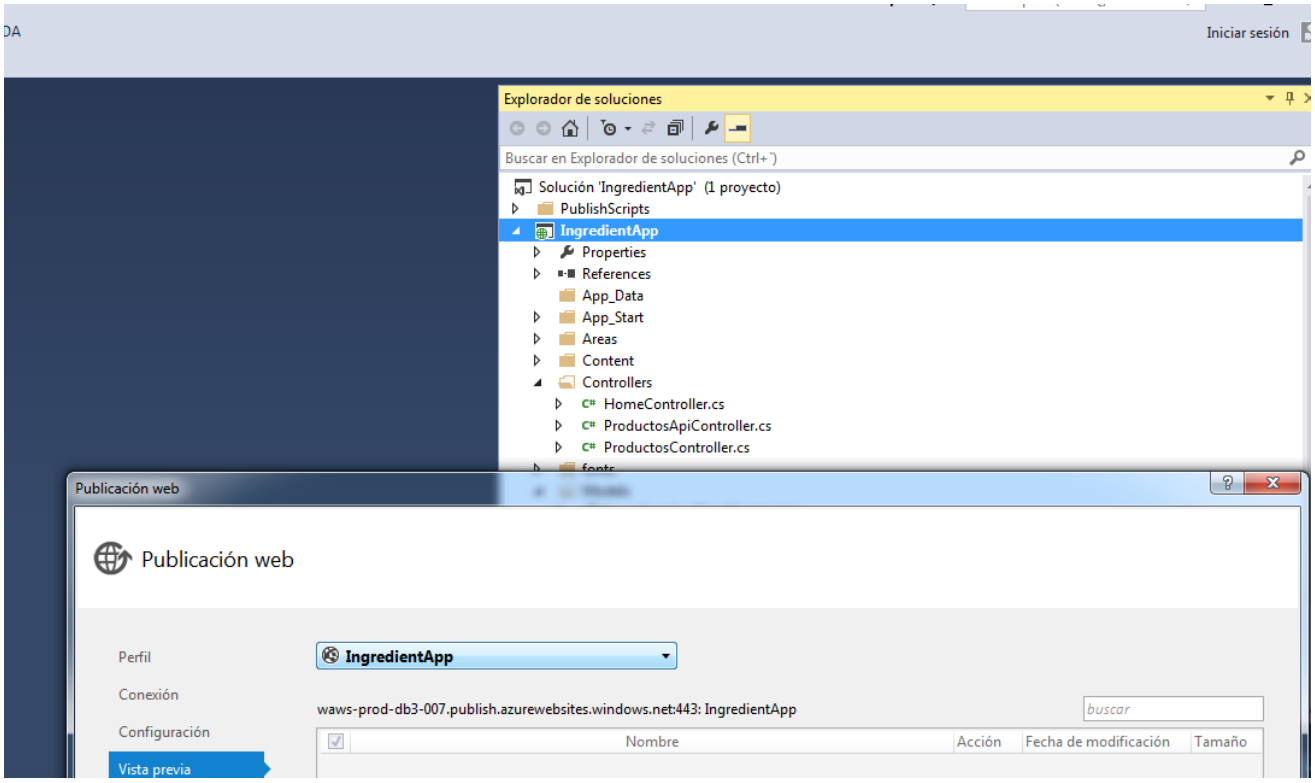


FIGURA 17 PUBLICAR SOLUCIÓ A WINDOWS AZURE

[5] Visual Studio http://www.visualstudio.com/

7.2 | Dispositius mòbils – Android

Tal com dèiem l'aplicació es desenvolupa a la plataforma Android.[6] Es una plataforma Open Source[7] i gratuïta, amb una comunitat molt activa.

Els dispositius es programen amb Android Development Tools[8] un pluggin per la plataforma de programació Eclipse[9]

L'aplicació fa us principal de l'Api d'Android i te com a suport d'emmagatzematge principal al dispositiu la base de dades lleugera Sqlite[10].

Tota la capa de comunicacions entre els dispositius mòbils i el servidor es fa amb serveis REST implementats al servidor en C#, i amb dades en format JSON[11].

Tots els aspectes formals de l'implementació es descriuen més endavant.

7.3 | Disseny relacional - ER

L'aplicació consta de dues bases de dades. Una allotjada a Windows Azure i un altre al dispositiu mòbil Android que fa córrer la mateixa.

L'estructura i utilització de cadascuna es detallen als punts següents.

7.3.1 | Base de dades global -Azure

La base de dades global s'allotja al núvol directament a Windows Azure. Conté una sola taula de productes. L'estructura es la següent

Camp	Tipus	Descripció
codigo	Cadena	Que es la clau principal i es on guardem el codi de barres dels productes
descripcion	Cadena	Es una breu descripció del producte. Es obligatori d'enregistrar
ingredientes	Cadena	Es un camp de text prou llarg per emmagatzemar el text dels ingredients dels productes. Es obligatori d'enregistrar.
fabricante	Cadena	No s'utilitza
id	Numèric	es un camp autoincremental que es fa servir per sincronitzar els dispositius. Cada vegada que un dispositiu comparteix un producte el servidor li dona un id. Aquest no es pot repetir i és el que es reparteix al diferents dispositius mòbils quan s'actualitzen

FIGURA 18 TAULA DE PRODUCTES AL SERVIDOR AZURE

[6] Android Developer http://developer.android.com/index.html
[7] Open Source http://es.wikipedia.org/wiki/C%C3%B3digo_abierto
[8] ADT http://developer.android.com/tools/sdk/eclipse-adt.html
[9] Eclipse https://www.eclipse.org/
[10] SQLITE http://www.sqlite.org/
[11] Json http://es.wikipedia.org/wiki/JSON

7.3.2 | Taula de productes de dades global – Local al dispositiu mòbil.

La base dades als dispositius es una base de dades SQlite.

Conté tres taules. Dues de les taules tenen una estructura molt similar a la del servidor Azure. La primera es diu **productosGlobal**. Aquesta taula en realitat una copia de la del servidor Azure. L'única diferència es que el camp **_id** no es autoincremental. Ja que el valor d'aquest camp s'assigna al servidor Azure.

El fet de mantenir aquesta taula al dispositiu de forma local es clau. Quan es fa una consulta dins d'un supermercat o en qualsevol altre context, es possible que no tinguem connexió a internet. O Potser la connexió disponible es molt lenta. Així la millor forma d'assegurar que la velocitat de les consultes sobre els productes es adequada, es mantenir aquesta taula al dispositiu.

7.3.3 | Taula de nous productes – Local al dispositiu mòbil.

Per la mateixa raó que al punt anterior es manté una taula de productes nous al dispositiu. Quan un producte no es troba a la base de dades del dispositiu es pot afegir a l'aplicatiu. Aquest nou producte s'afegeix en local.

Més tard quan l'usuari decideix compartir/ actualitzar la base de dades de productes. Aquests nous productes s'afegeixen a la base de dades Azure. D'aquesta manera es comparteixen automàticament per la resta d'usuaris. Quan un usuari actualitza l'aplicatiu es descarrega tots els productes nous, només els nous, que han introduït la resta d'usuaris.

Aquesta taula també que s'anomena **productosLocal**, també comparteix la mateixa estructura que la del servidor Azure. La diferencia es que el camp **_id**, no s'envia al servidor al moment d'actualitzar.

7.3.4 | Taula de paràmetres – Local al dispositiu mòbil.

Es manté una taula de paràmetres. En la versió actual de l'aplicatiu, només conte un camp **ultimaactualizacion**. Es guarda el darrer dia que es va actualitzar la base de dades.

Aqueta taula serviria per millores posteriors. Aquí es guardarien preferències d'usuari com "actualitzar automàticament", i qualsevol altre paràmetre que es necessités en possibles millores.

8 | IMPLEMENTACIÓ

En l'implementació de l'aplicatiu, com ja s'ha comentat, hi ha dos parts diferenciades. L'implementació en la plataforma Android. I el servidor al núvol.

Per l'implementació en Android s'ha fet servir l'entorn de programació Eclipse. I pel servidor central s'ha desenvolupat en c# i la part pública en Windows Azure.

Als següents punts es descriu més en detall els diferents entorns.

8.1 | Desenvolupament Android.

El desenvolupament en Android es basa en un model de Vista-Controlador. Per una banda tenim un disseny gràfic, que es la part que veu l'usuari. I per altre banda tenim el controlador. En Android les vistes s'anomenen Activities. I es representen en arxiu XML. La lògica es programa en classes Java.

Existeixen diferents eines visuals més intuïtives i fàcils per dissenyar les vistes que Eclipse, a més l'aspecte visual es molt més interessant de cara l'usuari. Es va pensar en una iteració posterior per millorar l'aspecte visual. Que finalment no va arribar per falta de temps. De totes formes l'aplicació es totalment funcional.

L 'avantatge de la separació total entre vista i controlador es que en qualsevol moment es pot canviar l'aspecte sense haver de canviar la lògica de l'aplicatiu.

Dins del IDE de programació la separació entre vistes i controladors es fa encara més palesa. Hi ha una carpeta per les vistes i un altre pels controladors. Cada vista conté un disseny o Layout, dins del qual es disposen el diferents elements visual. Aquests son els que veuen el usuaris.

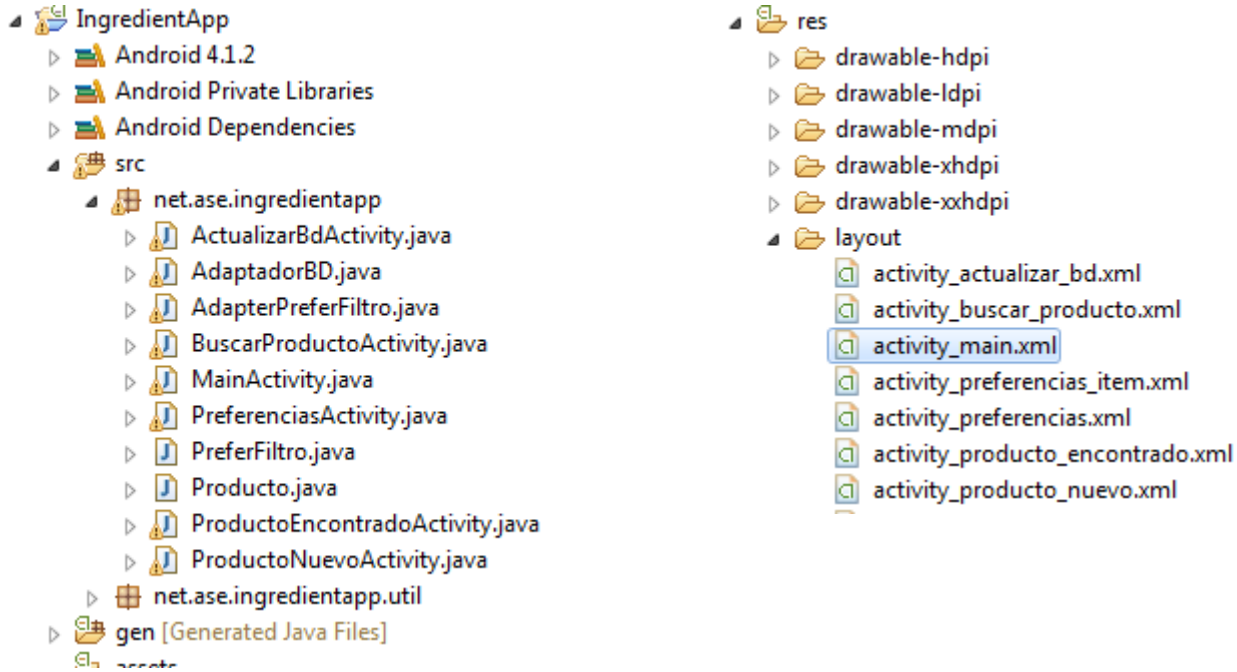
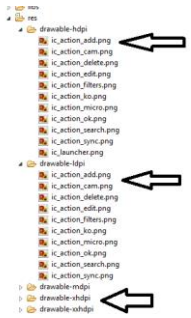


FIGURA 19 CONTROLADORS JAVA I ACTIVITIES EN XML DINS ECLIPSE

Un altre punt important es la facilitat d'internacionalització que brinda Android. Existeix un arxiu strings.xml on s'ubiquen tots els literals que fa servir l'aplicació. Aquest arxiu es pot externalitat i traduir amb certa facilitat. I augmenta la separació entre la lògica i l'aspecte visual.

Seria una millora de l'aplicació. Afegir diferents idiomes.

Figura 20 Arxiu de literals de l'aplicació



Totes les icones de l'aplicació que s'han fet servir estan en diferents mides. Així l'aspecte es pot acomodar segons la mida de la pantalla des dispositiu.

S'han utilitzat les de la pàgina http://www.icons4android.com/

FIGURA 21 MIDES DE LES ICONES

8.1.1 | Base de dades

Tot accés a la base de dades de l'aplicatiu; Sqlite. Està encapsulat en una sola classe: AdaptadorBD.java. que s'ajuda de la classe Android SQLiteOpenHelper.

En l'implementació s'han d'implementar dos mètodes obligatòriament. Oncreate() que s'executa la primera vegada que s'accedeix i s'encarrega de crear les taules.

I OnUpgrade() que serviria en actualitzar l'aplicatiu per fer canvis en l'estructura interna.

Aquí es creen tota la resta de mètodes de l'aplicatiu per accedir a preferències de l'usuari, productes, o estat de les actualitzacions.

8.1.2 | Pantalla principal

La pantalla principal mostra el menú d'inici de l'aplicatiu. On comença l'usuari.

Aquesta es l'implementació de l'aplicatiu. Està format per dos arxius

La vista: activity_main.xml

El controlador: MainActivity.java

Es un layout simple on es poden veure tres grans parts. Si premen en qualsevol punts anem a la pantalla de: buscar productes, establir filtres i actualitzar bases de dades.

FIGURA 22 PANTALLA PRINCIPAL

8.1.3 | Pantalla de recerca de productes

Els arxius que la conformes son activity_buscar_producto.xml per la vista i BuscarProductoActivity.java pel controlador

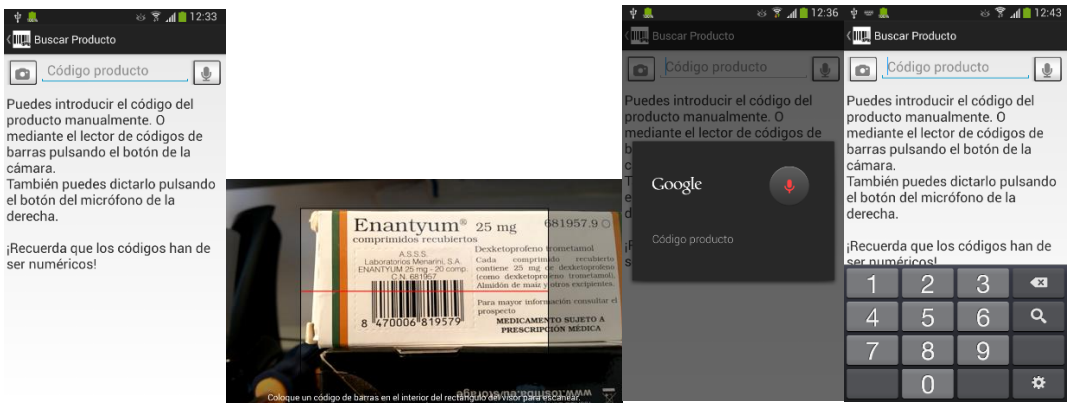


FIGURA 23 RECERCA DE PRODUCTES

Tal com es veu a les diferents imatges, es pot fer la recerca de diferents formes:

1. Llegint els codi de barres; botó de l'esquerra a la primera imatge
2. Es pot dictar el nombre de viva veu; botó dret a la primera imatge. Correspon a la tercera imatge.
3. Es pot fer servir el teclat. Configurat en aquest cas com numèric. Correspon a la quarta imatge.

Per reconèixer els codis de barres. Es fa servir la llibreria Zxing [12]. S'executa una activity de la llibreria Zxing per obtenir un resultat. Que en aquest cas es una cadena d'un codi de barres. Correspon a segona imatge. La línia vermella s'ha d'alinear amb el codi de barres.

Per fer el reconeixement de veu s'ha de tenir configurat el dispositiu per reconèixer veu i en aquest cas accés a internet. Ja que el servei es de Google per internet

En l'implementació. El controlador pot cridar a dos activities diferents. Una per reconèixer codis de barres i un altre per reconèixer veu. Quan es crea una activity amb la finalitat d'obtenir un resultat, se li dona un codi. En finalitzar l 'activity creada es crida una de mètode callback[13] que retorna el resultat.

```
btnEscanerCB = (ImageButton) this.findViewById(R.id.btnEscanerCB);
btnEscanerCB.setOnClickListener(new OnClickListener() {

    @Override
    public void onClick(View view) {

        Intent intent = new Intent("net.ase.ingredientapp.SCAN");
        intent.putExtra("SCAN_MODE", "ONE_D_FORMATS");
        startActivityForResult(intent, 0);


    }
});
```

Aquí veiem l'implementació que fem de la crida a la llibreria Zxing i li donem el codi 0.

FIGURA 24 RECERCA DE PRODUCTES - RECONEIXEMENT DE CODIS DE BARRES

[12] https://github.com/zxing/zxing
[13] http://es.wikipedia.org/wiki/Callback_%28inform%C3%A1tica%29

```java
/**
 * Fire an intent to start the speech recognition activity.
 */
private void startVoiceRecognitionActivity() {
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);

    // Specify the calling package to identify your application
    intent.putExtra(RecognizerIntent.EXTRA_CALLING_PACKAGE, getClass().getPackage().getName());

    // Display an hint to the user about what he should say.
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT, this.getString(R.string.txtBuscaProdHint));

    // Given an hint to the recognizer about what the user is going to say
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
            RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);

    // Specify how many results you want to receive. The results will be sorted
    // where the first result is the one with higher confidence.
    intent.putExtra(RecognizerIntent.EXTRA_MAX_RESULTS, 1);

    startActivityForResult(intent, VOICE_RECOGNITION_REQUEST_CODE);
}
```

Aquesta és la crida al reconeixement de veu. En aquest cas se li dona com a identificador de crida la constant VOICE_RECOGNITION_REQUEST_CODE.

FIGURA 25 RECERCA DE PRODUCTES - RECONEIXEMENT DE VEU

En l'implementació del mètode callback la signatura es sempre la mateixa i es aquí on intervenen el codis donats. Segons el mètode rebi un codi un altre sabem d'on torna.

```java
/**
 * Recibe los datos de la activitys que se lanzaron en para obtener un resultado.
 * Cada una de ellas tiene un código que las identifica.
 * En este caso hay dos
 * -el reconocimiento de voz
 * -el reconocimientod de códigos de barras.
 */
public void onActivityResult(int requestCode, int resultCode, Intent intent) {
        if (requestCode == 0) {
            if (resultCode == RESULT_OK) {
                String contenido = intent.getStringExtra("SCAN_RESULT");
                //String formato = intent.getStringExtra("SCAN_RESULT_FORMAT");

                if (contenido != null && contenido.length() > 0){
                    procesoBusquedaProducto(contenido);

                }

            } else if (resultCode == RESULT_CANCELED) {
                // Si se cancelo la captura.
            }
        } else if (requestCode == VOICE_RECOGNITION_REQUEST_CODE && resultCode == RESULT_OK) {
            // Fill the list view with the strings the recognizer thought it could have heard
            ArrayList<String> matches = intent.getStringArrayListExtra(
                    RecognizerIntent.EXTRA_RESULTS);

            txtBuscaProd.setText(matches.toString().replace(" ", "").replace("[", "").replace("]", ""));
            procesoBusquedaProducto(txtBuscaProd.getText().toString());

        }

    }
```

FIGURA 26 RECERCA DE PRODUCTES - IMPLEMENTACIO CALLBACK

8.1.3.1 PANTALLA DE PRODUCTE TROBAT

Si el producte es troba a la base de dades es mostra directament la fitxa del producte tot indicant si satisfà les preferències de l'usuari o no. Correspon a la vista activity_producto_encontrado.xml

FIGURA 27 PRODUCTE TROBAT

8.1.3.2 | PANTALLA DE PRODUCTE NO TROBAT

Si un producte no es troba a la BD es pregunta a l'usuari si es vol introduir. Si la resposta es afirmativa es mostra una fitxa de producte en blanc.

Aquesta pregunta es un dialog. Els dialogs en Android es poden personalitzar amb qualsevol layout propi. En aquest cas es un preestablert.

FIGURA 28 RECERCA DE PRODUCTES - PREGUNTA PER AFEGIR NOU PRODUCTE

Cas que l'usuari decideixi crear un nou producte es mostra un formulari per afegir un nou producte. En aquest formulari es volia introduir un OCR per ajudar a l'entrada de dades. Però no funcionava bé i es va descartar. A *ANNEX 1 | DESCARTANT OCR* hi ha un petit resum de les proves.

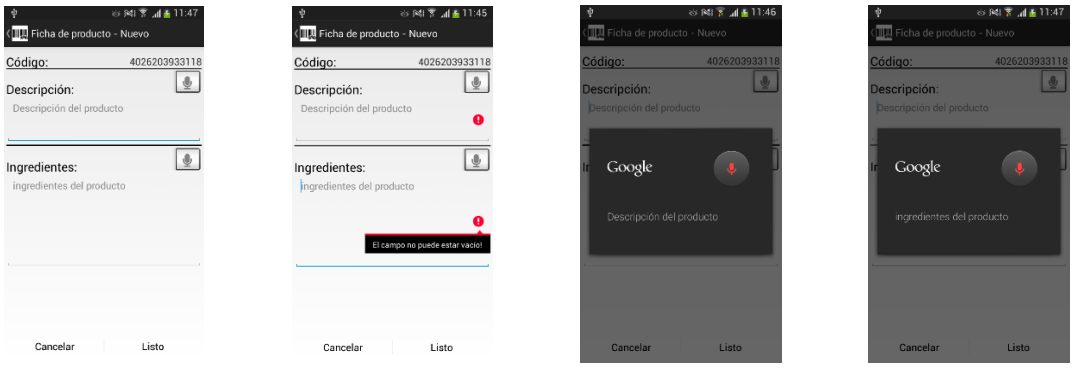


FIGURA 29 PANTALLES NOU PRODUCTE

L'implementació de la pantalla es fa per una banda amb el layout ctivity_producto_nuevo.xml per la part de la vista. I ProductoNuevoActivity.java pel controlador de la vista.

En aquesta vista es pot cridar a dos reconeixements de veus diferents(). Un per la descripció i un altre pels ingredients. Per diferenciar-los, tal com es va fer en la pantalla de recerca de codis, es dona un id diferent a cadascú. Així els diferenciem quan s'executa el callback de reconeixement de veu.

```
//Código del intent de reconocimiento de voz
private static final int VOICE_RECOGNITION_REQUEST_CODE_DESCRIPCION = 1234;
private static final int VOICE_RECOGNITION_REQUEST_CODE_INGREDIENTES = 4321;
```

Els camp de text son obligatoris d'emplenar. Per indicar a l'usuari que ha oblidat d'emplenar-los. Cridem els mètodes `setError`(txt) dels quadres de text. Correspon a l'imatge dos de la figura 29.

8.1.4 | Pantalla de preferències personals/ filtres

Aquesta pantalla es una llista de preferències. En la llista, que te persistència a la base de dades, es poden fer les operacions bàsiques CRUD - **C**reate, **R**ead, **U**pdate i **D**elete.

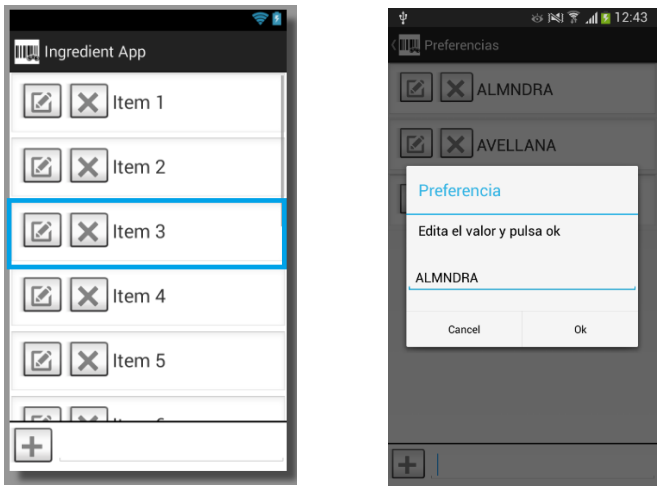


FIGURA 30 PANTALLES PREFERÈNCIES PERSONALS

Aquesta pantalla a diferència de la resta es composa de dues vistes. Una per la pantalla principal (activity_preferencias.xml), que conté el ListView. I un altre pels items del ListView (activity_preferencias_item.xml). A la figura 30 l'imatge dins del requadre pertany a l'item personalitzat del ListView.

Cada item de la llista conte amb dos botons per eliminar-lo o editar el contingut, això últim correspon a l'imatge dos de la figura 30.

L'implementació d'aquesta llista es bastant elaborada. D'una banda tenim un ListView amb elements personalitzats amb una item propi. Això en implica que cada vegada que hem de omplir els elements de la llista Android ha de fer una representació en objectes del XML dels items. S'aconsegueix això amb

Adapter. Aquest Adapter està implementat amb la classe AdapterPreferFiltro que estén de BaseAdapter. Quan es crea l'adapater se li passa com a paràmetre un Array que prèviament hem s'ha carregat amb les preferències des de la base de dades local.

Fins aquí tindríem un ListView estàtic lligat a un ArrayList. El propi ListView ens aporta una barra vertical per desplaçar-nos cas que els elements "surtin" de la pantalla.

Quan s'afegeix un element o modifiquem un element, es crida a notifyDataSetChanged() de l'adapter. Això fa que Android recarregui el ListView amb les dades del Array un altre cop. Així cada vegada que creem, eliminem o modifiquem un element. Es recarrega l'array i es notifica a Android que hi ha hagut canvis. Així aconseguim un ListView dinàmic lligat a la base de dades local. A mes la llista sempre està en ordre alfabètic per ajudar visualment a l'usuari.

També s'ha implementat un Dialog personalitzat per fer les modificacions de les preferències. Correspon a l'imatge dos de de la figura 30.

8.1.4 Pantalla d'actualitzacions

La pantalla d'actualitzacions tanca la darrera de les pantalles del menú principal. Les actualitzacions s'executen manualment. Tenen com a objectiu descarregar productes que altres usuaris han compartit, i compartir els creats pels propis usuaris.

Aquest potser un procés costós depenent del nombre d'elements que s'hagin d'enviar i de rebre. Android te un thread d'execució únic. I és en aquest thread on s'executa l'interfície gràfica, i qualsevol altre càlcul o procés que cridem.

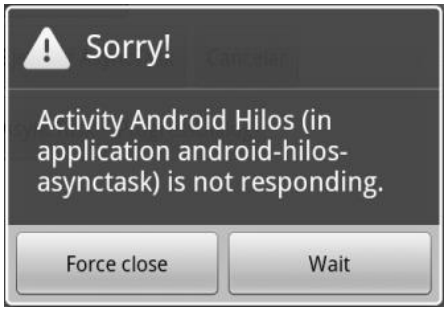


FIGURA 31 ERROR ANDROID NO RESPONDING

Android monitoritza totes les execucions de les aplicacions i cap d'aquestes triga més de 5 segons. Mostra un avís indicant si volem parar l'execució. Aquest és un avís que no pot donar-se a cap aplicació. Ja que fa que l'usuari no confiï en aquesta.

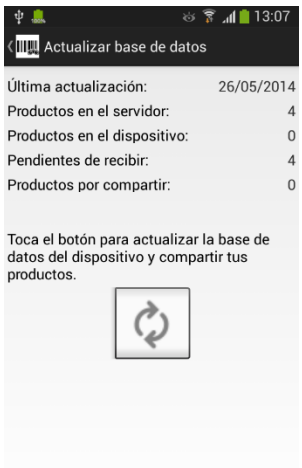


FIGURA 32 PANTALLA D'ACTUALITZACIONS

Per evitar que les tasques d'actualització generin aquest tipus d'error, s'han d'executar explícitament en un fil en segon pla (background). Per a tal fi Android disposa de la classe AsyncTask. És una classe que ens permet interactuar amb el fil (thread) principal d'execució i executar en un fil en background tasques amb cert pes.

La pantalla d'actualitzacions es composa de la vista en l'arxiu 'activity_actualizar_bd.xml' que mostra d'una forma clara l'estat d les actualitzacions. Es veu en concret:

- L'últim dia que es va actualitza el dispositiu
- El nombre de productes al servidor global – Azure
- El nombre de productes al dispositiu
- El nombre de productes pendents de rebre
- El nombre de productes pendents de compartir

Es defineixen quatre tasques asíncrones per fer les actualitzacions que estenen a AsyncTask. Mentre s'executen les actualitzacions no volem que l'usuari interactuï amb l'aplicatiu, però volem que vegi que s'està executant la tasca. Hem d'evitar que pensi que l'aplicatiu no respon a les ordres. Per això es mostrarà un dialog modal que va mostrant en cada moment el que fa i un progrés. De totes maneres en qualsevol moment l'usuari pot prémer el botó de tornar (back) i la tasca s'atura. Les tasques son:

- TareaAsincEstadoActulizacion: Calcula l'estat de les actualitzacions.
- TareaAsincEnviarProdsLocal: Envia els productes que ha creat l'usuari al servidor central.
- TareaAsincRecibirProdsGlobal:Reb els productes que li falten al dispositiu actual.
- TareaAsincInsertarProductosNuevos: Afegeix els productes a la base de dades local.

Totes les tasques tenen una estructura comuna de mètodes que defineix la classe AsyncTask:

- onPreExecute: S'executa abans de començar la tasca. Aquí definim el dialog modal a mostrar.
- onPostExecute: S'executa quan la tasca ha acabat. Aquí llencem la següent tasca per fer-les en sèrie. I mostrem les dades obtingudes.
- onProgressUpdate: Cada vegada que la tasca "avança" s'executa aquest mètode. I fa que el progress bar s'actualitzi i mostri l'usuari el progrés de la tasca global.
- doInBackground: Aquí s'executa el pes de la tasca. La connexió amb el servidor central.

Mentre el mètode doInBackground, tal com el seu nom indica, s'executa en segon pla. El altres tres s'executen al fil principal i per tant poden interactuar amb els controls que veu l'usuari.

En la classe de control 'ActualizarBdActivity.java' es on definim aquestes tasques. El que fan es establir una connexió amb el servidor central. I executar un mètodes allotjats al núvol. Més endavant, en un altre capítol, es detalla la programació del web service que s'ha desenvolupat en C#. Les crides les es fan en format JSON i criden a l'API REST que com dèiem es troba al servidor central.

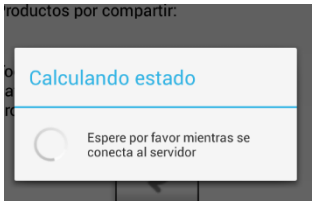


FIGURA 33 PROGESS BAR CALCUL ACTUALITZACIÓ

Cada tasca fa servir un progres bar diferent. Per calcular l'estat de l'actualització es mostra un dialog indefinit ja que no se sap quan pot trigar.

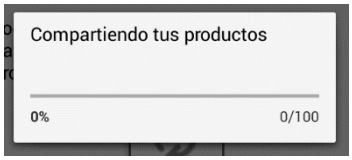


FIGURA 34 PROGRESS BAR COMPARTINT PRODUCTES

Per compartir productes de l'usuari sí es pot mostrar un progrés. A cada iteració la tasca executa OnProgressUpdate(...) i el dialog s'actualitza. Fent saber a l'usuari que la tasca avança.

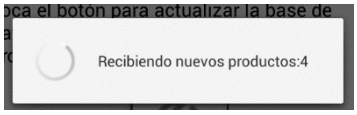


FIGURA 35 PROGESS BAR REBENT PRODUCTES

A l'hora de rebre, podem saber quants productes ens falten. Però no quan trigarem en rebre. Per això ara el dialog es indefinit.

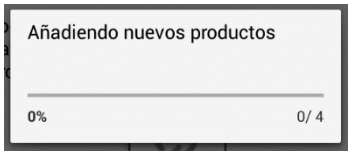


FIGURA 36 PROGRESS BAR AFEGINT NOUS PRODUCTES

Quan comencem a afegir els productes a la base de dades local. Sabem quants hem rebut i per tant podem fer un dialog que avança.

8.2 | Desenvolupament en C# - Windows Azure

La part del servidor central i la base de dades de suport global a l'aplicatiu s'ha construït sota Windows Azure. L 'infraestructura de la pròpia eina ens permet des de un mateix i únic panell de control web muntar màquines virtuals, pagines web, bases de dades compartides, serveis per a mòbils i moltes altres funcions.

Al nostre cas s'ha creat una base de dades i una petita web. La base de dades ens permet compartir amb tots els dispositius els productes. I la web ens permet mostrar des d'ordinadors convencionals o qualsevol navegador els productes que hi han.

A la mateixa web es pública una Api REST per tal que els dispositius executin els serveis web per entregar i rebre productes.

L'idea primera era de muntar una web elaborada amb estadístiques sobre producte més consultats, filtres definits pel usuaris etc. Tot i que la web està publicada, es només una eina més per depurar el funcionament de l'aplicatiu. De totes formes es consultable.

Tant la web MVC com l'Api rest s'ha fet en un únic projecte. Que després es publica a Windows Azure des del mateix Visual Studio. L'estructura de carpetes es mostra a continuació.

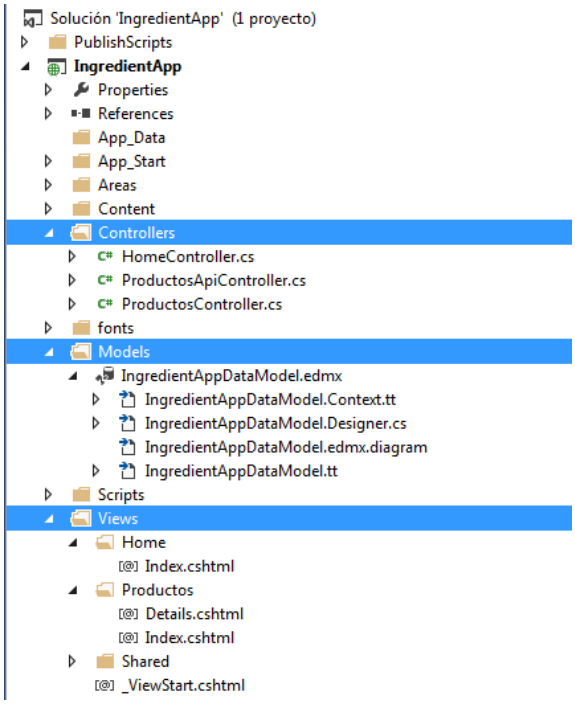


FIGURA 37 ESTRUCTURA DE CARPETAS DE LA SOLUCIÓ C#

Es pot veure a la figura com hi han tres parts diferenciades.

- Els controladors
- Els models
- Les vistes

L'Api REST es un controlador més.

8.2.1 | Web MVC de visualització

Veiem una mica mes en detall com funciona MVC.

Model: Son classes que interactuen amb la base de dades de l'aplicació. Al nostre cas hi ha només un model. Es l'entitat que representa la taula de productes global.

Vistes: Son les representacions que fem dels models. Les pàgines web.

Com que es una aplicació només de consulta en tenim tres:

- Home: que és la plana web d'inici
- Index: que és una plana web que representa un llistat de tots els productes.
- Details: aquesta plana web mostra els detalls d'un producte en concret.

Controllers: El controladors recullen les peticions del usuaris, processen les dades amb els models, i tornen a l'usuari una vista amb els resultats. Son les classes que tenen els mètodes. Aquests mètodes responen al verbs bàsics del protocol Http. Com son GET, POST, PUSH i DELETE.

Igual que fa Android MVC separa totalment la lògica de les dades. Així les vistes es poden personalitzar sense have de tocar el codi funcional de l'aplicatiu.

8.2.1.1 | ESTRUCTURA DE LA BASE DE DADES (MODEL)

La base de dades es crea directament des del panell de control de Windows Azure de forma visual.

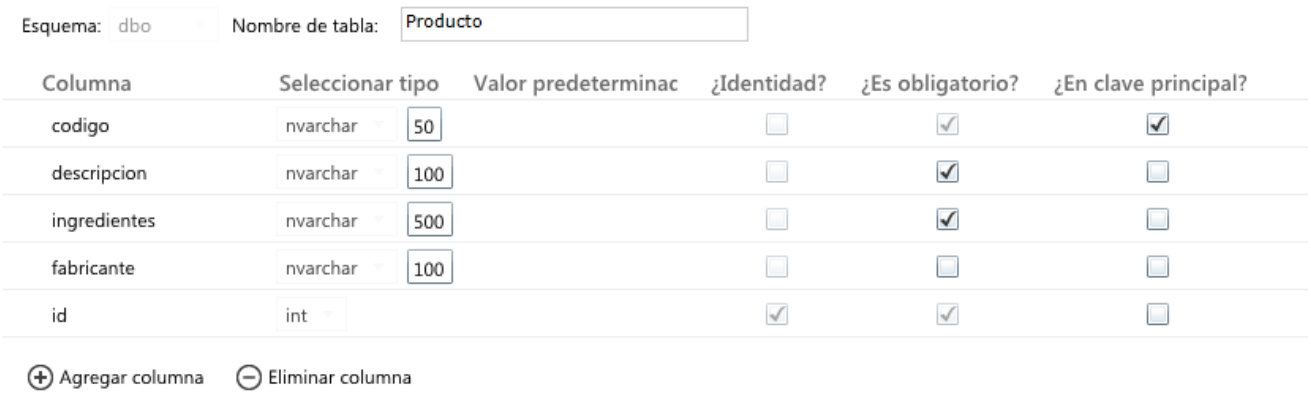


FIGURA 38 BASE DE DADES GLOBAL - AZURE

Una vegada creada importem al projecte de Visual Studio les entitats que es transformaran al nostres model de dades del projecte C#.

8.2.1.2 | VISTES

Hi han dues vistes tal com pels productes i una per la plana web principal. Aquestes son les que veu l'usuari quan entra a la pàgina web.

El arxiu que les conformes son Productos/Index.cshtml i Productos/Details.cshtml. Per la plana principal Home/Index.cshtml

Podem veure les vistes si accedim directament a la web publicada la direcció es:

http://ingredientapp.azurewebsites.net

Aquí accedirem a la vista principal. Des de aquí podem anar a la llista de productes i a la descripció de l'Api. L'aparença es bastant bàsica ja que només s'ha implementat coma eina de suport. L'Api no estaria a vista de tothom en una aplicació real.

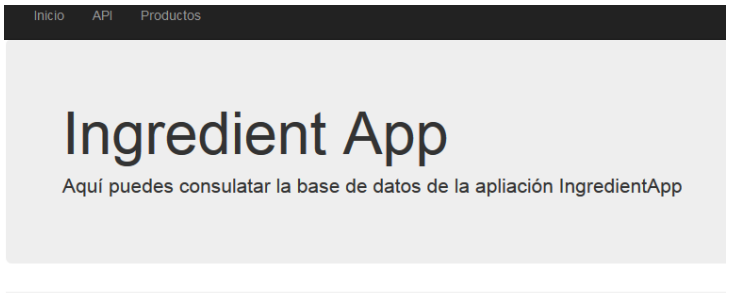


FIGURA 39 PLANA WEB DE INGREDIENT APP

Si anem a la vista de productes veurem un llistat del que hi ha actualment.

FIGURA 40 LLISTA DE PRODUCTES WEB

Si cliquem el link de la dreta accedim al detalls dels productes. Es mostra una fitxa individual.

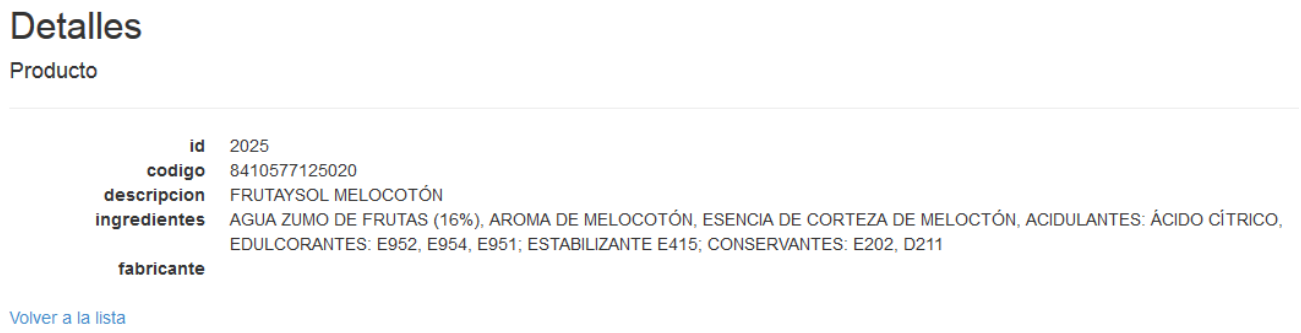


FIGURA 41 DETALLS D'UN PRODUCTE A LA WEB

8.2.1.3 | Controladors de la web

Les classes que reben els requests dels usuaris sons els controladors. Per una web tan senzilla només s'ha utilitzat un. Es l'arxiu ProductosController.cs

```
namespace IngredientApp.Controllers
{
    public class ProductosController : Controller
    {
        private IngredientApp_dbEntities db = new IngredientApp_dbEntities();

        // GET: Productos
        public ActionResult Index()
        {
            return View(db.Producto.ToList());
        }

        // GET: Productos/Details/5
        public ActionResult Details(string id)
        {
            if (id == null)
            {
                return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
            }
            Producto producto = db.Producto.Find(id);
            if (producto == null)
            {
                return HttpNotFound();
            }
            return View(producto);
        }
```

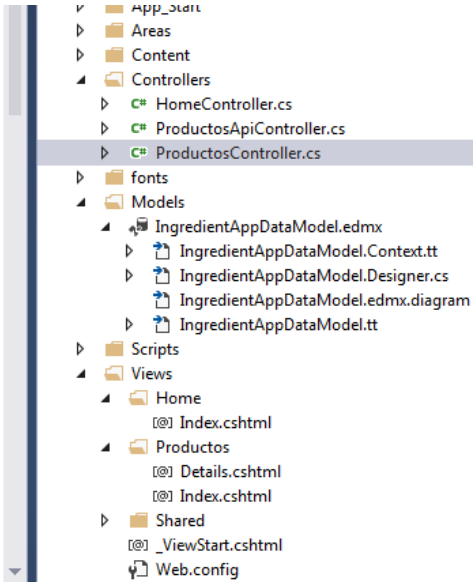


FIGURA 42DETALL DEL CONTROLADOR DE PRODUCTES WEB

Només hi han dos mètodes. Tots dos responen al verb http GET. El primer es el mètode Index() que no rep cap paràmetre. El que fa es enviar de tornada al l'usuari la vista de llista de productes amb tots els productes existents a la base de dades.

El segon que es pot veure a la figura 41 s'anomena Details(id). Rep com a paràmetre el identificador del producte a mostrar. Torna a l'usuari la vista de detalls amb les dades del productes escollit.

8.2.2 | Api REST

Per muntar aqueta Api que els dispositius Android executen utilitzem la mateixa web de consulta del punt anterior.

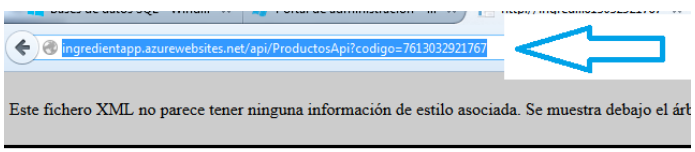
En realitat el funcionament és molt semblant al de una web MVC. Ara només hi ha un controlador, ja que no hi ha part gràfica. Els dispositius fan les crides directament al controlador i aquest els hi torna les dades.

El nostre cas l'Api la fem servir per actualitzar els dispositius mòbils e introduir nous productes a la base de dades global.

8.2.2.1 EL CONTROLADOR DE L'API.

Tots es codifica en un sol arxiu de controlador es ProductosApiController.cs

L'aplicatiu tant pot tornar dades en format JSON, com en format XML. Es en el moment d'executar el request quan se li diu al controlador el format escollit. En el nostre cas es JSON.



Si per exemple fem la crida des de un navegador convencional veurem les dades en XML

```
- <Producto>
    <codigo>7613032921767</codigo>
    <descripcion>TOMATE FRITO SOLÍS</descripcion>
    <fabricante i:nil="true"/>
    <id>2028</id>
  - <ingredientes>
      TOMATE Y CONCENTRADO DE TOMATE, ACEITE DE GIRASOL, AZÚCAR, ALMID
    </ingredientes>
  </Producto>
```

FIGURA 43 CRIDA AL SERVEI WEB DES DE EL NAVEGADOR

veiem els mètodes de l'Api REST que s'han fet servir per l'aplicació des dels els dispositius Android.

`GetTotalProducto()`: Aquest mètode torna el nombre total de productes existents a la base de dades. Es fa servir per calcular el nombre de productes pendents de rebre.

GetProductoFrom(int id): L'únic paràmetre que rep es un identificador del darrer productes que es va baixar el dispositiu. Llavors torna un Array amb la resta de productes a partir del que rep. Es el mètode que es crida per fer les actualitzacions.

PostProducto(Producto producto): Aquest mètode es fa servir per rebre nous productes des dels dispositius mòbils. Rep directament un objecte producte.

Aquest producte es rep via POST, de nou es pot rebre en format JSON o XML. El propi ASP. Net MVC disposa d'un element encarregat de serialitzar les dades per fer-les arribar al mètode en forma d'objecte.

Hi ha altres mètodes a l'Api que no es fan servir en l'aplicació. Son per actualitzar un producte, eliminar-lo, consultar etc. Es pot veure la descripció al menú Api de la web.

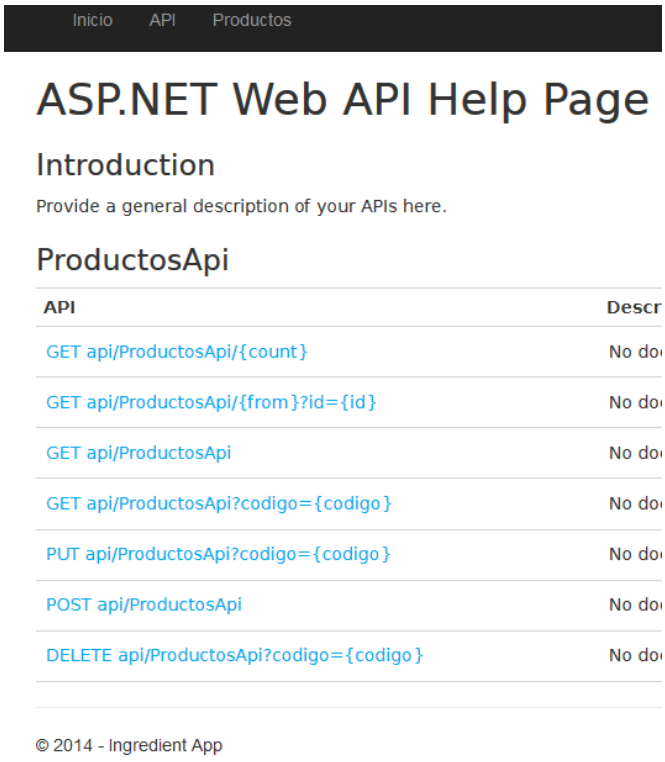


FIGURA 44 API WEB REST

9 | CONCLUSIONS

El desenvolupament del treball ha estat un repte considerable. Un punt fonamental subestimat al principi del treball ha sigut la corba d'aprenentatge de la pròpia plataforma Android. Els coneixements previs en altres plataformes han ajudat, però fer la "traducció" d'aquests a Android ha sigut una tasca llarga i costosa.

Per la part personal estic molt satisfet dels coneixements assolits sobre Android en tan poc temps. Com passa amb els primers programes fets en altres plataformes. Una vegada has acabat, tornaries a fer-los

des del principi. Es un dels principis DCU que comparteixo de forma absoluta. L 'iteració continua en l'anàlisi, disseny e implementació.

9.1 Objectius assolits

Els objectius inicials s'han complert parcialment. Una de les idees principals de l'aplicació era introduir els ingredients amb un OCR. Aquest part s'ha hagut de descartar per la poca fiabilitat en petits texts. A l'annex 1 s'expliquen els detalls.

Sense la part del OCR es molt probable que l'usuari decideixi no fer anar una aplicació d'aquests tipus. La feina per entrar els ingredients es considerable. Tot i que s'ha implementat el reconeixement de veu, que es una millora no establerta al principio del treball, i funciona correctament.

La resta de funcionalitats sí s'han completat. Es poden consultar i compartir productes i s'ha publicat una Web convencional, i una Api que es podria compartir amb altres aplicacions. D'aquesta manera es podria compartir la base de dades amb aplicacions afins. Aquest era un punt que es mostrava com a opcional al principi del treball.

L'objectiu principal i motivació d'escollir aquesta àrea de treball, aprendre el principis d'Android, s'ha assolit totalment. I tot i que encara falta molt per aprendre, ja tinc al cap nous projectes. El primers passos son els que costen més i aquest ja s'han fet. Professionalment he obert un nou camí amb molta demanda actual i estic molt satisfet.

9.2 Possibles millores

Tot i que penso que una aplicació d'aquest tipus es molt necessària, si no es parteix d'una base de dades ja conformada. Potser amb els fabricants, potser amb l'ajuda d'alguna gran plataforma de distribució (Carrefour, Alcampo, Caprabo). No crec que tingui una bona acceptació.

Una millora fonamental, o fins i tot una funcionalitat no establerta, es la connexió amb les xarxes socials. Si s'estén una idea en les xarxes socials qualsevol idea pot arribar a materialitzar-se. Poder saber qui ha introduït un producte, o marca un producte com erroni, etc. Es podria fer amb tecnologies d'identificació in accés d'usuaris com OAuth[14], OpenID[15].

La tecnologia Android es mòbil i actualment tot es mòbil. Crec que estar al dia amb Android per un professional es estar al dia en tecnologia.

FONTS CONSULTADES/ BIBLIOGRAFIA

- Curs Android | Salvador Gómez.
  - http://www.sgoliver.net/blog/?page_id=3011
- Videotutorial Android | Jesús Conde
  http://www.youtube.com/watch?v=2zJD4ASpfW8&list=TL89u2xmPDCkQbntFbcxbtJgkiLTkkf_jI

[14] OAuth http://es.wikipedia.org/wiki/OAuth
[15] OpenID http://es.wikipedia.org/wiki/OpenID

- Icones e imatges
  - http://findicons.com
  - http://www.icons4android.com/

- Llibreria per llegir codis de barres - Zxing
  - https://github.com/zxing/zxing
  - http://a3dany.blogspot.com.es/2013/08/zxing.html
- Llibreria per reconeixement de text OCR – Tesseract | Tess-Two
  - Navarro Santapau, Jaime. Repositori UOC
    - http://openaccess.uoc.edu/webapps/o2/handle/10609/18691?mode=full
  - Tesseract https://code.google.com/p/tesseract-ocr/
  - Tess-Two https://github.com/rmtheis/tess-two
  - OCR Test https://github.com/rmtheis/android-ocr
- Android Developer
  - http://developer.android.com/training/index.html
- ASP Net MVC & Api
  - http://www.asp.net/mvc
- Windows Azure
  - http://azure.microsoft.com/es-es/documentation/

ANNEX 1 | DESCARTANT OCR

Es va consultar un molt bon treball anterior d'un company[16]. Segons les conclusions del treball esmentat la millor opció per Android es fer servir la llibreria Tesseract[17], i en concret un fork anomenat d'aquesta anomenat Tess-two[18].

Després d'avaluar-la va descartar aquest via. Una de les primeres idees era ajudar-se del OCR al introduir els productes. D'aquesta manera s'aconseguia que l'usuari final no hagués d'introduir manualment ingredients. A més s'evitarien errors en la introducció dels mateixos.

Per descartar el OCR es va compilar la llibreria i es va baixar un exemple OCR-Test, i es va configurar el diccionari per reconèixer texts en espanyol. Es van provar diferents llistes d'ingredients, petits textos i es va acabar per descartar.

S'ha de dir que el context d'us es molt diferent al real. Es van fer les proves a casa meva amb condicions de llum correcte i molta calma. En una prova real a un supermercat a mà alçada i amb poc temps, crec que seria una mala experiència per l'usuari.

Veiem unes imatges de les proves:

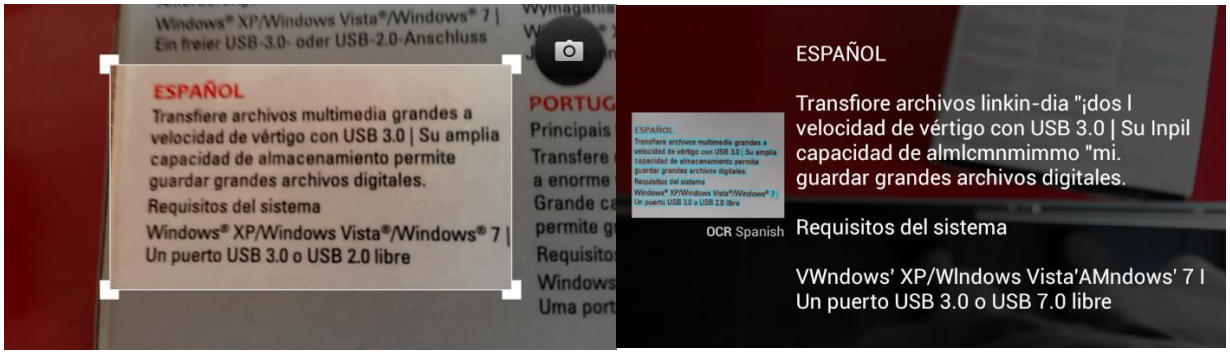


FIGURA 45 PROVES OCR 1

En aquest cas no es tracta d'una llista d'ingredients sinó d'un text descriptiu del producte. Tot i el text reconegut s'apropa a la realitat. No es el correcte.

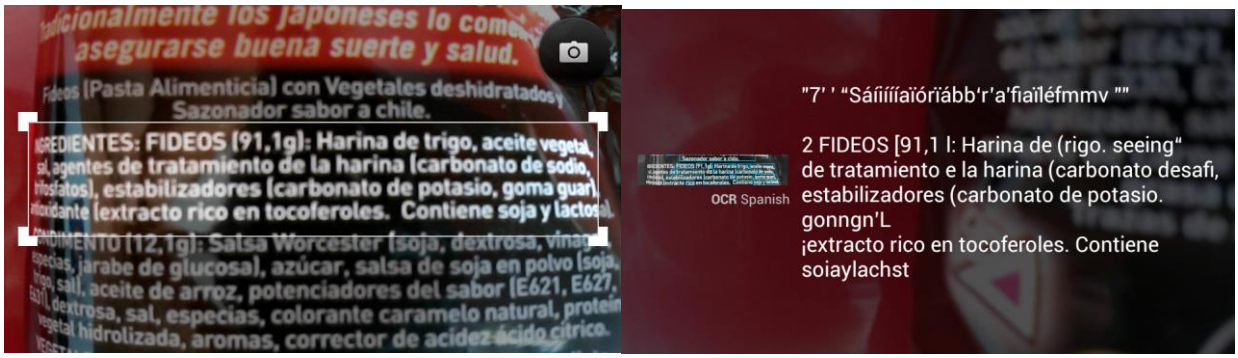


FIGURA 46 PROVES OCR 2

[16] Software de adquisición de imágenes y reconocimiento óptico de caracteres para Android http://openaccess.uoc.edu/webapps/o2/handle/10609/18691?mode=full
[17] Tesseract – OCR https://code.google.com/p/tesseract-ocr/
[18] Tess-two https://github.com/rmtheis/tess-two

El problema amb aquesta llista es el suport. Plàstic amb corbes i molt brillant que fa difícil apuntar la càmera correctament.

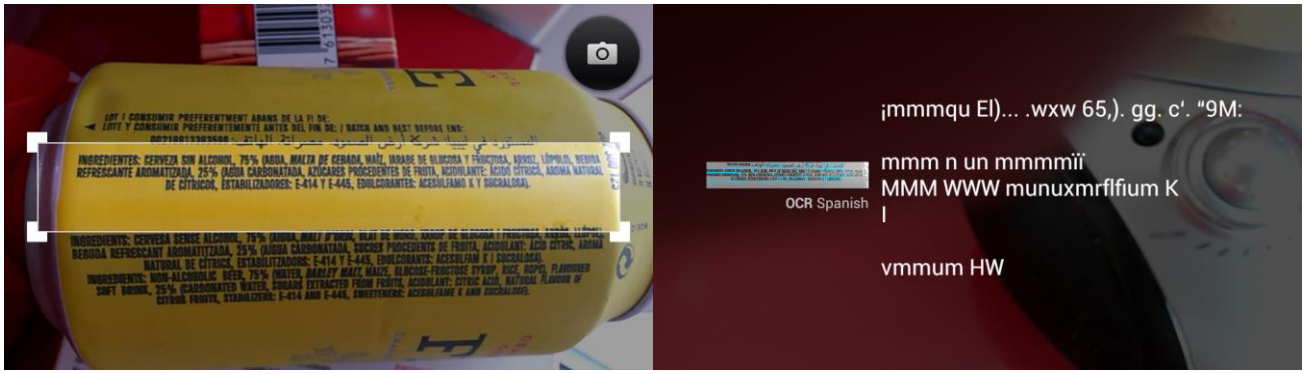


FIGURA 47 PROVES OCR 3

Potser en aquest cas el problema es el fons o la tipografia estreta. Malauradament en les llistes d'ingredients existeixen tot tipus de dissenys i tipografies.

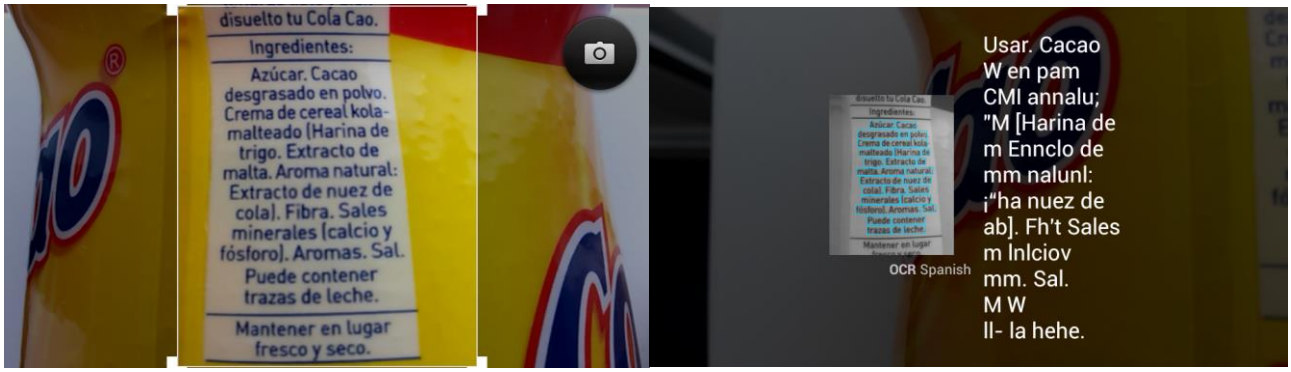


FIGURA 48 PROVES OCR 4

En aquest cas potser el problema l'orientació del text.

Després d'aquestes proves es va haver de descarta l'idea de fer servir aquesta funcionalitat en l'aplicatiu. Tot i que en altres textos i circumstàncies el seu us estigui recomanat.