Treball Final de Grau:

ANÀLISIS I DISSENY D’UN SISTEMA DE COMANDES DIGITAL PER A UN RESTAURANT

Author:
Alan Vives Invernón

Tutor:
Lluís Pérez Vidal

Departament:
Ciències de la computació

25 de juny de 2017
Resum

Els darrers anys el món de l’hostaleria cada vegada ha pres més protagonisme en la societat, la gent surt més de casa i cada vegada hi ha més restaurants. Aquest augment de competència ha forçat a molts restauradors a innovar per tal d’ofrir un producte diferent i un millor servei per tal de mantenir la seva posició respecte a la competència.
Amb l’incorporació dels telèfons intel·ligents en la societat, han estat molts els restauradors que han optat per l’introducció d’alguns d’aquests dispositius per tal d’agilitzar els processos. Tot i això, molts continuen pensant que l’ús de la llibreta i el paper ofereix més avantatges davant els softwares vigents al mercat.

Aquest projecte té com a objectiu desenvolupar una aplicació per a dispositius mòbils basada en Android per tal de realitzar comandes en un restaurant. Alhora també es pretén realitzar un programa d’escriptori en llenguatge Python 2 per tal de complementar les funcionalitats requerides en la presa de comandes.

És evident que no tots els establiments tenen les mateixes necessitats en el procés de realitzar una comanda, per aquest motiu el desenvolupament ha estat focalitzat en satisfacer els requisits d’un restaurant tradicional de forquilla. Més concretament s’ha tingut com a model de confecció el restaurant Les Espelmes de Valls. S’analitzarà el sistema actual i es proposarà una alternativa amb una mecànica d’ús familiar per als treballadors.

Al llarg d’aquesta memòria s’explica l’anàlisi previ requerit per tal d’entendre el funcionament del restaurant, l’ecosistema sobre el qual es desenvoluparà l’aplicació i les eines utilitzades per tal de satisfacer les necessitats dels usuaris.

L’aplicació obtinguda en aquest projecte no està preparada per a una implementació en un cas real, sinó que és un prototip en el qual cal continuar treballant. S’han indicat un seguit de millores que bé o per falta de temps o bé per la seva alta complexitat no s’han introduït en aquest prototip però caldria implementar en futures versions del software.
Índex

Taula de figures .......................... 5
Taula de taules .......................... 7

1 Introducció ........................................ 9
  1.1 Introducció ........................................ 9
  1.2 Motivació ........................................ 9
  1.3 Objectius ...................................... 10
  1.4 Abast .......................................... 10
  1.5 Fases de realització .................. 10

2 Estat de l’art .................................. 11
  2.1 Exemples d’aplicacions existents .... 11

3 Anàlisis previ .................................... 15
  3.1 Introducció .................................... 15
  3.2 Estructura del restaurant .............. 15
  3.2.1 Rols del personal ......................... 15
  3.2.2 Distribució del restaurant .......... 15
  3.2.3 Mode operacional del cambrer ..... 16
  3.2.4 Estructura d’una comanda .......... 17
  3.3 Android ..................................... 19
  3.3.1 Versions del sistema operatiu .... 19
  3.3.2 Arquitectura d’Android ............... 20
  3.3.3 Llenguatges de programació ...... 21
  3.4 Android Studio ............................. 23
  3.4.1 Estructura ................................. 23
  3.4.2 Arxius Java ............................... 24
  3.4.3 Arxius XML ............................... 26
  3.5 Python ...................................... 27

4 Aplicació ........................................ 28
  4.1 Eines de desenvolupament .......... 28
    4.1.1 Android Studio ....................... 28
    4.1.2 Dropbox ............................... 28
    4.1.3 Python i Emacs ....................... 29
  4.2 Requisits per executar l’aplicació . 29
  4.3 Requisits funcionals dels usuaris .... 29
    4.3.1 Cambrer ................................ 29
    4.3.2 Propietari ............................. 30
    4.3.3 Cuiners ................................ 30
  4.4 Casos d’ús ................................. 30
    4.4.1 Cambrer ................................ 31
    4.4.2 Propietari ............................. 32
  4.5 Interfície gràfica ......................... 33
    4.5.1 Pantalla principal .................... 34
Índex

4.5.2 Introducció d’una nova taula .......................................................... 35
4.5.3 Edició de la comanda ................................................................. 36
4.6 Estructura de les bases de dades ....................................................... 38
4.7 Sincronització ................................................................................. 38
4.8 Gestió de les dades en l’ordinador .................................................... 39

5 Implementació ....................................................................................... 42
5.1 Funcions .......................................................................................... 42
5.1.1 Bases de dades ............................................................................ 42
5.1.2 Creació de llistes lliscants .......................................................... 44
5.1.3 Buscador de plats ....................................................................... 48
5.1.4 Sincronització ............................................................................ 50
5.1.5 Tractament de dades amb python .............................................. 52

6 Planificació ........................................................................................... 54
6.1 Diagrama de Gantt .......................................................................... 54

7 Costos .................................................................................................... 56
7.1 Recursos humans ............................................................................ 56
7.2 Material ......................................................................................... 56
7.3 Cost total del projecte ...................................................................... 57

8 Impacte ambiental ................................................................................ 58
8.1 Impacte ambiental en l’execució del projecte .................................... 58
8.2 Impacte ambiental de l’implementació ............................................. 58

9 Feina futura ........................................................................................... 59

10 Conclusions, resultats i perspectives de futur ..................................... 60

Bibliografia .............................................................................................. 63
Índex de figures

2.1 Aplicació TPV 123 Hosteleria [9] ........................................ 12
2.2 Aplicació GlopDroid [5] .................................................. 12
2.3 Aplicació Olicomanda [8] ................................................ 13

3.1 Croquis de la distribució .................................................. 16
3.2 Exemples de comandes .................................................. 18
3.3 Partició d’Android ......................................................... 19
3.4 Pila de software d’Android .............................................. 20
3.5 Estructura del projecte en Android Studio ....................... 23
3.6 D’esquerra a dreta: Navega endarrere, retorn a la pantalla principal, aplicacions utilitzades recentment ........................................... 24

4.1 Logotips de les eines utilitzades ...................................... 28
4.2 Diagrames de casos d’ús .................................................. 30
4.3 Pantalla principal de l’aplicació ....................................... 34
4.4 Introducció de les dades d’una nova taula ......................... 35
4.5 Creació d’una comanda .................................................. 36
4.6 Revisió d’una comanda creada ........................................ 37
4.7 Esquema de la sincronització dels fitxers ......................... 39
4.8 Captura de la vista de taules ........................................... 39
4.9 Factura de la taula ......................................................... 40
4.10 Llistat d’estadístiques d’disponibles ............................... 40
4.11 Resums estadístics d’interès .......................................... 41

5.1 Visa de la simple_list_item_2 ........................................ 45
5.2 Buscador de plats ......................................................... 50

6.1 Diagrama de Gantt ......................................................... 55
Índex de taules

3.1 Llistat de versions d'Android .................................................. 19
6.1 Durada d'hores previstes pel projecte ........................................ 54
7.1 Desglossament dels costos en recursos humans ......................... 56
7.2 Desglossament dels costos en material .................................... 57
7.3 Cost total del projecte ............................................................... 57
Glossari

- **API**: *Application Programming Interface*, conjunt de subrutines, funcions i procediments que ofereix una biblioteca per a ser utilitzat per un altre software.

- **dpi**: *dots per inch*, en català abreviat ppp (punts per polzada). És una unitat de mesura per a resolucions.

- **Hardware**: conjunt d’elements físics d’un sistema informàtic.

- **Kernel**: Núcli del sistema operatiu. Responsable de facilitar als diferents programes accés al hardware.

- **MIT**: *Massachusetts Institute of Technology*, institut tecnològic de Massachusetts.

- **UTF-8**: *8-bit Unicode Transformation Format*, és un mètode de codificació de caràcters de longitud variable per a Unicode.

- **Software**: conjunt d’instruccions que permeten al usuari interactuar amb el hardware i realitzar tasques.

- **XML**: *Extensible Markup Language*, és un metallenguatge de marques desenvolupat pel W3C. Aquest es proposa com un estàndard per a l’intercanvi d’informació estructurada entre diferents plataformes.
Capítol 1

Introducció

1.1 Introducció

El sector de l’hostaleria és un dels més immobilitzats de l’actualitat. Molts dels restaurants segueixen amb els mètodes tradicionals tant a dins com a fora la cuina. A les botigues d’aplicacions es troben diferents aplicacions que prometen realitzar aquesta tasca, però es queden cures de personalització o funcionalitat. Per aquest motiu si es vol instal·lar un sistema de comandes digital cal contractar una empresa perquè personalitzi el seu software a les necessitats del restaurant i no tots els negocis s’atreveixen a fer front a una inversió d’aquesta magnitud.

L’objectiu d’aquest projecte és desenvolupar una sistema digital per realitzar comandes en un restaurant, i adaptar-lo als requeriments funcionals del restaurant Les Espelmes de Valls. L’ecosistema sobre el qual es desenvoluparà aquest projecte es divideix en dos: Android per tal de programar un dispositiu mòbil amb el qual es prendran les comandes, i Python per tal de rebre aquestes comandes i gestionar-les adequadament.

1.2 Motivació

Al llarg dels meus últims anys, he tingut molt clar que vull ser enginyer. Però en l’àmbit familiar la gastronomia sempre ha estat molt present. La restauració és una feina molt dura, són moltes hores de feina en la que hi poses tot l’esforç possible i no saps com l’endemà et pots tornar a aixecar per una pallissa igual. Aquest fet el visco des de molt a prop ja que pares, tíets, cosins i germans treballen en un mateix negocí familiar, el restaurant Les Espelmes de Valls.

La meva tasca dins del restaurant conjuntament amb la meva germana és una mica semblant a una navalla suïssa, gestionem les reserves, rebem a la gent a l’entrada, fem cafès, rentem coping i passen les notes a l’ordinador. Aquesta última tasca ocupa molts de temps i és la més empipadora ja que amb les presses i l’estres del moment fan que la lletra de moltes comandes sigui confusa o pràcticament illegible, per aquest motiu des de fa temps hem estat pensant en introduir un sistema de comandes digital per al restaurant però no s’ha arribat a implementar mai.

Com a enginyer industrial vaig realitzar l’assignatura d’informàtica els primers anys d’estudis i me’n recordo d’una frase que he anat sentint al llarg dels anys: Si saps programar en un llenguatge, és fàcil aprendre un altre de nou, tots s’assemblen. Bé, estic disposat en posar a prova aquest fet de primera mà i com a enginyer, molt motivat per aprofundir en un camp tan interessant com són les aplicacions Android.
1.3 Objectius

Automatitzar la gestió de comandes d’un restaurant. Per tal d’assolir aquest fet es desenvoluparà una aplicació en l’ecosistema Android per a dispositius mòbils i un programa en Python per a un ordinador de sobretaula que permetin:

- Prendre una comanda des de diferents dispositius mòbils.
- Realitzar una gestió de les comandes intuïtiva i fàcil d’entendre per al usuari.
- Incorporar totes les funcions bàsiques que es poden necessitar en la presa de comandes.
- Establir una comunicació entre el dispositiu mòbil i l’ordinador de sobretaula.
- Poder modificar les comandes des del ordinador.
- Incorporar en l’ordinador un análisis estadístic de les dades.

Amb l’incorporació d’aquestes funcions es pretén disminuir els temps d’espera dels comensals i maximitzar la productivitat dels empleats al llarg d’un servei en eliminar la tasca d’introduir les notes manualment a l’ordinador.

1.4 Abast

El projecte pretén crear una aplicació amb totes les funcionalitats bàsiques per a prendre nota en un restaurant i establir una connexió entre la base de dades del dispositiu mòbil i un programa d’ordinador. El programa està pensat per gestionar totes les comandes preses des dels diferents dispositius mòbils.

1.5 Fases de realizació

En aquest apartat es farà menció a les diferents fases que es s’han seguit en el desenvolupament del treball.

- Estudi previ: Investigació sobre la tecnologia disponible i estat de la competència.
- Anàlisis de les funcions bàsiques que ha d’implementar l’aplicació.
- Disseny que definirà l’aspecte visual de l’aplicació.
- Programació d’aquesta
- Proves per tal de comprovar el seu correcte funcionament.
- Documentació d’aquesta tant a nivell de codi com en document per tal de facilitar-ne la lectura i la posterior edició.
Capítol 2

Estat de l’art

És habitual en aquest tipus d’oficis rebre alguna queixa sobre el servei d’atenció al client degut a un excessiu temps d’espera. Aquest fet pot estar desencadenat per diferents factors: o bé el cambrer està atent a altres taules i no té temps per atendre unes altres, també és possible que el cambrer pensi que ha notificat a cuina la realització dels plats d’una taula sense haver-lo fet, o la cuina no aconsegueix endreure la comanda realitzada pel cambrer i es generen confusions. Però com que aquestes problemàtiques sempre han estat presents en qualsevol empresa d’aquest sector, és normal que s’hagin plantegiat diferents alternatives tecnològiques per tal d’agilitzar el procés i minimitzar el número d’errors cometes.

En l’època de la telefonia mòbil intel·ligent s’ha creat una oportunitat per a diferents empreses per oferir una aplicació que pugui solucionar aquesta problemàtica. Tot seguit s’analitzaran algunes d’aquestes i els seus respectius serveis.

2.1 Exemples d’aplicacions existents

En l’actualitat predominen 2 sistemes operatius en els dispositius mòbils, Android i iOs. Com que per motius posteriormen exposats aquest treball està centrat en l’ecosistema Android, només s’analitzaran aplicacions pertanyents en aquest ecosistema.

En la botiga d’aplicacions d’Android (Google Play Store) hi han més de 2,5 M d’aplicacions disponibles per als usuaris de la plataforma, així doncs, és obvi que la aplicació que es pretén desenvolupar no és la primera del sector i ja hi han aplicacions que ofereixen serveis similars. Cercant entre aquest mar d’aplicacions s’han trobat bastantes aplicacions focalitzades en la presa de comandes en restaurants però en descarregar aquestes moltes d’elles no permeten al usuari veure com és la mecànica de funcionament sense subscriure’s en els seus serveis mensuals. Per raons òbvis questes no s’han analitzat i només s’han tingut en compte les que són gratuïtes o incorporen un període de prova previ a la compra. Les més destacables són:

- TPV 123 Hosteleria
- GlopDroid
- Oñicmanda
- FoodZaps
TPV 123 Hosteleria mostra les taules i els plats en forma de graella amb uns icones molt grans, això permet reduir el nombre de possibles errors comuns en prendre una comanda. Un cop finalitzada la comanda permet fer una revisió d’aquesta en forma de llista amb la possibilitat d’editar el número de plats directament des d’aquesta. Com a part negativa, no permet fer una diferenciació dels primers i segons plats de la taula.
Preu: 95.59€/any

GlopDroid utilitza la visualització en forma de graella només en la disposició de taules, mentre que opta per mostrar els plats i la comanda en forma de llista. També ofereix la possibilitat de comentar els plats o separar la comanda segons la zona de cuina a on va dirigir. Malgrat que ofereix bastanta personalització dels plats, la utilització d’aquesta és lenta i confusa.

Preu del producte TPV Hosteleria Android
Capítol 2. Estat de l’art

13

(a) Llistat de taules  
(b) Llistat de plats  
(c) Vista de la comanda

Figura 2.3: Aplicació Oficomanda [8]

Oficomanda es decanta per la visualització en forma de graella per a les taules i els plats, malgrat això, no permet afegeir comentaris als plats i la visualització d’aquests es pot confondre fàcilment amb les categories.

Aquesta té un preu inicial de 100€ per a l’adquisició de la primera llicència del producte i 80€ per l’adquisició de llicències addicionals. Per al manteniment d’aquestes cal realitzar un pagament mensual de 9€/més per la primera llicència i 7€/més per al manteniment de llicències addicionals.

(a) Llistat de taules  
(b) Llistat de plats  
(c) Vista de la comanda

Figura 2.4: Aplicació FoodZaps [4]

Finalment l’aplicació FoodZaps ofereix una vista per a les taules en forma de graella i per a la revisió de la comanda en forma de llista, mentre que en la vista de selecció de plats es pot escollir si es vol en forma de graella o en forma de llista.

Aquesta també incorpora diferents filters que permeten veure tots els plats del establiment o només
les categories escollides, un buscador de plats pel seu nom i addició de comentaris en cadascun dels plats.

Com a funcionalitat extra, aquesta també permet realitzar un control d’inventari i un resum dels ingressos diaris.

Preu: 39,90€/mes

Tal i com es pot veure, moltes de les aplicacions existents al mercat tenen bastants inconvenients que impossibiliten l’expansió d’aquests sistemes en front a les llibretes de notes tradicionals. També cal destacar que d’aquestes opcions analitzades, TPV 123 i FoodZaps són les úniques que permeten la integració d’un ordinador per tal de realitzar factures i modificacions més cómodament.
Capítol 3

Anàlisis previ

3.1 Introducció

En quest capítol es realitza un anàlisis dels diferents pilars bàsics que conformen aquest projecte així com són el funcionament d’un restaurant i l’estructura d’Android i Python.

3.2 Estructura del restaurant

3.2.1 Rols del personal

Per tal d’entendre el funcionament del restaurant, primerament cal veure com s’organitza i quins són els diferents rols dins del restaurant.

• Cuiners: Diferenciat segons la seva especialitat (primers, segons, brasa o postres).
• Renta plats: S’encarreguen de rentar els plats i estris de cuina.
• Cambrers: Separats per menjadors, cada cambrer porta el control únicament de les taules del seu menjador.
• Passa-plats: Ajuden als cambrers a passar els plats a les diferents taules.
• Barra: Introdueixen les notes dels cambrers al ordinador, gestionen les reserves, renten les copes i fan cafès i combinats.

3.2.2 Distribució del restaurant

En la figura 3.1 es mostra un croquis del restaurant, en aquest es pot veure com la complexa distribució dels menjadors. El restaurant té una capacitat mitjana d’unes 120 persones i les taules estan organitzades de la següent manera:

Menjador 1: Taules de la 1 a la 7.

Menjador 2: Taules de la 21 a la 25, a més a més d’un reservat amb el nom de 200.
La numeració del menjador 3 es complementa amb la del menjador 4 i l’ordre de les taules està anomenat en sentit antihorari i segons el cambrer es troba les taules quan surt de la cuina. En aquests menjadors el número d’una taula pot anar seguit d’un guió amb la lletra P, M o F corresponent a abreviacions de Passadís, Mig i Finestra.


Menjador 4: Complementant amb el menjador 3 trobem 34, 35 amb les variants P,M i F, 36 també amb les variants P,M,F.

Menjador 5: La numeració de les taules va de la 101 a la 106.

En època d’estiu s’afegeix un nou menjador a prop del menjador 1 que correspondria a una terrassa formada per 9 taules anomenades de la 01 a la 09.

Cal destacar que la numeració depèn de la disposició de les taules al restaurant i si un dia cal ajuntar dos taules consecutives, la numeració de la nova taula anirà a criteri del cambrer però mantenint una lògica amb la exposada anteriorment.

![Croquis de la distribució](image)

Veient el croquis es pot veure que amb la complexa distribució de menjadors, quan el cambrer es troba en la bodega o ajudant a un company a realitzar alguna tasca és difícil mantenir contacte visual amb les taules i saber el progrés d’aquestes.

### 3.2.3 Mode operacional del cambrer

Quan comensals entren al restaurant, aquests són atesos pel responsable el qual els identifica amb el nom de la reserva si és necessari, els acompanya a la taula corresponent i els entrega les cartes. Un cop entregades les cartes, aquests passen a mans del cambrer corresponent:

1. S’entreguen les cartes.
2. El cambrer s’atansa a la taula i realitza la comanda.
3. Aquest retorna a la cuina per entregar la comanda.
4. Es van a buscar les begudes i el pica pica de la taula.
5. Es porten els entrants.
6. Es reclamen els segons a la cuina i es retiren els plats.
7. Es porten els segons plats.
8. Es retiren els plats i s’entrega la carta de postres.
9. Es pren comanda de postres i s’entrega aquesta a la cuina.
10. Es porten els postres a la taula.
11. Es passa la nota a ordinador i es revisen els ítems.
12. Es passa la nota a la taula i es cobra.

D’aquest llistat d’accions que ha de dur a terme el cambrer, les més costoses pel que fa al temps que requereixen són: la presa Cal tenir en compte que mentre es realitzen aquest seguit d’operacions també s’estan atenent a altres taules simultàniament i no és estrany l’oblit d’alguna taula o d’anotar algun ítem en la factura d’aquesta.

3.2.4 Estructura d’una comanda

La presa de comanda en un restaurant, tot i no ser el pilar d’aquest si no es realitza correctament o cada cambrer utilitza el seu propi format, es poden desencadenar confusions i malentèsos que interrompén el bon funcionament del restaurant. Aquest fet repercuteix al client directament ja sigui augmentant els temps d’espera per tal de ser servits o bé no rebé els plats tal i com els havien demanat. Cada restaurant adopta un format diferent de comanda, fet que pot provocar disputes sobre quins són els elements que ha d’incorporar aquesta i com ha d’estar indicat però totes les solucions són vàlides si el restaurant treballa amb un criteri comú. Tal i com s’ha indicat en l’inici d’aquest projecte, es pren com a referència el restaurant Les Espèlmes.

L’estructura bàsica d’una comanda:
Els fulls utilitzats per a realitzar la comanda incorporen un sistema de còpia i tenen impresos alguns camps per complir com són:

- Número de taula
- Número de comensals
- Cambrer
- Hora en que s’ha realitzat la comanda.
- Habitació (en el cas que s’utilitzin en un hotel)

Malgrat tenir totes les opcions, per tal d’optimitzar el temps només s’introduix l’informació essencial per al procés que és el de la taula.

En el cos de la comanda es diferencia clarament dos apartats separats per una ratlla horitzontal, aquesta separa els entrants i els segons plats. Aquesta separació és necessària per tal de permetre a cuina diferenciar diferents temps de comanda i no passar tots els plats a la vegada. Al costat dels plats que ho requereixen es realitza una anotació per tal d’indicar el punt de coció, guarnicions o modificacions específiques pel client així com: sense sal, salsa a part, no incorporar un ingredient...
Normalment en la banda dreta de la comanda, a l’alçada de la ratlla separadora s’indica la quantitat de pa que vol la taula i si el volen tootrat o sense tootrar, però aquesta ubicació pot canviar segons el cambrer que realitza la comanda.
Finalment en la part inferior de la comanda s’indica el beure, en el cas que no hi hagi espai suficient
s’anota pels laterals d’aquesta.

La comanda de postres es pren en un full senzill sense còpia, fet que obliga al cambrer en realitzar una còpia manualment d’aquests juntament amb ítems extra que hagi demanat la taula i calgui incorporar en la factura. En la figura 3.2 es poden veure dos exemples de comandes preses al restaurant al llarg d’un servei de cap de setmana. Tal i com es pot apreciar, degut a la gran quantitat de feina i la pressió per estar atent a més taules fan que la lletra no sigui molt clara i sovint es troben tatrcons i correccions sobre aquestes.

![Figura 3.2: Exemples de comandes](image)

**Taula: 35-P**

- **Primers**
  1 Saltejat de bolets

- **Segons**
  1 Milfulls amb espàrrecs
  1 Suquet de rap

- **Pa:** 2 Torrades

- **Begudes**
  1 Aigua 1L
  2 Copa de vi

**Taula: 104**

- **Primers**
  1 Amanida gratinada
  1 Espàrrecs amb pasta de full
  1 Carpaccio de tomàquet

- **Segons**
  1 Magret d’ànec (amb patates fregides)
  1 Graellada de carns
  1 Graellada vegetal

- **Pa:** 4 Torrades

- **Begudes**
  1 El perro verde
  1 Aigua 1L
3.3 Android

Android és un sistema operatiu inicialment pensat per a dispositius mòbils llançat per Android Inc. amb la col·laboració econòmica de Google, que posteriorment acabaria comprant la companyia l’any 2005. Aquest sistema operatiu es caracteritza per estar basat en un *kernel* Linux sota una llicència de codi obert.

3.3.1 Versions del sistema operatiu

Des del seu llançament al mercat l’any 2008 incorporat en el dispositiu HTC Dream, el sistema no ha parat d’evolucionsar. Si es compara la primera versió d’Android amb la actual, es pot veure com hi ha hagut una gran evolució del sistema operatiu principalment degut al seu èxit en el mercat.

Les diferents versions d’Android, a part de rebre un número superior a la versió anterior, reben el nom de postre, ordenat alfabèticament.

<table>
<thead>
<tr>
<th>Nom</th>
<th>Número</th>
<th>Nivell d’API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apple Pie</td>
<td>1.0</td>
<td>1</td>
</tr>
<tr>
<td>Banana Bread</td>
<td>1.1</td>
<td>2</td>
</tr>
<tr>
<td>Cupcake</td>
<td>1.5</td>
<td>3</td>
</tr>
<tr>
<td>Donut</td>
<td>1.6</td>
<td>4</td>
</tr>
<tr>
<td>Eclair</td>
<td>2.0 - 2.1</td>
<td>5 - 7</td>
</tr>
<tr>
<td>FroYo</td>
<td>2.2 - 2.2.3</td>
<td>8</td>
</tr>
<tr>
<td>Gingerbread</td>
<td>2.3 - 2.3.7</td>
<td>9 - 10</td>
</tr>
<tr>
<td>Honeycomb</td>
<td>3.0 - 3.2.6</td>
<td>11 - 13</td>
</tr>
<tr>
<td>Ice Cream Sandwich</td>
<td>4.0 - 4.0.4</td>
<td>14 - 15</td>
</tr>
<tr>
<td>Jelly Bean</td>
<td>4.1 - 4.3.1</td>
<td>16 - 18</td>
</tr>
<tr>
<td>KitKat</td>
<td>4.4 - 4.4.4, 4.4W - 4.4W.2</td>
<td>19 - 20</td>
</tr>
<tr>
<td>Lollipop</td>
<td>5.0 - 5.1.1</td>
<td>21 - 22</td>
</tr>
<tr>
<td>Marshmallow</td>
<td>6.0 - 6.0.1</td>
<td>23</td>
</tr>
<tr>
<td>Nougat</td>
<td>7.0 - 7.1.2</td>
<td>24 - 25</td>
</tr>
</tbody>
</table>

Taula 3.1: Llistat de versions d’Android

Figura 3.3: Partició d’Android
Malgrat la ràpida actualització del software, molts dispositius mòbils llençats amb una versió no s’han actualitzat a versions més recents. Això provoca que en l’actualitat conviuguin diferents versions. Aquest fet és molt important a l’hora de dissenyar una aplicació ja que aquesta estarà dissenyada per a una versió del sistema operatiu, podent fer servir les funcions d’aquella versió, però les versions amb una API més baixa que la versió escollida no podràn utilitzar l’aplicació.

### 3.3.2 Arquitectura d’Android

![Diagrama de la pila de software d’Android](image)

En la figura 3.4 es mostra un esquema de l’arquitectura de Android. Tal i com es pot veure, aquesta està dividida en forma de capes.

Si analitzem aquestes de forma ascendent, la capa més baixa és el *kernel* de Linux. La funció d’aquesta és connectar i transmetre informació entre els diferents components de hardware així com la càmera, el panell tàctil o el xip de wi.

Una capa superior al *kernel* hi ha el *Hardware Abstraction Layer (HAL)* o Capa d’abstracció de hardware. Aquesta capa funciona com a interfàcie entre el software i el hardware del dispositiu. Així doncs si una aplicació vol accedir a un element del hardware del dispositiu no ho fa directament, sinó que ho fa mitjançant el mòdul del HAL corresponent al component amb el qual es vol interactuar.

Més amunt trobem l’*Android Runtime (ART)* o temps d’execució d’Android i les Llibreries natives C/ C++. El temps d’execució d’Android és una màquina virtual la funció de la qual és transformar
les comandes rebudes per l’aplicació en instruccions de llenguatge màquina que són executades per el
processador del dispositiu. Abans de la API 21 (Android Lollipop), Dalvik era el temps d’execució
utilitzat pel sistema operatiu aquesta utilitza una compilació just-in-time (JIT) el qual compila el
codi cada vegada que s’inicia una aplicació, mentre que ART utilitza una compilació ahead-of-time
(AOT), el qual crea un arxiu posterior a l’instal·lació de l’aplicació que s’utilitzarà en obrir aquesta.
Fet que disminueix l’ús del processador i augmenta la duració de la bateria. Cal notar que si una
aplicació funciona bé amb ART, aquesta també funcionarà amb Dalvik, però no a l’inversa.
En el mateix nivell que l’ART es troben les **llibraries natives** C/C++ compilades en el codi natiu
del processador. Algunes d’aquestes són:

- SLG: motor de gràfics 2D
- SQLite: motor de base de dades relacionals
- SSL: proporciona serveis d’encryptació *Secure Socket Layer*
- Media Framework: llibreria que suporta codec’s de reproducció i gravació en multitud de formats
d’àudio, vídeo i imatges (MPEG4,MP3,JPG, PNG...)

El **Java API Framework** són el conjunt de funcions disponibles per als programadors mitjançant
una API i escrites en Java que permeten crear una aplicació de forma ràpida i fàcil. Algunes de les
funcions més importants són:

- Activity Manager
- Location Manager
- Notification Manager
- Content Provider

Finalment en la capa superior es troben les **aplicacions** ja siguin les pre-instal·lades del sistema com
les instal·lades per l’usuari posteriorment.

### 3.3.3 Llenguatges de programació

Per tal de programar aplicacions en Android existeixen diferents llenguatges de programació que
segons el coneixement del programador i els requeriments de l’aplicació és més adient una o altre.

**Android Studio**

La plataforma de programació oficial de Google per a programar en Android, el llenguatge de programa-
ció que aquest utilitza és Java conjuntament amb XML, el primer és un llenguatge de programació
orientat a objectes l’objectiu del qual és que els programadors només hagin d’escriure el programa
una sola vegada i aquest es pugui executar en qualsevol dispositiu. Aquest té suport per XML, un
llenguatge basat en etiquetes que permet crear i organitzar la posició dels diferents elements que mos-
strarà la pantalla. És el llenguatge més utilitzat per al desenvolupament d’aplicacions Android, i en ser
e el llenguatge oficial permet trobar informació d’aquest amb molta facilitat. Per exemple, en la pàgina
oficial d’Android es troben tutorials, forums i vídeos sobre cada API d’Android i aquesta informació
està pensada per a ser implementada en Android Studio.
**Basic4Android**

Plataforma de programació d’aplicacions Android en la qual el llenguatge de programació és Visual Basic. Aquest és un llenguatge de programació molt més gràfic que Java, fet que li proporciona a aquesta plataforma la corba d’aprenentatge més elevada en aquest camp. Com a inconveniènt, per tal d’utilitzar aquest software s’ha d’adquirir una llicència amb un cost de 59€.

**App inventor**

Plataforma primerament impulsada per Google i posteriorment mantinguda pel MIT. Aquesta està basada en un llenguatge de desenvolupament gràfic, la mecànica d’ús consisteix en arrastrar i deixar anar blocs que realitzen una acció particular, per lo que no cal escriure codi per a realitzar una aplicació. L’interfície gràfica d’aquesta permet crear una aplicació senzilla en una hora, fet que permet introduir gent sense base de programació en el món de la creació d’aplicacions Android.

Finalment s’ha optat per utilitzar Android Studio per aquest projecte. És a dir, el llenguatge de programació utilitzat serà Java conjuntament amb XML. Com que el programador en el moment de realitzar el treball no tenia coneixements de cap dels llenguatges exposats, els punts que han decantat la decisió cap aquesta plataforma són: la gratuïtat d’aquesta i el fet de ser l’eina de desenvolupament oficial d’Android.
3.4 Android Studio

3.4.1 Estructura

- **AndroidManifest.xml**: Proporciona informació sobre l’aplicació al sistema Android que ha de tenir en compte a l’hora d’executar el codi. Entre altres items trobem:
  - Declara els permisos que ha de tenir l’aplicació.
  - Descriu els components de l’aplicació així com activitats, serveis, receptors de missatges..
  - Declara el nivell mínim d’API que requereix l’aplicació per a funcionar.
  - Enumera les biblioteques a les que està vinculada l’aplicació.

- **Carpeta Java**: S’hi localitzen els fitxers Java creados en el desenvolupament de l’aplicació. Dins d’aquesta carpeta s’han diferenciat mitjançant subcarpetes els fitxers de les activitats de l’aplicació i els fragments, classes (Families), i adaptadors (Adapters).

- **Carpeta res**: Directori principal de recursos els quals són fàcilment accessibles mitjançant la classe R. Els diferents recursos es distribueixen de la següent forma:
  - drawable: S’hi emmagatzemen els icones i imatges utilitzats en l’aplicació.
- layout: Conté els fitxers XML que defineixen l’interfície gráfica de les diferents activitats.
- menu: Conté els fitxers XML relacionats amb la barra superior de cada activitat (ToolBar).
- mipmap: Carpeta molt similar a drawable, l’única diferència és que els gràficst inclosos en aquesta carpeta són re-escalats segons la densitat gràfica del dispositiu. També és on es troba l’icona principal de l’aplicació. Segons les densitats de pantalla es diferencien:
  * Medium dots per Inch (mdpi): Densitats de 160 dpi.
  * High dots per Inch (hdpi): Densitats de 240 dpi.
  * Extra high dots per inch (xhdpi): Densitats de 340 dpi.
  * Extra extra high dots per inch (xxhdpi): Densitats superiors a 480 dpi.
- values: Valors utilitzats en l’aplicació:
  * colors.xml: Colors definits mitjançant notació hexadecimal.
  * dimens.xml: Dimensions.
  * strings.xml: Textos.
  * styles.xml: Estils.
- Gradle Scripts: Carpeta on es troben els fitxers Gradle que permeten compilar i construir l’aplicació. El fitxer més important és el build.gradle (Module: app) que és on es configuren les opcions de compilació del mòdul.

### 3.4.2 Arxius Java

**Activitats**

Una activitat en Android és lo que col·loquialment anomenem una pantalla de l’aplicació. Una aplicació està formada per diferents activitats entre les quals es mou l’usuari. En la part inferior del dispositiu es troben tres botons ja siguin físics o en pantalla que permeten anar a l’activitat anterior, anar a la pantalla principal del dispositiu i navegar entre les últimes aplicacions utilitzades.

![Figura 3.6: D’esquerra a dreta: Navega endarrere, retorn a la pantalla principal, aplicacions utilitzades recentment](image)

Per tal de crear una activitat, es crea una classe que hereti de la classe AppCompatActivity tal i com es mostra en la figura[3.1] Cal sobreescriure el mètode onCreate en el qual mitjançant setContentView relaciona l’activitat amb el fitxer XML corresponent al layout d’aquesta.
Capítol 3. Anàlisis previ

Listing 3.1: Creació d’una nova activitat

```java
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //codi
    }
}
```

Fragments

Es pot pensar el concepte de fragment com una secció modular d’una activitat, és una sub-activitat que permet ser reutilitzada en diferents activitats. Aquest fet de re-aprofitament és el concepte més important dels fragments, un cop s’ha realitzat el primer, aquest es pot utilitzar en diferents activitats o realitzar petits canvis a ells per crear-ne de nous. Per tal de crear un fragment cal crear una subclass que hereti de la classe Fragment tal com es mostra en el codi3.2. Cal sobrescriure el mètode onCreateView en el qual es fa referència al fixer XML on es troba l’aspecte visual del fragment, en aquest cas anomenat fragment1.

Listing 3.2: Creació d’un nou fragment

```java
public class Fragment1 extends Fragment {
    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
    @Nullable Bundle savedInstanceState) {
        View view;
        view = inflater.inflate(R.layout.fragment1,container);
        //Codi
        return view;
    }
}
```

Adaptadors

Un adaptador és un objecte que actua com un pont entre l’AdapterView i les dades a mostrar. Aquest agafa les dades proporcionades i a cada element li assigna el layout corresponent. Aquest es complementa amb un ListView o GridView que mostraran els layouts creats per l’adaptador. Android proporciona un seguit d’adaptadors senzills que permeten estendre les seves funcionalitats per crear adaptadors que s’ajustin a les necessitats. Els més utilitzats són:

- ArrayAdapter: Subministra les dades a un control de selecció a partir d’un array.
- SimpleAdapter: Permet tractar amb dades estàtiques i els contenidors de cada ítem definits en un fixer XML. Cada element de l’ArrayList correspon a una fila de la llista.
- SimpleCursorAdapter: S’utilitza per introduir columnes d’un cursor obert d’una base de dades a la llista.
3.4.3 Arxius XML

Layout

Els arxius que es troben en la carpeta layout defineixen l’estil de cada activitat. Per a cada activitat li correspon un fitxer XML situat dins aquest directori. Per a cada arxiu cal definir un layout, és un contenidor que defineix l’orde amb el que s’organitzen els ítems d’aquest. Hi han diferents tipus de layout, així com:

- **RelativeLayout**: Cada element dins d’aquest layout es col·loca de forma relativa. La posició pot estar referenciada al layout o a altres elements dins d’aquest. Algunes de les comandes més utilitzades són:
  - `android:layout_alignParentTop`: El valor és un booleà, "true"o "false". Si el valor del atribut és true, aquest fa coincidir el marge superior del ítem amb el marge superior del layout.
  - `android:layout_alignParentRight`: El valor és un booleà, "true"o "false". Si el valor del atribut és true, aquest fa coincidir el marge dret del ítem amb el marge dret del layout.

- **LinearLayout**: Introdueix els ítems un darrera l’altre en format horitzontal o en format vertical, segons s’hagi definit amb la propietat android:orientation.

- **GridLayout**: Permet definir una taula amb files i columnes, i els ítems s’ubiquen en les cel·les referenciant el número de fila i columna corresponent.

- **FrameLayout**: És el més simple de tots, alinea els ítems al extrem superior esquerra de la pantalla. Aquest està pensat per contenir un sol ítem, ja que si s’introdueixen més d’un aquests es superposaran. Normalment s’utilitza per a incloure fragments.

Les vistes que conformen el layout poden ser de diferents tipus, les més utilitzades són:

- **TextView**: Mostra text al usuari.

- **ImageView**: Mostra una imatge al usuari.

- **EditText**: Una subclasse de TextView que permet al usuari editar el text d’aquest ítem.

- **Button**: Representa un botó que el pot pulsar.

- **ImageButton**: Similar al Button però mostra una imatge en el botó en comptes de text.

- **CheckBox**: Un tipus de botó que té dos estats, marcat i sense marcar.

Menu

Aquesta carpeta conté el layout del menú superior del dispositiu. En aquest s’identifica un element principal <menu> el qual conté elements <item>. Aquests elements <item> corresponen a les diferents opcions del menú i es pot modificar la seva aparença i posició dins del menú.
Capítol 3. Anàlisi previ

Values

La carpeta values conté els arxius colors.xml, strings.xml, styles.xml. Aquests arxius contenen informació sobre colors, strings, estils entre d’altres. L’avantatge d’aquests és que definint les propietats una sola vegada, permeten utilitzar-les en qualsevol arxiu XML de l’aplicació.

3.5 Python

Creat per Guido van Rossum a finals de la dècada dels 80, Python és un llenguatge de programació fàcil d’aprendre i proporciona una lectura còmode i comprehensible. Entre les diferents característiques d’aquest es destaquen:

- És un llenguatge dinàmic, és a dir no és necessari declarar quin tipus de dada contindrà una variable.
- És de codi obert i multi plataforma disponible en Linux, Windows, Macintosh, Solaris, OS/2, etc.
- És un llenguatge orientat a objectes
- És un llenguatge d’alt nivell, no cal preocupar-se pels detalls de baix nivell així com la gestió de la memòria utilitzada pel programa.
- És poden crear múltiples programes. No és un llenguatge específic per a treballar en un sol camp.
- Té una sintaxi molt visual i clara degut a l’indentació de caràcter obligatori.

En conjunt Python és un llenguatge en ple desenvolupament, i una opció molt interessant per a realitzar programes que s’executin en qualsevol màquina. Algunes de les empreses que utilitzen Python són Yahoo, Google, NASA, Red Hat, etc.
Capítol 4

Aplicació

4.1 Eines de desenvolupament

Per tal de poder realitzar l’aplicació i consegüentment poder completar el projecte satisfactòriament és necessari la utilització d’una sèrie d’eines tant a nivell de software com a nivell de hardware. Les seleccionades per realitzar el projecte són:

- Android Studio
- Dropbox
- Python
- Editor de text Emacs
- Mòbil Android

![Logotips de les eines utilitzades](image)

4.1.1 Android Studio

La versió d’Android Studio utilitzada per al desenvolupament d’aquest treball és la 2.3 llençada el 24 de febrer del 2017.

4.1.2 Dropbox

Per tal de sincronitzar les dades dels dispositius mòbils amb l’ordinador, s’ha utilitzat l’API de Dropbox disponible en la referència [3]. Aquesta a més a més de donar suport a dispositius Android també disposa d’una llibreria per a Python, fet que permetrà integrar aquest servei en els dos dispositius...
amb facilitat. Sumat a aquest fet, l’estesa implementació de Dropbox en el mercat i el seu servei gratuït ( fins a un límit d’emmagatzematge) fa que sigui la plataforma de sincronització preferida en front d’altres alternatives més complexes d’utilitzar així com phpMyAdmin.

4.1.3 Python i Emacs

Finalment, per tal de programar el programa encarregat de rebre l’informació i analitzar-la s’ha optat per utilitzar Python juntament amb l’editor de text Emacs. Aquesta elecció és deguda al suport dels serveis de Dropbox per aquest llenguatge i la familiarietà de l’autor amb aquest degut a la formació rebuda en l’ETSEIB. La versió de Python utilitzada és la 2.7.6. Malgrat l’existència de Python 3, s’ha optat per utilitzar Python 2 ja que l’API de Dropbox no és suportada en la versió 3 del llenguatge.

4.2 Requisits per executar l’aplicació

Com que la plataforma escollida per a desenvolupar l’aplicació és Android, cal disposar d’un smartphone corrent el sistema operatiu Android amb una versió igual o superior a Android 4.2 Jelly Bean (API 17 o superior). Una connexió sense fils d’internet o bé disposar de connexió a internet mitjançant dades móbils als dispositius de sala. També caldrà disposar d’un compte de Dropbox per tal de realitzar la sincronització i un ordinador amb Python 2 instal·lat per a gestionar les comandes realitzades amb els dispositius móbils. En l’ordinador també cal tenir instal·lades les llibreries matplotlib i Dropbox. Per a saber com fer-ho, veure el capítol 5.1.5

4.3 Requisits funcionals dels usuaris

A continuació s’exposen els requisits funcionals de cada tipus d’usuari per tal de facilitar el màxim possible el desenvolupament del seu rol en el restaurant.

4.3.1 Cambrer

- Crear una nova comanda.
- Eliminar una comanda.
- Visualitzar les comandes realitzades anteriorment.
- Afegir i eliminar plats en la comanda.
- Poder realitzar comentaris en els plats.
- Consultar la carta de plats i begudes.
- Poder reclamar els segons plats d’una taula a cuina.
- Marcar la taula com a cobrada.
4.3.2 Propietari

- Tenir una visió global de totes les comandes realitzades en un dia.
- Veure el total facturat en un servei i la quantitat de plats realitzats.
- Poder editar les comandes prèviament a l’impressió del compte per al client.
- Disposar d’una base de dades de negocis amb les dades corresponents per tal de poder realitzar una factura si és demanat.
- Disposar d’un historial de factures.

4.3.3 Cuiners

- Poder veure de forma clara i concisa la comanda impresa.

4.4 Casos d’ús

Mitjançant el diagrama de casos d’ús es representa de forma gràfica com el cambrer i el propietari interactuen amb el sistema.

![Diagramas de casos d’ús](image)

Figura 4.2: Diagrames de casos d’ús

Per tal de comprendre millor el diagrama a continuació es detallen els diferents casos d’ús.
### 4.4.1 Cambrar

<table>
<thead>
<tr>
<th>Nom</th>
<th>Crear comanda</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descripció</td>
<td>Permet crear una nova taula.</td>
</tr>
<tr>
<td>Pre-condició</td>
<td>No aplicable.</td>
</tr>
<tr>
<td>Post-condició</td>
<td>L'usuari ha creat una nova taula.</td>
</tr>
</tbody>
</table>

**Flux principal**

1. L'usuari sol·licita afegir una taula
2. El sistema mostra la pantalla amb uns paràmetres d’introducció
3. L’usuari introduceix els paràmetres sol·licitats
4. El sistema valida els paràmetres introduïts
5. El sistema mostra l’element creat
6. Finalitza el cas d’ús

**Flux alternatiu**

El sistema mostra un missatge d’error si els camps introduïts són incorrectes.

<table>
<thead>
<tr>
<th>Nom</th>
<th>Eliminar comanda</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descripció</td>
<td>Permet eliminar una taula.</td>
</tr>
<tr>
<td>Pre-condició</td>
<td>Ha creat una taula prèviament.</td>
</tr>
<tr>
<td>Post-condició</td>
<td>L’usuari ha eliminat una taula.</td>
</tr>
</tbody>
</table>

**Flux principal**

1. L’usuari sol·licita eliminar una taula
2. El sistema elimina la taula i totes les dades associades a ella.
3. El sistema notifica al usuari que el procés ha finalitzat satisfactoriament.
4. Finalitza el cas d’ús

**Flux alternatiu**

L’usuari cancel·la l’eliminació de la taula.

<table>
<thead>
<tr>
<th>Nom</th>
<th>Modificar comanda</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descripció</td>
<td>Permet modificar els elements d’una taula.</td>
</tr>
<tr>
<td>Pre-condició</td>
<td>Ha creat una taula prèviament.</td>
</tr>
<tr>
<td>Post-condició</td>
<td>L’usuari ha modificat els ítems de la taula.</td>
</tr>
</tbody>
</table>

**Flux principal**

1. L’usuari selecciona la taula que vol modificar.
2. L’usuari selecciona l’element a afegir.
3. El sistema afegixe l’element seleccionat a la comanda.
4. El sistema mostra en pantalla el nom del ítem seleccionat.
5. Finalitza el cas d’ús.

**Flux alternatiu**

- L’usuari selecciona la vista en forma de llista.
- El sistema mostra els ítems corresponents a la taula en forma de llista.
- L’usuari clica sobre l’ítem a modificar.
- El sistema mostra en pantalla una finestra amb les diferents modificacions que es poden realitzar.
- L’usuari realitza la modificació.
- El sistema notifica la modificació realitzada.
### Nom
Demanar segons plats

### Descripció
Permet enviar a cuina un missatge reclamant els segons plats d’una taula.

### Pre-condició
Haver enviat la comanda prèviament a cuina.

### Post-condició
L’usuari ha sol·licitat els segons plats de la taula.

### Flux principal
1. L’usuari selecciona la taula desitjada.
2. L’usuari selecciona l’opció de sol·licitar segons plats.
3. El sistema mostra en pantalla que el procés s’ha realitzat correctament.
4. Finalitza el cas d’ús.

### Flux alternatiu
- L’usuari cancel·la l’operació.

---

### 4.4.2 Propietari

### Nom
Modifier comanda

### Descripció
Permet modificar una comanda creada des de qualsevol dispositiu.

### Pre-condició
Haver creat la comanda prèviament.

### Post-condició
La comanda ha estat modificada.

### Flux principal
1. L’administrador selecciona la taula que vol editar.
2. El sistema mostra totes les dades relacionades amb la comanda.
3. L’administrador realitza els canvis necessaris.
4. El sistema notifica que l’acció ha estat completada.
5. Finalitza el cas d’ús.

### Flux alternatiu
- L’administrador cancel·la l’operació.

---

### Nom
Realitzar factura

### Descripció
Permet relacionar una factura amb un negoci.

### Pre-condició
Haver creat la comanda prèviament.

### Post-condició
S’ha relacionat un número de factura amb un negoci.

### Flux principal
1. L’administrador selecciona la taula.
2. El sistema mostra totes les dades relacionades amb la comanda.
3. L’administrador selecciona l’opció de realitzar factura.
4. El sistema mostra tots els negocis dels que es disposen les dades.
5. L’administrador selecciona el negoci desitjat.
6. El sistema relaciona les dades del negoci amb el número de factura.

### Flux alternatiu
- El negoci no es troba en el sistema.
- L’administrador afegeix les dades del negoci.
- El sistema guarda les dades en el sistema.
Capítol 4. Aplicació

<table>
<thead>
<tr>
<th>Nom</th>
<th>Consulta estadístiques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descripció</td>
<td>Permet consultar les estadístiques de les ventes diàries.</td>
</tr>
<tr>
<td>Pre-condició</td>
<td>Ha realitzat alguna venda.</td>
</tr>
<tr>
<td>Post-condició</td>
<td>Es mostren en pantalla diferents indicadors estadístics.</td>
</tr>
<tr>
<td>Flux principal</td>
<td>1. L'administrador selecciona l’opció estadística desitjada.</td>
</tr>
<tr>
<td></td>
<td>2. El sistema calcula aquesta i la mostra en pantalla.</td>
</tr>
<tr>
<td>Flux alternatiu</td>
<td>• L’administrador cancel·la l’operació.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Nom</th>
<th>Consulta històrial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Descripció</td>
<td>Permet consultar dades de diferents dies ja tancats.</td>
</tr>
<tr>
<td>Pre-condició</td>
<td>Ha facturat en diferents dies prèviament.</td>
</tr>
<tr>
<td>Post-condició</td>
<td>Es mostren en pantalla dades generals sobre el dia desitjat.</td>
</tr>
<tr>
<td>Flux principal</td>
<td>1. L’administrador selecciona l’opció de veure l’històrial.</td>
</tr>
<tr>
<td></td>
<td>2. El sistema mostra els dies dels que es disposa informació.</td>
</tr>
<tr>
<td></td>
<td>3. L’administrador selecciona el dia desitjat.</td>
</tr>
<tr>
<td></td>
<td>4. Es mostra l’informació en pantalla.</td>
</tr>
<tr>
<td>Flux alternatiu</td>
<td>• L’administrador cancel·la l’operació.</td>
</tr>
</tbody>
</table>

### 4.5 Interfície gràfica

En el transcurs del desenvolupament de l’interfície gràfica s’han tingut en compte les diferents funcionalitats a implementar i en tenir una estructura el més amigable possible per al usuari. També com a afegit, s’ha intentat que l’aparença global d’aquesta anés en harmonia amb l’estil del restaurant. En el següent apartat es mostra el disseny de les diferents pantalles que conformen l’aplicació i una breu explicació d’aquestes.
4.5.1 Pantalla principal

![Image](image.png)

Figura 4.3: Pantalla principal de l’aplicació

Compuesta per 2 elements: Una barra superior i un llistat de taules en forma de graella.
En la barra superior apareixen dos opcions:

1. Correspon a l’acció d’afegir una taula i redirigeix a la vista corresponent la qual es troba en el l’apartat 4.5.2.

2. Correspon a l’acció de sincronitzar les dades de l’aplicació.

En seleccionar un ítem del llistat de taules, el sistema redirigeix l’usuari a la pantalla corresponent per tal d’editar la taula seleccionada, la vista es troba en l’apartat 4.5.3.

La barra superior és una toolbar, la qual trobarem en totes les activitats però els ítems que apareixen en ella canvien segons l’activitat.
4.5.2 Introducció d’una nova taula

![Diagrama de pantalla mostrant la introducció d’una nova taula](image)

(a) Vista del usuari  (b) Anàlisi de la vista

Figura 4.4: Introducció de les dades d’una nova taula

En aquesta pantalla es mostren els camps necessaris per a l’introducció d’una nova taula al sistema. Una cop el clica el botó guardar, es retorna a la pantalla principal on es mostra la taula creada.
4.5.3 Edició de la comanda

(a) Categories

(b) Plats de la categoria Entrants

(c) Anàlisis de la vista

Figura 4.5: Creació d’una comanda
Capítol 4. Aplicació

Aquestes pantalles tenen una estructura molt semblant. En la barra superior s’observa una seva poques diferències entre aquestes dos activitats, en l’activitat on es troben els plats hi apareix una lupa, la qual permet realitzar cerques d’ítems que es troben dins la categoria seleccionada prèviament. En aquesta barra també es troba un menú desplegable on es localitzen altres funcions no tan utilitzades com són:

- Eliminar la taula
- Demanar segons

A sota de la barra hi ha una zona on es mostra informació rellevant de la taula així com en quin número de taula s’efectuarà la comanda, l’estat actual de la taula (E,S) segons si aquesta es troba en fase d’entrants o segons plats i quin és l’últim ítem seleccionat en aquesta. En una barra de color d’aurat s’ubiquen els diferents controls més freqüents que es poden necessitar en realitzar la comanda, d’esquerra a dreta són:

1. Eliminar l’últim plat inserit.
2. Realitzar un comentari sobre l’últim plat seleccionat.
3. Inserir un separador per diferenciar primers-segons-postres o altres categories.
4. Llistar la comanda de la taula.
5. Enviar la comanda un cop finalitzada.

Quan es demana llistar la comanda, es mostra una llista amb els plats i els respectius comentaris. En cas de voler modificar un plat es pita sobre aquest i es mostra un menú amb les diferents opcions.

![Figura 4.6: Revisió d’una comanda creada](image-url)
4.6 Estructura de les bases de dades

Taules: [Numero, Persones, Estat]

- Numero: Representa el número de la taula. La numeració de les taules no són números íntegrament, sinó que també incorporen lletres com P, M i F per diferenciar Passadís, Mig i Finestra. És un *string*.
- Persones: Número de comensals en la taula. És un enter.
- Estat: Estat en el que es troba la taula E, S corresponent a Entrants, Segons. És un *string*.

Comanda: [Taula, Persones, Plat, Observacions]

- Taula: Número de la taula. És un *string*.
- Persones: Número de persones de la taula. És un enter.
- Plat: Nom del plat. És un *string*.
- Observacions: Comentari opcional sobre el plat. És un *string*.

Plats: [Id, Plat, Preu, Família, Categoria].

- Id: Codi amb el qual s’identifica el plat. És un enter.
- Plat: Nom del plat. És un *string*.
- Preu: Preu unitari del plat. És un número decimal.
- Família: Lletra que permet ordenar els plats amb més precisió que les categories. Per exemple, E correspon a entrants, C correspon a carns, P correspon a peix... És un *string*.
- Categoria: Nom que permet realitzar una separació genèrica dels plats. Per exemple Primers, Segons, Postres. És un *string*.

4.7 Sincronització

En la part superior dreta de la pantalla principal de l’aplicació també es troba un icona de sincronitzar, si es pitja aquesta icona, s’inicia un procés de sincronització que consisteix en enviar totes les comandes creades per si hi ha hagut alguna modificació posterior i també es rep la de nou la base de plats actualitzada en cas que hi hagi hagut alguna incorporació o modificació d’aquests. La sincronització amb l’ordinador central del local no necessàriament s’ha de realitzar amb l’icona de la pantalla principal, sinó que un cop s’ha realitzada la comanda, es pot pitjar el botó enviar de la fígura aquest envia la comanda cap a l’ordinador central del local mitjançant una API de Dropbox. Per tal de poder realitzar la sincronització de les bases de dades entre els diferents dispositius s’ha seguit el següent esquema:
Figura 4.7: Esquema de la sincronització dels fitxers

En l’esquema s’observen 3 fitxers diferents amb noms diferents:

- **DBbackup\_PC\_A**: Correspon a la base de dades del dispositiu A.
- **DBbackup\_PC\_B**: Correspon a la base de dades del dispositiu B.
- **DBbackup\_PC**: Correspon a la base de dades del ordinador principal.

S’ha pres com a referencia la convivència de dos dispositius prenent nota simultàniament, però és possible afegir tants dispositius com sigui necessari. Només cal diferenciar-los amb el nom de la base de dades atorgat a ells i en el programa del ordinador.

### 4.8 Gestió de les dades en l’ordinador

Un cop s’han enviat les comandes, l’ordinador s’encarrega de recopilar les comandes provenents de diferents dispositius mòbils per la realització de factures i l’anàlisi de dades posterior. El programa utilitzat per a realitzar aquesta feina està programat en llenguatge **Python 2** i es requereix d’aquest en un entorn Linux per a fer-lo funcionar correctament. Per tal de gestionar les dades correctament, l’aplicació s’ha separat en diferents apartats:

**Llistat de taules**

En aquesta pantalla es mostren les taules preses amb els diferents dispositius mòbils. La pantalla està dividida en dos apartats, en la part superior d’aquesta hi ha un llistat de les diferents opcions disponibles així com veure les estadístiques (E) o refrescar la pàgina (R), a sota es mostren en forma de graella les taules creades fins al moment.

Figura 4.8: Captura de la vista de taules
**Vista de una comanda**

Aquesta vista permet visualitzar de forma agrupada i simple la comanda d’una taula, molt semblant a la factura que rebrà el client en finalitzar l’àpat. De cada plat s’especifica el nom, el nombre d’unitats, el preu per unitat i el total, finalment es fà la suma i s’aplica l’IVA.

![Factura de la taula](image)

**Figura 4.9: Factura de la taula**

**Estadístiques** En la vista 4.10 es mostra un llistat dels diferents resums estadístics es poden efectuar, conjuntament amb el control que cal introduir al programa i s’utilitzin llistats, i gràfics segons l’opció escollida.

![Llistat d’estadístiques d’disponibles](image)

**Figura 4.10: Llistat d’estadístiques d’disponibles**
Capítol 4. Aplicació

(a) Llistat de plats demanats

(b) Resum diari

(c) Formatge de plats

Figura 4.11: Resums estadístics d’interès
Capítol 5

Implementació

En aquest capítol s’explicaran les diferents dificultats que s’han tingut en l’hora de desenvolupar l’aplicació. Tal i com s’ha exposat anteriorment, el desenvolupament de l’aplicació es realitzarà mitjançant Android Studio, i el llenguatge utilitzat és Java conjuntament amb XML.

5.1 Funcions

Abans de començar a programar, cal definir quines funcions es volen implementar i de quina manera. Un cop aquestes estan definides, es buscarà el disseny que més s’adiguï perquè l’usuari utilitzi aquestes funcions.

5.1.1 Bases de dades

Tal i com s’ha vist en el punt 3.3.2, Android disposa d’un sistema de base de dades integrat anomenat SQLite. S’utilitzarà aquest per tal de crear les bases de dades necessàries: una base de dades de plats, una altra de taules i finalment una de comandes. Per tal de poder gestionar aquestes, s’ha creat una classe anomenada BaseHelper la qual estén les funcions de SQLiteOpenHelper. Aquesta conté funcions que utilitzen comandaments execSQL per tal d’interactuar amb les bases de dades. Les comandes execSQL utilitzades en el projecte són:

```
1  execSQL("CREATE TABLE NomDeLaTaula,(Param1,INTEGER,Param2,Text,Param3,INTEGER,Param4,Text,,
Paran5,Text")")
2  execSQL("SELECT,* FROM NomDeLaTaula")
3  execSQL("DELETE, FROM NomDeLaTaula WHERE NomParametre,=,valor_del_parametre")
4  execSQL("UPDATE, NomDeLaTaula,SET, NomParameter,=,valor_del_parametre,WHERE, NomParameterCondicio,=,valor_del_parametre_condicio")
5  execSQL("DROP, TABLE, IF, EXISTS, NomDeLaTaula")
```

La primera crea una nova taula amb el nom i els paràmetres indicats.
La segona selecciona tots els paràmetres de cada element de la taula. S’utilitza per a recórrer la taula.
La tercera elimina els elements que tenen el paràmetre indicat.
La quarta modifica un paràmetre dels elements de la taula que compleixin una condició.
La cinquena elimina la taula.
La classe BaseHelper permet crear funcions que agilitzin la modificació de les bases de dades en el transcurs del programa. En crear la classe cal sobrescriure els mètodes `onCreate` i `onUpgrade`. El primer s'executa automàticament en crear un fitxer SQLite i la seva funció és crear les taules desitjades dins d'aquest, el segon es crida quan s'ha modificat la base de dades i es vol actualitzar. Per tal de cridar aquest cal posar l'últim argument en instal·lar un element pertanyent a la classe BaseHelper superior al que hi havia anteriorment.

Algunes de les funcions més importants creades en el BaseHelper són:

**Listing 5.2: Clase BaseHelper**
```java
public void insert_taula(String taula, int persones, String estat) {
    ContentValues contentvalues = new ContentValues();
    contentvalues.put("Numero", taula);
    contentvalues.put("Persones", persones);
    contentvalues.put("Estat", estat);
    //long insert (String table_name,String nullColumnHack,ContentValues values_to_insert)
    (this.getWritableDatabase()).insert("Taules",null,contentvalues);
}
```

En cridar aquesta funció amb els arguments (Nom de la taula, número de persones, estat de la taula) s'introduceix una nova taula en la taula Taules.

**Listing 5.3: Introducción d’una nova taula a la base de dades**
```java
public void insert_comanda(String taula, int persones, String plat, String observacions) {
    ContentValues contentvalues = new ContentValues();
    contentvalues.put("Taula", taula);
    contentvalues.put("Persones", persones);
    contentvalues.put("Plat", plat);
    contentvalues.put("Observacions", observacions);
    (this.getWritableDatabase()).insert("Comanda",null,contentvalues);
}
```

En cridar aquesta funció, amb els arguments (Nom de la taula, número de persones, nom del plat, observacions del plat) permet introduir dades ràpidament en la taula Comanda.

**Listing 5.4: Eliminar l’últim ítem de la comanda**
```java
public boolean delete_ultim_Comanda(SQLiteDatabase db) {
    db.execSQL("DELETE FROM Comanda WHERE rowid = (SELECT MAX(rowid) FROM Comanda)");
    return true;
}
```

Aquesta funció permet eliminar l’últim ítem introduït en la taula Comanda.

**Listing 5.5: Comentari en l’últim ítem**
```java
public boolean updateComentariUltim(SQLiteDatabase db,String comentari){
    String sql="UPDATE Comanda SET Observacions="/"comentari="",where_roid=",(SELECT MAX(roid) FROM Comanda)"
    db.execSQL(sql);
    return true;
}
```

Aquesta funció permet introduir un comentari en l’últim plat introduït en la base de dades ed manera molt cómode en el desenvolupament del codi.

A continuació es detalla el procediment utilitzat per introduir una nova taula en la base de dades de taules utilitzant la classe BaseHelper. Aquest procediment és molt semblant per a realitzar altres operacions com modificar o eliminar elements de la base de dades.
Primerament es crea un element pertanyent a la classe BaseHelper anomenat baseHelper. Un cop s’ha creat l’element BaseHelper, s’utilitza la funció definida en la classe per a introduir una nova taula i finalment s’envia un missatge al usuari en forma de Toast indicant que la taula ha estat inserida.

Cal tenir en compte que en aquest cas l’usuari no és conscient de quin tipus d’input accepta la base de dades o aquest es pot equivocar en l’introducció d’aquestes introduint un caràcter inesperat. Per aquest motiu cal vigilar que les variables introduïdes en la base de dades no provoquin un error. Aquest control es realitza mitjançant el mètode try/catch. Aquest mètode funciona de la següent manera: el codi s’executa amb normalitat dins del camp try, si al llarg de l’execució d’aquest es produeix un error, s’executa el codi situat dins de catch i es segueix amb les següents línies del codi amb normalitat. Però si en el transcurs de l’execució del codi de try no es produeix cap error, el codi situat dins del mètode catch no s’executa. Un exemple clar de l’utilitat d’aquest mètode es troba en l’introducció del número de persones en la creació d’una nova taula, si l’usuari introduïx la paraula errònia com que en la base de dades s’ha declarat el camp com a numèric no produeixia un error i es reiniciaria l’aplicació. Tal i com es veu en la figura 5.7 amb l’incorporació de la parella try/catch si s’introduïx una dada errònia es mostraria un toast al usuari informant que alguna dada ha estat mal introduïda.

5.1.2 Creació de llistes lliscant

Crear una llista lliscant (ListView) en Android és relativament fàcil, els passos a seguir són:

1. Inserir una ListView en la posició desitjada de l’activitat.
2. Crear una ArrayList amb els elements que es volen mostrar en la ListView.
3. Crear un adaptador (ListAdapter), el qual s’encarregarà d’adaptar la ArrayList a una ListView.
4. Establir l’adaptador al ListView que prèviament s’ha declarat.
5. Per últim i de manera opcional, s’afegeix un listener (onItemClickListener) a la llista per tal de realitzar alguna acció en clicar els elements de la llista.
Es pot veure un clar exemple en la pantalla que mostra el llistat de plats que conformen la comanda. En aquest cas s’utilitza un `ArrayAdapter` amb la distribució anomenada `simple_list_item_2` la qual permet la introducció d’un text principal i un de secundari en cada cel·la.

![Imatge de l’adaptador simple](image.png)

**Figura 5.1:** Vista de la `simple_list_item_2`

Però aquest fet es complica quan es vol que la llista incorpori unes cel·les personalitzades o incorporar una funció extra així com una filtració dinàmica dels elements de la llista. En aquests casos cal crear un adaptador personalitzat en comptes d’utilitzar els ja predefinits.

En la figura 5.8 es mostra el codi utilitzat per a crear l’adaptador dels plats anomenat `GridPlatsAdapter`. Aquest serà l’encarregat de mostrar els plats en forma de grella i realitzar un filtratge d’aquests. A continuació es mostra el codi utilitzat per a crear aquest adaptador amb filtratge dinàmic dels plats.
Listing 5.8: Adaptador de plats personalitzat

```java
public class GridPlatsAdapter extends RecyclerView.Adapter<GridPlatsAdapter.PlatsViewHolder>
    implements Filterable{

    ArrayList<Plats> plaAL = new ArrayList<Plats>();
    Context ctx;
    String t_id;
    int i=0;
    ArrayList<Plats> filterList;
    PlatsCustomFilter filter;
    TextView textView;

    public GridPlatsAdapter(ArrayList<Plats> pllist, Context ctx, String taula_id, TextView txt_view) {
        this.ctx = ctx;
        this.plaAL = pllist;
        this.t_id = taula_id;
        textView=txt_view;
        this.filterList=pllist;
    }

    @Override
    public PlatsViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        View view = LayoutInflater.from(parent.getContext()).inflate(R.layout.recuadre_gris, parent, false);
        PlatsViewHolder platsViewHolder = new PlatsViewHolder(view, ctx, plaAL, t_id);
        return platsViewHolder;
    }

    @Override
    public void onBindViewHolder(PlatsViewHolder holder, int position) {
        Plats PLA = plaAL.get(position);
        String nomdelplat=PLA.get_nom();
        holder.plat_nom.setText(nomdelplat);
        holder.setItemClickListenerPlats(new ItemClickListener_Plats() {
            @Override
            public void onItemClick(View v, int pos, BaseHelper baseHelper) {
                baseHelper = new BaseHelper(v.getContext(),"DB",null,1);
                baseHelper.insert_comanda(t_id,2,plaAL.get(pos).get_nom(),"");
                //S'assigna el nom de l'ultim producte clicat al textview del fragment1
                String nomplat=plaAL.get(pos).get_nom();
                textView.setText(nomplat);
            }
        });
    }

    @Override
    public int getItemCount() {return plaAL.size();}

    @Override
    public Filter getFilter() {
        if (filter==null){
            filter=new PlatsCustomFilter();
        }
        return filter;
    }

    public static class PlatsViewHolder extends RecyclerView.ViewHolder implements View.OnClickListener{
        Button plat_nom;
        ArrayList<Plats> plats = new ArrayList<Plats>();
        Context ctx;
        String t_id;
    }
```
Capítol 5. Implementació

```java
public PlatsViewHolder(View view, Context ctx, ArrayList<Plats> plats, String t_id) {
  super(view);
  this.plats = plats;
  this.ctx = ctx;
  this.t_id = t_id;
  plat_nom = (Button) view.findViewById(R.id.recuadre_gris);
  plat_nom.setOnClickListener(this);
}

@Override
public void onClick(View v) {
  this.itemClickListenerPlats.onItemClick(v,getLayoutPosition(),baseHelper);
}

public void setItemClickListenerPlats(ItemClickListener_Plats ic){
  this.itemClickListenerPlats = ic;
}

//INNER CLASS
class PlatsCustomFilter extends Filter{

  @Override
  protected FilterResults performFiltering(CharSequence constraint) {
    FilterResults results= new FilterResults();
    if (constraint != null && constraint.length()>0){
      constraint=constraint.toString().toUpperCase();
      ArrayList<Plats> filters= new ArrayList<>();
      for(int j=0;j<filterList.size();j++){
        if (filterList.get(j).get_nom().toUpperCase().contains(constraint)) {
          Plats p = new Plats(filterList.get(j).get_id(),
                                filterList.get(j).get_nom(),filterList.get(j).get_preu(),
                                filterList.get(j).get categoria());
          filters.add(p);
        } else {
          results.count=filters.size();
          results.values=filters;
        }
      }
    } else {
      results.count=filterList.size();
      results.values=filterList;
    }
    return results;
  }

  @Override
  protected void publishResults(CharSequence constraint, FilterResults results) {
    plaAL = (ArrayList<Plats>)results.values;
    notifyDataSetChanged() notifica que les dades de l'ArrayList han canviat i
    cal refreshar l'adaptador
    notifyDataSetChanged();
  }
}
```
En utilitzar aquest adaptador, pren com a arguments: una llista de plats que pertanyen a aquesta categoria "pllist", el context des del qual es crida `ctx", el número de la taula "taula_id" i el `TextView` de l’activitat contenidora del adaptador.

El mètode `onCreateViewHolder` crida al contenidor on es dipositarà el contingut dels elements de la llista.

El mètode `getItemCount` torna un número que correspon a la mida de la llista. molt important per saber el número de contenidors que caldrà crear per tal d’introduir tots els ítems.

El mètode `onBindViewHolder` assigna a cada ítem un contenidor anteriorment creat, i associa a cada contenidor un `ItemClickListener` que s’encarregarà d’escoltar si l’element ha estat clicat, i en cas afirmatiu s’executaràn les línies de codi definides dins del mètode.

### 5.1.3 Buscador de plats

El mètode que s’utilitza per a realitzar cerques en Android s’anomena `Search View`. Aquesta està integrada en l’`actionBar`, una barra en la part superior de l’aplicació. Per tal d’afegir l’`ActionBar`, cal afegir en el fitxer XML de l’activitat on es vol introduir l’`ActionBar` les següents línies de codi [5.9] i també cal crear un segon fitxer XML que incorporarà l’estructura de la `toolbar` en si. Cada ítem en aquest fitxer és un element que mostra l’`actionBar`, en l’exemple [5.10] es poden distingir dos ítems: el primer correspon a la `searchView` la qual conté l’atribut `app:showAsAction=ifRoom|collapseActionView"` el qual indica que quan es seleccioni l’icona corresponent, la opció de buscar ocuparà tota l’`actionBar`. El segon element del fitxer correspon a l’opció Elimina la taula que estarà ocultada i no es veu directament en l’`actionBar`.

![Listing 5.9: Instancia d’una Toolbar en un fitxer XML](image)

Un cop s’ha creat la `searchview`, cal instanciar-lo en el fitxer Java de l’activitat mitjançant el mètode `onCreateOptionsMenu`, en el qual cal sobre escriure els mètodes `onQueryTextSubmit` i `onQueryTextChange`. El primer es cridarà quan s’executi la cerca final, és a dir quan es polsi el botó cercar, el segon correspon a una cerca dinàmica que es produeix cada vegada que s’introduceix o s’esborra algun caràcter en la cerca.
Com que en aquest cas la base de dades és relativament petita i es desitja ràpida en el procés, el mètode que interessa més és el `onQueryTextChange`. En aquest mètode es pot veure com crida el mètode `getFilter` de l’adaptador de plats 

El funcionament de la cerca és el següent: Primerament es crea un element de la classe `PlatsCustomFilter la qual estén de la classe Filter, quan s’introdueix la primera lletra en l’espai de cerca, s’executa el mètode `performFiltering` sobreescrit de la classe `PlatsCustomFilter` el qual comprova quins són els plats que contenen els caràcters introduïts en el `SearchView` i els introduceix en l’`ArrayList "filters". Un cop s’ha acabat la cerca, es passen els elements de l’`ArrayList "filters" a un element `FilterResults` i es crida el mètode `publishResults` el qual introduirà els resultats de la cerca en la llista que agafa l’adaptador per a mostrar els plats des del principi. Un cop s’ha canviat la llista del adaptador per la nova llista filtrada, es crida el mètode `notifyDataSetChanged` per tal que es refresqui l’adaptador amb els nous values.

Listing 5.11: Tractament del text introduït en la `SearchView`

```java
@override 
public boolean onCreateOptionsMenu(Menu menu){ 
    getMenuInflater().inflate(R.menu.toolbar_plats, menu); 
    MenuItem searchItem = menu.findItem(R.id.action_search); 
    SearchView searchView = (SearchView) MenuItemCompat.getActionView(searchItem); 
    searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
        @Override 
        public boolean onQueryTextSubmit(String query) {
            //Cerca final.
            return false; 
        }
        @Override 
        public boolean onQueryTextChange(String query) {
            //El text ha canviat.
            adapter.getFilter().filter(query); 
            return false; 
        }
    });
    return true; 
}
```
5.1.4 Sincronització

Tal com s’ha establert anteriorment, la sincronització s’ha realitzat mitjançant la API de Dropbox. Per tal de poder utilitzar-la cal realitzar un seguit de passos previs:

Donar a l’aplicació els següents permisos:

Listing 5.12: Permisos requerits per l’aplicació

```xml
1 <uses-permission android:name="android.permission.INTERNET" />
2 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

El primer atorga a l’aplicació permís per a connectar-se a la xarxa.

El segon atorga el permís de saber l’estat de la xarxa. Aquest és molt important, ja que permet saber si estem connectats correctament a la xarxa o hi ha algun problema en aquesta. A continuació cal complir les següents llibreries:

Listing 5.13: Llibreries a complir

```java
1 compile files ('libs/json_simple-1.1.jar')
2 compile files ('libs/dropbox-android-sdk-1.6.3.jar')
```

Figura 5.2: Buscador de plats
Per tal d’accedir a una compte de Dropbox s’utilitza l’instància [5.14], aquesta té en compte els valors APP_KEY, APP_SECRET i ACCESSTOKEN únics per a cada compte de Dropbox per tal de verificar l’usuari i tenir permisos de lectura i escritura. En aquestes línies s’han substituït aquestes claus per cadenes de "x" per motius de privatitat.

**Listing 5.14: Instància inici sessió Dropbox**

```java
1  AndroidAuthSession session = buildSession();
2  DropboxAPI = new DropboxAPI<AndroidAuthSession>(session);
3  static DropboxAPI<AndroidAuthSession> dropboxAPI;
4  private static final String APP_KEY = "xxxxxxxxxxxxxxxxxx";
5  private static final String APP_SECRET = "xxxxxxxxxxxxxxxxxx";
6  private static final String ACCESSTOKEN = "xxxxxxxxxxxxxxxxxx";
7  private DropboxAPI.UploadRequest request;
8  private AndroidAuthSession buildSession(){
9      AppKeyPair appKeyPair = new AppKeyPair(APP_KEY, APP_SECRET);
10     AndroidAuthSession session = new AndroidAuthSession(appKeyPair);
11     session.setOAuth2AccessToken(ACCESSTOKEN);
12     return session;
13  }
```

En clicar enviar la comanda, el programa executa la pujada de la base de dades a la direcció específica dins de la compte de Dropbox. Per tal de pujar un fitxer es crida la funció UploadToDropboxFromPath(5.15) la qual pren com a arguments 2 Strings, el primer correspon a la direcció on es troba la base de dades i el segon és la carpeta de Dropbox on es guardarà el fitxer.

Cal tenir en compte que la base de dades es genera automàticament en el directori /databases dins de la carpeta de l’aplicació, per tal d’accedir cal utilitzar el codi: String datapath = Environment.getDataDirectory() + "//data//" + çom.alanvives.espelmesapp" + "/databases/" + "DB"

**Listing 5.15: Pujar un fitxer a Dropbox**

```java
1  private void UploadToDropboxFromPath(String uploadPathFrom, String uploadPathTo)
2  {
3      Toast.makeText(getActivity(), "Enviant la comanda ...", Toast.LENGTH_SHORT).show();
4      final String uploadPathF = uploadPathFrom;
5      final String uploadPathT = uploadPathTo;
6      Thread th = new Thread(new Runnable(){
7          public void run(){
8              File tmpFile = null;
9              try{
10                 tmpFile = new File(uploadPathF);
11             }catch (Exception e) {
12                e.printStackTrace();
13            }
14              FileInputStream fis = null;
15              try{
16                 fis = new FileInputStream(tmpFile);
17             }catch (FileNotFoundException e) {e.printStackTrace();}
18             try{
19                 dropboxAPI.putFileOverwrite(uploadPathT, fis, tmpFile.length(), null);
20             }catch (Exception e) {}
21             (getActivity()).runOnUiThread(new Runnable() {
22                 @Override
23                 public void run() {
24                     Toast.makeText(getActivity(), "Comanda enviada!", Toast.LENGTH_SHORT).show();
25                 }
26             });
27     });
28     th.start();
29  }
```
Per tal d’aconseguir una comunicació bidireccional entre l’ordinador i els dispositius mòbils, s’introduex la possibilitat d’actualitzar la base de dades de plats des de l’ordinador i que aquesta sigui en viu als dispositius mòbils. Així doncs cal definir una funció que permeti descarregar un fitxer localitzat dins de Dropbox. Aquesta queda definida en el codi[16]. Una problemàtica d’aquest procés és que el fitxer que es rep amb la base de dades actualitzada no es pot sobreescriví directament sobre la base de dades que està amb ús, per aquest motiu, quan s’accedeix en la base de dades de plats es fa referència al fitxer DBbackup_PC.sqlite en comptes del DBbackup_PC_A corresponent al dispositiu. Però en termes d’ús no suposa cap inconvenient més enllà que citar una base de dades o una altra.

Listing 5.16: Descarregar un fitxer de Dropbox

```java
private void DownloadFromDropboxFromPath (String downloadPathTo, String downloadPathFrom){
    DropboxDownloadPathTo = downloadPathTo;
    DropboxDownloadPathFrom = downloadPathFrom;
    (MainActivity.this).runOnUiThread(new Runnable() {
        @Override
        public void run() {
            Toast.makeText(MainActivity.this, "Syncronizing", Toast.LENGTH_SHORT).show();
            Thread th = new Thread(new Runnable() {
                public void run() {
                    File file = new File(DropboxDownloadPathTo + DropboxDownloadPathFrom.substring(DropboxDownloadPathFrom.lastIndexOf('.')));
                    if (file.exists()) file.delete();
                    try {
                        FileOutputStream outputStream = new FileOutputStream(file);
                        MainActivity.dropboxAPI.getFile(DropboxDownloadPathFrom, null, outputStream, null);
                        (MainActivity.this).runOnUiThread(new Runnable() {
                            @Override
                            public void run() {
                                Toast.makeText(MainActivity.this, "Files successfully synchronized!", Toast.LENGTH_SHORT).show();
                            }
                        });
                        } catch (Exception e) {
                            e.printStackTrace();
                        }
                    } catch (Exception e) {
                        th.start();
                    }
                }
            });
        }
    });
}
```

5.1.5 Tractament de dades amb python

Prèviament, cal instal·lar les llibreries matplotlib i Dropbox. Si fent ús del comandament pip, cal introduir els següents comandaments al terminal:

Listing 5.17: Instal·lació de la llibreria matplotlib

```bash
pip install matplotlib
```

Listing 5.18: Instal·lació de la llibreria de Dropbox

```bash
pip install dropbox
```
Si aquests comandaments no funcionen, cal introduir el següent comandament per tal de descarregar la funció pip.

Listing 5.19: Com obtenir la funció pip

```python
1  python get-pip.py
2
```

Totes les funcions necessàries per tal de realitzar el tractament estadístic de les dades es troben en el fitxer `estadístiques.py`, on s’han utilitzat les llibraries `matplotlib.pyplot`, `sqlite3` i `operator` per tal d’extreure les dades i generar les gràfiques corresponents. Per tal que l’ús del programa sigui més cómode a la vista i l’usuari no es pugui confondre amb dades d’operacions anteriors, entre comandament i comandament de l’usuari es neteja la pantalla del terminal utilitzant el comandament `os.system('clear')` havent importat prèviament `os`.
Capítol 6

Planificació

Tot projecte requereix d’una planificació prèvia sobre les diferents tasques que cal realitzar i els recursos assignats a aquestes.
És clar que en el desenvolupament d’una aplicació intervenen persones de diferents camps realitzant tasques específiques i a les quals estan especialitzats, aquest fet reduiria molt el temps de desenvolupament d’una aplicació i n’augmenta la qualitat.

6.1 Diagrama de Gantt

Per tal de planificar el temps disponible per a realitzar el projecte, aquest s’ha descompost en diferents tasques que caldrà anar completant. Les hores expressades en la taula ?? són un valor orientatiu i poden variar lleugerament respecte els valors reals.

<table>
<thead>
<tr>
<th>Tasca</th>
<th>Durada</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inici del projecte: Recerca de tema i tutor</td>
<td>5 hores</td>
</tr>
<tr>
<td>Planificació</td>
<td>5 hores</td>
</tr>
<tr>
<td>Recerca d’informació</td>
<td>30 hores</td>
</tr>
<tr>
<td>Anàlisis dels requisits de l’aplicació</td>
<td>15 hores</td>
</tr>
<tr>
<td>Estudi del llenguatge</td>
<td>60 hores</td>
</tr>
<tr>
<td>Disseny de l’aplicació</td>
<td>40 hores</td>
</tr>
<tr>
<td>Implementació</td>
<td>130 hores</td>
</tr>
<tr>
<td>Proves</td>
<td>15 hores</td>
</tr>
<tr>
<td>Memòria del projecte</td>
<td>75 hores</td>
</tr>
<tr>
<td>Defensa del projecte</td>
<td>5 hores</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>380 hores</strong></td>
</tr>
</tbody>
</table>

Taula 6.1: Durada d’hores previstes pel projecte

A continuació es mostra un diagrama de Gantt que il·lustra de forma aproximada les dates límit per a les diferents tasques en les què s’ha separat el projecte. Cal fer mención que el fet que una tasca tingui un marge de temps més elevat, no necessàriament significa que se li hagin dedicat més hores ja que les hores diàries dedicades al desenvolupament del projecte han anat variant segons la càrrega de treball d’altres assignatures que s’estan cursant i projectes personals.
Figura 6.1: Diagrama de Gantt
Capítulo 7

Costos

7.1 Recursos humanos

Com que al llarg d’aquest projecte una sola persona ha estat encarregada de realitzar totes les tasques, s’assigna un únic preu per hora. S’agafa com a referència el preu per hora mig representatiu d’un enginyer junior que és igual a 20€/hora. En la taula 7.1 es pot veure en més detall quantes hores s’han dedicat a cada tasca i el preu d’aquesta.

<table>
<thead>
<tr>
<th>Tasca</th>
<th>Durada[horas]</th>
<th>Cost[€]</th>
</tr>
</thead>
<tbody>
<tr>
<td>ESTUDI PREVI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Estudi del funcionament d’un restaurant</td>
<td>20</td>
<td>400,00</td>
</tr>
<tr>
<td>Estudi de mercat</td>
<td>10</td>
<td>200,00</td>
</tr>
<tr>
<td>Estudi de les diferents eines de desenvolupament</td>
<td>15</td>
<td>300,00</td>
</tr>
<tr>
<td>ESTUDI DELS LLENGUATGES DE PROGRAMACIÓ</td>
<td>60</td>
<td>1200,00</td>
</tr>
<tr>
<td>Java</td>
<td>40</td>
<td>800,00</td>
</tr>
<tr>
<td>Xml</td>
<td>15</td>
<td>300,00</td>
</tr>
<tr>
<td>Python</td>
<td>5</td>
<td>100,00</td>
</tr>
<tr>
<td>TASQUES DE PROGRAMACIÓ</td>
<td>190</td>
<td>3800,00</td>
</tr>
<tr>
<td>Aplicació Android</td>
<td>140</td>
<td>2800,00</td>
</tr>
<tr>
<td>Programa Python</td>
<td>50</td>
<td>1000,00</td>
</tr>
<tr>
<td>INTRODUCCIÓ DE LES DADES DEL RESTAURANT</td>
<td>5</td>
<td>100,00</td>
</tr>
<tr>
<td>MEMÒRIA</td>
<td>90</td>
<td>1800,00</td>
</tr>
<tr>
<td>TOTAL</td>
<td>390</td>
<td>7800,00</td>
</tr>
</tbody>
</table>

Taula 7.1: Desglossament dels costos en recursos humans

7.2 Material

Per a la realització d’aquest treball s’han utilitzat un seguit d’elements materials i paquets de software sense els quals no s’hagués pogut realitzar aquest. Algunes d’aquests elements s’han escollit a criteri personal com són el telèfon intel·ligent o l’ordinador, però d’altres varen imposar-se per l’estructura del treball com són Android Studio o el paquet ofimàtic de Microsoft. L’amortització de tots els materials ofimàtics s’ha considerat de 4 anys, ja que aquests poden ser utilitzats per a futurs projectes. Així doncs, si el preu inicial de l’ordinador és de 700€, aquest té una amortització de 4 anys i la durada del projecte ha estat de 5 mesos, correspon a un valor proporcional de 72.92€. El mateix càlcul s’aplica per al telèfon mòbil i el software informàtic amb uns preus respectius de 200€ i 150€. Seguidament es
Capítol 7. Costos

mostra en més detall el cost del material utilitzat.

<table>
<thead>
<tr>
<th>Recurs material</th>
<th>Cost (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amortització ordinador</td>
<td>72.92</td>
</tr>
<tr>
<td>Amortització telèfon mòbil</td>
<td>20.83</td>
</tr>
<tr>
<td>Material oficina</td>
<td>50.00</td>
</tr>
<tr>
<td>Amortització software informàtic</td>
<td>15.63</td>
</tr>
<tr>
<td>Connexió internet</td>
<td>250.00</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>359.38</strong></td>
</tr>
</tbody>
</table>

Taula 7.2: Desglossament dels costos en material.

### 7.3 Cost total del projecte

En la taula 7.3 es mostra un resum dels costos exposats anteriorment en les taul·les 7.1 i 7.2 tenint en compte un marge del 3% del cost del projecte per a imprevistos i l’IVA aplicat a aquest.

<table>
<thead>
<tr>
<th>Concepte</th>
<th>Cost (€)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recursos humans</td>
<td>7800.00</td>
</tr>
<tr>
<td>Material</td>
<td>359.38</td>
</tr>
<tr>
<td>Imprevistos (5%)</td>
<td>387.97</td>
</tr>
<tr>
<td><strong>Subtotal</strong></td>
<td><strong>8147.35</strong></td>
</tr>
<tr>
<td>IVA (21%)</td>
<td>1710.94</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>10258.29</strong></td>
</tr>
</tbody>
</table>

Taula 7.3: Cost total del projecte
Capítol 8

Impacte ambiental

En aquest apartat s’analitzarà l’impacte ambiental sota dos punts de vista diferents: l’impacte produït pel treball final de grau en sí, i l’impacte de l’aplicació del projecte en un futur.

8.1 Impacte ambiental en l’execució del projecte

El desenvolupament d’una aplicació mòbil no genera residus de manera directa, com que originalment no es disposava d’un dispositiu Android per al desenvolupament d’aquest, s’ha reutilitzat un dispositiu vell que fins llavors estava en desús. Els únics recursos que s’ha consumit de manera indirecte són l’electricitat per tal d’alimentar l’ordinador i el dispositiu mòbil, i fulls de paper per tal de realitzar esquemes i anotacions durant l’etapa de disseny.

8.2 Impacte ambiental de l’implementació

L’usuari necessita d’un smartphone o tableta per tal d’executar l’aplicació, en conjunt amb un ordinador. Aquests aparells consumixen electricitat, però cal tenir en compte que segons l’implementació actual ja hi ha un ordinador per tal de poder introduir i imprimir les comandes, i el consum dels dispositius mòbils es pot veure molt reduït si aquests es configuren adequadament.

Si bé és cert que amb aquest nou mètode de presa de comandes introduït utilitza paper per imprimir les comandes en la cuina i les factures al client, però la quantitat de paper utilitzat per a imprimir una comanda és molt més reduït que si es prenguessin a mà degut a un format més compacte i organitzat. Cal destacar que amb el sistema actual mitjançant blocs de notes amb duplicat, s’utilitza sempre la mateixa quantitat de paper indistintament del número de comensals, i freqüentment s’escriu una comanda 2 vegades degut a problemes d’espai o per tal de fer-la més entendible per a la cuina. Es podria estudiar en un futur la viabilitat ambiental d’introduir un tercer dispositiu a la cuina que mostrés les comandes i interactués amb els cuiners per tal d’estalviar nombroses impressions.
Capítol 9

Feina futura

Una aplicació pot anar incorporant millores progressivament quasi gairebé sense acabar mai. Per aquest motiu, en aquest apartat es llisten algunes de les funcions que bé o per falta de temps o per falta de recursos no s’han pogut implementar en la aplicació vigent, però es pretén introduir-les en una actualització futura d’aquesta.

- En el procés d’enviament de la comanda, incorporar una impressora com a receptora perquè aquesta imprimeixi les comandes dins de cuina.

- Incorporar la opció de demanar 2n plats d’una taula a cuina des del dispositiu, imprimint un paper a l’impressora.

- Segons en l’estat que es trobi la taula, diferenciar-la amb un color diferent tant en l’aplicació com en l’ordinador.

- En el llistat de la comanda, poder inserir un nou plat enmig de la llista.

- En el programa d’escriptori, poder editar els plats de les comandes creades.

- Mostrar suggeriments de comentaris quan es va a comentar un plat.

- Introduir la hora com un nou paràmetre de la taula per tal de realitzar un seguiment estadístic en finalitzar la jornada.

- Implementar la integració d’una base de dades de clients per si aquests volen una factura del servei.
Capítol 10

Conclusions, resultats i perspectives de futur

L’objectiu d’aquest projecte ha estat el de construir un sistema de comandes digital per a un restaurant per tal d’optimitzar el procés de presa de comandes. A efectes pràctics aquest objectiu s’ha assolit satisfactoriament, doncs és possible realitzar comandes amb el dispositiu mòbil, sincronitzar-les amb un temps raonable amb una base de dades d’un ordinador i gestionar-les des d’aquest. Pel que refereix a efectes pràctics, aquest sistema proposat presenta certes limitacions en el procés així com l’impressió de comandes i numeració de factures, el cobrament de les taules un cop finalitzat el servei o l’edició de comandes des del ordinador.

A nivell personal aquest projecte ha requerit un gran esforç per tal d’entendre el procés de desenvolupament d’aplicacions Android en llenguatge Java i la complexitat de la sincronització i transferència d’àrxius entre diferents plataformes. Tot i que la sincronització de l’aplicació s’ha realitzat amb l’API de Dropbox, s’ha vist que aquesta no és la millor opció, i l’utilització d’una eina com phpMyAdmin lauria permés una millor gestió de les bases de dades. Un dels majors problemes en el transcurs del treball ha estat en l’API de Dropbox, aquesta sobreescriu el fitxer de la base de dades cada vegada que es realitza una sincronització, eliminant la versió anterior i substituint-la per la última. Aquest sistema de treball impedeix l’edició d’aquestes des del ordinador i des del telèfon simultàniament ja que en realitzar les sincronitzacions la última en transmetre l’ordre eliminarà a la primera sense haver guardat els canvis realitzats. Amb l’implementació de phpMyAdmin es solucionsaria aquest fet ja que només existiria una sola base de dades editable en un servidor online.

Malgrat les dificultats exposades anteriorment i la falta d’algunes funcionalitats de l’aplicació, estic satisfet amb els resultats obtinguts i molt motivat per seguir amb el desenvolupament d’aquesta aplicació i en un futur realitzar més aplicacions a nivell personal.
Agraïments

Primerament m’agradaria agrair als meus pares per tal de permetre'm cursar els estudis d’enginyeria, també a la meva germana Ester qui amb la seva opinió ha ajudat en decidir sobre alguns dels aspectes visuals de l’aplicació.

També volria agrair a la meva parella Jaquelina per suport donat al llarg del desenvolupament d’aquest projecte.

Al meu tiet Joan Albert pel telèfon intel·ligent cedit que ha permès un millor desenvolupament del projecte.

Finalment, però no menys important al tutor del projecte Lluís Pérez Vidal pels consells i suport donat al llarg de la durada del treball.
Bibliografia


