Universitat Oberta de Catalunya

Enginyeria Tècnica en Informàtica de Gestió

Treball Final de Carrera

Desenvolupament d'aplicacions per a dispositius mòbils

Projecte "Les Meves Series"

-Memòria Final-

Alumne:

Marc Baldevey Quilez

Consultors:

Jordi Almirall López

Marc Domingo Prieto

Gener de 2013

Índex

1- Introducció i Context....6
1.1 - Introducció....6
1.2 - Justificació del projecte....6
1.3 - Descripció del projecte....6
1.4 - Objectius....7
1.5 - Planificació....9
1.5.1 - Calendari del projecte....9
1.5.2 - Diagrama de la planificació....11
1.6 - Eines utilitzades....12
1.7 - Llenguatges de Programació utilitzats....12
2 - Requisits de l'aplicació....13
2.1 – Usuaris....13
2.2 - Requisits funcionals....13
2.3 - Requisits d'informació....14
3 - Anàlisis del sistema....15
3.1 – Diagrames de casos d'ús....15
3.2 - Descripció textual dels casos d'ús....17
3.2.1 - CU01 - Registrar-se....17
3.2.2 - CU02 – Iniciar Sessió....18
3.2.3 - CU03 – Finalitzar Sessió....18
3.2.4 - CU04 – Explorar Series....19
3.2.5 - CU05 – Veure series preferides....19
3.2.6 - CU06 – Veure capítols pendents....20
3.2.7 - CU07 – Afegir serie....21
3.2.8 - CU08 – Veure informació de la serie....21
3.2.9 - CU09 – Cercar Serie....22
3.2.10 - CU10 – Afegir Serie preferida....22
3.2.11 - CU010.2 – Eliminar Serie preferida....23
3.2.12 - CU11– Veure Llistat de capítols....23
3.2.13 - CU12 – Marcar capítol vist....24
3.2.14 - CU12.2 – Desmarcar capítol vist....25
3.2.15 - CU13 – Afegir capítol....25
3.2.16 - CU14 – Veure informació capítol....26
3.2.17 - CU15– Veure enllaços de visualització online....26
3.2.18 - CU16 – Veure enllaços de descàrrega....27
3.2.19 - CU17 – Veure online....27
3.2.20 - CU18 – Descarregar....28
4 – Disseny....29
4.1 - Arquitectura bàsica de funcionament....29
4.2 – Diagrama de classes....31
4.2.1 – Pantalla inicial, registre i identificació....31
4.2.2 – Altres pantalles....32
4.3 - Diagrames de seqüència....33

4.3.1 - Afegir Serie a preferides....33
4.3.2 - Eliminar Serie de preferides....34
4.3.3 - Veure capítol online....35
4.3.4 - Notificar Usuaris quan hi ha nous capítols....36
4.4 - Disseny de la persistència....37
4.4.1 - Diagrama de la base de dades....37
4.4.2 - Model relacional de la base de dades....38
4.5 - Prototipatge....39
4.5.1 - Pantalla principal....40
4.5.2 - Pantalla de Les meves Series....40
4.5.3 - Pantalla de capítols pendents....40
4.5.4 - Pantalla veure totes les series....41
4.5.5 - Pantalla Serie....41
4.5.6 - Pantalla Capítol....42
4.5.7 - Pantalla Temporada....42
4.5.8 - Pantalla Enllaços de descàrrega....43
4.5.9 - Pantalla Enllaços de visualització online....43
5 – Implementació (Procés de creació d'una aplicació per Android)....44
5.1 - Descàrrega i instal·lació 45
5.2 - Les parts bàsiques d'un programa per Android....45
5.2.1 - Les classes....45
5.2.2 - El Layout....46
5.2.3 - Els drawable....46
5.2.4 - Els String....47
5.2.5 - El Manifest 47
5.3 - La pantalla inicial....48
5.3.1 - El Layout XML....48
5.3.2 - L'Activity....49
5.3.3 - Editar el Manifest....51
5.4 - La pantalla Explorar Series....53
5.4.1 - El Layout XML....53
a) ListView....53
b) Els elements del llistat....54
5.4.2 – L'Activity....57
a) Creant menús i submenús....57
b) Tasques en segon plà....59
c) Realizar operacions de xarxa....60
5.4.3 – L'Adapter....62
5.5 – Guardar info BBDD local 63
5.5.1 - La classe BBDD....64
5.5.2 - Mètode per guardar series a la base de dades....65
5.5.3 - Mètode per comprovar si existeix una serie en concret a la base de dades 66
5.5.4 - Mètode per obtenir un llistat amb totes les series....66
5.6 - Servei GCM....67

5.6.1 – IntentService....67
5.6.2 – El registre....68
5.6.4 – La recepció de missatges....68
5.7 - La part servidor....69
5.7.1 - Els arxius php....69
a) Obtenir llistat de totes les series....69
b) Afegir serie a la base de dades....69
c) Obtenir informació de la serie....70
d) Obtenir els capítols d'una serie....70
e) Guardar els capítols de les series que segueix un usuari....71
f) Funcions d'usuari....72
5.8 - La base de dades....73
5.9 – Captures de pantalla....74
5.9.1 – Pantalla d'autenticació....74
5.9.2 – Pantalla de registre....74
5.9.3 – Pantalla inicial....75
5.9.4 – Pantalla d'explorar series....75
5.9.5 – Pantalla d'informació d'una serie....76
5.9.6 – Pantalla d'informació de capítol....77
5.9.7 – Pantalla d'enllaços de descàrrega....77
5.9.8 – Pantalla de capítols pendents....78
5.9.9 – Pantalla per afegir nova serie....78
5.9.10 – Notificació de capítol nou....79
6 – Conclusions....80
7 - Bibliografia....81
8 - Glossari de termes....82

Índex d'il·lustracions

Il·lustració 1: Diagrama de planificació....10
Il·lustració 2: Diagrama de casos d'us. Part Servidor....14
Il·lustració 3: Diagrama de casos d'us. Part Client....15
Il·lustració 4: Arquitectura bàsica de funcionament....28
Il·lustració 5: Diagrama de classes. Pantalla inicial, registre i identificació....31
Il·lustració 6: Diagrama de classes. Altres pantalles....32
Il·lustració 7: Diagrama de seqüència. Afegir serie a preferides....33
Il·lustració 8: Diagrama de seqüència. Eliminar serie de preferides....34
Il·lustració 9: Diagrama de seqüència. Veure capítol online....35
Il·lustració 10: Diagrama de seqüència. Notificar usuaris....36
Il·lustració 11: Diagrama de persistència....37
Il·lustració 12: Prototip. Pantalla principal....40
Il·lustració 13: Prototip. Pantalla les meves series....40
Il·lustració 14: Prototip. Pantalla capítols pendents....41
Il·lustració 15: Prototip. Pantalla veure totes les series....42
Il·lustració 16: Prototip. Pantalla Serie....42
Il·lustració 17: Prototip. Pantalla capítol....43
Il·lustració 18: Prototip. Pantalla temporada....43
Il·lustració 19: Prototip. Pantalla enllaços....44
Il·lustració 20: Prototip. Pantalla visualització online....45
Il·lustració 21: Exemple de carpetes "layout" per a diferent mides de pantalla....47
Il·lustració 22: Exemple de carpetes "drawable" per a diferents mides de pantalla....47
Il·lustració 23: Cicle de vida d'un Activity....51
Il·lustració 24: Exemple d'un item del llistat....57
Il·lustració 25: Captura. Pantalla d'autenticació....75
Il·lustració 26: Captura. Pantalla de registre....75
Il·lustració 27: Captura. Pantalla inicial....76
Il·lustració 28: Captura. Pantalla explorar series, odre alfabètic....76
Il·lustració 29: Captura. Pantalla explorar series. Ordre cadena....76
Il·lustració 30: Captura. Pantalla explorar series, exemple de cerca....76
Il·lustració 31: Captura. Pantalla Serie. Carregant....77
Il·lustració 32: Captura. Pantalla serie....77
Il·lustració 33: Captura. Pantalla info. capítol....78
Il·lustració 34: Captura. Pantalla enllaços de descàrrega....78
Il·lustració 35: Captura. Pantalla capítols pendents....79
Il·lustració 36: Captura. Pantalla afegir serie....79
Il·lustració 37: Captura. Pantalla notificació capítol nou....80

1- Introducció i Context

1.1 - Introducció

El sistema operatiu per a dispositius mòbils Android, des de que va a aparèixer per primera vegada l'any 2007 ha tingut una enorme acceptació per part dels usuaris i consumidors de dispositius mòbils, convertint-se en l'actualitat en el sistema operatiu per a telèfons intel·ligents més estès arreu del mon i guanyant quota de mercat en el terreny de les tauletes.

és per aquesta raó i per el fet que la programació per aquest sistema operatiu no necessita d'una inversió inicial molt elevada, que s'ha escollit Android com la plataforma sobre la que desenvolupar el present Projecte de Final de Carrera.

1.2 - Justificació del projecte

El present Treball de Final de Carrera es troba dins l'àmbit del Desenvolupament d'aplicacions per a dispositius mòbils. L'objectiu del qual és la implementació d'una aplicació per algun dels sistemes operatius per a dispositius mòbils presents en l'actualitat. Concretament s'ha triat desenvolupar l'aplicació per a sistemes Android, degut al fet que es tracta d'un entorn proper i conegut el qual porto utilitzant des de fa diversos anys. A demés el fet que la programació per a aquest sistema operatiu utilitzi el llenguatge Java, el qual també és força conegut i que no és necessària una inversió en material ni software per a començar a desenvolupar a ajudat a l'elecció d'aquest sistema.

El present treball desenvolupa una aplicació per gestionar i permetre la descàrrega i la visualització de series de televisió en el propi dispositiu. S'ha escollit aquest tema degut a que hi ha un gran interès en el seguiment de series de televisió estrangeres per les quals no existeix cap manera de visualitzar-les fins que no s'emeten en el nostre país, a demés del fet que al seguir nombroses series simultàniament és molt útil el poder-ne fer un a gestió dels capítols vistos i disposar d'un mètode per ser notificat quan es disposa d'un nou capítol per a veure. En el moment d'iniciar el projecte no existia cap aplicació disponible al Play Store que reunís aquestes característiques de forma conjunta.

1.3 - Descripció del projecte

El Treball Final de carrera que s'ha realitzat és una aplicació per a dispositius mòbils amb sistema operatiu Android que permet als usuaris gestionar de les series de televisió que vol veure o que està seguint en un moment donat. Proporciona una solució per poder explorar series, és a dir, cercar en un llistat totes les series disponibles a la base de dades i fer uns seguiment de les series que l'usuari hagi marcat com a preferides. Aquest seguiment inclou el fet de poder marcar capítols com a vistos per conèixer en tot moment quin és el següent capítol que encara no s'ha vist, així com un sistema per a notificar als usuaris a través de l'enviament de notificacions "**push**" quan apareix un nou capítol d'alguna de les series que està seguint, d'aquesta manera no és necessari per part de l'usuari anar consultant els capítols d'una serie per saber si n'hi ha algun de nou.

A més, un cop seleccionat algun capítol, l'usuari també pot procedir a la seva descàrrega, ja sigui a través de serveis descàrrega directa o mitjançant arxius torrent,

els quals, depenent de les aplicacions que es tinguin descarregades, permet fer-ho directament al dispositiu o de forma remota a l'ordinador personal. També és possible, gràcies a la utilització d'una aplicació externa, visualitzar el capítol online, sense necessitat de fer-ne la descàrrega.

A continuació es descriuran el components principals del sistema:

- Una aplicació per a dispositius amb Android: Aquesta aplicació és la part principal del sistema, és la que permet interactuar amb l'usuari a través de la interfície gràfica que proporciona. Des d'aquí l'usuari pot consultar el llistat de series, marcar-les com a preferides per a fer-ne el seguiment, marcar els capítols com a vistos, realitzar-ne la descàrrega o visualització, afegir noves series, etc... Està codificada en Java utilitzant l'SDK d'Android mitjançant l'entorn de desenvolupament Eclipse en la seva versió Juno.
- Una base de dades: És la que permet la persistència de dades, és a dir, emmagatzemar de forma permanent tota la informació que necessita el sistema per a funcionar. Entre d'altres dades, es guarden les series, els capítols, enllaços i usuaris. Per a la seva implementació s'ha utilitzat MySQL.
- La part servidor: El servidor és la part que fa de pont entre l'aplicació del dispositiu i la base de dades. Consta de diferents classes que es criden des del dispositiu i que realitzen diferents consultes de a la base de dades. Ens permet guardar, consultar o modificar la informació que s'emmagatzema la base de dades, així com altres funcions com per exemple notificar al usuaris de que hi ha nous capítols per visualitzar. Per a la seva codificació s'ha utilitzat el llenguatge PhP mitjançant l'aplicació SublimeText2.
- Servei Google Cloud Messaging: Tot i ser un servei extern al sistema, és important destacar-ne la seva utilitat, ja que permet a través de peticions del nostre servidor enviar notificacions "**push**" als usuaris de l'aplicació.

1.4 - Objectius

A continuació s'esmentaran alguns dels objectius principals del Treball Final de carrera que s'ha realitzat.

- Aprofundir i dur a la practica els coneixements adquirits en numeroses assignatures de la carrera d'Enginyeria Tècnica en Informàtica de Gestió. Com per exemple:

  - Programació orientada a l'objecte: Bàsica per el desenvolupament de l'aplicació principal, ja que la programació per Android està basada en Java.

  - Estructura de la informació: De gran ajuda per a entendre com s'estructura la informació en un entorn informàtic i aprendre a treballar amb estructures com Cadenes, Taules, Arbres, etc...

- Bases de dades: Imprescindible per aprendre a treballar amb bases de dades, ja que ha sigut una part troncal del sistema que hem desenvolupat.

- Interfícies multimèdia: Ha permès aprendre a crear i millor interfícies per tal que fossin accessibles i fàcils d'utilitzar per l'usuari.

- Tècniques de desenvolupament de programari: Ha ajudat a comprendre com funciona el procés de creació d'una aplicació des de zero, des de l'anàlisi de requisits fins a la seva implementació.

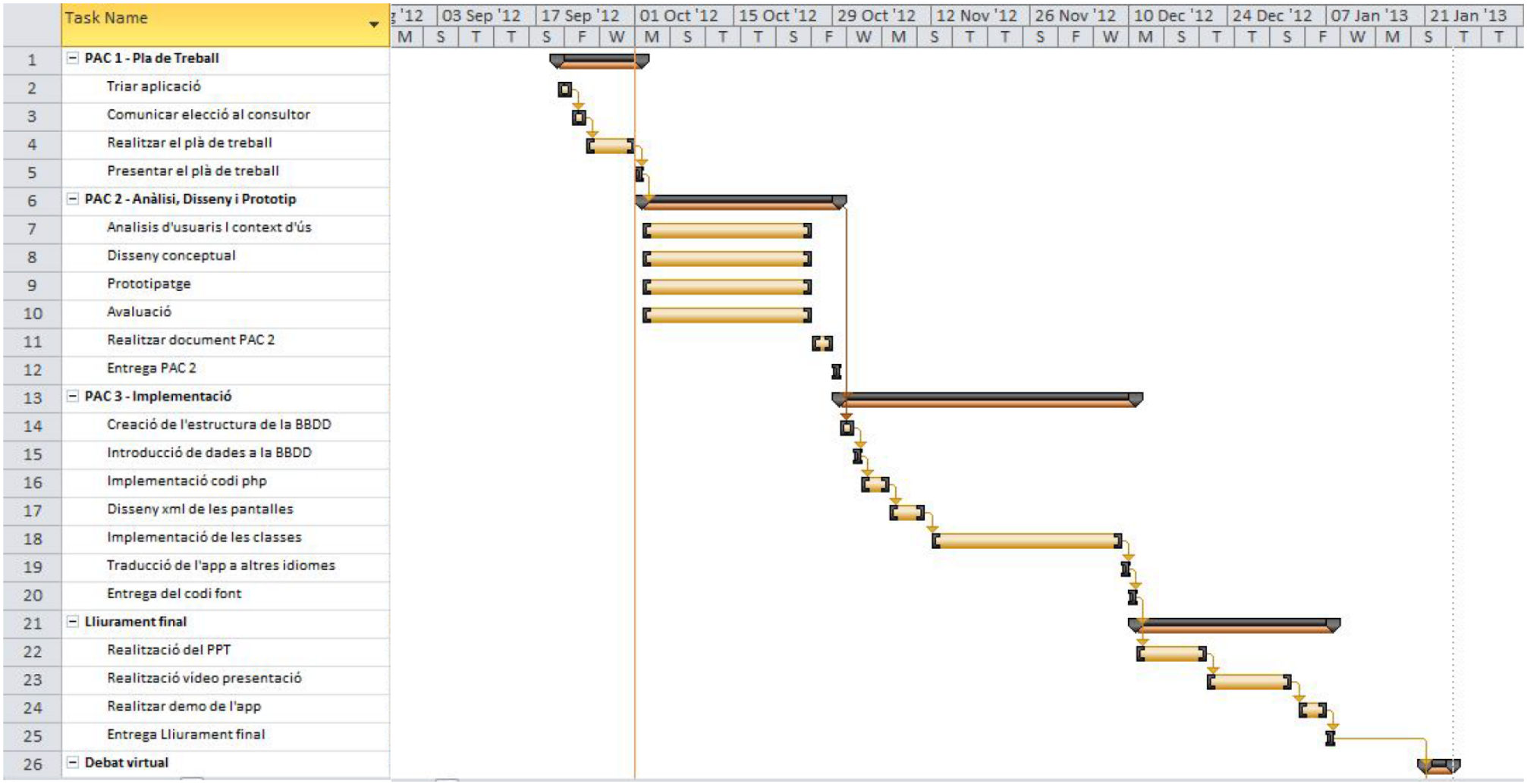
- Aprendre nous llenguatges de programació com per exemple PhP, que per al desenvolupament de la part del servidor ha sigut una eina indispensable.
- Aprendre a realitzar una aplicació per a sistemes operatius Android. Tot i que ja es coneixia com es realitza una aplicació amb Java, el fet de realitzar una aplicació per a Android ha sigut tot un repte. Aprendre quines són les parts bàsiques del programa, com interactuen, com s'utilitzen certs elements del sistema, com es treballa amb la seva interfície gràfica...
- Aprendre noves tecnologies com per exemple XML o JSON per a l'enviament de missatges estructurats, GCM per enviar notificacions Push, treballar amb CronoJobs, muntar un servidor local per a provar certes funcionalitats utilitzant WampServer...

1.5 - Planificació

1.5.1 - Calendari del projecte

Nom	Durada	Inici	Fi
PAC 1 – Pla de Treball	**8 dies**	**20/09/2012**	**21/09/2012**
Triar l'aplicació	2 dies	22/09/2012	21/09/2012
Comunicar elecció al consultor	2 dies	24/09/2012	23/09/2012
Realitzar el pla de treball	6 dies	20/09/2012	30/09/2012
Presentar el pla de treball	1 dies	01/10/2012	01/10/2012
PAC 2 – Anàlisi, Disseny i Prototip	**20 dies**	**02/10/2012**	**29/10/2012**
Anàlisi d'usuaris i context d'ús	18 dies	02/10/2012	25/10/2012
Disseny conceptual	18 dies	02/10/2012	25/10/2012
Prototipatge	18 dies	02/10/2012	25/10/2012
Avaluació	18 dies	02/10/2012	25/10/2012
Realitzar document PAC 2	2 dies	26/10/2012	28/10/2012
Entrega PAC 2	1 dies	29/10/2012	29/10/2012
PAC 3 - implementació	**30 dies**	**30/10/2012**	**10/12/2012**
Creació de l'estructura de la BBDD	2 dies	30/10/2012	31/10/2012
Introducció de dades a la BBDD	1 dies	01/11/2012	01/11/2012
Implementació del codi php	2 dies	02/11/2012	05/11/2012
Disseny xml de les pantalles	5 dies	06/11/2012	10/11/2012
Implementació de les classes	21 dies	12/11/2012	08/12/2012
Traducció de l'app a altres idiomes	1 dies	09/12/2012	09/12/2012
Entrega del coi font	1 dies	10/12/2012	10/12/2012
Lliurament Final	**20 dies**	**11/12/2012**	**07/01/2013**
Realització del PPT	8 dies	11/12/2012	20/12/2012
Realització del vídeo de presentació	8 dies	21/12/2012	01/01/2013
Realitzar demo de l'app	3 dies	03/01/2013	06/01/2013
Entrega Lliurament Final	1 dies	07/01/2013	07/01/2013
Debat virtual	**4 dies**	**21/01/2013**	**24/01/2013**

1.5.2 - Diagrama de la planificació



Il·lustració 1: Diagrama de planificació

1.6 - Eines utilitzades

Per a la realització del Treball de final de carrera s'han utilitzat les següents eines:

- **Eclipse Juno**, l'entorn Integrat de Desenvolupament (IDE) d'Oracle. És l'IDE per excel·lència a l'hora de programar per Android.
- **AIDE**, IDE disponible per a dispositius mòbils que permet desenvolupar aplicacions Android directament al dispositiu.
- **Android SDK Manager**, permet gestionar totes les versions disponibles del SDK d'Android, així com d'altres eines de desenvolupament i SDK's.
- **Android AVD Manager**, permet la gestió i configuració de diferents AVD (Android Virtual Device o Dispositiu Virtual Android) per tal de poder provar l'aplicació en diferents configuracions de dispositius (pantalles, versió del S.O., etc).
- **WampServer**, sistema integrat per a Windows que utilitza Apache com a servidor web, MySQL com gestor de bases de dades i (en el nostre cas) PhP com a llenguatge de programació. Utilitzat per a poder provar les funcionalitats de la part servidor de forma local.
- **SublimeText2:** Editor de textos que permet resaltar en diferents colors el codi depenent del llenguatge utilitzat. Molt eficient per la seva lleugeresa i facilitat d'ús. S'ha emprat per a codificar els arxius PhP.
- **Chrome i Firefox**, navegadors d'Internet utilitzat per a gestionar i provar la part del servidor.
- **LibreOffice**, editor de textos utilitzar per la realització dels documents.
- **Adobe Photoshop**, per a la creació d'elements gràfics del sistema i icones de l'aplicació.
- **Justinmind Prototyper**, per a la realització dels prototips.
- **Microsoft Visio**, per a dissenyar els diagrames.
- **Diferents dispositius mòbils**, per tal de provar l'aplicació en dispositius reals.

1.7 - Llenguatges de Programació utilitzats

Per a la realització del projecte ha sigut necessari més d'un llenguatge de programació:

- **Java**: El principal llenguatge de programació utilitzat. Juntament amb l'SDK d'Android és el que ens ha permès crear l'aplicació que s'executarà en els dispositius mòbils, és a dir la part "client".
- **Php**: Ha sigut el llenguatge emprat per tal de crear les funcionalitats de la part del servidor.
- **SQL**: Llenguatge emprat per tal de construir, gestionar i realitzar consultes a la base de dades remota.

2 - Requisits de l'aplicació

2.1 – Usuaris

Aquesta aplicació consta d'un tipus usuari principal, que és el que farà ús de l'aplicació, però també podem considerar dos tipologies d'usuari més, que tot i que no interactuaran directament amb l'aplicació poden tenir certa importància per al bon funcionament del sistema, a continuació procedirem a descriure'ls:

- **Usuari estàndard:** Podrà realitzar, un cop registrat, totes les funcions de la nostra aplicació. Consultar series, afegires al llistat de preferides, marcar capítols vistos, descarregar capítols...
- **Usuari Gestor de la base de dades:** No farà ús de l'aplicació per a dispositius mòbils, sinó que serà l'encarregat de comprovar periòdicament la base de dades per tal de fer-ne un manteniment. Comprovar que no hi hagi series duplicades, que no s'hagi introduït informació erronea o introduir dades noves si fos necessari.
- **Usuari Administrador:** Podrà realitzar les mateixes accions que l'usuari gestor de la base de dades. A més a més també serà l'encarregat de gestionar la part de la base de dades relacionada amb els usuaris, com per exemple, afegir o eliminar algun usuari si aquest fa un mal ús de l'aplicació.

2.2 - Requisits funcionals

Tot i que durant la fase d'anàlisi es van proposar molts requisits que hauria de tenir l'aplicació al final no ha sigut possible implementar-los tots, a continuació s'esmenten els requisits que s'ha implementat:

- **Registrar-se:** Permet a l'usuari registrar-se al sistema per fer ús de totes les funcionalitats existents.
- **Identificar-se:** Un cop registrat l'usuari podrà identificar-se per tal d'accedir al sistema.
- **Fer un seguiment de les series preferides:** La funcionalitat principal d'aquesta tasca és el fet de poder marcar diferents series com a preferides per a fer-ne un seguiment, és a dir conèixer els capítols existents o saber si n'apareixen de nous per a poder descarregar-los, marcar capítols com a vistos per saber quin toca veure, etc.
- **Cercar Series:** Permet cercar series per diferents criteris, d'aquesta manera es poden trobar noves series que l'usuari no coneixia pero que reuneixen unes característiques, temàtica o actors que poden ser del seu interes.
- **Descarregar capítols de forma remota:** Permet descarregar un capítol de forma remota a un ordinador on es tingui instal·lat algun software que permeti aquesta

característica com pot ser µTorrent.

- **Descarregar capítols al mateix dispositiu:** Permet descarregar de forma local el capítol, ja sigui a traves de descarrega directa o utilitzant serveis P2P que disposin d'aplicació per Android.
- **Afegir una nova serie:** Permet a l'usuari afegir una nova serie a la base de dades remota.
- **Afegir un nou capítol:** Permet afegir un capítol nou d'alguna de les series que ja existeixen a la base de dades.

2.3 - Requisits d'informació

La informació que s'ha considerat necessària per tal de poder donar d'alta tots els elements és la següent:

Series

- **Nom:** El nom de la serie.
- **Cadena:** El nom de la cadena on s'emet la serie.
- **Temporades:** el número de temporades de que disposa la serie.

Capítols

- **Nom:** Nom del capítol.
- **Temporada:** Temporada a la que pertany el capítol.
- **Número:** Número del capítol dins de la temporada.
- **Data:** Data en que es va emetre el capítol per primera vegada.

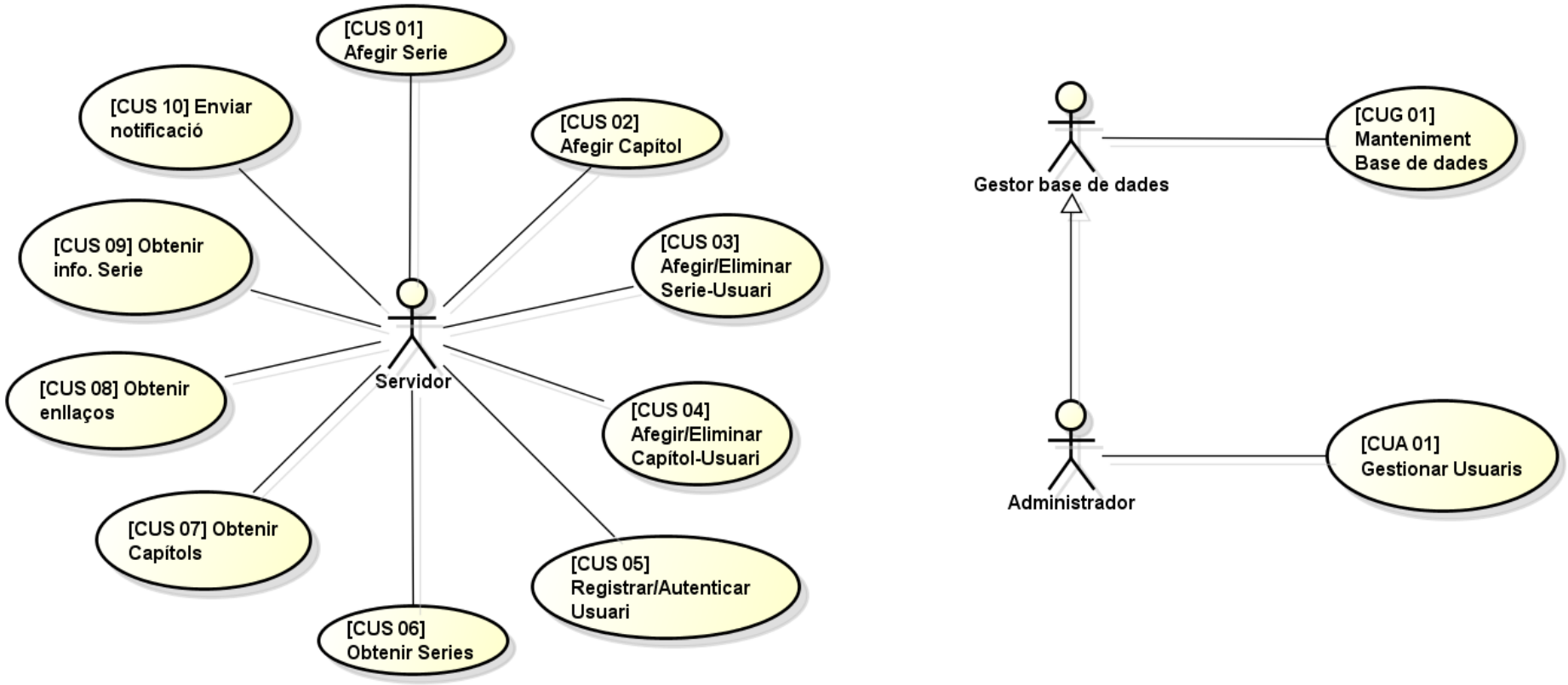
Usuari

- **Nom:** El nom complert de l'usuari.
- **Email:** El correu electrònic de l'usuari, s'utilitzara com a identificador.
- **Password:** Contrasenya amb la qual voldrà identificar-se.

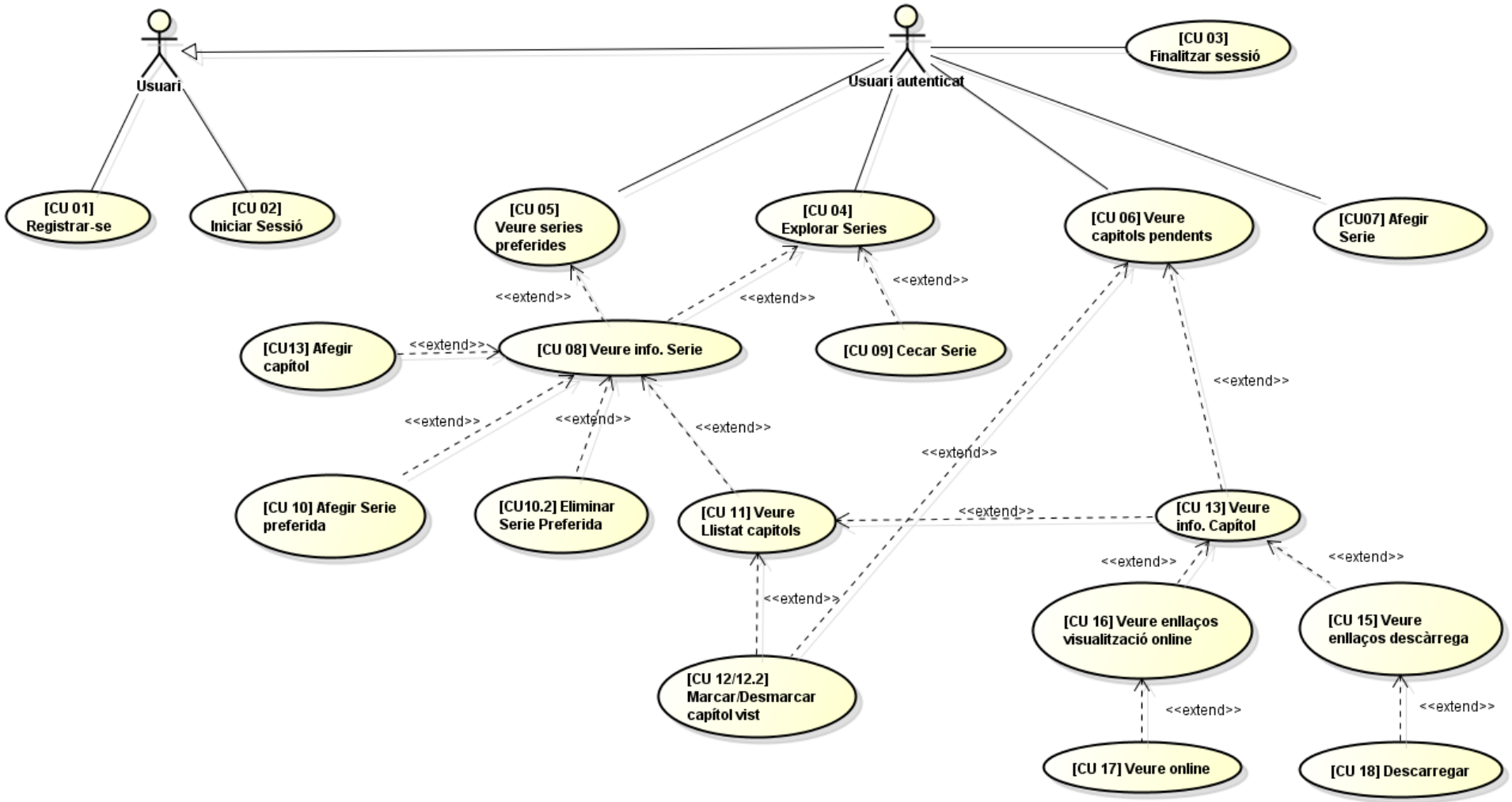
Aquests requisits podran ser modificats a mesura que l'aplicació es vagi actualitzant, podent afegir camps nous a alguns dels elements.

3 - Anàlisis del sistema

3.1 – Diagrames de casos d'ús



Il·lustració 2: Diagrama de casos d'us. Part Servidor



Il·lustració 3: Diagrama de casos d'us. Part Client

3.2 - Descripció textual dels casos d'ús

A continuació es descriuran els casos d'ús de forma textual indicant els atributs més important. En alguns casos, algunes funcions que es podrien haver considerat com a casos d'ús independents, s'ha decidit incloure'ls com a fluxos alternatius d'altres casos d'ús per tal de no complicar l'estructura ni la gràfica. Alguns exemples són el fet d'ordenar el llistat de series, refrescar els capítols pendents, etc..

3.2.1 - CU01 - Registrar-se

Codi	CU01
Nom	Registrar-se
Descripció	Permet a l'usuari registrar-se en el sistema per a poder-hi accedir posteriorment.
Actors	Usuari
Precondicions	L'usuari no ha d'estar registrat en el sistema
Flux Normal	a) A l'iniciar l'aplicació apareix la pantalla d'autenticació b) L'usuari tria l'opció de registrar-se c) S'omplen les dades demanades d) Es prem sobre el botó de registrar-se e) Finalitza el registre
Flux Alternatiu	En cas que alguna de les dades no sigui correcta e1.1) L'aplicació avisa que alguna de les dades no és correcta e1.2) L'usuari corregeix les dades e1.3) Es prem sobre el botó de registrar-se e1.4) Finalitza el registre En cas que l'usuari ja estigues registrat e2.1) L'aplicació avisa que l'usuari ja estava registrat e2.2) Es torna a la pantalla d'autenticació
Postcondicions	L'usuari queda registrat al sistema

3.2.2 - CU02 – Iniciar Sessió

Codi	CU02
Nom	Iniciar Sessió
Descripció	Permet a l'usuari iniciar sessió al sistema per tal de poder utilitzar l'aplicació.
Actors	Usuari
Precondicions	L'usuari no ha d'estar autenticat en el sistema
Flux Normal	a) A l'iniciar l'aplicació apareix la pantalla d'autenticació b) L'usuari omple les dades c) Es prem sobre el botó de "Login" d) Finalitza l'autenticació
Flux Alternatiu	En cas que alguna de les dades no sigui correcta o l'usuari no estigui registrat al sistema d1.1) L'aplicació avisa que alguna de les dades no és correcta d1.2) L'usuari corregeix les dades d1.3) Es prem sobre el botó de "Login" d1.4) Finalitza l'autenticació
Postcondicions	L'usuari queda autenticat al sistema

3.2.3 - CU03 – Finalitzar Sessió

Codi	CU03
Nom	Finalitzar Sessió
Descripció	Permet a l'usuari finalitzar la sessió oberta.
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat en els sistema
Flux Normal	a) En qualsevol de les pantalles l'usuari prem el botó de menú b) L'usuari tria l'opció de finalitzar la sessió c) Es finalitza la sessió d) Es torna a la pantalla d'autenticació
Flux Alternatiu	
Postcondicions	La sessió queda finalitzada

3.2.4 - CU04 – Explorar Series

Codi	CU04
Nom	Explorar Series
Descripció	Permet a l'usuari explorar totes les series accedint a un llistat d'aquestes.
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema
Flux Normal	a) Al menú principal es tria l'opció d'Explorar Series b) Es mostra el llistat de series
Flux Alternatiu	Hi ha hagut un error de connexió b1.1) L'aplicació avisa que hi ha hagut un error de connexió b1.2) Es torna al menú principal L'usuari vol ordenar el llistat de series c) L'usuari prem l'opció d'ordenar d) S'escull l'opció d'ordenació que convé e) Es mostra el llistat de totes les series ordenat pel criteri escollit
Postcondicions	Es mostra per pantalla un llistat amb totes les series

3.2.5 - CU05 – Veure series preferides

Codi	CU05
Nom	Veure Series Preferides
Descripció	Permet a l'usuari veure un llistat amb les series que segueix.
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i ha d'estar seguint alguna serie
Flux Normal	a) Al menú principal es tria l'opció de Les Meves Series b) Es mostra el llistat de series que es segueixen
Flux Alternatiu	
Postcondicions	Es mostra per pantalla un llistat amb les series que l'usuari segueix

3.2.6 - CU06 – Veure capítols pendents

Codi	CU06
Nom	Veure capítols pendents
Descripció	Permet a l'usuari veure un llistat amb el primer capítol no vist de cada una de les series que segueix.
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema
Flux Normal	a) Al menú principal es tria l'opció de Veure capítols pendents b) Es mostra el llistat de capítols pendents de veure
Flux Alternatiu	No hi ha capítols pendents de veure: a1.1) Es mostra un llistat buit Es vol refrescar el llistat per actualitzar la informació: b1.1) Es prem sobre el botó de refrescar b1.2) Es torna a mostrar el llistat amb la nova informació Es vol consultat la base de dades per si han sortit nous capítols: b2.1) Es prem sobre el menú d'opcions b2.2) S'escull l'opció "Comprovar nous capítols" b3.3) Es torna a mostrar el llistat amb els nous capítols (si n'hi ha)
Postcondicions	Es mostra per pantalla un llistat amb tots els capítols pendents de veure

3.2.7 - CU07 – Afegir serie

Codi	CU07
Nom	Afegir serie
Descripció	Permet a l'usuari afegir una nova serie a la base de dades
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema.
Flux Normal	a) Al menú principal es tria l'opció d'"Afegir nova serie" b) EL sistema obre una nova pantalla amb el formulari a omplir. c) L'usuari omple el formulari. d) L'usuari prem el botó "Afegir". e) El sistema afegeix la serie a la base de dades.
Flux Alternatiu	No s'han introduït tots els camps: e1.1) El sistema avisa que falta algun camp per omplir. e1.2) L'usuari omple el camp que falta.. e1.3) L'usuari prem el botó "Afegir". e1.4) El sistema afegeix la serie a la base de dades.
Postcondicions	La serie queda afegida a la base de dades

3.2.8 - CU08 – Veure informació de la serie

Codi	CU08
Nom	Veure informació de la serie
Descripció	Permet a l'usuari veure informació d'una serie en concret
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla d'Explorar series (CU04) o bé Veure series preferides (CU05)
Flux Normal	a) Es prem sobre alguna de les series del llistat b) S'obre una nova pantalla c) El sistema carrega la informació de la serie des de la base de dades d) Es mostra la informació de la serie
Flux Alternatiu	Hi ha hagut un error de connexió d1.1) El sistema avisa que hi ha hagut un error de connexió d1.2) Es torna a la pantalla anterior
Postcondicions	Es mostra per pantalla un la informació de la serie seleccionada.

3.2.9 - CU09 – Cercar Serie

Codi	CU09
Nom	Cercar Serie
Descripció	Permet a l'usuari cercar una serie en concret
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla d'Explorar series (CU04)
Flux Normal	a) Es prem l'icona de cercar b) A la barra de cerca que apareix es tecleja el nom de la serie que es vol cercar c) A mesura que es va escrivint s'actualitza el llistat amb les series que contenen el text introduït
Flux Alternatiu	
Postcondicions	Es mostra per pantalla el llistat amb les series coincidents amb el text

3.2.10 - CU10 – Afegir Serie preferida

Codi	CU10
Nom	Afegir Serie preferida
Descripció	Permet a l'usuari afegir una serie al llistat de series preferides
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla de Veure informació de Serie (CU07)
Flux Normal	a) Es prem sobre el botó d'Afegir serie b) El sistema afegeix la serie al llistat de series preferides
Flux Alternatiu	
Postcondicions	La serie queda afegida al llistat de preferides

3.2.11 - CU010.2 – Eliminar Serie preferida

Codi	CU010.2
Nom	Eliminar Serie preferida
Descripció	Permet a l'usuari eliminar una serie del llistat de series preferides
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla de Veure informació de Serie (CU07)
Flux Normal	a) Es prem sobre el botó d'Eliminar serie b) El sistema pregunta si l'usuari està segur d'eliminar la serie del llistat c) L'usuari confirma l'acció d) El sistema elimina la serie del llistat de preferides
Flux Alternatiu	b1.3) L'usuari cancel·la l'acció b1.4) Es cancel·la l'acció d'eliminar la serie del llistat de preferides
Postcondicions	La serie queda eliminada del llistat de preferides

3.2.12 - CU11– Veure Llistat de capítols

Codi	CU11
Nom	Veure llistat de capítols
Descripció	Permet a l'usuari veure el llistat de capítols disponibles d'una serie
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla de Veure informació de Serie (CU07)
Flux Normal	a) Es prem sobre alguna de les temporades de la serie b) S'expandeix el llistat de temporades i es mostren els capítols corresponents a la temporada premuda
Flux Alternatiu	L'usuari vol amagar el llistat de capítols b1.1) Es torna a prémer sobre el nom d'alguna de les temporades ja obertes b1.2) Es contrau el llistat de capítols de la temporada
Postcondicions	Es mostren per pantalla els capítols de la serie

3.2.13 - CU12 – Marcar capítol vist

Codi	CU12
Nom	Marcar capítol vist
Descripció	Permet a l'usuari marcar un capítol com a vist
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla de Veure informació de Serie (CU07), el capítol no ha d'estar marcat com a vist i la serie a la que pertany ha d'estar al llistat de preferides
Flux Normal	a) Es prem sobre alguna de les temporades de la serie b) S'expandeix el llistat de temporades i es mostren els capítols corresponents a la temporada premuda c) Es prem sobre la casella de verificació de la part dreta d) El capítol queda marcat com a vist e) Es mostra un missatge informant del canvi
Flux Alternatiu	La seria a la qual pertany el capítol no es troba en el llistat de preferides c1.1) El sistema avisa que la serie ha d'estar al llistat de preferides per poder marcar algun dels seus capítols com a vist c1.2) El sistema demana a l'usuari si vol afegir la seria al llistat de preferides c1.3) Si l'usuari accepta, la serie queda afegida al llistat de preferides i es pot tornar a començar el cas d'ús
Postcondicions	El capítol queda marcat com a vist

3.2.14 - CU12.2 – Desmarcar capítol vist

Codi	CU12
Nom	Desmarcar capítol vist
Descripció	Permet a l'usuari desmarcar un capítol com a vist
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla de Veure informació de Serie (CU07), el capítol ha d'estar marcat com a vist.
Flux Normal	a) Es prem sobre alguna de les temporades de la serie b) S'expandeix el llistat de temporades i es mostren els capítols corresponents a la temporada premuda c) Es prem sobre la casella de verificació de la part dreta d) El capítol queda marcat com a no vist e) Es mostra un missatge informant del canvi
Flux Alternatiu	
Postcondicions	El capítol queda marcat coma no vist

3.2.15 - CU13 – Afegir capítol

Codi	CU13
Nom	Afegir capítol
Descripció	Permet a l'usuari afegir un capítol a la serie
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla de Veure informació de Serie (CU07).
Flux Normal	a) Es prem sobre l'icona amb el símbol "suma" del menú de l'aplicació b) El sistema mostra la pantalla amb el formulari a omplir c) L'usuari omple el formulari d) Es prem el botó afegir e) El capítol queda afegit
Flux Alternatiu	Alguna de les dades introduïdes no és correcta. d1.1) El sistema avisa que algun dels camps no és correcte. d1.2) L'usuari corregeix els camps. d1.3) Es prem el botó afegir d1.4) El capítol queda afegit
Postcondicions	El capítol queda afegit a la serie

3.2.16 - CU14 – Veure informació capítol

Codi	CU14
Nom	Veure informació capítol
Descripció	Permet a l'usuari veure la informació detallada d'un capítol
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla de Veure informació de Serie (CU07) amb la llista de capítols expandida, o bé a la pantalla Veure capítols pendents (CU06).
Flux Normal	a) Es prem sobre algun dels capítols del llistat b) S'obre una nova pantalla c) Es mostra la informació detallada del capítol
Flux Alternatiu	
Postcondicions	El mostra la informació del capítol.

3.2.17 - CU15– Veure enllaços de visualització online

Codi	CU15
Nom	Veure enllaços de visualització online
Descripció	Permet a l'usuari veure els enllaços de visualització online d'un capítol en concret.
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla de Veure informació de capítol (CU13).
Flux Normal	a) Es prem sobre el botó "Veure online". b) S'obre una nova pantalla amb un llistat d'enllaços
Flux Alternatiu	
Postcondicions	Es mostren els enllaços de visualització online del capítol.

3.2.18 - CU16 – Veure enllaços de descàrrega

Codi	CU16
Nom	Veure enllaços de descàrrega
Descripció	Permet a l'usuari veure els enllaços de descàrrega d'un capítol en concret.
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla de Veure informació de capítol (CU13).
Flux Normal	a) Es prem sobre el botó "Descarregar". b) S'obre una nova pantalla amb un llistat d'enllaços de descàrrega
Flux Alternatiu	
Postcondicions	Es mostren els enllaços de descàrrega del capítol.

3.2.19 - CU17 – Veure online

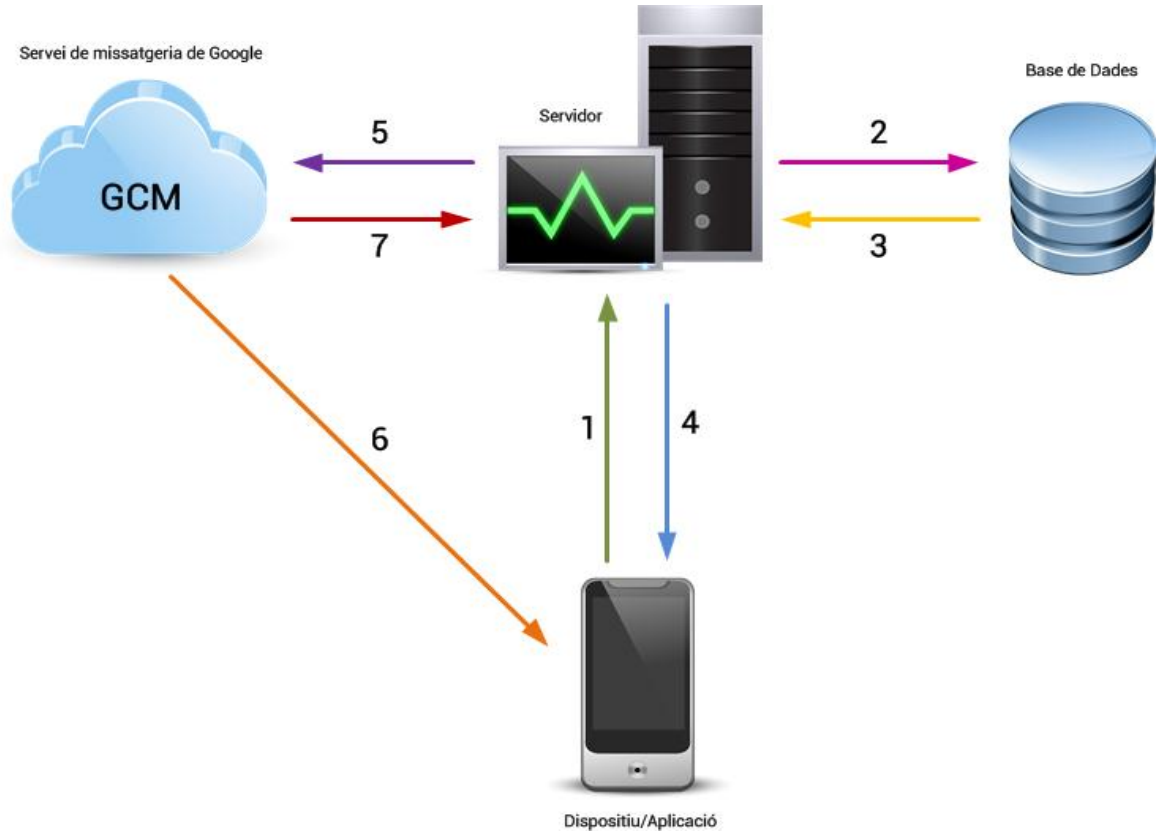
Codi	CU17
Nom	Veure online
Descripció	Permet a l'usuari veure el capítol que desitja directament online, és a dir, sense descarregar el capítol.
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla de "Veure enllaços de visualització online" (CU14).
Flux Normal	a) L'usuari prem sobre algun dels servidors de visualització online b) El sistema expandeix el llistat d'enllaços. c) L'usuari prem sobre algun dels enllaços disponibles. d) S'obre l'aplicació externa "Hubi" per visualitzar el capítol. e) L'usuari prem la tecla "Play" de l'aplicació externa. d) Es reprodueix el capítol
Flux Alternatiu	L'aplicació "Hubi" no està instal·lada: c1.1) El sistema avisa a l'usuari que és necessària l'aplicació per poder visualitzar el capítol i pregunta a l'usuari si la vol descarregar. c1.2) Si l'usuari accepta, s'obre el Google Play per a descarregar-la. c1.3) L'usuari descarrega i instal·la l'aplicació. c1.4) Es procedeix a realitzar el flux normal. Si l'usuari no accepta la descàrrega de l'aplicació: c1.2.1) Es tanca l'avís i es torna a la pantalla de "Veure enllaços de visualització online" (CU14).
Postcondicions	S'inicia la reproducció del capítol.

3.2.20 - CU18 – Descarregar

Codi	CU18
Nom	Descarregar
Descripció	Permet a l'usuari descarregar el capítol. Depenent de les aplicacions que tingui instal·lades al dispositiu, es pot descarregar al dispositiu o remotament a l'ordinador personal.
Actors	Usuari
Precondicions	L'usuari ha d'estar autenticat al sistema i s'ha de trobar a la pantalla de "Veure enllaços de descàrrega" (CU15).
Flux Normal	a) L'usuari prem sobre algun dels servidors de descàrrega b) El sistema expandeix el llistat d'enllaços. c) L'usuari prem sobre algun dels enllaços disponibles. d) Es demana amb quina aplicació instal·lada es vol procedir a realitzar la descàrrega e) L'usuari escull a l'aplicació d) S'inicia l'aplicació escollida per procedir amb la descàrrega
Flux Alternatiu	No hi ha cap aplicació instal·lada que permeti la descàrrega: c1.1) El sistema avisa a l'usuari que no hi ha cap aplicació instal·lada que permeti la descàrrega d'aquell tipus d'arxiu. c1.2) Es torna a la pantalla de "Veure enllaços de descàrrega" (CU15).
Postcondicions	S'inicia la descàrrega del capítol.

4 – Disseny

4.1 - Arquitectura bàsica de funcionament



Il·lustració 4: Arquitectura bàsica de funcionament

En l'esquema anterior es mostra l'esquema bàsic de funcionament de l'aplicació, es poden observar les tres part principals de les quals consta, és a dir:

- El dispositiu mòbil, des d'on s'executarà l'aplicació.
- El servidor, on s'emmagatzemen els scripts php, que s'encarreguen de realitzar diferents funcions.
- La base de dades, on s'emmagatzemen totes les dades necessàries, tant d'usuaris com de series i capítols.

A demés també podem observar que el servidor es comunica amb el servei de missatgeria en el núvol de Google (o Google Cloud Messaging – GCM) per tal de poder enviar notificacions Push al dispositiu.

Un exemple de com es comuniquen les diferents parts de l'aplicació podria ser:

1. Qualsevol canvi o petició d'informació que realitzi l'usuari en l'aplicació, s'enviarà al servidor per tal que executi l'script pertinent.
2. Aquest script demanarà, guardarà o eliminarà (depenent del tipus d'acció que vulgui realitzar l'usuari) informació a la base de dades a través d'un query.
3. La base de dades retornarà la informació demanada o la confirmació de que s'ha realitzat el canvi de forma correcta al servidor
4. El servidor retornarà, en format JSON la informació demanada o la confirmació de que s'ha realitzat el canvi de forma correcta a l'aplicació, i aquesta s'encarregarà de mostrar-ho per pantalla.

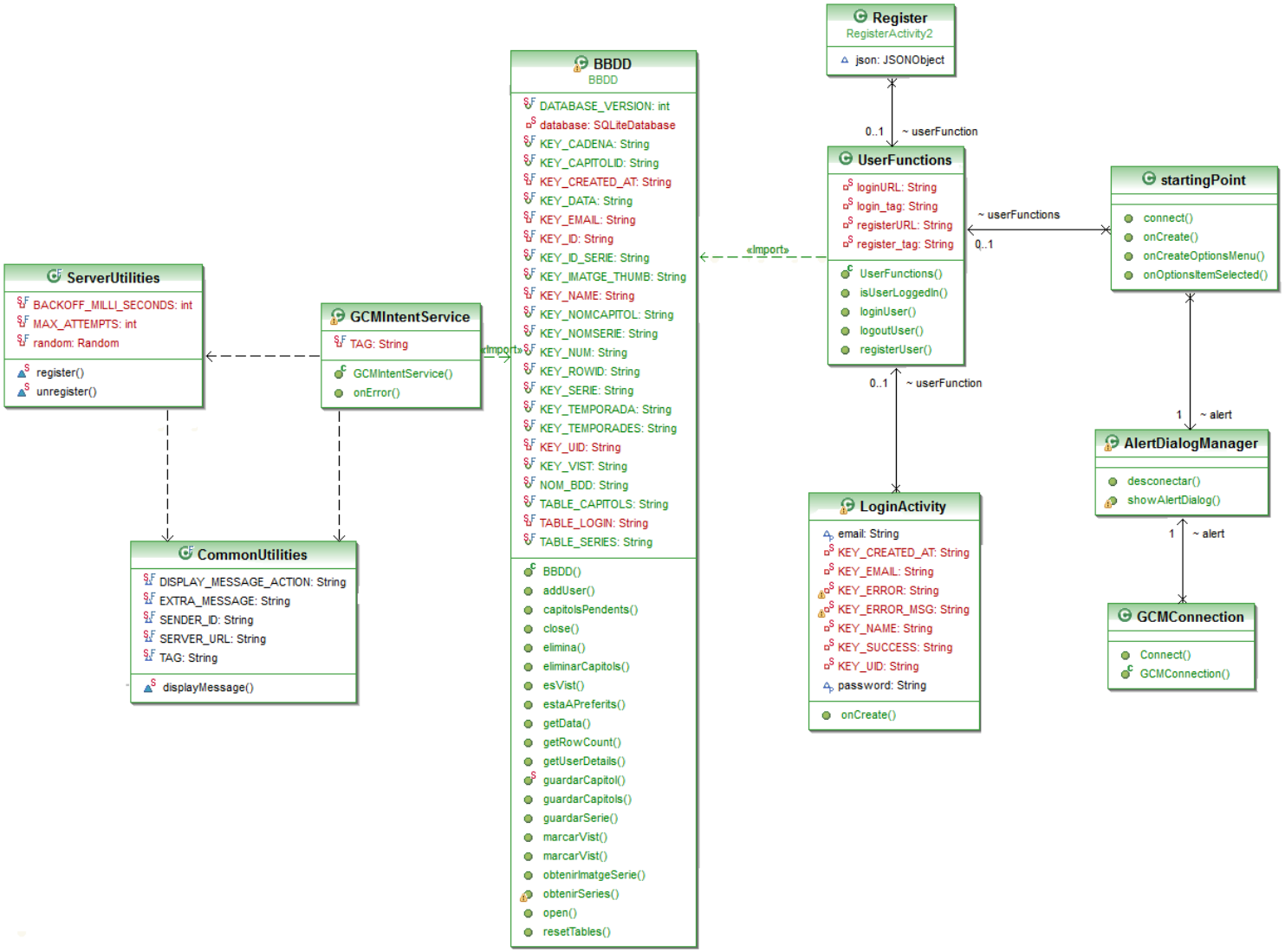
En el cas de les notificacions "push", en la nostra aplicació es fan servir per comunicar a l'usuari que s'ha afegit a la base de dades un nou capítol d'alguna de les series que està seguint. Per a realitzar tal funció es procedeix de la següent manera:

- El servidor va executant un script cada cert temps que s'encarrega de comprovar si hi ha algun capítol nou, un cop el troba comprova que tingui algun enllaç disponible, si el té cerca tots els usuaris que estan seguint aquella serie i procedeix a l'enviament de la notificació.
- Per a fer-ho notifica al GCM quins usuaris han de rebre la notificació i quin ha de ser el missatge (5).
- Aleshores el GCM envia a tots els usuaris (Multidifusió) que han de ser notificats el missatge assignat (6).
- I per ultim, el propi GCM, notifica al servidor (7) quin ha sigut el resultat de l'enviament de la notificació, és a dir, a quins usuaris s'ha enviat, si s'ha fet correctament, els errors que hi hagi pogut haver, etc...

Cal tenir en compte que el sistema de missatgeria de Google només notifica al servidor que el missatge ha sigut enviat, és a dir, no disposa de cap funcionalitat per comprovar que el missatge ha sigut rebut correctament. Per tal de poder contemplar aquest fet, l'aplicació hauria d'implementar un sistema en el qual la base de dades guardés l'últim missatge enviat pel GCM, d'aquesta manera, l'aplicació podria comparar l'últim missatge rebut amb l'últim missatge enviat per tal de recuperar possibles errors de recepció. Aquesta versió de l'aplicació no contempla aquest fet.

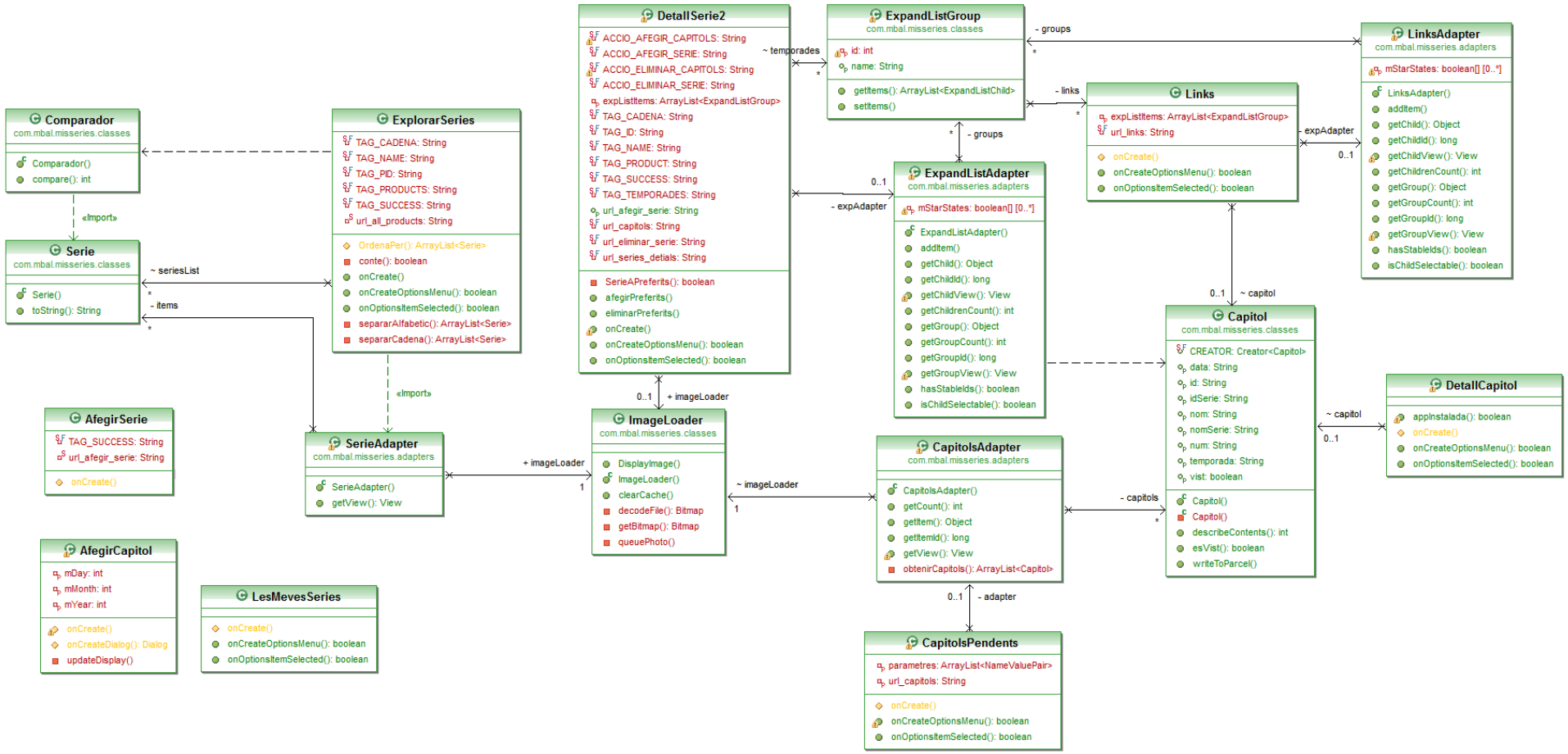
4.2 – Diagrama de classes

4.2.1 – Pantalla inicial, registre i identificació



Il·lustració 5: Diagrama de classes. Pantalla incial, registre i identificació

4.2.2 – Altres pantalles

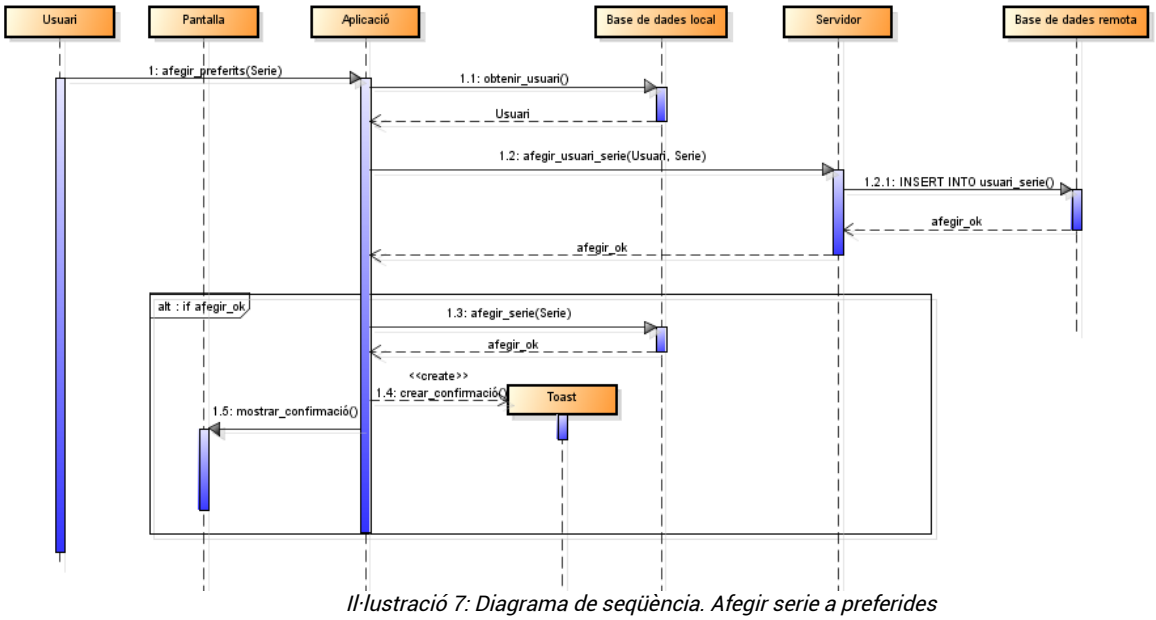


Il·lustració 6: Diagrama de classes. Altres pantalles

4.3 - Diagrames de seqüència

A continuació mostrarem els diagrames de seqüencia d'algunes de interaccions entre els diferents elements del sistema. Només esmentarem les més representatives, o aquelles en les que intervenen més elements del sistema, d'aquesta manera ens podrem fer una idea de com actuen els diferents elements sense haver de mostrar els diagrames de seqüencia de totes les interaccions.

4.3.1 - Afegir Serie a preferides

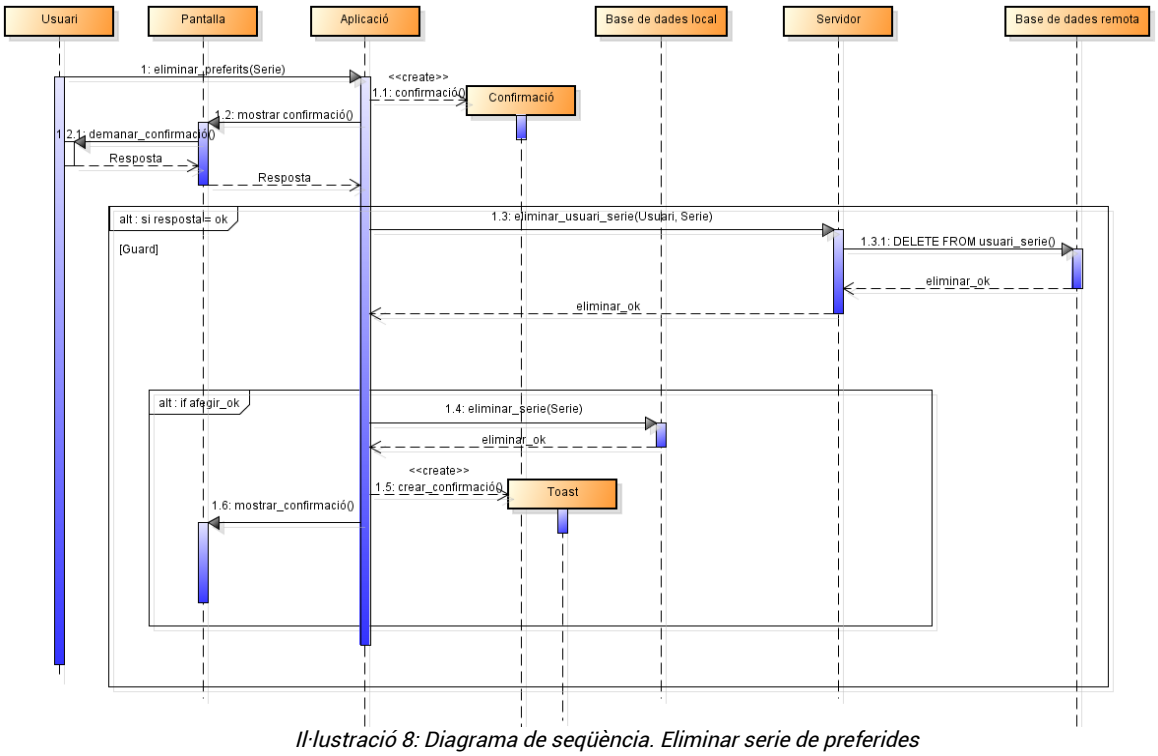


Il·lustració 7: Diagrama de seqüència. Afegir serie a preferides

Seqüencia textual:

- L'usuari (que es troba a la pantalla d'informació de la serie) prem el botó d'afegir Serie al llistat de preferides.
- L'aplicació demana a la base de dades local la informació de l'usuari.
- L'aplicació sol·licita al servidor que afegeixi una nova combinació d'Usuari-Serie.
- El servidor executa el query necessari a la base de dades perquè afegeixi el nou camp a la taula corresponent.
- Si el procés s'ha realitzat amb èxit l'aplicació crea un **Toast** de confirmació i el mostra per pantalla.

4.3.2 - Eliminar Serie de preferides

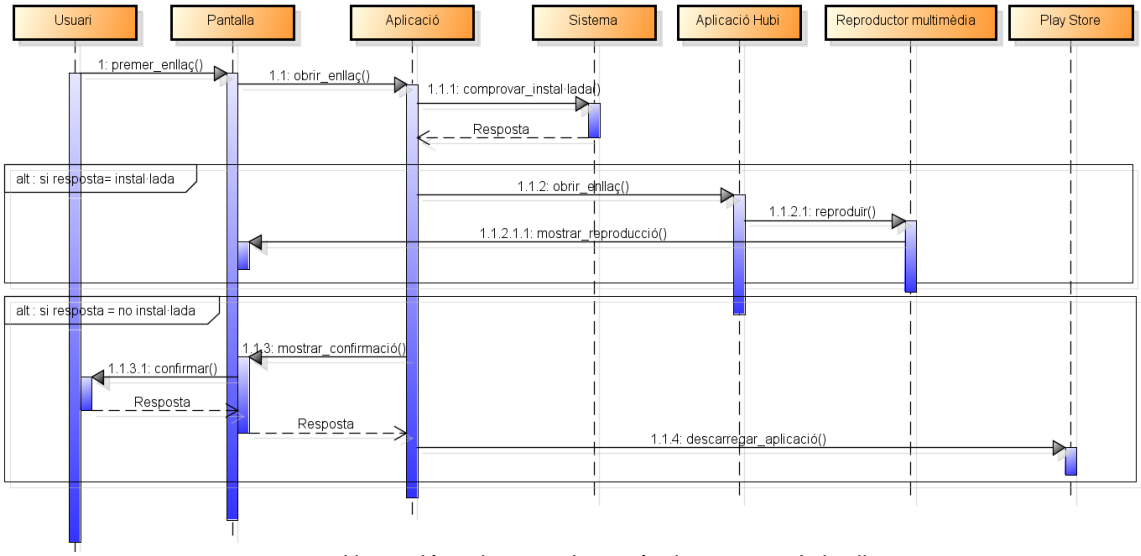


Il·lustració 8: Diagrama de seqüència. Eliminar serie de preferides

És una seqüència molt similar a l'anterior, amb la diferencia de que quan es vol eliminar una serie del llistat de series preferides, prèviament l'aplicació demana a l'usuari si realment desitja realitzar l'acció. Textualment queda així:

- L'usuari (que es troba a la pantalla d'informació de la serie) prem el botó d'eliminar Serie del llistat de preferides.
- L'aplicació mostra per pantalla la confirmació de realització de l'acció.
- Si l'usuari accepta:
- L'aplicació demana al servidor que elimini la combinació d'Usuari-Serie
- El servidor executa el query necessari a la base de dades perquè elimini el camp de la taula corresponent.
- Si el procés s'ha realitzat amb èxit l'aplicació crea un **Toast** de confirmació i el mostra per pantalla.

4.3.3 - Veure capítol online

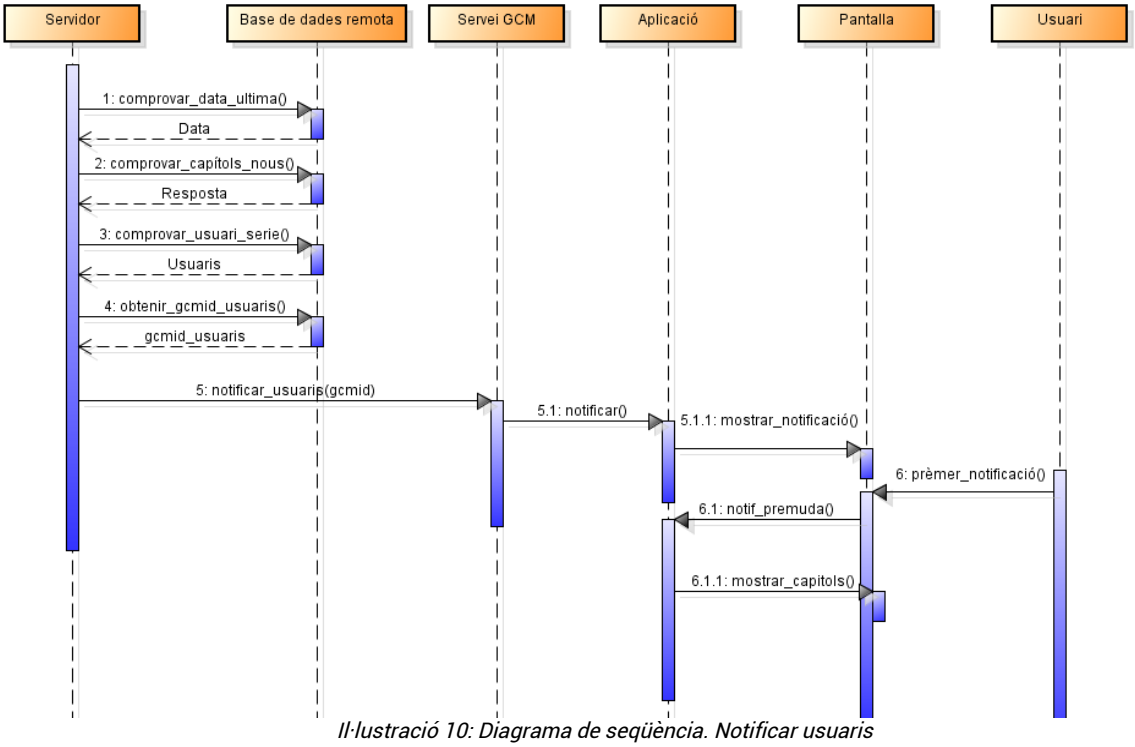


Il·lustració 9: Diagrama de seqüència. Veure capítol online

Seqüència textual:

- L'usuari prem sobre l'enllaç que vol reproduir
- L'aplicació comprova si l'aplicació "Hubi" (necessària per veure els capítols online) està instal·lada.
- En cas afirmatiu:
- Obre l'enllaç amb l'aplicació "Hubi" i es reprodueix el capítol.
- En cas negatiu:
- Pregunta a l'usuari si la vol descarregar.
- Si l'usuari accepta, s'obre el Play Store per a descarregar l'app.

4.3.4 - Notificar Usuaris quan hi ha nous capítols



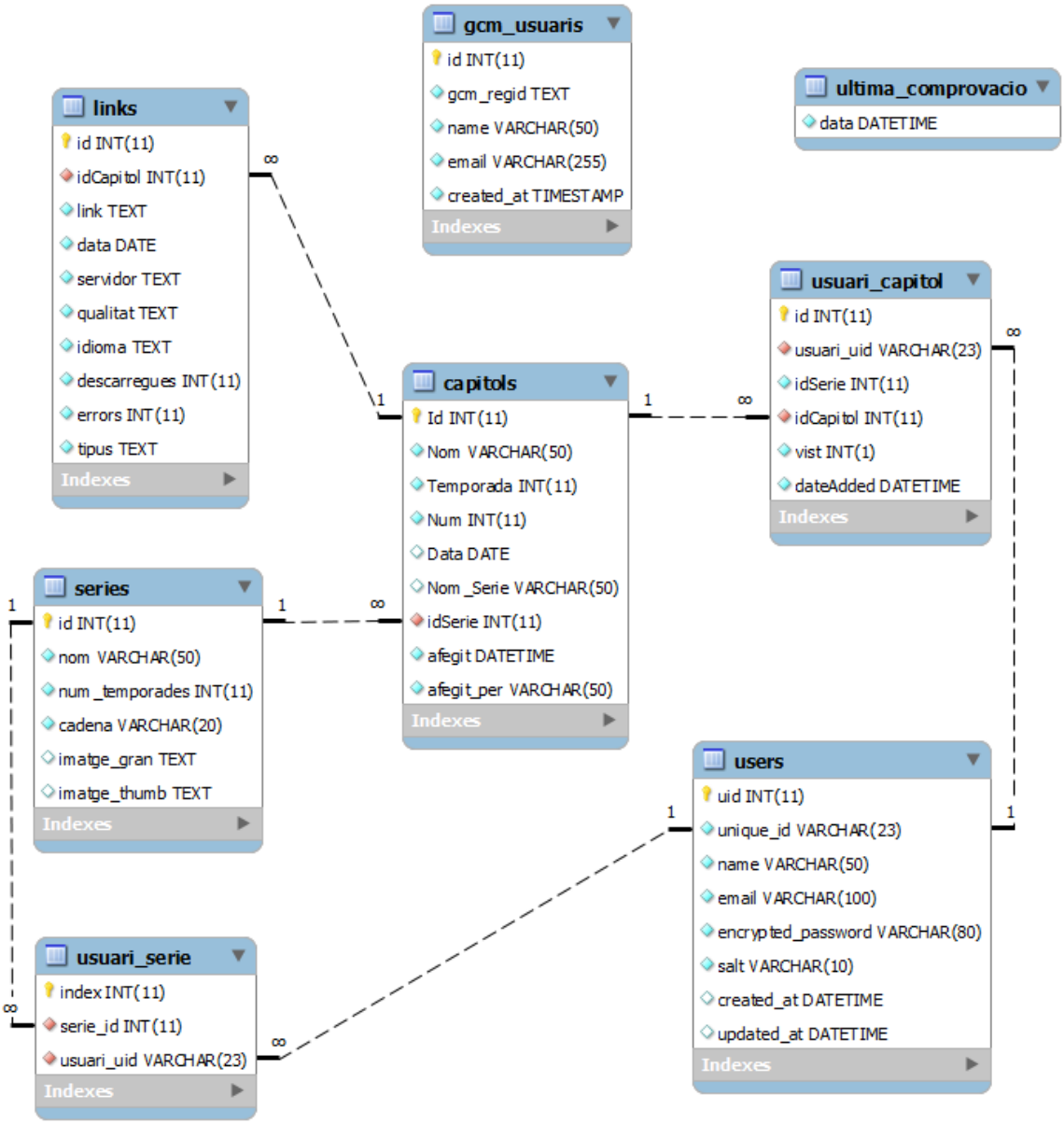
Il·lustració 10: Diagrama de seqüència. Notificar usuaris

Aquesta és una seqüencia iniciada per el servidor, la seva descripció textual és la següent:

- El servidor demana a la base de dades la data de la última vegada que es va comprovar si hi havia capítols nous.
- El servidor demana ala base de dades si hi ha capítols afegits posteriorment a la data retornada.
- Si n'hi ha, el servidor demana a la base de dades els usuaris que segueixen la serie.
- El servidor demana la id del Google Cloud Management dels usuaris que segueixen la serie.
- El servidor indica al servei GCM que vol notificar a certs usuaris.
- El servei GCM notifica als usuaris indicats.
- L'aplicació rep la notificació i la mostra per pantalla.
- Si l'usuari prem la notificació, s'obre el llistat de capítols pendents per tal de poder visualitzar els que han aparegut nous.

4.4 - Disseny de la persistència

4.4.1 - Diagrama de la base de dades



Il·lustració 11: Diagrama de persistència

4.4.2 - Model relacional de la base de dades

SERIES (id, nom, num_temporades, cadena, imatge_gran, imatge_thumb)

La taula SERIES és l'encarregada de guardar la informació sobre una serie, tal com el número que la identifica, el nom de la serie, el número de temporades de que disposa, la cadena on s'emet o emetia originalment i dos imatges, una gran per la pantalla d'informació de la serie i una petita per els llistats de series.

CAPITOLS (Id, Nom, Temporada, Num, Data, Nom_Serie, idSerie, afegit, afegit_per)
idSerie és clau forana de SERIES (id)

La taula CAPITOLS s'encarrega de guardar informació sobre els capitols, entre aquesta informació podem trobar el número que identifica el capítol, el seu nom, la temporada a la que pertany, el número de capítol dins de la temporada, la data en que es va emetre per primera vegada, el nom i l'dentificador de la serie i l'informació de quan i qui va afegir el capítol.

LINKS (id, idCapitol, link, data, servidor, qualitat, idioma, descarregues, errors, tipus)
idCapitol és clau forana de CAPITOLS (Id)

La taula LINKS guarda informació sobre els enllaços d'una serie, com per exemple, l'dientificador de l'enllaç, l'identificador del capítol al que pertany, la data en que es va afegir, el servidor de descàrregues en el que està allotjat l'enllaç, l'idioma i la qualitat del capítol, el número de vegades que s'ha descarregat, el número de vegades que s'ha marcat l'enllaç com a erroni i el tipus de descàrrega (directa, torrent, etc...).

USERS (uid, unique_id, name, email, encrypted_password, salt, created_at, updated_at)

La taula USERS desa informació sobre els usuaris que s'han registrat a l'aplicació. Aquesta informació és la típica en qualsevol registre, és a dir, nom, correu electrònic i contrasenya (que s'encripta amb una clau per seguretat). A més a més es genera un número únic d'usuari aleatori i es desa la data en que es va registrar i la data en que s'ha modificat l'usuari.

USUARI_CAPITOL (id, usuari_uid, idSerie, idCapitol, vist, dateAdded)

idCapitol és clau forana de CAPITOLS (Id)

usuari_uid és clau forana de USERS (unique_uid)

La taula USUARI_CAPITOL guarda les relacions entre usuaris i capitols, és a dir, tots aquells capítols de les series que segueix un usuari. Es guarden camps tals com els identificadors de relació, d'usuari, de la serie a la qual pertany el capítol i del capítol en sí, així com si l'usuari ha vist el capítol i la data en que s'ha afegit la relació.

USUARI_SERIE (index, serie_id, usuari_uid)

serie_id és clau forana de SERIES (id)

usuari_uid és clau forana de USERS (unique_uid)

De la mateixa manera que la taula anterior, la taula USUARI-SERIE desa les relacions entre usuaris i series, és a dir, totes aquelles series que segueix un usuari en concret. Guarda els identificadors de la relació, de l'usuari i de la serie.

GCM_USUARIS (id, gcm_regid, name, email, crated_at)

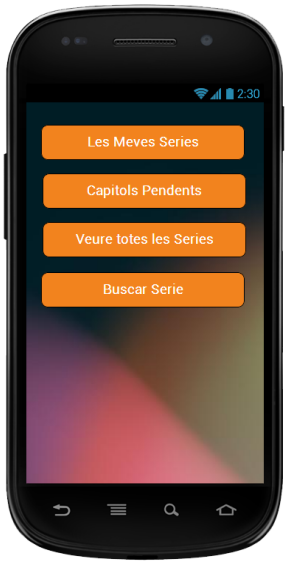
La taula GCM_USUARIS és l'encarregada de guardar informació sobre l'us del servei Google Cloud Messaging, en desa els camps nom i correu de l'usuari així com l'identificador de registre del servei GCM, que és el que es fa servir per enviar els missatges a través del servei.

ULTIMA_COMPROVACIÓ (data)

La taula ULTIMA_COMPROVACIÓ només conté un camp, que és la data en que s'ha comprovat per última vegada si hi havia capítols nous. Cada cop que es fa la comprovació s'actualitza aquesta camp.

4.5 - Prototipatge

4.5.1 - Pantalla principal

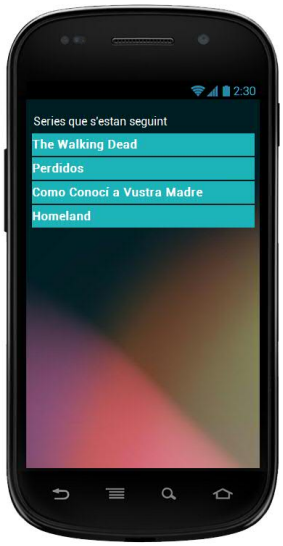


Il·lustració 12: Prototip. Pantalla princial

Aquesta és la pantalla inicial de l'aplicació, des d'aquesta es pot accedir a les funcionalitats principals a través dels quatre botons disposats per a tal finalitat:

- Les Meves Series
- Capítols pendents
- Veure totes les Series
- Buscar Serie
- Introduir nova serie

4.5.2 - Pantalla de Les meves Series

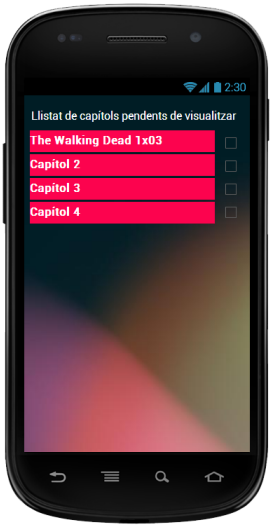


Il·lustració 13: Prototip. Pantalla les meves series

En aquesta pantalla es veurà el llistat de totes les series que s'han marcat com a preferides, un cop seleccionada alguna de les series es mostrarà la pantalla d'informació de les series.

De la mateixa manera que la pantalla de capítols pendents, al prémer el botó "Menú" apareixeran opcions com per exemple ordenar el llistat o filtrar-lo per diferents criteris.

4.5.3 - Pantalla de capítols pendents



Il·lustració 14: Prototip. Pantalla capítols pendents

En aquesta pantalla es mostren, de les series marcades com a preferides, el primer dels capítols que encara no s'han vist. Per exemple, si d'una serie ja s'ha vist el primer capítol i ja estan disponibles el segon i tercer, només apareixerà el segon, quan aquest es marqui com a vist, apareixerà el tercer.

Al prémer sobre algun dels capítols es mostrarà la pantalla d'informació d'aquest, permetent descarregar-lo o visualitzar-lo.

A la dreta de cada capítol hi ha una casella de verificació per a marcar-lo com a vist.

4.5.4 - Pantalla veure totes les series

Il·lustració 15: Prototip. Pantalla veure totes les series

En aquesta pantalla es mostra el llistat de totes les series disponibles, en un primer moment apareixen en ordre alfabètic i sense filtrar, però des del botó "Menú" del dispositiu es podran modificar aquestes opcions.

Un cop seleccionada alguna de les series s'obrirà la pantalla d'informació de la serie, que descriurem mes endavant.

4.5.5 - Pantalla Serie

Il·lustració 16: Prototip. Pantalla Serie

Aquesta pantalla ens mostra informació sobre la serie, amb camps tals com nom, Cadena i número de temporades.

A continuació es mostren les temporades de la serie, aquest llistat és desplegable, és a dir, un com premuda alguna de les temporades se'n mostraran, a la mateixa pantalla, els capítols d'aquesta, els quals es podran prémer o bé marcar com a vistos.

La última opció és el botó que permet afegir la serie al llistat de preferides o bé eliminar-la d'aquest llistat si ja és una serie preferida.

4.5.6 - Pantalla Capítol

Il·lustració 17: Prototip. Pantalla capítol

Aquesta pantalla ens mostra informació sobre un capítol, amb camps tals com nom, a quina temporada pertany, el número del capítol o la data d'emissió, a sota d'aquests caps apareix la opció "més info..." que permet ampliar la informació del capítol, com per exemple una breu descripció.

A continuació es disposa de diferents botons que permeten accedir a les descàrregues del capítol, als enllaços per visualitzar-la online o bé els enllaços a serveis de lloguer de series, tals com Waki, Youzee, etc..

4.5.7 - Pantalla Temporada

Il·lustració 18: Prototip. Pantalla temporada

Aquesta pantalla només s'ha inclòs per substituir el desplegable de temporades de la pantalla Serie, ja que amb l'eina de prototipatge que s'ha utilitzat no hi havia la opció de crear un menú desplegable.

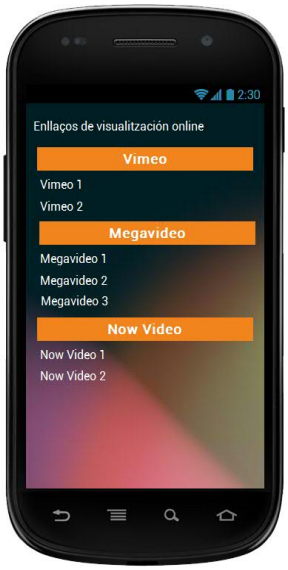
4.5.8 - Pantalla Enllaços de descàrrega

Il·lustració 19: Prototip. Pantalla enllaços

En aquesta pantalla apareixen els enllaços de descàrrega d'un capítol, ja sigui per descàrrega directa o bé arxius .torrent.

Es podran ordenar per tipus de descàrrega, pàgina d'on provenen els enllaços, qualitat, idioma, etc...

4.5.9 - Pantalla Enllaços de visualització online



Il·lustració 20: Prototip. Pantalla visualització online

En aquesta pantalla apareixen els enllaços de visualització online d'un capítol.

Es podran ordenar per pàgina d'on provenen els enllaços, qualitat del vídeo, idioma, etc...

5 – Implementació (Procés de creació d'una aplicació per Android)

A continuació es procedirà a descriure a grans trets el procés de creació d'una aplicació per a dispositius Android. La finalitat d'aquest apartat no busca ser una guia de desenvolupament per Android, de fet només es tocarà molt per sobre algunes de les parts bàsiques del procés de creació d'aplicacions per aquest S.O, sinó que el que es pretén mostrar la fase d'implementació del projecte .

D'aquesta manera s'aprofitarà per explicar diferents parts de l'aplicació mentre es resumeixen els conceptes bàsics apresos durant la realització del projecte.

5.1 - Descàrrega i instal·lació

Abans de començar a programar una aplicació per Android necessitem descarregar i instal·lar totes les eines necessàries. Aquestes són: El JDK de Java, l'IDE Eclipse, l'Android ADT, l'SDK d'Android i l'AVD Mànager.

JDK de Java

- http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Descarregar i instal·lar la última versió de "Java Platform (JDK)", durant la realització del projecte aquesta era la versió 7u10.

Anteriorment era necessari descarregar, instal·lar i configurar la resta de software manualment. Ara això ja no es necessari, ja que des de la pàgina d'Android Developers s'ha afegit un paquet amb tot el programari necessari.

Per a descarregar-lo hem d'anar a http://developer.android.com/sdk/index.html i prémer sobre download. I a http://developer.android.com/sdk/installing/bundle.html tenim totes les instruccions de com instal·lar aquest paquet.

5.2 - Les parts bàsiques d'un programa per Android

El codi d'una aplicació per Android consta de moltes parts diferents, i a mesura que augmenta la dimensió del nostre projecte o els dispositius amb els quals és compatible la nostra aplicació també augmenten les parts de les que disposa la codificació. Ara bé, hi ha algunes parts que són indispensables o almenys molt recomanables en qualsevol aplicació, per molt bàsica que sigui. Aquestes parts són:

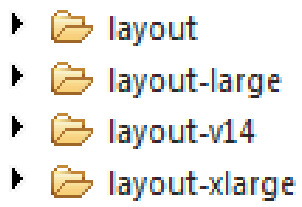
5.2.1 - Les classes

Com en tots els programes realitzats amb un llenguatge orientat a objectes, com en el nostre cas és Java, la part principal d'aquests són les classes, en aquestes és on es programen les funcionalitats de l'aplicació, des de classes que implementen les pantalles de l'aplicació, fins a classes que s'encarreguen de realitzar certes funcions com per exemple donar un format concret a un text, carregar una imatge, etc. Passant per classes que donen estructura, és a dir, s'utilitzen com a plantilla per a crear objectes.

La utilització de les classes ens aporten nombrosos beneficis, n'esmentem un quants:

- Evitar la redundància de codi, és a dir, si hem de realitzar una acció diverses vegades en un programa, podem delegar a una classe en concret la realització d'aquesta acció, aixi només haurem de cridar al mètode d'aquesta cada cop que necessitem dur a terme aquesta tasca.
- Reducció d'erros, si sabem que una classe funciona de manera correcta, estem segurs que cada cop que la instanciem i n'utilitzem algun dels seus mètodes, aquests funcionaran de manera estable.
- Reutilització, a partir d'una classe ben definida en podem reutilitzar o estendre el codi per tal d'afegir-hi funcionalitats, per exemple, si tenim una classe de tipus persona, la podem estendre per crear una classe tipus treballador afegint només els atributs necessaris (sou, lloc de treball, etc..)

5.2.2 - El Layout



Il·lustració 21: Exemple de carpetes "layout" per a diferent mides de pantalla

Mentre que les classes que implementen una pantalla descriuen el funcionament d'aquesta, són els layouts els encarregats de definir l'aspecte que tindrà la pantalla. Aquests layouts estan escrits en llenguatge xml i en aquests es descriuen aspectes com el color, la forma, el tipus i mida de lletra, o de quins elements té la pantalla, com estan distribuïts o com es comporten.

Un dels avantatges de poder definir el disseny de la pantalla a través dels layouts és que podem separar la part del codi que defineix el comportament de la part que descriu l'aspecte, d'aquesta manera podem modificar el disseny de la pantalla sense haver de preocupar-nos de modificar el comportament.

El fet de disposar dels layouts també ens permet crear diferents layouts per a cada tipus de pantalla, depenent de la mida, la resolució, la densitat de píxels, etc...

5.2.3 - Els drawable

En aquest apartat és on col·loquem tots els elements gràfics que voldrem utilitzar. Com per exemple l'icona de l'aplicació, les icones d'alguns botons del menú, els fons, etc... És aquí on també podrem definir (a través d'xml) com es mostren certs elements del sistema, com per exemple de quin color volem que siguin els botons quan els premem, estan seleccionats, en "repòs", etc...

A més a més podem definir diferents "drawables" depenent del tipus de pantalla o la versió del sistema operatiu, d'aquesta manera en un pantalla de mida més gran o major densitat de píxels podem fer que les icones que es mostrin també tinguin una mida superior.

Il·lustració 22: Exemple de carpetes "drawable" per a diferents mides de pantalla

5.2.4 - Els String

Tot i ser una part completament prescindible en una aplicació Android és molt recomanable utilitzar aquest recurs.

L'arxiu Strings no és res més que un llistat on es poden afegit tots els textos, etiquetes, missatges, etc, que es mostraran per pantalla, d'aquesta manera podrem recollir en un sol document tots aquests elements i recuperar-los en el nostre codi utilitzant la funció `getString(R.string.el_nostre_string)` . Aquest fet és molt útil per varies raons:

- Si en la nostra aplicació apareix moltes vegades el mateix text, per exemple un botó que tingui el nom "Acceptar", i en un moment donat decidim modificar-lo, per exemple perquè passi a tenir el nom "OK", no cal que anem element per element modificant-ne el nom, sinó que modificant-lo una sola vegada en l'arxiu strings en tenim prou, per tant estalviem temps, esforços i possibles errors en modificar algun dels textos.
- Per altra banda, és un recurs molt important si volem fer l'aplicació compatible amb diferents idiomes. Es pot crear diferents carpetes "values", una per a cada idioma i guardar-hi els arxius "string" per tal que, depenent de l'idioma del dispositiu, faci servir l'string d'una carpeta o una altra.
- A demés també ho podem fer servir per crear Arrays d'Strings, Strings quantitatius (plurals), donar'ls-hi un format o estil concrets, etc.

5.2.5 - El Manifest

Aquest és l'arxiu on es guardarà la informació general de la nostra aplicació, sense quest arxiu l'aplicació no podria mai funcionar, per tant, és una part fonamental d'un projecte Android. Entre d'altres coses, el manifest inclou elements imprescindibles com:

- El nom del paquet de l'aplicació.
- Descriu els seus components (Activities, Services, receivers...).
- Defineix de quins permisos gaudeix l'aplicació per interactuar amb elements del sistema.
- Defineix el nivell mínim d'API d'Android necessari.

Tot i que al crear per primera vegada el projecte ja es genera el manifest, hem d'anar molt en compte d'anar afegint totes les Activities que anem creant o els permisos que anem necessitant per tal que tot funcioni correctament.

5.3 - *La pantalla inicial*

En aquest subapartat veurem, seguint la creació de la pantalla principal, aspectes com l'estructura del layout d'una pantalla, elements típics com els botons, la creació d'un Activity, la crida a una Intent per obrir una nova pantalla, etc..

5.3.1 - El Layout XML

Com ja hem indicat en l'apartat anterior, el layout és el que defineix l'estructura visual de la interfície d'usuari, ja sigui per una pantalla, un widget, un element, etc..

Hi ha dos maneres de declarar un layout, a través de l'xml o dinàmicament en temps d'execució. En aquest apartat ens centrarem en la primera, doncs és la manera més senzilla i potser la més típica a l'hora de donar forma a les interfícies.

Per a crear l'estructura de la nostra pantalla inicial, simplement hem d'indicar quin tipus de layout volem fer servir i anar afegint elements en llenguatge xml d'Android en el nostre layout. Començarem indicant-li a l'arxiu que es tracta d'un layout lineal de la següent manera:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center_horizontal"
    android:orientation="horizontal"
    android:weightSum="100" >
</LinearLayout>
```

D'aquest fragment de codi en podem desgranar la següent informació:

- **layout_width:** Ens indica quina amplada tindrà l'objecte. **Fill_parent** indica que ocupara tot l'ample disponible de l'element que el conté (en aquest cas la pantalla). Altres possibles valors podrien ser **wrap_content,** és a dir que l'objecte seria tant gran com la mida dels seus elements interiors o bé introduir un valor fix.
- **layout_heigh:** El mateix que en el cas anterior però per indicar l'alçada.
- **gravity**: La disposició dels elements (centrat, a la dreta, esquerra, etc...)
- **weightSum:** Ens serveix per distribuir elements dins d'aquest layout, és a dir, podem definir un pes a cada element per tal que entre tots sumin 100. Veurem més endavant com fer-ho amb el paràmetre **layout_weight**.

Un cop tenim definit el layout ja podem començar a introduir-hi els elements, sempre immediatament abans que l'etiqueta `</LinearLayout>`.

Començarem col·locant-hi un botó de la següent manera:

```
<Button android:id="@+id/principal_LesMevesSeries"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dip"
```

```
    android:text="@string/les_meves_series" />
```

A part de les expressions que em vist al crear el layout, en aquest fragment podem veure:

- **id:** És l'identificador de l'element, farem servir aquest nom per referir-nos a ell en el codi de l'aplicació.
- **layout_marginTop**: Indica el marge superior que volem que tingui el nostre element.
- **text:** És el text que volem que tingui el nostre element, cal indicar que `@string/les_meves_series` significa que volem que cerqui el valor de l'string anomenat "les_meves_series" al recurs d'strings que hem esmentat en l'apartat anterior. Prèviament l'haurem incorporat a l'arxiu strings.xml. (`<string name="les_meves_series">Les Meves Series</string>`).

Ara només ens queda anar afegint els botons que necessitem de la mateixa manera.

5.3.2 - L'Activity

L'**Activity** és un dels components més típics en una aplicació Android, és l'encarregat de proporcionar una pantalla amb la qual l'usuari pot interactuar, com per exemple per visualitzar un mapa, fer una trucada o enviar un correu.

Per començar a construir el nostre **Activity** principal ho farem creant una nova classe.

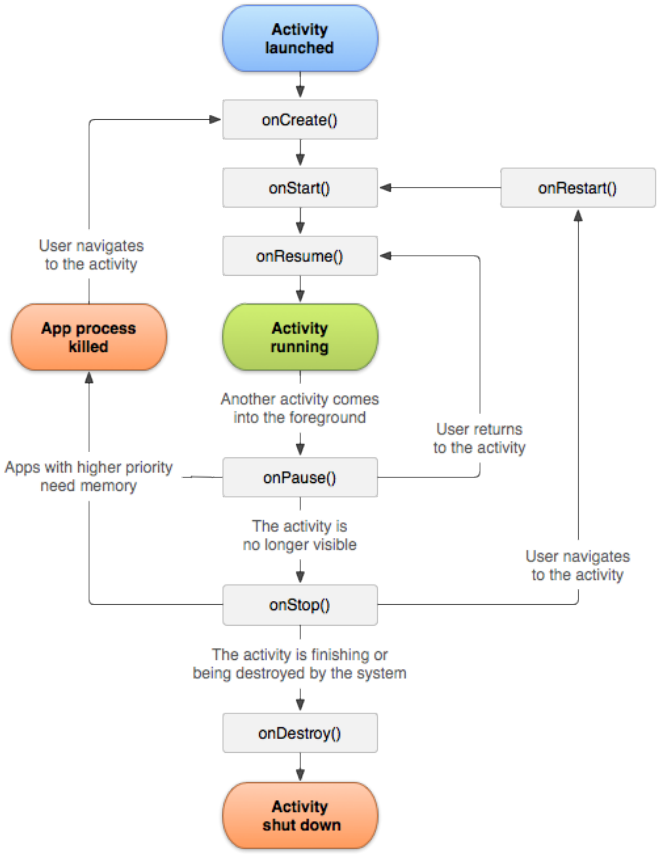
```
public class MenuPrincipal extends Activity{
}
```

D'aquesta manera estem indicant que la nostra classe estendrà les propietats d'un **Activity**.

A continuació es declararan totes les variables que necessitem així com els components que es mostraran per pantalla, que en el cas de la pantalla principal, seran els botons.

```
Button btnViewSeries, btnLesMevesSeries,  btnVeureBBDD, btnCapitols, btnAfegir;
```

Hi ha diversos mètodes que les subclasses d'Activity poden implementar, com poden ser **onCreate(), onPause(), onStart(), on ReStart(), onDestroy()**, etc. Que defineixen el cicle de vida d'un **Activity**. Gràficament el cicle de vida d'un **Activity** té aquest aspecte:



Il·lustració 23: Cicle de vida d'un Activity

En el nostre cas només ens centrarem en el mètode **onCreate()**, que és el que es crida quan l'**Activity** s'inicialitza.

```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
}
```

Si l'**Activity** està sent reinicialitzada després d'haver-la tancat el paràmetre savedInstanceState conté les dades de l'estat en que es trobava l'**Activity**.

Dins aquest mètode és on realitzarem la majora de funcions de la nostra pantalla principal, tals com inicialitzar variables, indicar el comportament dels botons o la més important, definir el layout de la interfície d'usuari que hem definit anteriorment, cridant el mètode `setContentView(R.layout.main_screen);` . Per a inicialitzar les variables, o en el nostre cas els botons que es mostraran per pantalla ho farem de la següent manera:

```
btnViewSeries = (Button) findViewById(R.id.btnViewProducts);
btnLesMevesSeries = (Button) findViewById(R.id.principal_LesMevesSeries);
btnVeureBBDD = (Button) findViewById(R.id.principal_VeureBBDD);
btnCapitols = (Button) findViewById(R.id.principal_veureCapitolsPendents);
btnAfegir =(Button) findViewById(R.id.principal_afegir);
```

El mètode **findViewById** indica que volem cercar el View* per la seva identificació, la qual havíem indicat mentre creàvem el layout (android:id="@+id/...) .

*View: Representa els components de la interfície d'usuari. Ocupa una àrea rectangular a la pantalla i és el responsable de dibuixar i gestionar els esdeveniments del component

Un cop tenim els components inicialitzats, només ens falta descriure com es comportaran quan l'usuari els premi, per això farem servir un mètode típic dels View anomenat **setOnClickListener().** És l'encarregat de que cada cop que es premi sobre l'element en qüestió es passi a executar el codi que es trobi en aquest mètode.

```
btnViewSeries.setOnClickListener(new View.OnClickListener() {
     public void onClick(View view) {
         Intent i = new Intent(getApplicationContext(), ExplorarSeries.class);
         startActivity(i);
     }
});
```

Ara vegem en detall que és el que estem indicant que faci el botó quan el premem. El que li estem indicant es que crei un nou **Intent** anomenat i lligat a la classe *ExplorarSeries* i que després l'inicii.

Un **Intent** no és res més que una manera d'invocar components, ho podríem interpretar com la intenció de realitzar alguna acció, depenent de quina acció sigui i a qui va dirigida, els SO o l'aplicació reaccionaran d'una manera en concret. En el nostre cas estem indicant que la nostra "intenció" és inicialitzar un nou Activity i que aquest és la classe ExplorarSeries, per tant el que farà l'aplicació serà obrir una nova pantalla d'Explorar Series.

Hauríem de procedir de la mateixa manera per a cada un dels botons que havíem declarat, si és que volem que facin quelcom quan siguin premuts.

5.3.3 - Editar el Manifest

Com ja havíem indicat anteriorment qualsevol **Activity** que creem ha d'estar degudament declarada en el nostre AndroidManifest.xml, per tal que l'aplicació sigui capaç de interactuar amb l'**Activity** en qüestió. A demés en aquest cas es tracta de la pantalla principal del nostre programa per tant haurem d'afegir alguna propietat extra en el manifest.

```
<activity
   android:name=".MenuPrincipal"
   android:label="@string/title_activity_starting_point" >
   <intent-filter>
       <action android:name="android.intent.action.MAIN" />
       <category android:name="android.intent.category.LAUNCHER" />
   </intent-filter>
</activity>
```

Estem indicant el següent:

- Android:name: indica quina és la classe a la que ens estem referint. En aquest cas és MenuPrincipal. Si la classe no es trobés en el paquet arrel , hauríem

d'indicar també el nom del paquet (nom_del_paquet.MenuPrincipal).

- Android:label: indica com volem que s'anomeni l'**Activity**, en aquest cas obtenim el nom de l'arxiu strings.xml (vegeu apartat 5.2.4).
- Action: indica l'acció que es realitzarà, action.MAIN indica que serà l'**Activity** principal.
- Category: conté informació addicional del tipus de component que tractarà l'intent. En aquest cas estem indicant que l'**Activity** serà la tasca inicial de l'aplicació.

5.4 - La pantalla Explorar Series

En aquesta pantalla ja es començaran a veure elements més avançats de la programació per Android. Veurem altres elements típics d'una pantalla, com poden ser els ListView, TextView, ImageView, veurem com es treballa amb els adapter per donar-li forma als llistats, s'aprendrà a obtenir informació d'una plana web, crear una tasca en segon pla, crear menús i submenús, diàlegs d'alerta, etc...

Bàsicament el que mostra aquesta pantalla és un llistat vertical de totes les series disponibles a la base de dades. Aquest llistat, per a cada serie, dóna informació sobre el nom de la serie, la cadena on s'emet, i una petita imatge de la serie.

5.4.1 - El Layout XML

Ja hem vist en l'apartat anterior com crear el layout que donarà forma a la nostra interfície d'usuari, ara en centrarem en altres elements que es poden incorporar en un pantalla típica d'una aplicació.

a) ListView

Un **ListView**, com el seu nom pot donar a entendre, és una manera de mostrar contingut en un llistat vertical, cada un dels seus elements es pot seleccionar per separat. La manera com es mostra el llistat (només nom, nom i imatge, nom i quadre de validació) o els elements que conté la llista ve donat per l'**adapter** associat a aquest **ListView** (més endavant es veurà com treballar amb adapters).

Per a crear el **ListView** ho farém de la mateixa manera que hem creat el botó en l'apartat anterior, en aquest cas el codi dins el nostre **layout** serà el següent.

```
<ListView android:id="@+id/all_prod_list"
   android:layout_width="fill_parent"
   android:divider="@null"
   android:dividerHeight="0dp"
   android:paddingLeft="2dip"
   android:paddingRight="2dip"
   android:layout_height="wrap_content"/>
```

Ja hem vist que són els camps id, layout_width i layout_heigt, en aquest exemple tanim la oportunitat de descriure alguns camps més:

- divider: defineix l'aspecte que tindrà la divisió entre element i element del nostre **ListView.** En aquest cas, null vol dir que no hi haurà cap divisor entre elements.
- dividerHeight: el gruix de la línia que separa els elements.
- paddingLeft: indica el marge exterior esquerre que volem que tingui el nostre **ListView** de la mateixa manera que paddinRight indica el marge dret.

b) Els elements del llistat

Un cop ja tenim definit el nostre **ListView** ens falta indicar com volem que es vegin els seus elements. Per defecte, un **ListView** mostra els elements com una simple línia de text, però en la nostra aplicació ens interessa que mostri informació extra com per exemple la cadena on s'emet la serie i una petita imatge de la serie. Per a fer-ho crearem un nou arxiu xml.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/list_selector"
    android:orientation="horizontal"
    android:padding="5dip" >

      <!--  Imatge de la serie a l'esquerra -->
      <LinearLayout android:id="@+id/thumbnail"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:padding="3dip"
        android:layout_alignParentLeft="true"
        android:background="@drawable/image_bg"
         android:layout_marginRight="5dip">

             <ImageView
              android:id="@+id/item_image"
               android:layout_width="50dip"
               android:layout_height="50dip"
               />

      </LinearLayout>

      <!-- Nom de la serie-->
    <TextView
        android:id="@+id/item_title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignTop="@+id/thumbnail"
        android:layout_toRightOf="@+id/thumbnail"

        android:textColor="#040404"
        android:typeface="sans"
        android:textSize="15dip"
        android:textStyle="bold"/>

      <!-- Cadena on s'emet la serie -->
    <TextView
        android:id="@+id/item_cadena"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/item_title"
        android:textColor="#343434"
        android:textSize="10dip"
        android:layout_marginTop="1dip"
        android:layout_toRightOf="@+id/thumbnail"
         />
```

```
    <!-- Fletxa a la dreta -->
    <ImageView android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@drawable/arrow"
      android:layout_alignParentRight="true"
      android:layout_centerVertical="true"/>

</RelativeLayout>
```

Com es pot veure en aquest cas el layout que hem utilitzat és **RelativeLayout**, és a dir que les posicions dels seus descendents (child) es descriuen en relació als altres o al seu pare (parent). Per tant això ens permet utilitzar propietats com **layout_toRightOf** (indica que l'element va a la dreta d'un altre element), **layout_alignParentRight** (l'element va a la part dreta de l'element que el conté), **layout_alignParentLeft** (l'element va a la part esquerra de l'element que el conté), etc..

En aquest fragment de codi, podem veure altres elements que encara no havíem utilitzat. Com per exemple Videotext que implementa un text simple o **Imaginaire** que s'utilitza per mostrar imatges.

Una de les propietats que mereixen més atenció és la del Relative**Layout** anomenada **background,** que és l'encarregada de definir el fons d'un element. En aquesta propietat es pot observar que s'escull un valor que es troba en la carpeta drawable, com havíem comentat en apartats anteriors, és en aquesta carpeta on es pot definir com es mostren alguns dels elements del sistema, doncs si indiquem en l'xml que `android:background="@drawable/list_selector"`, estem dient que el fons d'aquest layout es mostri tal com definim en l'arxiu "image_bg" que es troba en la carpeta "drawable". Vegem més en detall el contingut d'aquest arxiu:

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
<item
    android:state_selected="false"
    android:state_pressed="false"
    android:drawable="@drawable/gradient_bg" />
<item android:state_pressed="true"
    android:drawable="@drawable/gradient_bg_hover" />
<item android:state_selected="true"
    android:state_pressed="false"
    android:drawable="@drawable/gradient_bg_hover" />
</selector>
```

En aquest fragment veiem tres propietats ítem:

- La primera indica com volem que es mostri l'element quan no està ni seleccionat ni premut, i que volem que en aquest cas es mostri com descriu l'arxiu "gradient_bg"
- La segona indica com volem que es mostri quan l'element està premut, utilitzant l'arxiu "gradient_bg_hover"
- La tercera indica com volem que es mostri quan està seleccionat i premut, utilitzant el que descriu l'arxiu "gradient_bg_hover"

Vegem doncs el codi que conté un d'aquests dos arxius:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
   <gradient
      android:startColor="#f1f1f2"
      android:centerColor="#e7e7e8"
      android:endColor="#cfcfcf"
      android:angle="270" />
</shape>
```

Simplement estem indicant que es tracta d'una forma rectangular (android:shape) i li estem aplicant un color, en aquest cas un gradient que definim amb els paràmetres **startColor, centerColor, endColor i angle**.

Ja tenim definit l'aspecte dels nostres elements del **ListView**, aquests es mostraran de la següent manera:

Il·lustració 24: Exemple d'un item del llistat

5.4.2 – L'Activity

a) Creant menús i submenús

Ja hem vist com crear una pantalla bàsica en l'apartat anterior, aquesta només mostrava una serie de botons que portaven a altres pantalles. Ara ens centrarem en veure com es pot crear un menú d'opcions a la part superior de la pantalla, on a part dels botons també aparegui el títol de la pantalla i l'icona de la nostra aplicació, així com un menú que s'obri al prémer el botó "Menú" del dispositiu.

Per a crear aquests menús ens ajudarem de l'eina "ActionBarSherlock" que ofereix un munt d'eines relacionades amb menús i ens facilita la feina enormement a l'hora de crear-los.

Per utilitzar aquesta eina, la nostra classe ha d'estendre la classe **SherlockActivity** en comptes de l'**Activity** normal com havíem fet el la pantalla principal. De la mateixa manera que s'havia creat el mètode **OnCreate()**, per a la creació dels menús hem de fer servir el mètode **OnCreateOptionsMenu()**:

```
public boolean onCreateOptionsMenu(Menu menu) {
	super.onCreateOptionsMenu(menu);
	// Es creen els menús
	// Afegim un títol i un subtítol
	getSupportActionBar().setTitle("Explorar Series");
	getSupportActionBar().setSubtitle("Des d'aquí podrà consultar qualsevol
serie");
	// Afegim el menú d'ordenació
	SubMenu subMenu1 = menu.addSubMenu(0, 2, 1, "Ordenar");
	// Amb els seus submenús
	subMenu1.add(0, 3, 0, "A-Z");
	subMenu1.add(0, 4, 0, "Z-A");
	subMenu1.add(0, 5, 0, "Cadena");
	subMenu1.add(0, 6, 0, "Data últim episodi");
	MenuItem subMenu1Item = subMenu1.getItem();
	// El menú només es mostrara a la actionBar si hi ha espai i només
	// mostrarà text
	subMenu1Item.setShowAsAction(MenuItem.SHOW_AS_ACTION_IF_ROOM|
			MenuItem.SHOW_AS_ACTION_NEVER);
	// La barra de cerca serà colapsable
	menu.add(0, 1, 1, "Cercar")
		.setIcon(R.drawable.ic_action_search)
		.setActionView(R.layout.colapsable_auto_edit)
		.setShowAsAction(MenuItem.SHOW_AS_ACTION_ALWAYS
				| MenuItem.SHOW_AS_ACTION_COLLAPSE_ACTION_VIEW);
		search = (AutoCompleteTextView) menu.getItem(1).getActionView();
		search.addTextChangedListener(textWatcher);
		// Afegim l'opció de desconnectar-se
		SubMenu subMenu4 = menu.addSubMenu(0, 7, 0, "Desconectar-se");
		MenuItem subMenu4Item = subMenu4.getItem();
		// Li assignem una icona
		subMenu4Item.setIcon(R.drawable.ic_compose);
		return true;
	}
```

Comentem alguna de les funcions:

- **getSupportActionBar.setTitle** i **setSubtitle** ens permet crear i assignar un text al títol i subtítol de la pantalla.
- Per afegir un submenú al nostre menú ho podem fer amb la funció **menu.add (int groupId, int itemId, int order, CharSequence title)**. Fixem-nos que a l'hora de crear els submenús passem una serie de paràmetres, en el nostre cas només farem ús de **itemId** que determina un número identificador per al submenú i de **title**, que indica el text que tindrà.
- **.setIcon** permet establir una icona per al submenú.
- **.setShowAsAction** determina la manera com volem que es mostri el submenú. Per exemple **SHOW_AS_ACTION_IF_ROOM** indica que mostri l'opció al menú superior si hi ha espai i **SHOW_AS_ACTION_WITH_TEXT** mostraria a més de l'icona, el text de l'opció.
- El menú cercar és un cas especial, ja que és colapsable, és a dir, que al principi només és una icona però al prémer s'expandeix en una barra de cerca. Per això fem servir la funció `.setActionView(R.layout.colapsable_auto_edit)`. I afegim `search.addTextChangedListener(textWatcher)` per tal de disposar d'un mètode per controlar en tot moment quan el text de la barra canvia.

Ja tenim el menú i els seus submenús creats, ara hem de disposar d'alguna manera de realitzar alguna acció quan les opcions del menú es premen, per això crearem el mètode:

```
public boolean onOptionsItemSelected(MenuItem item) {
	switch(item.getItemId()){
		case 1: //la barra de cerca
			return true;
		case 3: //Ordenar ascendent
			return true;
		case 4: //Ordenar descendent
			return true;
	}
	return false;
}
```

Com hem identificat cada submenú i opció amb un número ara podem distingir gracies a **item.getItemId()** quina ha sigut l'opció premuda. A dins de cada **case** es realitzaran les accions desitjades.

b) Tasques en segon plà

En alguns casos no desitjarem que alguna tasca es realitzi en primer plà sinó que voldrem que la tasca es vagi realitzant de manera asíncrona en segon plà mentre la part principal del programa està realitzant alguna altra acció. A demés si volem realitzar una operació de xarxa estem obligats a fer-ho en segon plà.

Per a fer-ho crearem una nova classe (dins de la classe en la que estàvem treballant) que extengui **AsyncTask**.

```
class LoadAllProducts extends AsyncTask<String, String, String>
```

Una tasca asíncrona té tres métodes principals, **onPreExecute()**, per realitzar una acció abans de fer la tasca, **doInBackground()** on es fa la tasca en segon plà i **onPostExecute()**, per realitzar una acció després de fer la tasca. En el nostre cas abans de realitzar la tasca en segon plà, crearem i mostrarem un diàleg que indiqui a l'usuari que s'està realitzant l'acció i després de realitzar la tasca amagarem aquest diàleg.

```
class LoadAllProducts extends AsyncTask<String, String, String> {

     protected void onPreExecute() {
          super.onPreExecute();
          //Es crea el Dialog
          pDialog = new ProgressDialog(ExplorarSeries.this);
          //Se li assigna un missatge
          pDialog.setMessage(getString(R.string.wait_for_series));
          pDialog.setIndeterminate(false);
          //Es fa Cancelable, per si no acaba de carregar poder cancel·lar
          pDialog.setCancelable(true);
          pDialog.setOnCancelListener(new DialogInterface.OnCancelListener() {
                    public void onCancel(DialogInterface pDialog) {
                          finish(); //Si es cancel·la tanca l'Activitat i torna
                    }                   a la pantalla anterior
             });
             pDialog.show(); //Mostra el dialeg
        }
        protected String doInBackground(String... args) {
             //ACCIÓ A REALITZAR
            return null;
        }

        protected void onPostExecute(String file_url) {
             pDialog.dismiss(); //Amaga el diàleg
        }
    }
```

Per a crear el diàleg d'alerta hem fet servir les funcions:

- **new ProgressDialog:** per a crear el diàleg.
- **.setMessage():** per establir el missatge que volem que aparegui.
- **.setCancelable():** perquè el diàleg es pugui cancel·lar.
- **show() i dismiss():** per mostrar-lo i amargar-lo.

El mètode **setOnCancellListener()** fa que quan es cancel·li el diàleg tanqui la pantalla sencera i torni a la pantalla anterior.

c) Realizar operacions de xarxa

Per tal d'obtenir el llistat de totes les series disposem d'un script PhP que cerca a la base de dades totes les series disponibles i en retorna una cadena de text en format **JSON** que podrem tractar a posteriori.

Com ja s'ha esmentat per a poder realitzar una operació de xarxa ho hem de fer en la tasca asíncrona (segon plà) que acabem de crear.

Per a realitzar l'operacio de xarxa treballarem amb objectes **JSON** i ens ajudarem d'una classe anomentada JSONParser obtinguda d'internet de la qual en comentarem el funcionament de manera general.

```
if(method == "POST"){
   // request method is POST
   // defaultHttpClient
   DefaultHttpClient httpClient = new DefaultHttpClient();
   HttpPost httpPost = new HttpPost(url);
   httpPost.setEntity(new UrlEncodedFormEntity(params));
   HttpResponse httpResponse = httpClient.execute(httpPost);
   HttpEntity httpEntity = httpResponse.getEntity();
   is = httpEntity.getContent();

}else if(method == "GET"){
   // request method is GET
   DefaultHttpClient httpClient = new DefaultHttpClient();
   String paramString = URLEncodedUtils.format(params, "utf-8");
   url += "?" + paramString;
   HttpGet httpGet = new HttpGet(url);

   HttpResponse httpResponse = httpClient.execute(httpGet);
   HttpEntity httpEntity = httpResponse.getEntity();
   is = httpEntity.getContent();
}
```

El codi anteriror és només un fragment de la classe, basicament la seva funcio és realitzar una petició a una plana web de la seguüent manera:

- Crea un nou client http.
- Depenent de si volem obtenir o enviar informació crea un nou **httpPost** o **httpGet** amb la url a la que volem fer la petició.
- En cas de que vulguem enviar informació estableix l'httpEntity* del **httpPost**.
- S'executa el client
- S'obté l'httpEntity de la resposta.

*__httpEntity__: Entitat que pot ser enviada o rebuda en una operació HTTP. Per exemple en una petició POST poden ser el parametres de la petició (www.exemple.com/get.php**?nom=Marc**) i per a un GET, la resposta obtinguda.

Per tant quan necessitem fer una petició només necessitem fer la crida a aquest mètode indicant-li la url a la que volem fer la petició, si és "GET" o "POST" i els paràmetres, guardant el resultat en un objecte **JSON.**

```
JSONObject json = jParser.makeHttpRequest(url_all_products, "GET", params);
```

Vegem quina estructura té una resposta en format **JSON:**

{"series":[
{"id":"1","nom":"The Walking Dead","cadena":"HBO","imatge_thumb":""},
{"id":"2","nom":"Braking Bad","cadena":"AMC","imatge_thumb":""},
{"id":"3","nom":"Perdidos","cadena":"HBO","imatge_thumb":""}],
"success":1}

Veiem que la informació està dividida en diferents camps identificats amb un nom i que contenen un valor. En aquest exemple podem veure el camp "series" i el camp "success". Dins el camp "series" la informació està estructurada de manera similar, és a dir, amb diferents camps identificats per un nom i el seu valor.

Treballant amb JSONs amb Android podem recuperar la informació de manera molt senzilla, simplement hem d'obtenir el valor del camp amb funcions tals com `int success = json.getInt("success")`, que indica que volem obtenir el valor (un enter) guardat en el camp amb nom "success", `products = json.getJSONArray("series")`, que indica que volem obtenir la cadena que es guarda en el camp amb nom "series".

Un cop obtinguda aquesta cadena podem recuperar els valors de forma similar, és a dir guardant cada element de la cadena en un **JSONObject** i obtenint els valors dels camps amb `String id = c.getString("id")`.

Per tant el codi (que anira en el mètode **doInBackground()**) quedaria d'aquesta manera:

```
JSONObject json = jParser.makeHttpRequest(url_all_products, "GET", params);
int success = json.getInt("success"); //Comprovant que l'obtenció sigui
                                        correcta
     if (success == 1) {
        products = json.getJSONArray("series");//Obtenint l'array de productes
        //Bucle per TOTS els productes
        for (int i = 0; i < products.length(); i++) {
            JSONObject c = products.getJSONObject(i);
           // Guardant cada json item en una variable
            String id = c.getString("id");
            String name = c.getString("nom");
            String cadena=c.getString("cadena");
            String imatge_thumb=c.getString("imatge_thumb");
            //Creant un nou objecte de Serie amb les dades obtingudes
            Serie serie = new Serie(id, name, cadena);
            serie.setImatge_thumb(imatge_thumb);
            //Afegint la Serie a l'ArrayList
            seriesList.add(serie);
        }
    }
```

És a dir, que itera totes les series de la resposta, va desant els paràmetres i crea un nou objecte de Serie (classe que no veurem en detall que representa una serie) que es guarda en una cadena de series.

5.4.3 – L'Adapter

Ja tenim la pantalla del llistat de series creada, amb els menús i submenús, i hem obtingut totes les series en una tasca en segon plà, ara només queda aprendre com mostrar questa informació en un llistat on els seus elements tinguin l'aparença que hem dissenyat amb l'xml.

Aquesta tasca la realitza el que s'anomena un **Adapter,** que no és més que un objecte encarregat de donar accés al les dades dels elements i és el responsable de definir la manera com es mostren aquests elements.

Per a crear l'**Adapter** començarem creant una nova classe que extengui, en aquest cas, **ArrayAdapter**. Com hem comentat l'**Adapter** s'encarrega tant de definir quin aspecte tindran els nostres elements com les dades que contindran, per tant hem de definir-ho en el que és el mètode més important d'un **Adapter**, el **getView()**.

En la nostra aplicació, el llistat de series es mostren agrupades alfabèticament, és a dir, totes les que el seu nom comença per A en un bloc, les que comença per B en un altre, etc.. Ara bé, per a simplificar l'exemple no crearem els elements separadors. Aixi doncs, el mètode **getView()** i el constructor de l'**Adapter** tindran un aspecte tal com:

```
public SerieAdapter(Context context,ArrayList<Serie> items) {
	//Inicialitza els valors de l'Adapter
	super(context,0, items);
	this.context = context;
	this.items = items;
	vi =
(LayoutInflater)context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
	imageLoader=new ImageLoader(context);
}
public View getView(int position, View convertView, ViewGroup parent) {
	View v = convertView;
	//S'obté l'item
	final Serie i = items.get(position);
	if (i != null) {
		//S'atribueix cada element al seu View, definit en l'xml
		v = vi.inflate(R.layout.list_row, null);
		final TextView title = (TextView)v.findViewById(R.id.item_title);
		final TextView id = (TextView) v.findViewById(R.id.item_id);
		final TextView cadena = (TextView)
v.findViewById(R.id.item_cadena);
		final ImageView thumb_image = (ImageView)
v.findViewById(R.id.item_image);
		//Es mostren els valors de Serie als camps corresponents
		if (title != null) title.setText(i.getNom());
		if(id!=null) id.setText(i.getId());
		cadena.setText(i.getCadena());
		imageLoader.DisplayImage(i.getImatge_thumb(),thumb_image);
	}
	return v;
}
```

Com es pot observar l'estructura del **getView()** és bastant senzilla, primer de tot obtenim la Serie de la posició que volem mostrar, per tant l'atribut "items" (que obtenim en el constructor) ha de ser un **ArrayList** de Series. En segon lloc, si la Serie obtinguda no és buida, assignarem cada **View** que haviem definit en l'xml a una variable i per últim

assignem valors a aquestes variables amb la informació de la Serie que hem obtingut en el primer pas.

Ja tenim definit l'**Adapter**, ara només queda fer-lo servir en la nostra classe d' "Explorar Series" per tal de que es mostri el llistat correctament. En l'apartat anterior, en el mètode **doInBackround()** de la tasca asíncrona, havíem omplert un **ArrayList** de Series anomenat seriesList (`seriesList.add(serie);`), que serà el llistat que férem servir juntament amb l'**Adapter** per tal de crear el **ListView**.

Aquesta assignació de les dades del **ListView** la farem en el mètode **onPostExecute()**, és a dir, quan hagi acabat la tasca en segon pla. És tan simple com:

```
//Es genera la vista amb l'adapter
SerieAdapter adapter = new SerieAdapter(ExplorarSeries.this, seriesList);
//S'assigna l'adapter a la llista
my_ListView.setAdapter(adapter);
```

Es crea un nou adapter am la informació de l'**ArrayList<Series>** seriesList i s'assigna l'**Adapter** al **ListView**.

Evidentment, perquè tot això funcioni hem d'haver definit el mètode onCreate() en la nostra classe com en l'exemple de la pantalla principal, on hem de definir i donar valor a certes variables tals com:

```
//Estableix el contingut de la pantalla
setContentView(R.layout.all_products);
//ArrayList on es guardaran les Series
seriesList = new ArrayList<Serie>()
// Get listview
my_ListView = (ListView) findViewById(R.id.all_prod_list);
```

També serà en l'**onCreate()** on executarem la tasca en segon pla.

```
// Carregant series en segon pla.
new LoadAllProducts().execute();
```

5.5 – Guardar info BBDD local

Una de les funcionalitats que utilitza l'aplicació és el fet de guardar informació a la base de dades local, és a dir la que es troba en el nostre dispositiu. D'aquesta manera tenim un mètode per guardar informació en un espai que ens permet un accés i consulta molt més ràpids que no pas en el cas d'una base de dades remota.

Algunes de les funcionalitat que implementen el fet de guardar informació a la base de dades són:

- Guardar el llistat de series que l'usuari segueix, és a dir el llistat de series preferides.
- Guardar el llistat de capitols de les series que l'usuari segueix, així com marcar quins capítols estan vistos i quins no.
- Guardar informació sobre la sessió actual, és a dir, un cop l'usuari inicia sessió a l'aplicació es guarden les seves credencials a la base de dades local i quan tanca la sessió s'esborren. D'aquesta manera sempre podem saber si l'usuari

està identificat o no i en cas de que fos necessari recuperar les seves credencials.

- Afegir noves combinacions de usuari-capítol i usuari-serie. Quan s'afegeixen aquestes relacions a la base de dades remota, hem de passar com a paràmetre l'identificador d'usuari que podem recuperar de la base de dades local.

Per a poder realitzar totes aquestes funcions és necessària la creació d'una classe que gestioni totes les creacions, insercions, eliminacions... de taules i columnes de la base de dades.

A continuació veurem l'estructura i alguna de les parts més importants d'aquesta classe i com s'utilitzen les seves funcionalitats.

5.5.1 - La classe BBDD

Hem anomenat a la nostra classe "BBDD", aquesta serà la classe que gestionarà totes les operacions que fem en la base de dades local. En aquest cas, com no es tracta d'una classe que implementi una pantalla no és necessari que estengui la classe **Activity.**

El primer que ens trobem en aquesta classe és la declaració de les variables que utilitzarem en la creació, consulta i inserció de dades. Per simplificar l'exemple només és mostraran les variables i funcions referents a la taula de series.

```
public static final String NOM_BDD = "SeriesBbdd";
public static final int DATABASE_VERSION = 1;
// Taula Series
public static final String TABLE_SERIES = "taula_series";
public static final String KEY_ROWID = "id";
public static final String KEY_NOMSERIE = "nom_serie";
public static final String KEY_CADENA = "cadena";
public static final String KEY_TEMPORADES = "temporades";
public static final String KEY_IMATGE_THUMB = "imatge_thumb";
```

Podem veure que declarem en Strings estàtics:

- El nom de la base de dades i la seva versió.
- El nom de la taula
- El nom de les columnes de la taula

Dins de la classe que acabem de crear es crearà una nova classe privada i estàtica que anomenarem DbHelper.

```
private static class DbHelper extends SQLiteOpenHelper {
	public DbHelper(Context context) {
		super(context, NOM_BDD, null, DATABASE_VERSION);
	}
```

```
	@Override
	public void onCreate(SQLiteDatabase db) {
		db.execSQL("CREATE TABLE " + TABLE_SERIES + " (" + KEY_ROWID
			+ " INTEGER PRIMARY KEY, " + KEY_NOMSERIE
			+ " TEXT NO NULL, " + KEY_CADENA + " TEXT NO NULL, "
			+ KEY_IMATGE_THUMB + " TEXT NO NULL, "
			+ KEY_TEMPORADES + " INTEGER NOT NULL); ");
		}

	@Override
	public void onUpgrade(SQLiteDatabase db,int oldVersion, int newVersion){
		//Elimina les taules si exisitien
		db.execSQL("DROP TABLE IF EXISTS " + TABLE_SERIES);
		//Torna a crear les taules
		onCreate(db);
	}
}
```

Podem observar que és en l'**onCreate()** on s'execute la comanda de creació de la taula. I es crea un mètode que elimina i torna a crear les taules en cas que canviem la versió de la base de dades.

A continuació es defineixen els mètodes principals de la classe BBDD, que són el constructor i els mètodes d'obrir i tancar la base de dades.

```
public BBDD(Context c) {
	context = c;
}

public BBDD open() throws SQLException {
	helper = new DbHelper(context);
	database = helper.getWritableDatabase();
	return this;
}

public void close() {
	helper.close();

}
```

Vegem alguns mètodes que ens donaran una aproximació de com es treballa amb la base de dades local.

5.5.2 - Mètode per guardar series a la base de dades

```
public long guardarSerie(int idSerie, String nom, String cadena, String
                         imatge_thumb,int num_temporades) {
	ContentValues cv = new ContentValues();
	cv.put(KEY_ROWID, idSerie);
	cv.put(KEY_NOMSERIE, nom);
	cv.put(KEY_CADENA, cadena);
	cv.put(KEY_IMATGE_THUMB, imatge_thumb);
	cv.put(KEY_TEMPORADES, num_temporades);

	return database.insert(TABLE_SERIES, null, cv);
```

```
}
```

- Rep com a paràmetres la informació de la serie
- Guarda els paràmetres en un contenidor de valors (ContentValues)
- Insereix una nova columna a la base de dades amb aquests valors.

5.5.3 - Mètode per comprovar si existeix una serie en concret a la base de dades

```
public boolean estaAPreferits(int i) {
	boolean resultat = false;
	String[] columnes = new String[] { KEY_ROWID, KEY_NOMSERIE, KEY_CADENA,
				KEY_TEMPORADES };
	Cursor c = database.query(TABLE_SERIES, columnes, KEY_ROWID + "=" + i,
				null, null, null, null);
	if (c.moveToFirst() != false) {
		resultat = true;
	}
	return resultat;
}
```

- Rep com a paràmetres l'identificador de la serie.
- Crea un booleà anomenat "resultat" que al principi és fals
- Crea una nou array d'Strings amb els noms de les columnes
- Crea un nou **Cursor*** executant la consulta a al base de dades, és a dir cercant aquelles files que tinguin com a id el paràmetre que hem passat a la funció.

  ***Cursor**: Apuntador a les dades retornades per la consulta

- Intenta moure el cursor a la primera fila, si pot fer-ho vol dir que hi ha resultats i per tant que si que existeix la serie que volíem comprovar, per tant "resultat" passa a ser cert. Si no aconsegueix desplaçar-se a la primera fila "resultat" roman essent fals.
- Retorna el resultat.

5.5.4 - Mètode per obtenir un llistat amb totes les series

```
public ArrayList<HashMap<String, String>> obtenirSeries() {
	String[] columnes = new String[] { KEY_ROWID, KEY_NOMSERIE, KEY_CADENA,
		KEY_TEMPORADES };
	Cursor c = database.query(TABLE_SERIES, columnes, null, null, null,
		null, null);
	int iRow = c.getColumnIndex(KEY_ROWID);
	int inomSerie = c.getColumnIndex(KEY_NOMSERIE);
	int iCadena = c.getColumnIndex(KEY_CADENA);
	int iTemporades = c.getColumnIndex(KEY_TEMPORADES);
	ArrayList<HashMap<String, String>> series = new
```

```
            ArrayList<HashMap<String, String>>();
        // adding each child node to HashMap key => value
        for (c.moveToFirst(); !c.isAfterLast(); c.moveToNext()) {
            HashMap<String, String> map = new HashMap<String, String>();
            map.put("id", c.getString(iRow));
            map.put("nom", c.getString(inomSerie));
            map.put("cadena", c.getString(iCadena));
            // adding HashList to ArrayList
            series.add(map);
        }
        return series;
}
```

- Crea una nou array d'Strings amb els noms de les columnes
- Crea un nou **Cursor** executant la consulta a al base de dades, és a dir cercant totes les files de la taula.
- Crea un nou **ArrayList** on es guardaran les series obtingudes.
- Itera el Cursor per totes les files guardant a l'**ArrayList** els valors de la serie.
- Retorna l'**ArrayList**.

5.6 - Servei GCM

Una altra de les funcionalitats de l'aplicació és el fet de poder rebre notificacions dels nous capítols que van apareixent de les series que segueix un usuari. Per a poder realitzar aquesta funcionalitat ens hem ajudat d'un servei de Google anomenat Google Cloud Messaging.

En la part del servidor disposem d'una classe php que comprova els nous capítols i utilitza el GCM per a notificar als usuaris afectats. Ja hem explicat en més detall aquest fet en els diagrames de seqüència, ara comentarem amb una mica més de detall com s'ha implementat aquest servei en el dispositiu.

A diferencia de les altres classes i funcionalitats que hem anat explicant, en aquest cas no comentarem el codi en detall, ja que és força complex i extens i només faria que embolicar innecessàriament el document actual.

5.6.1 – IntentService

Un **Service** és un component d'una aplicació que permet realitzar accions de llarga durada sense haver d'interactuar amb l'usuari ni proporcionar una interfície d'usuari, en altres paraules, és una manera de que certes funcionalitats de l'aplicació s'estiguin sempre executant encara que l'aplicació no estigui oberta.

Ja hem comentat que un **Intent** era qualsevol intenció de realitzar una acció, per part d'una aplicació en concret o d'un servei extern. Per tant l'**IntentService** no és més que una manera "d'escoltar" certs **Intents** per tal de capturar aquells que ens interessen i realitzar certa acció, és a dir que disposarem d'un component de l'aplicació, encara que aquesta no estigui oberta, que comprovarà en tot moment si es rep un missatge des de GCM per tal de mostrar-lo i realitzar les accions pertinents.

5.6.2 – El registre

Quan un usuari faci servir l'aplicació per primera vegada en un dispositiu en concret, la nostra aplicació realitzarà el registre d'aquest dispositiu al servei GCM, atorgant-li un número de registre que es guardarà a la base de dades remota i a partir d'aquell moment serà el que s'utilitzarà per enviar notificacions a aquell dispositiu en concret.

5.6.4 – La recepció de missatges

Quan el dispositiu rep un missatge és l'**IntentService** l'encarregat de capturar-lo. Un cop rep el missatge s'execute el mètode **onMessage()**, que és el que s'encarrega de tractar-lo. El missatge en si és un Array amb informació del nou capítol. El mètode onMessage(), sempre i quant l'usuari tingui la sessió iniciada, crea un nou objecte de Capítol amb la informació obtinguda, el guarda a la base de dades i crida al mètode **generateNotification()** que s'encarrega de mostrar a l'usuari que ha aparegut un nou capítol.

5.7 - La part servidor

Un dels elements més importants de l'aplicació és la part del servidor. A diferencia de la part que s'executa al dispositiu, la del servidor no disposa d'una interficie d'usuari gràfica, sinó que consta de diferents classes codificades en llenguatge php que es criden remotament des del dispositiu.

Totes aquestes classes es troben allotjades en el servei d'allotjament gratuit 000Webhost.

5.7.1 - Els arxius php

Com ja s'ha comentat, la part del servidor, consta majoritàriament d'una serie de classes php que executen un codi que consulta, modifica o afegeix informació a la base de dades remota. A continuació descriurem alguns del arxius i en comentarem la seva funcionalitat.

a) Obtenir llistat de totes les series

Nom	get_all_products.php
Descripció	Obté un llistat de totes les series de la base de dades.
Paràmetres	No necessita rebre cap paràmetre
Acció	Si ha rebut els paràmetres correctament, executa el query "*INSERT INTO series(nom, num_temporades, cadena) VALUES ('$nom', '$temporades', '$cadena')*" per tal d'introduir a la base de dades la nova serie.
Resposta	Retorna en format JSON el resultat de la inserció.

b) Afegir serie a la base de dades

Nom	add_serie.php
Descripció	Afegeix una nova serie a la base de dades.
Paràmetres	Ha de rebre paràmetres amb informació de la serie, tals com el nom, la cadena i el número de temporades.
Acció	Executa el query "*SELECT *FROM series*", per tal d'obtenir totes les files de la taula series. En acabat recorre totes les files obtingudes i en va guardant el valor de cada columna en un Array que s'introdueix en un JSON.
Resposta	Retorna el JSON amb el llistat de les series o el missatge d'error en el cas que hagi fallat.

c) Obtenir informació de la serie

Nom	get_series_details.php
Descripció	Obté informació detallada d'una serie en concret.
Paràmetres	Ha de rebre com a paràmetre el número que identifica la serie (id).
Acció	Executa el query "*SELECT *FROM series WHERE id = $id*" , per tal d'obtenir la fila de la taula series de la serie que volem consultar . En acabat guarda el valor de cada columna en un Array que s'introdueix en un JSON.
Resposta	Retorna el JSON amb la informació de la serie o el missatge d'error en el cas que hagi fallat.

d) Obtenir els capítols d'una serie

Nom	get_capitols.php
Descripció	Obté tots els capítols s'una serie en concret
Paràmetres	Ha de rebre com a paràmetre el número que identifica la serie (idSerie).
Acció	Executa el query "*SELECT *FROM capitols WHERE idSerie = $idSerie*" , per tal d'obtenir les files de la taula capítols on coincideixi l'id de la serie de la qual volem obtenir els capítols . En acabat recorre totes les files obtingudes i va guardant el valor de cada columna en un Array que s'introdueix en un JSON.
Resposta	Retorna el JSON amb el llistat de capítols de la series o el missatge d'error en el cas que hagi fallat.

e) Guardar els capítols de les series que segueix un usuari

Aquesta classe no s'executa de forma independent, sinó que disposa d'una funció que es crida des de la classe que guarda les series que segueixen els usuaris, ja que sempre que vulguem guardar una nova relació serie-usuari també necessitarem guardar la relació capítols_de_la_serie-usuari.

Nom	add_capitol_user.php
Descripció	Guarda el llistat de capítols d'una serie en concret en la taula usuari_capitol, que és l'encarregada de guardar tots els capítols de les series que segueixen els usuaris.
Paràmetres	Ha de rebre com a paràmetre el número que identifica la serie (serie_id) així com l'identificador d'usuari (usuari_uid).
Acció	Executa el query "*SELECT * FROM capitols WHERE idSerie = $serie_id*" , per tal d'obtenir les files de la taula capítols on coincideixi l'id de la serie de la qual volem obtenir els capítols. En acabat recorre totes les files obtingudes i va insertant capítol a capítol una nova relació usuari-capítol, executant el query "*INSERT INTO usuari_capitol(usuari_uid, idSerie, idCapitol, dateAdded) VALUES('$usuari_uid', $serie_id, $idCapitol, NOW())*"
Resposta	Retorna el JSON amb el resultat de la inserció.

f) Funcions d'usuari

Com la classe anterior, aquesta tampoc s'executa de forma independent, sinó que disposa de diverses funcions que són cridades des d'altres classes del servidor o des d'altres funcions de la pròpia classe.

Nom	db_functions.php
Descripció	Executa diverses funcions relacionades amb usuaris, com per exemple registrar-se, obtenir informació d'un usuari, comprovar si existeix un usuari...
Paràmetres	Cada una de les funcions ha de rebre paràmetres diferents.
Acció	Es comentaran les accions de cada una de les funcions de la classe: • storeGCMUser: guarda usuaris del servei GCM a la taula gcm_usuaris, executant el query "INSERT INTO gcm_usuaris(name, email, gcm_regid, created_at) VALUES('$name', '$email', '$gcm_regid', NOW())" • StoreUser: Registra un usuari al sistema guardant-ne les dades a la base de dades, encripta el password per seguretat. Executa el query "*INSERT INTO users(unique_id, name, email, encrypted_password, salt, created_at) VALUES('$uuid', '$name', '$email', '$encrypted_password', '$salt', NOW())*". • GetUserByEmailAndPassword: Obté un usuari a partir del seu email i contrasenya. Comprova que el passowrd sigui correcte desencriptant-lo previament. • IsUserExisted: Comprova si un usuari existeix cercant el seu email a la base de dades, executant el query "*SELECT email from users WHERE email = '$email*" • HashSSHA: Encripta la contrasenya i la retorna encriptada i la clau d'encriptació. • CheckhashSSHA: Desencripta la contrasenya. • GetAllUsers: Retorna tots els usuaris del servei GCM executant el query "select * FROM gcm_usuaris"
Resposta	Cada funció retorna un resultat diferent.

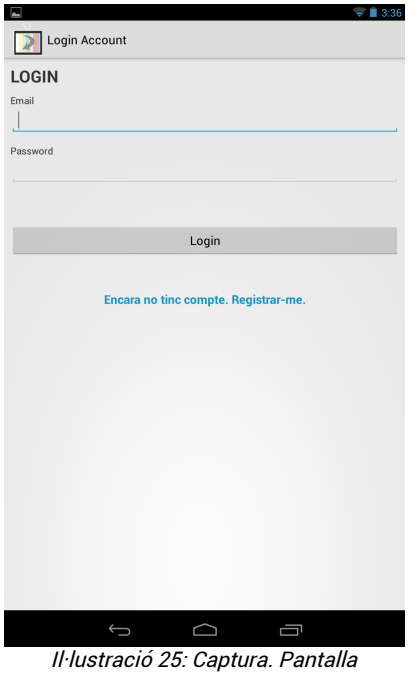
5.8 - La base de dades

Un altre element fonamental de l'aplicació és la base de dades. És l'encarregada de guardar tota la informació necessària per tal que l'aplicació pugui consultar, guardar o modificar totes les dades requerides perquè tot funcioni de forma correcta. Disposa de taules per guardar informació sobre usuaris, series, capítols, enllaços, relacions entre usuaris i capítols i entre usuaris i series (series i capítols que els usuaris segueixen).

De la mateixa manera que les classes php, la base de dades es troba allotjada en el servei d'allotjament gratuït 000Webhost i com a sistema de gestió de base de dades utilitza MySQL.

5.9 – Captures de pantalla

5.9.1 – Pantalla d'autenticació



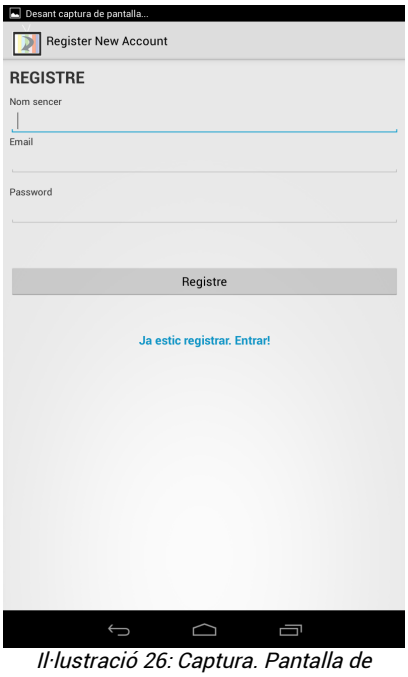
Il·lustració 25: Captura. Pantalla d'autenticació

Quan l'usuari entra per primera vegada a l'aplicació aquesta és la primera pantalla que visualitza, des d'aquí es pot procedir a l'autenticació per poder accedir al sistema.

Com es pot veure, consta de dos camps de text on l'usuari ha d'introduir les seves credencials i procedir prement el botó "Login".

Si l'usuari encara no s'ha registrat al sistema pot prémer sobre el text "Encara no tinc compte. Registrar-me" per procedir al registre.

5.9.2 – Pantalla de registre

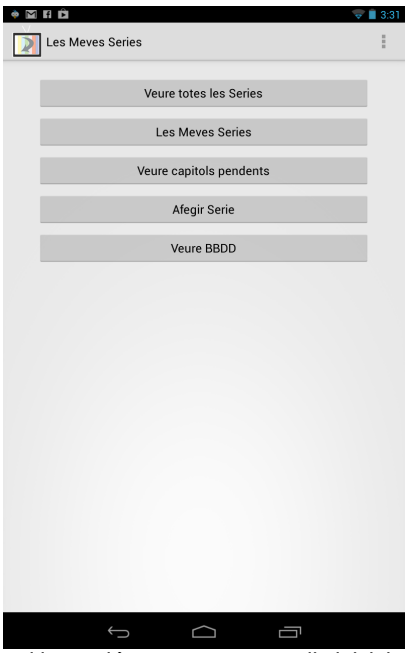


Il·lustració 26: Captura. Pantalla de registre

Aquesta és la pantalla des de la qual l'usuari podrà crear un compte d'usuari, per a fer-ho haurà d'introduir la informació requerida, és a dir, el seu nom sencer, el seu correu electrònic i la contrasenya amb la qual voldrà accedir al sistema. Per a procedir al registre s'haurà de prémer el botó "Registre".

Si l'usuari ja disposa d'un compte, haurà de prémer sobre el text "Ja estic registrat. Entrar!" per accedir a la pantalla d'autenticació.

5.9.3 – Pantalla inicial

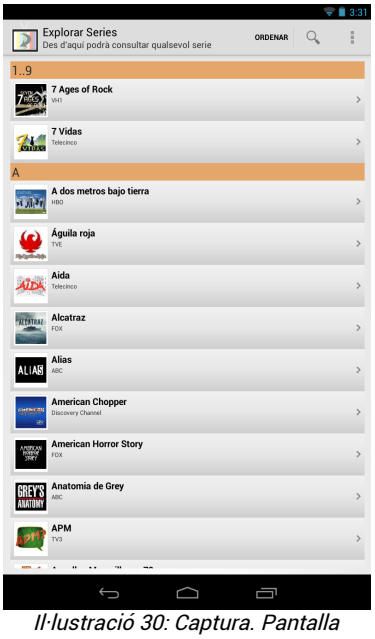


Il·lustració 27: Captura. Pantalla inicial

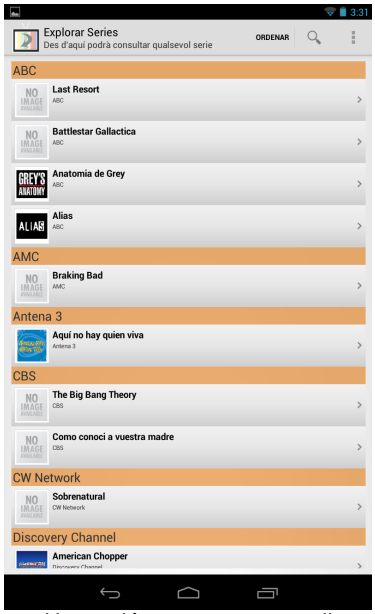
Aquesta és la pantalla inicial, on es troba el menú principal de l'aplicació. Trobem les següents opcions:

- Veure totes les series: Permet accedir al llistat complert de series.
- Les Meves Series: permet accedir al llistat de series preferides.
- Veure capítols pendents: permet accedir a la pantalla on es mostren els capítols que l'usuari encara no ha vist.
- Afegir serie: serveix per poder afegir una nova serie a la base de dades
- Veure BBDD: element auxiliar per poder veure el contingut de la base de dades. Només té sentit per al desenvolupador.

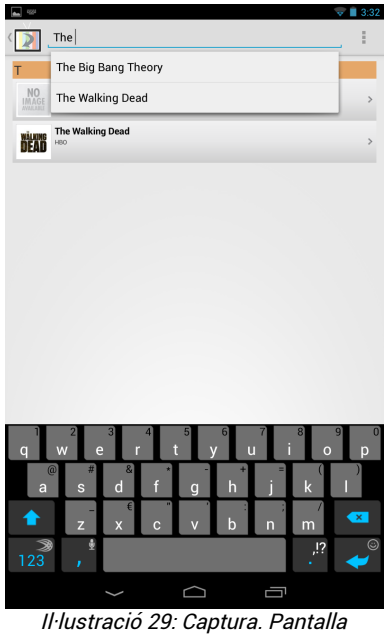
5.9.4 – Pantalla d'explorar series



Il·lustració 30: Captura. Pantalla explorar series, odre alfabètic



Il·lustració 28: Captura. Pantalla explorar series. Ordre cadena



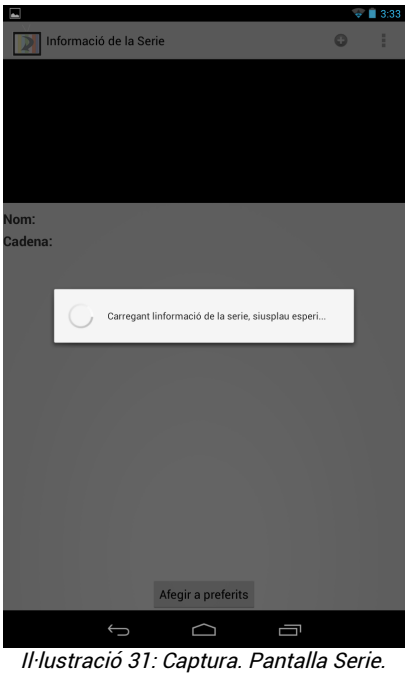
Il·lustració 29: Captura. Pantalla explorar series, exemple de cerca

Aquesta és la pantalla que permet explorar el llistat de totes les series, com es pot observar, de cada serie se'n mostra el nom, la cadena a la que pertany i una petita imatge al costat esquerre.

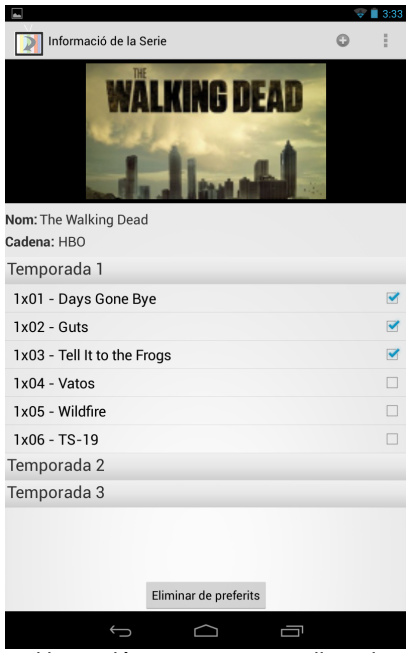
En el menú superior disposem de dues opcions principals, la primera és la que permet ordenar el llistat per diferents criteris, en la segona imatge podem veure aquest mateix llistat ordenat per cadena en comptes de per ordre alfabètic.

L'altra opció del menú superior és la icona de la lupa, que permet fer una cerca dins del llistat de series, com podem observar en la tercera imatge, és una barra de cerca que autocompleta el text, és a dir a mesura que l'usuari va escrivint escrivint va suggerint elements de la llista per completar la cerca. A més el llistat de series en sí es va adaptant a aquestes suggerències.

5.9.5 – Pantalla d'informació d'una serie



Il·lustració 31: Captura. Pantalla Serie. Carregant



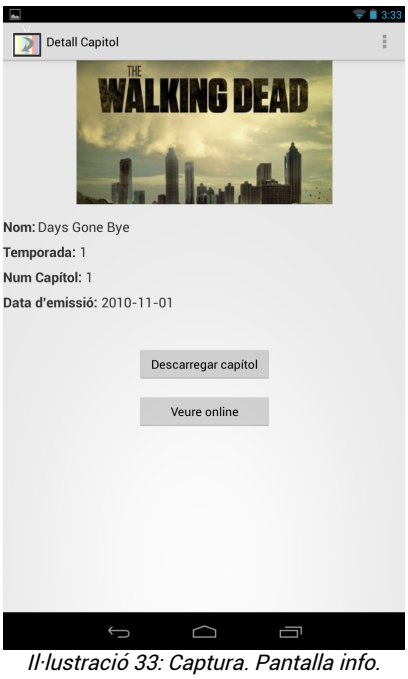
Il·lustració 32: Captura. Pantalla serie

Aquesta pantalla permet veure informació ampliada d'una serie, en podem visualitzar el nom, la cadena i un llistat amb les temporades, si es prem sobre alguna de les temporades, s'expandeix i es mostren els capítols d'aquesta. Com es pot observar en la primera captura de pantalla, mentre no s'ha obtingut tota la informació de la serie de la base de dades, apareix un diàleg que ens informa de que s'està carregant la informació.

Cada capítol disposa d'un requadre de validació que permet marcar-lo com a vist, només es pot fer quan la series es troba al llistat de preferides, la qual cosa es pot fer prement el botó "Afegir a preferits" de la part inferior.

En el menú superior es disposa d'una icona amb el símbol positiu que permet obrir una nova pantalla per afegir un nou capítol a la serie.

5.9.6 – Pantalla d'informació de capítol

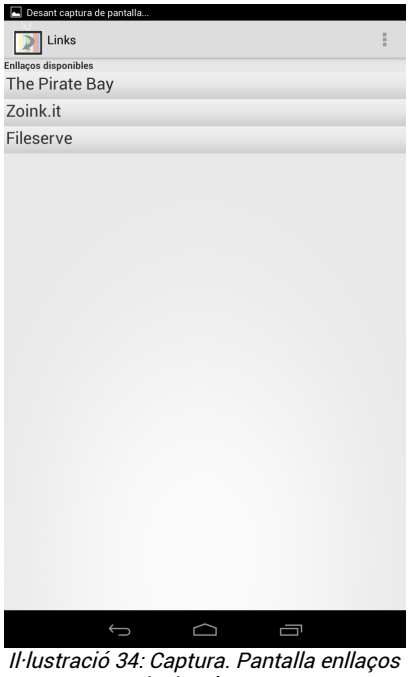


Il·lustració 33: Captura. Pantalla info. capítol

En aquesta pantalla es pot observar informació sobre un capítol, hi apareix una imatge de la serie, el nom del capítol, la temporada a la que pertany, el número de capítol dins de la temporada i la data d'emissió.

Hi ha disponibles dos botons, un per descarregar el capítol i l'altre per a veure'l online.

5.9.7 – Pantalla d'enllaços de descàrrega

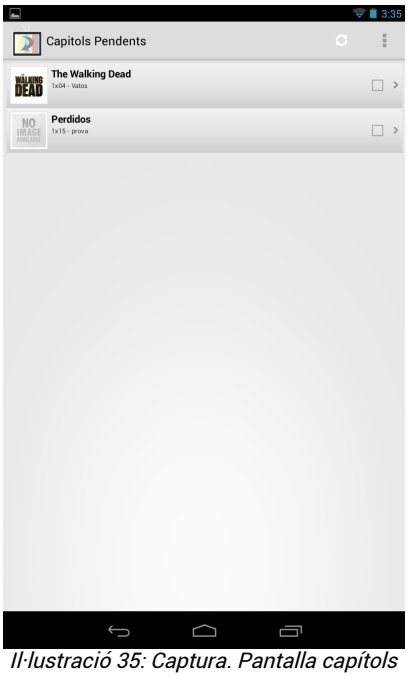


Il·lustració 34: Captura. Pantalla enllaços de descàrrega

En aquesta pantalla es mostren els enllaços de descàrrega disponibles per un capítol en concret. Aquests enllaços estan ordenats per Servidor de descàrrega, un cop es prem un dels servidors, s'expandeix el llistat i es mostren els enllaços per aquell servidor.

Quan es prem algun dels enllaços s'obre un dialeg preguntant amb quina aplicació es vol procedir a realitzar la descàrrega.

5.9.8 – Pantalla de capítols pendents



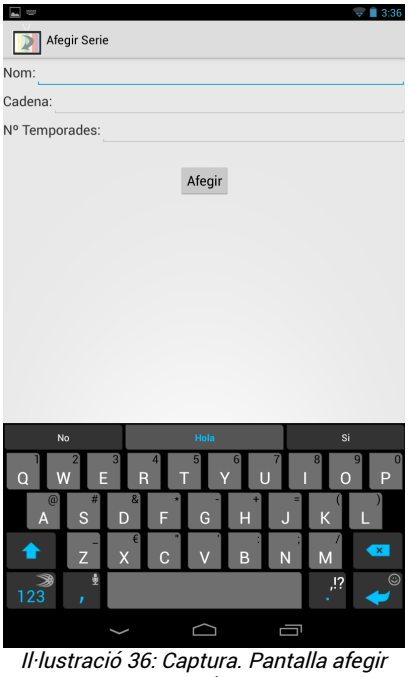
Il·lustració 35: Captura. Pantalla capítols pendents

Des d'aquesta pantalla es poden visualitzar els capítols pendents per veure, és a dir, de cada serie que segueix l'usuari el primer capítol que encara no s'ha vist. De cada capítol se'n mostra el nom i una petita imatge de la serie, el nom del capítol, la temporada i el número dins de la temporada. Cada dun dels capítols disposa d'una caixa de confirmació que permet marcar un capítol com a vist.

En el menú de la part superior disposem d'una icona que permet refrescar la pàgina, per si s'ha marcat un capítol com a vist, que mostri el següent capítol. A demés també es disposa d'un menú desplegable amb una opció per mirar d'obtenir nous capítols de la base de dades remota.

Un cop es prem sobre algun dels capítols del llistat, s'obre la pantalla d'informació del capítol.

5.9.9 – Pantalla per afegir nova serie

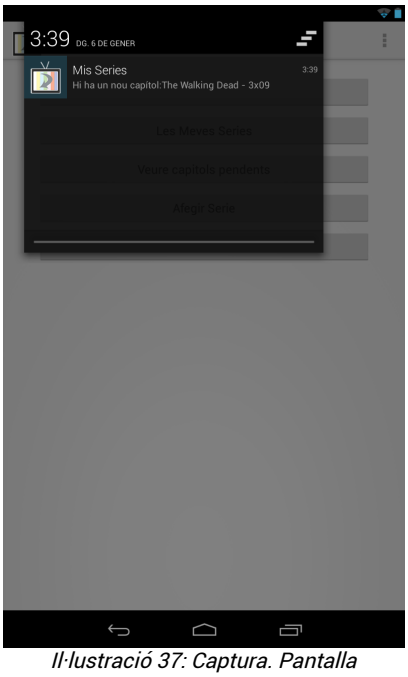


Il·lustració 36: Captura. Pantalla afegir serie

Aquesta pantalla permet a l'usuari afegir una nova serie, demana informació tal com el nom, cadena i número de temporades de la serie.

Una vega introduïdes totes les dades es procedeix al guardat premen sobre el botó "Afegir". Si al fer-ho manca algun camp, l'aplicació avisarà a l'usuari quin és el camp que falta per omplir i no permetrà el guardat fins que totes les dades estiguin complertes.

5.9.10 – Notificació de capítol nou



Il·lustració 37: Captura. Pantalla notificació capítol nou

En aquesta captura es pot apreciar com es visualitza quan es rep la notificació de que ha aparegut un capítol nou per a alguna de les series que segueix l'usuari. Inclou el nom de la serie, i la temporada i número del capítol.

Un cop es prem sobre la notificació s'obre la pantalla de capítols pendents.

6 – Conclusions

Durant la realització del projecte s'ha implementat una aplicació per a dispositius móbils amb sistema operatiu Android per a poder gestionar, descarregar i visualitzar les series que segueix un usuari. Tot i que no s'han pogut implementar totes les funcionalitats que s'havien proposat en un principi, l'aplicació és del tot funcional, permet realitzar les accions proposades més importants i l'estructura dels sistema està del tot implementada i és funcional al cent per cent.

S'ha pogut comprovar la facilitat amb la que s'ha pogut treballar amb sistemes completament diferents en una arquitectura client-servidor utilitzant un sistema d'intercanvi de dades estructurat com és JSON, fent molt senzilla aquesta comunicació gràcies a la simplicitat del format.

A nivell personal, ha sigut un procés molt gratificant, doncs durant la realització del projecte s'han aprés un munt d'aspectes que en un principi es desconeixien, des d'elements purament tècnics o de funcionament fins a nous llenguatges de programació. El fet de veure com ha sigut possible començar des de cero en el món de la programació per dispositius mòbils i aconseguir realitzar una aplicació amb cara i ulls ha sigut sens dubte una experiència molt bona.

7 - Bibliografia

http://www.androidhive.info/

Diferents tutorials i exemples explicats de manera molt detallada de com realitzar certes funcions en dispositius Android.

http://developer.android.com/index.html

Font principal d'informació per a desenvolupadors Android, amb documentació sobre les classes i el disseny de la interfície, així com exemples d'us d'aquestes classes.

http://thenewboston.org/list.php?cat=6

200 videotutorials curts de programació per Android, començant des dels conceptes més bàsics i amb una corba d'aprenentatge molt ben ajustada.

http://android.cyrilmottier.com/?p=574

Alguns exemples i tutorials sobre alguns aspectes del desenvolupament per Android, sobretot en quant a interfícies gràfiques.

http://stackoverflow.com/questions/tagged/android

Lloc web de preguntes i respostes col·laboratiu on són els usuaris els que formulen i responen les qüestions. Font d'informació imprescindible per resoldre els dubtes que van sorgint durant el procés d'implementació.

http://www.androiduipatterns.com/

Tutorials i exemples de com dissenyar i millorar interfícies d'usuari per a Android.

Android. Guía para desarrolladores – ABLESON, Frank. SEN, Robi. KING, Chris (Anaya Multimedia – ISBN:8441529582)

8 - Glossari de termes

Android: Sistema operatiu per a dispositius mòbils basat en Linux. Desenvolupat per la Open Handset Alliance, liderada per Google. La primera versió va ser llançada el 21 d'Octubre de 2008, des d'aleshores han aparegut diverses versions, i segons la firma IDC actualment domina un 75% del mercat s'Smartphones.

Activity: Un Activity és una pantalla única que l'usuari pot visualitzar en un moment donat. S'encarrega de crear una finestra on es pot disposat la interfície d'usuari definida en l'xml. Normalment es tracta d'una pantalla completa, però també pot ser una pantalla flotant o una pantalla incrustada en una altra.

Array (Vector): És una estructura de dades, que està constituïda per diferents elements i que l'accés a aquests es fa per indexació. Normalment tots els elements d'un Array o Vector són del mateix tipus.

Eclipse: Entorn de desenvolupament integrat (IDE) per a desenvolupar en diferents llenguatges com C++, COBOL, Python, o Java. Actualment es troba en la versió 4.2 (Juno).

GCM (Google Cloud Messaging): És un servei de Google que permet l'enviament de dades a través del núvol. Permet enviar dades entre usuaris o des d'un servidor a un o diversos usuaris a través de notificacions **Push.** Això permet, entre d'altres coses, que l'aplicació no hagi d'estar constantment contactant amb el servidor per a comprovar si hi ha noves dades, sinó que és el servidor que envia les dades tan bon punt aquestes estan disponibles.

Intent: És la manera que té el S.O Android de descriure la intenció de realitzar alguna acció (obrir un enllaç, començar un Activity, obrir el dial del telèfon,...), depenent de quina acció sigui i a qui va dirigida, els SO o l'aplicació reaccionaran d'una manera en concret.

Java: És un llenguatge de programació multi-plataforma orientat a objectes creat l'any 1990. Es considera un llenguatge potent, flexible i senzill de programar. Gaudeix dels avantatges típics dels llenguatges orientats a objectes, és a dir, l'herència, l'encapsulament i el polimorfisme.

JSON (JavaScript Object Notation): Estàndard obert basat en text dissenyat per a intercanvi de dades llegibles per les persones. Gràcies a que és un estàndard simple se n'ha generalitzat molt el seu ús, sobretot com a alternativa a l'xml.

Multidifusió: o Multicast en anglès. És l'enviament d'informació a diferents destinataris de forma simultània.

Play Store: Botiga d'aplicacions, jocs, pel·lícules, llibres i música per a dispositius amb sistema operatiu Android.

Push (notificació Push): Estil de comunicacions a Internet on la petició s'origina en el servidor, en comptes d'originar-se en el client. La missatgeria instantània és un clar exemple de la utilització d'aquest estil de comunicació, on el servidor envia a l'usuari el missatge tan bon punt el té disponible, en comptes de ser l'usuari el que va comprovant si el servidor disposa de nous missatges per enviar.

PhP: Llenguatge de programació per a la creació de pàgines web de contingut dinàmic. S'executa a la part del servidor i aquest genera la pàgina web resultant. Permet la connexió amb diferents bases de dades.

Query: Petició a una base de dades per tal d'obtenir, emmagatzemar o modificar la informació.

Script: Arxiu d'ordres o guió que es interpretat per tal de realitzar certes accions com combinar components, interactuar amb el sistema operatiu o amb l'usuari. Php és un exemple de llenguatge que utilitza scripts executats des de la part servidor.

Service: En Android, un Service és component d'una aplicació que permet realitzar accions de llarga durada sense haver d'interactuar amb l'usuari ni proporcionar una interfície d'usuari, en altres paraules, és una manera de que certes funcionalitats de l'aplicació s'estiguin sempre executant encara que l'aplicació no estigui oberta.

SQL (Structured Query Lenguage): Llenguatge d'accés a bases de dades relacionals que permet especificar diversos tipus d'operacions. Com per exemple fer consultes per recuperar, crear o modificar informació de la base de dades.

Toast: En Android, un Toast és una manera de proporcionar informació sobre la realització d'una operació en una petita finestra emergent a la part inferior de la pantalla. Entre d'altres aspectes, se'n pot editar el text i la durada.

View: Representa els components de la interfície d'usuari. Ocupa una àrea rectangular a la pantalla i és el responsable de dibuixar i gestionar els esdeveniments del component

XML (eXtensible Markup Lenguage): Metallenguatge extensible organitzat per etiquetes que permet l'intercanvi d'informació estructurada entre diferents plataformes.