

Taula de contingut

1	Introducció, motivacions, propòsits i objectius del projecte	6
1.1	Introducció	6
1.2	Motivacions i propòsits	7
1.3	Objectius del projecte.....	7
1.4	Abast.....	8
2	Estudi de viabilitat.....	9
2.1	Coneixements previs	9
3	Metodologia.....	10
3.1	Scrum.....	10
3.1.1	Rols de Scrum	13
3.1.2	Sprint	14
3.1.3	Reunions de Scrum.....	15
3.1.4	Documentació	18
4	Marc de treball i conceptes previs	19
4.1	Entorn de treball.....	19
4.2	Eines de treball	20
4.3	Conceptes previs	21
4.3.1	Que és iOS?	21
4.3.2	Arquitectura per capes.....	22
4.3.3	Desenvolupament per iOS.....	23
4.3.4	Eines de desenvolupament: iOS SDK.....	24
4.3.5	Control de versions	29
4.3.6	Com apliquem Scrum	29
5	Planificació i requisits del sistema.....	31

6	Estudis i decisions.....	33
7	Anàlisi i disseny del sistema	35
8	Implementació i proves.....	38
8.1	Sprint 1	38
8.1.1	Històries d'usuari detallades	38
8.1.2	Descarregar portades publicacions del dia	39
8.1.3	Mostrar portades publicacions del dia	42
8.1.4	Diagrama d'evolució del Sprint	43
8.1.5	Incidències destacables	43
8.1.6	Reunió final Sprint.....	44
8.1.7	Demostració del producte al client	45
8.2	Sprint 2	46
8.2.1	Histories d'usuari detallades	46
8.2.2	Descarregar una publicació concreta	47
8.2.3	Mostrar una publicació descarregada	50
8.2.4	Diagrama d'evolució del Sprint	51
8.2.5	Incidències destacables	51
8.2.6	Reunió final Sprint.....	52
8.2.7	Demostració del producte al client	53
8.3	Sprint 3	54
8.3.1	Històries d'usuari detallades	54
8.3.2	Més informació a la llista de publicacions.....	55
8.3.3	Deslligar el procés de descàrrega i visualització.....	56
8.3.4	Guardar les descarregues en el dispositiu.....	57
8.3.5	Mostrar una publicació descarregada (paginada)	59
8.3.6	Llistat publicacions descarregades	60

8.3.7	Diagrama d'evolució del Sprint	61
8.3.8	Incidències destacables	61
8.3.9	Reunió final Sprint	62
8.3.10	Demostració del producte al client.....	63
8.4	Sprint 4	64
8.4.1	Històries d'usuari detallades	64
8.4.2	Navegació per seccions de publicació	65
8.4.3	Canvi imatges JPG alta resolució per PDF.....	67
8.4.4	Adaptar vistes al nou disseny	67
8.4.5	Eliminar publicacions.....	71
8.4.6	Veure darreres publicacions d'una publicació	73
8.4.7	Diagrama d'evolució del Sprint	78
8.4.8	Incidències destacables	78
8.4.9	Reunió final Sprint	79
8.4.10	Demostració del producte al client.....	80
8.5	Sprint 5	82
8.5.1	Històries d'usuari detallades	82
8.5.2	Comunicacions amb XML-RPC.....	83
8.5.3	Plugin EPQuiosc.....	83
8.5.4	Plugin EPUuaris	86
8.5.5	Adaptar models per treballar amb XML-RPC	92
8.5.6	Identificar usuari a El Quiosc	94
8.5.7	Identificar-me	98
8.5.8	Desconnectar-me	98
8.5.9	Descarrega gratuïta	99
8.5.10	Anar a	99

8.5.11	Veure fitxa d'usuari.....	99
8.5.12	Diagrama d'evolució del Sprint.....	100
8.5.13	Incidències destacables	100
8.5.14	Reunió final Sprint	102
8.5.15	Demostració del producte al client.....	103
8.6	Sprint 6	105
8.6.1	Històries d'usuari detallades	105
8.6.2	Encapsular publicació ZIP	106
8.6.3	Opcions de configuració de l'aplicació	107
8.6.4	Navegació per pàgines i seccions	108
8.6.5	Mode eliminar publicacions descarregades	110
8.6.6	Buscar publicació per data	111
8.6.7	Control disponibilitat de connexió	112
8.6.8	Utilització de la aplicació offline.....	113
8.6.9	Mode mono/multi publicació.....	114
8.6.10	Diagrama d'evolució del Sprint.....	116
8.6.11	Incidències destacables	116
8.6.12	Reunió final Sprint	117
8.6.13	Demostració del producte al client.....	117
8.7	Sprint 7	120
8.7.1	Històries d'usuari detallades	120
8.7.2	Personalització de l'aplicació per Presència.....	123
8.7.3	Diagrama d'evolució del Sprint	125
8.7.4	Incidències destacables.....	125
8.7.5	Demostració del producte al client	126
9	Implantació	130

10	Conclusions	135
11	Treball futur	136
12	Bibliografia	137

1 Introducció, motivacions, propòsits i objectius del projecte

1.1 Introducció

Hermes Comunicacions és el primer grup de comunicació escrita en català, que compta amb 54.000 exemplars cada dia al carrer i prop de 600.000 lectors a internet. És la responsable de les publicacions: El Punt Avui (edicions Nacional i Comarques Gironines), Presència, L'Econòmic, Sortim, Catalonia Today i El 9 Esportiu.

Actualment, es disposa de tres vies per difondre les diferents publicacions:

- ❖ **Paper:** El format tradicional de tota la vida. De venda als quioscos o repartit a els subscriptors.
- ❖ **Web:** L'accés mitjançant internet a les diferents notícies i opinions publicades als productes en paper. En destaquem el “elpuntavui.cat”, un web de lliure accés, on trobem la informació publicada a les dues edicions del producte (nacional i comarques gironines) organitzada poble a poble i amb continguts addicionals com blocs, foto-galeries, vídeos, agenda, entre d'altres.
- ❖ **El Quiosc:** Un web, d'accés exclusiu als subscriptors de les publicacions del Grup Hermes. L'objectiu del qual és facilitar les diferents edicions dels productes en format PDF, per tal de llegir-les, consultar-les o desar-les als seus ordinadors.

Aquest projecte pretén desenvolupar l'aplicació i entorn necessaris per la creació d'una quarta via de difusió dels diferents productes, facilitant-ne la visualització mitjançant dispositius mòbils. Concretament ens centrarem en els dispositius mòbils d'Apple basats en el sistema operatiu iOS.

1.2 Motivacions i propòsits

El gran èxit dels dispositius d'Apple i de les seves apps despertava l'interès de l'autor del projecte en aprendre el llenguatge i el framework emprats en el desenvolupament d'aquestes.

Analitzades les dades de trànsit (veure 7.Estudis) dels diferents webs del grup, reafirmem la gran penetració dels dispositius d'Apple entre els dispositius mòbils utilitzats.

El plus de realitzar un projecte que tingui una aplicació pràctica un cop finalitzat el desenvolupament. I que al mateix temps pugui aportar alguna cosa positiva a l'empresa, ens varen dur a pensar en la possibilitat de desenvolupar una "app" per distribuir digitalment les publicacions a dispositius mòbils amb sistema operatiu iOS.

1.3 Objectius del projecte

L'objectiu del projecte consisteix en desenvolupar una plataforma per distribuir digitalment els diferents productes del grup a diferents dispositius mòbils. Concretament a els dispositius d'Apple (iPad, iPhone i iPod Touch) basats en el sistema operatiu iOS.

L'aplicació ens ha de permetre navegar per les diferents publicacions i cercar la data desitjada mitjançant l'hemeroteca.

L'usuari s'haurà de poder identificar amb les seves dades per tal de poder tenir accés a les publicacions a les que està subscrit. Un cop identificat, podrà descarregar la publicació al seu dispositiu per desplaçar lliurement i sense dependre de la connexió a la xarxa.

Caldrà poder gestionar les publicacions descarregades per tal de facilitar-ne la lectura o l'eliminació.

Per tal de poder accedir a les publicacions i altres dades necessàries per el correcte funcionament de l'aplicació, caldrà adaptar el sistema de que disposa l'empresa. La idea inicial del projecte és nodrir-se dels continguts del web *E/ Quiosc*, que disposa de totes les publicacions en format PDF.

1.4 Abast

L'abast del projecte és el d'assolir els següents objectius:

- Aprendre a desenvolupar correctament aplicacions per dispositius iOS.
És a dir, obtenir coneixements sobre el llenguatge Objective-C, Cocoa Touch i com s'estructura una "app".
- Desenvolupar un software compatible amb els diferents dispositius iOS que ens permeti obtenir i visualitzar les diferents publicacions a les que l'usuari està subscrit.
- Adaptar el sistema actual per tal de poder servir els contingut necessaris a la aplicació i identificar l'usuari en cas que sigui necessari.

2 Estudi de viabilitat

Som conscients que el mercat ens ofereix solucions semblants a l'aplicació que volem desenvolupar. Proporcionat gestió de documents en PDF i fins i tot el seu allotjament. Pretenem reduir els costos que ens suposarien l'adquisició d'un software d'aquest tipus, adaptant-nos a les nostres necessites i als sistemes dels que disposem.

El grup disposa d'una plataforma de distribució de publicacions en format digital a través del web “El Quiosc” i “El 9 Esportiu”. Per tant, s'intentarà adaptar aquesta plataforma, per tal d'aprofitar al màxim els recursos disponibles.

Per tal de poder publicar una aplicació a la “App Store” (l'única forma de distribució legal d'aplicacions natives per a iOS), Apple requereix l'adquisició d'una llicència de desenvolupament. Aquesta llicència també dona accés a la descàrrega de l'entorn desenvolupament més actual i té un cost de 99\$ anuals en la seva versió “estàndard”.

Estimem que el temps per el desenvolupament de l'aplicació serà de 5 mesos.

Tenint en compte els factors esmentats anteriorment i que el desenvolupament d'aquest projecte ens permetrà augmentar la difusió dels nostres productes sense haver d'afegir una despesa extra a les esmentades en els punts anteriorment, creiem que és un projecte molt viable.

2.1 Coneixements previs

Com s'ha exposat a l'apartat “abast del projecte”, l'autor no té coneixements sobre el desenvolupament d'aplicacions per iOS, ni dels frameworks utilitzats, ni del llenguatge base emprat. Cal destacar que a l'empresa tampoc hi ha ningú amb aquets coneixements, necessaris per orientar-me en el desenvolupament mitjançant l'SDK per iOS.

3 Metodologia

Degut a que ens trobem davant d'un nou entorn per als desenvolupadors i per al client, preveiem que aniran sorgint canvis durant el transcurs del projecte. Per tant decidim utilitzar una metodologia flexible als canvis de requeriments. Les metodologies àgils són les que millor s'ajusten al desenvolupament que es vol fer.

Les metodologies àgils compleixen quatre principis bàsics:

- Els individus i les seves interaccions per sobre dels processos i les eines.
- El programari que funciona per sobre de la documentació exhaustiva.
- La col·laboració amb el client per sobre de la negociació de contractes.
- La resposta davant del canvi per sobre de seguir un pla tancat.

Dins el conjunt de metodologies àgils, hem escollit *Scrum*. Tot i que és una metodologia pensada per a treball en equip, pretenem adaptar-la al nostre cas.

3.1 Scrum

És un marc de treball en el qual es van aplicant de manera regular un conjunt de bones pràctiques per treballar col·laborativament o en equip i obtenir el millor resultat possible d'un projecte.

Aquestes pràctiques es complementen les unes amb les altres i la seva selecció té l'origen en un estudi sobre la manera de treballar de grups altament productius.

Durant la utilització de *Scrum*, es realitzen entregues parcials i regulars del producte final, prioritzades per el benefici que aportin al receptor del projecte. Per aquest motiu, és especialment indicat per projectes en entorns complexes, on es necessiten obtenir resultats immediats, on els requisits són canviants o poc definits, on la innovació, la competitivitat, la flexibilitat i la productivitat són fonamentals.

També s'utilitza per resoldre situacions en que no s'està entregant el que es necessita, quan les entregues s'allarguen més del previst, els costos es disparen o la qualitat no és acceptable, quan es necessita capacitat de reacció davant la competència, quan la moral dels equips és baixa i la rotació alta, quan és necessari identificar i solucionar ineficiències sistemàticament o quan es requereix treballar utilitzant un procés especialitzat en el desenvolupament de productes.

Els rols principals dins *Scrum* són:

- *Scrum Master*
- *Scrum Team*
- *Product Owner*

Altres rols auxiliars dins l'equip:

- *Stakeholders*
- *Directors*

Durant cada *Sprint*, un període fix d'una durada definida per l'equip d'entre una i quatre setmanes, l'equip crea un increment de software potencialment lliurable. El conjunt de característiques que forma part de cada *Sprint* està identificat al *Product Backlog*, que és el conjunt de requeriments a alt nivell prioritzats que defineixen la feina a fer. Els elements dels *Product Backlog* que formen part del *Sprint* es determinen durant la reunió de *Sprint Planning*. Durant aquest reunió el *Product Owner* identifica els elements del *Product Backlog* prioritaris que vol veure realitzats durant el *Sprint*. Posteriorment, l'equip de desenvolupament determina la quantitat de d'elements que es pot comprometre a completar. Durant l'*Sprint*, només pot modificar el *Sprint Backlog* l'equip de desenvolupament en cas de no poder completar-lo durant aquest cicle.

Scrum permet la creació d'equips auto organitzats impulsant la co-localització de tots els membres de l'equip, i la comunicació verbal entre tots els membres i disciplines involucrats en el projecte.

Un dels principis clau que reconeix *Scrum* és la possibilitat que durant el projecte els clients puguin canviar d'idea sobre el que volen i que els requeriments no poden ser fàcilment definits completament al principi del projecte. Per tant, entenent que el problema no pot ser entès i definit completament, es centra en maximitzar la capacitat de l'equip per lliurar ràpidament i respondre als requeriments emergents.

3.1.1 Rols de Scrum

Scrum Master

És la persona responsable de fer seguir les normes de *Scrum* tant dins el grup com envers l'organització. La seva tasca principal és eliminar els obstacles que es trobi l'equip per arribar a l'objectiu de cada *Sprint*. No és el líder de l'equip, ja que aquest s'autogestiona. Protegeix l'equip de qualsevol influència que el pugui distreure.

En el nostre cas el rol de *Scrum Master*, està repartit entre el tutor del projecte (al mateix temps director adjunt del departament d'informàtica) i l'autor del projecte.

Molts dels problemes sorgits durant el desenvolupament han estat relacionats amb la tecnologia utilitzada i aquest no podien ser resolts per el tutor, per tant era el mateix autor el que havia d'intentar solucionar-los.

Mentre que problemes d'indefinició del producte o d'arquitectura de l'aplicació si que han pogut ser tractats amb el tutor.

Scrum Team o equip de desenvolupament

L'equip de desenvolupament inclou tots els participants en el desenvolupament del programari. Sol ser d'un nombre reduït entre 3 i 6 membres, amb els coneixements necessaris per realitzar l'increment "lliurable" del producte. És a dir, té la responsabilitat de lliurar el producte amb el resultat dels Sprints.

Al tractar-se d'un projecte individual, l'equip de desenvolupament estarà format exclusivament per l'autor d'aquest.

Product Owner

El *Product Owner*, representa la veu del client. S'assegura que el resultat del treball de l'*equip de desenvolupament* s'adequa des de la perspectiva de

negoci. Escriu “històries d’usuari”, les prioritza i les col·loca en el *Product Backlog*.

En aquest projecte, la funció del *Product Owner* serà encarnada per el director del departament d’informàtica.

Rols auxiliars

Són aquells que no tenen un intervenció freqüent en el procés *Scrum*. Un dels aspectes claus en qualsevol desenvolupament àgil, és el d’involucrar en el procés a usuaris, experts del negoci i *stakeholders*.

Els Stakeholders (clients, proveïdors, comercials, ...) Fan possible la realització del projecte, són els que obtindran el seu benefici acordat que justifica el projecte. Només intervenen en el procés durant la revisió dels *Sprints*.

Un altre rol auxiliar important és el *Director*. És l’encarregat de muntar l’equip i l’entorn on es desenvoluparà el projecte.

3.1.2 Sprint

El *Sprint* és el període durant el qual es realitza l’increment del producte. Sol tenir una durada fixa i determinada per l’equip de desenvolupament. Fixar la durada facilita a l’equip un ritme constant de treball i l’abast dels objectius associats al *Sprint* es respectin per part del client durant la seva execució. Al final de cada *Sprint*, el *Scrum Team* presentarà l’increment aconseguit sobre el producte, que de ser “potencialment lliurable” al client. Es recomana no modificar els objectius del *Sprint* a no ser que els canvis siguin imprescindibles per assegurar l’èxit del projecte.

Com ja hem mencionat, en el projecte diferenciem dues etapes en el que respecte a la disponibilitat d’hores de dedicació al projecte. Per tant el temps emprat en els diferents *Sprints* i la seva durada han variat al llarg del desenvolupament.

Durant la primera etapa, acordem fixar la durada dels *Sprints* en períodes de 3 setmanes amb un dedicació setmanal de 14 hores. Reservem un marge de

temps per dedicar al Daily Scrum i a possibles reunions, ens quedem amb una dedicació de 40 hores/Sprint.

A la segona etapa, amb la implicació total de l'empresa, el nombre d'hores de dedicació setmanal creix fins a 40 hores. Mantenim la durada del Sprint en tres setmanes. Descomptant el temps per el Daily Scrum i possibles reunions, ens quedem amb una dedicació de 110 hores/Sprint.

3.1.3 Reunions de Scrum

Daily Scrum

Durant l'*Sprint* es realitza un reunió diària sobre l'estat del projecte, normalment al començament de la jornada. Ha de seguir unes regles concretes:

- La reunió ha de començar puntualment.
- És oberta a tothom, però només hi té veu el *Scrum Team*.
- Durada fixa de 15 minuts.
- Tots els assistents han d'estar drets.
- La reunió s'ha de fer al mateix lloc i hora durant tot l'*Sprint*.

Durant la reunió cada membre de l'equip contesta a tres preguntes:

- Que has fet des de la darrera reunió?
- Que faràs fins la propera?
- Quins problemes t'impedeixen arribar als teus objectius?

En el nostre cas, la *Daily Scrum* no ha sigut pròpiament una reunió diària, sinó més aviat una recopilació a nivell individual de les tasques que s'han dut a terme diàriament i quines estaven previstes de realitzar la pròxima sessió.

Degut a no poder tenir-hi una dedicació diària durant la primera etapa del projecte, el *Daily Scrum* m'ha permès recordar com havia deixat el projecte i amb que havia de seguir treballant a l'hora de reprendre'l.

Scrum de Scrums

Es realitza diàriament, normalment després del *Daily Scrum* i tindrà una durada i forma similar a aquesta.

- Les reunions permeten als grups d'equips que treballen en un mateix projecte discutir el seu treball, especialment solapaments i integracions.
- Només hi assistirà un representant de cada equip.

Les preguntes a respondre durant la reunió són:

- Que ha fet l'equip des de la darrera reunió?
- Que farà l'equip fins que ens tornem a trobar?
- Hi ha algun problema que endarrereixi o destorbi l'equip?
- Estàs apunt de posar quelcom en el camí d'un altre equip?

En el nostre cas, no s'han format grups dins l'equip de desenvolupament, per tant no s'han realitzat aquest tipus de reunions.

Sprint Planning Meeting

Com el nom indica, serveix per planificar el *Sprint* que s'iniciï. Per tant es durà a terme cada vegada que es comenci un nou *Sprint* i es decideix:

- Prioritzar la feina assignada al *Sprint* actual amb el *Product Owner*.
- L'equip de desenvolupament estima el temps necessari per realitzar cada element assignat al *Sprint*.
- En funció de l'esforç disponible (hores * membres equip * durada sprint) es selecciona la feina que es farà.
- Es comunica la decisió al *Product Owner*.
- Vuit hores com a límit per *Sprints* d'un mes

Sprint Review Meeting

Es realitza al finalitzar el cicle de Sprint i amb els següents objectius:

- Revisar els objectius que s'han pogut aconseguir i els que no.
- Presentar els resultats obtinguts de l'increment dels producte als interessats. Normalment amb una demostració.

Sprint Retrospective

Es realitza després de cada cicle de *Sprint* i té una durada de quatre hores, tot l'equip de desenvolupament opina sobre com ha funcionat l'equip durant el *Sprint*. L'objectiu de la reunió és identificar els aspectes positius i els que s'han de millorar per optimitzar el rendiment de l'equip.

En el nostre cas, les reunions de *retrospectiva de Sprint*, no tenen sentit.

3.1.4 Documentació

Product Backlog

És un document d'alt nivell per tot el projecte. Conté descripcions genèriques dels requeriments, funcionalitats i problemes a solucionar prioritzats segons el criteri del *Product Owner*.

S'anirà construint durant els *Sprints*. És públic i tothom pot fer-hi aportacions, tot i que només pot ser modificat per el *Product Owner*.

Conté estimacions d'alt nivell, tant del valor per el negoci de cada element com de l'esforç necessari per realitzar-los. Aquestes estimacions, ajuden al *Product Owner* a anar reajustant les prioritats i les assignacions als *Sprints*.

Sprint Backlog

És un document detallat on es defineixen les tasques a realitzar per aconseguir els objectius assignats a el *Sprint* actual. Les tasques es detallen per a que la seva durada sigui petita (menor de dos dies). Aquest refinament dels requeriments del *Sprint* pot derivar en la variació del esforç necessari per assolir l'objectiu assignat a el *Sprint*. Aquestes tasques no són assignades explícitament, sinó que els mateixos membres del *Scrum Team*, les van prenent del *Sprint Backlog* tal com es van realitzant.

Burn down

És una gràfica que mesura la quantitat de requeriments del *Product Backlog* assignats al *Sprint* actual que estan pendents de ser finalitzats.

S'obté dibuixant en un quadrant (Hores x Dies) una "recta ideal" que comença a (Hores Sprint i Zero) i acaba a (Zero, Dies Sprint). Cada dia del Sprint es revisa la feina aconseguida i es van pintant els "punts reals" de la (#Hores Pendents, #Dia Actual). La "recta real" s'obté unint els "punts reals" obtinguts. Ens permetrà anticipar diàriament quina quantitat de la feina planificada per el *Sprint* es podrà assolir realment.

4 Marc de treball i conceptes previs

4.1 Entorn de treball

L'entorn de treball serà el departament d'informàtica de l'empresa Hermes Comunicacions, situada a l'emblemàtic edifici de *La Farinera Teixidor* de Girona.



La Farinera Teixidor és un edifici modernista projectat per *Rafael Masó i Valentí* l'any 1910 al carrer Santa Eugenia, 42, de Girona. Va ser reformat l'any 1999 i actualment és la seu del diari *El Punt Avui*.

El departament d'informàtica es troba instal·lat a la segona planta de l'edifici i està format per 9 treballadors organitzats en dues seccions, sistemes i programació.

4.2 Eines de treball

Per tal de poder realitzar el desenvolupament de l'aplicació, l'empresa m'ha facilitat un ordinador Mac. Concretament un Mac Pro amb processador Intel i 2GB de RAM.

Per realitzar els testos necessaris en dispositiu real, l'empresa em facilitat un *iPad 2* amb connectivitat 3G, capacitat de 16GB i amb la darrera versió del sistema iOS. En aquests moments la 5.1.1.



A nivell personal apporto un *iPhone 4* amb connectivitat 3G i capacitat de 16GB i un iPad de la primera generació. Tots dos amb la versió 5 de iOS per poder realitzar les proves en altres tipus de dispositius.

4.3 Conceptes previs

4.3.1 Que és iOS?

iOS és el nom amb el que es coneix actualment el sistema operatiu dissenyat per Apple per els seus dispositius d'última generació (iPhone, iPod Touch, iPad i Apple TV).

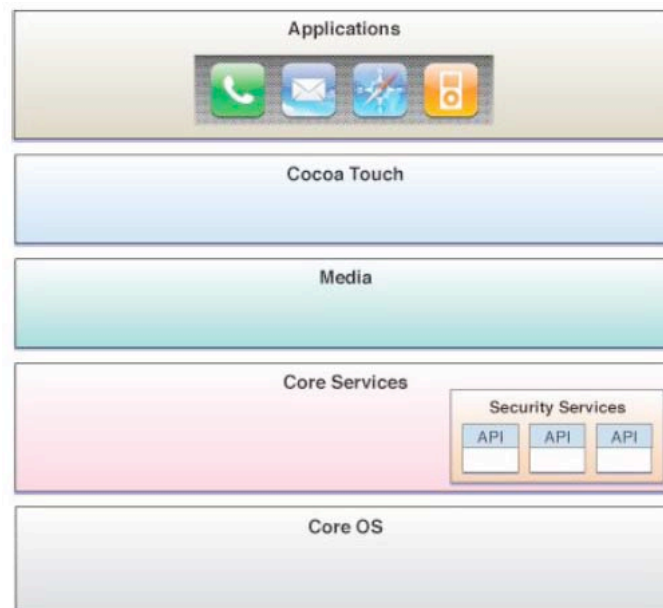
El desenvolupament inicial es base en el nucli del sistema operatiu Mac OS X (DarwinBSD), però a diferència del sistema original, la interfície gràfica ha estat redissenyada per adaptar-se a les necessitats d'un entorn mòbil, on les necessitats de l'usuari són lleugerament diferents.

La nova interfície gràfica està basada en el sistema de manipulació directa, utilitzant gestos multi-tàctils. Els elements de control bàsics consisteixen en botons, interruptors i "lliscadors".

En aquells moments degut a l'ús exclusiu en el telèfon d'Apple, se li dona el nom de iPhone OS. Però amb l'aparició de nous dispositius i la publicació de noves versions, el sistema es rebateja com a iOS.

4.3.2 Arquitectura per capes

El sistema iOS, està basat en una arquitectura per capes. Podríem definir quatre capes.



Esquema arquitectura per capes iOS

Core OS

És nucli del sistema. Com ja hem comentat, és una modificació del kernel de Mac OS X.

Core Services

Aquesta capa, inclou serveis essencials com accedir al sistema de fitxers, tipus de dades de baix nivell, sockets de xarxa, entre d'altres. Igual que el Core OS són escrites en C.

Media

A mesura que ens desplacem a nivells superiors, trobem tecnologies mes avançades. Utilitzen una barreja de interfícies basades en C i Objective-C.

Conté tecnologies fonamentals utilitzades per donar suport al dibuix 2D i 3D, audio i vídeo. Inclou les tecnologies basades en C *OpenGL*, *Quartz* i *Core Audio* i el motor d'animació *Core Animation*.

Cocoa Touch

Aquesta capa, conté els *frameworks* que ens proporcionen la infraestructura fonamental utilitzada per qualsevol aplicació.

Alguns d'aquestes frameworks essencials són:

- Framework UIKit: Proporciona la infraestructura visual per a la nostre aplicació. Finestres, vistes, controls, taules i els controladors que gestionen aquets objectes.
- Framework Foundation: Proporciona suport a la orientació a objectes per col·leccions de dades, gestió de fitxers, operacions de xarxa, entre d'altres.
- A aquest mateix nivell trobem altres frameworks que ens proporcionen accés a informació de l'agenda, contactes, fotografies, acceleròmetre i altres característiques hardware del dispositiu.

L'origen de qualsevol aplicació és la capa Cocoa Touch i concretament el Framework UIKit. Quan es planteja quines tecnologies addicionals són necessàries, es recomana començar per els frameworks a més alt nivell i anar baixant a mesura que sigui necessari.

4.3.3 Desenvolupament per iOS

Les aplicacions instal·lades, resideixen a la "Home" de l'usuari, juntament amb les altres aplicacions del sistema. Una vegada s'inicia una aplicació, aquesta és la única que es troba corrent en el sistema a part del *kernel* i alguns dimonis.

Durant l'execució l'aplicació ocupa la totalitat de la pantalla i és el focus d'atenció de l'usuari. Quan l'usuari prem el botó de "Home", l'aplicació que en segon pla i el sistema torna a la pantalla de "Home".

Com ja hem comentant, la manera en que els usuaris interactuen amb les aplicacions és diferent de com ho fan en Mac OS X, per tant la manera de desenvolupar les aplicacions també ho serà.

En una aplicació per iOS no existeix el concepte de finestres de documents, per mostrar el contingut. Tota la informació es mostra en una única finestra. Això a portat a la creació de vistes i controls per tal de presentar aquesta informació de manera organitzada.

La manera de tractar els “events” també és lleugerament diferent a com ho fem en una aplicació d'escriptori. S'introdueix la idea dels *Touch Events* (“events” tàctils). Un event d'aquest tipus pot esdevenir en qualsevol moment i en combinació amb un o més “events” addicionals. Els gestos, poden ser identificats per detectar interaccions simples amb el contingut, com seleccionar o arrastrar elements o gestos més complexes com els realitzats per fer zoom o rotar un element.

Més enllà de considerar l'estructura bàsica de l'aplicació, s'ha de pensar en com l'utilitzarà l'usuari. Les aplicacions ha de ser clares i centrades en que necessita l'usuari en cada moment. Els usuari d'aquests dispositius necessiten accedir a la informació de forma ràpida i sense haver de navegar entre capes i capes de l'aplicació. Un disseny simple que destaquí la informació que necessita l'usuari és essencial.

4.3.4 Eines de desenvolupament: iOS SDK

iOS Software Developer Kit, és un paquet per a desenvolupadors, preparat per Apple i distribuït gratuïtament perquè qualsevol programador pugui desenvolupar les seves pròpies aplicacions per a iOS. Incorpora totes les interfícies, eines i recursos necessaris per desenvolupar aplicacions iOS des de qualsevol ordinador Mac.

Alguns dels components clau del SDK són:

XCode Tools

Proporciona les eines que permeten el desenvolupament d'aplicacions iOS. Està compost per els següents elements:

- **XCode:** Entorn de desenvolupament integrat que gestiona els diferents projectes d'aplicació. Permet editar, compilar, llançar i depurar el codi. Està integrat amb moltes altres eines i és l'aplicació d'us principal durant el desenvolupament.
- **Interface Builder:** Una eina que s'utilitza per construir la interfície d'usuari de manera gràfica. Els objectes de la interfície utilitzats són guardats en un fitxer amb un format especial i carregats en temps d'execució.
- **Instruments:** Una eina d'anàlisi de comportament i “debugging”. S'utilitza per reunir informació sobre el funcionament de l'aplicació en temps d'execució i detectar problemes potencials.

iOS Simulator

És una aplicació que simula un dispositiu iOS (iPad o iPhone), permeten provar les aplicacions de manera local en qualsevol ordinador Macintosh.

iOS Reference Library

El SDK inclou documentació de referència per iOS per defecte i permet descarregar versions més completes, incloent exemples de codi.

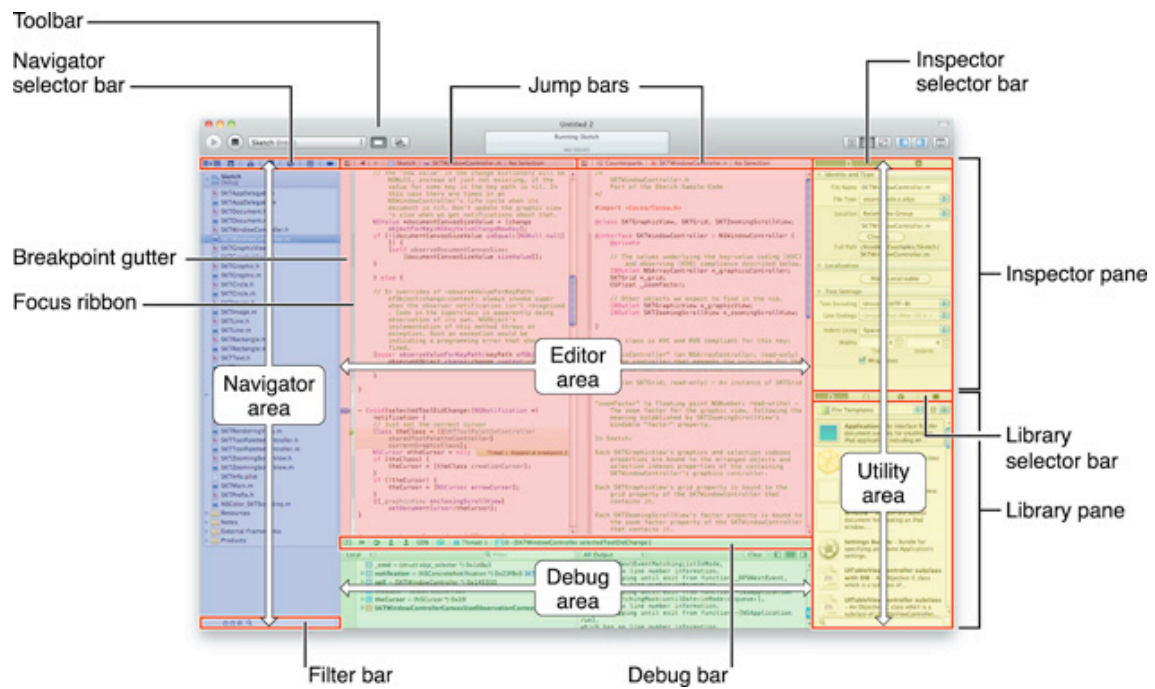
Tot i que el SDK proporciona el software necessari per desenvolupar aplicacions, i que XCode i Instruments permeten interactuar directament amb un dispositiu adjunt per executar i depurar el codi en un terminal físic. El desenvolupament en un dispositiu real, requereix adquirir una llicència de desenvolupament “Apple Developer Program” i configurar el terminal per aquest propòsit. La llicència té un cost de 99\$ anuals.

XCode

És un entorn de desenvolupament que disposa de totes les eines necessàries per crear i gestionar projectes iOS i els seus fitxers, compilar el codi en un executable, llançar i depurar el codi, bé en el simulador o en un dispositiu real. XCode incorpora un gran nombre de característiques per fer més fàcil el desenvolupament d'aplicacions iOS, com són:

- Un sistema de gestió de projectes per definir productes software
- Un sistema d'edició de codi que inclou característiques com la sintaxis remarcada a color, auto completat de codi i indexat.
- Un visor de documentació avançat amb sistema de cerca.
- Compiladors GCC per C, C++, Objective-C, Objective-C++ i Objective-C 2.0.
- Depuració a nivell de font utilitzant GDB.
- Compilació predictiva que accelera el procés.
- Suport per “snapshots” de projectes, que permeten una forma lleugera de gestionar el codi font.
- Eines d'anàlisi d'aplicacions.

Per crear una aplicació iOS es comença per crear un nou projecte XCode. Un projecte, gestiona tota la informació associada a l'aplicació, incloent els fitxers font, la configuració de compilació i la resta d'elements.



Interfície XCode

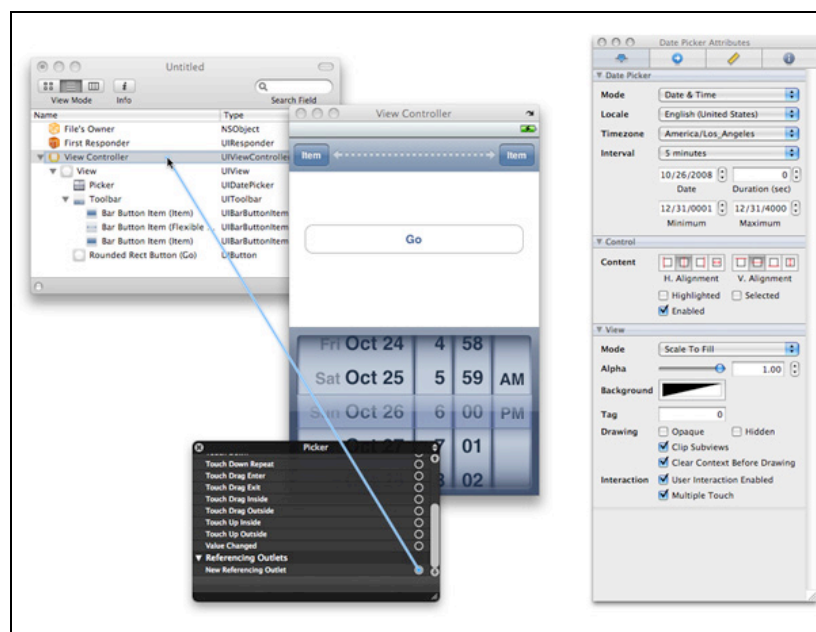
Interface Builder

És l'eina que utilitzem per dissenyar visualment la interfície d'usuari de la nostra aplicació.

Ens permet crear les diferents vistes, arrossegant diferents components preconfigurats. Inclou controls com caixes de text, botons, calendaris o interruptors.

Una vegada situats els elements podem modificar les seves propietats gràcies amb *Inspector*. També ens permetrà definir la relació entre aquests components i el nostre codi.

Aquesta informació la guardarem en fitxers *Nib* o en *Storyboards*, que contindran tota la informació necessària perquè el framework *UIKit* pugui recrear els objectes en temps d'execució de l'aplicació.



Captura Interface Builder

Amb el llançament de la versió 4 de XCode, Apple integra les diferents eines Interface Builder, Instruments i XCode en una sola aplicació.

4.3.5 Control de versions

Per tal de gestionar el codi que anem generant, utilitzem *Subversion*. Un sistema de control de versions de codi lliure. Hem escollit aquest perquè és el que utilitzem habitualment a l'empresa.

A mesura que el projecte avança, ens adonem que per tal de gestionar més eficientment les diferents apps personalitzades que anem generant, hem de dividir el projecte en dues parts, una part comuna per totes les versions i una part específica que contindrà els fitxers i configuracions que variïn d'una app a l'altra.

Per tant a l'hora de generar una nova aplicació, primer farem un “check-out” del repositori on tinguem la part comuna de l'aplicació. Per seguidament crear un nou repositori que contindrà la part específica.

Aquesta estructura ens permet mantenir molt còmodament el codi de l'aplicació sense obligar-nos a retocar totes les aplicacions una per una.

4.3.6 Com apliquem Scrum

Tot i que ja ho hem anat comentat durant l'explicació de la metodologia, volem deixar clar com hem adaptat i aplicat *Scrum* en el projecte.

En una primera reunió amb en Marc Massot (que desenvolupa algunes de les funcions del *Scrum Master* i tutoritza el projecte) i en Josep Madrenas (que desenvolupa el rol de *Product Owner*) recollim quines són les funcionalitats que ha de tenir el producte i les organitzem en el *Product Backlog*.

Una vegada tenim prioritzades les històries d'usuari del *Product Backlog*, escollim les més prioritàries i les assignem a un *Sprint*. L'equip de desenvolupament format únicament per l'autor del projecte va desenvolupant les diferents tasques en un període que anomenem *Sprint* d'una durada d'aproximadament 3 setmanes. Cal destacar que durant el transcurs del projecte, les hores de dedicació durant el *Sprint* han variat.

Per tal de portar un control sobre la feina que hem fet i la que ens falta per fer, utilitzem el *Sprint Backlog*. En el nostre cas, no és res més que una pissarra on

disposem de les històries assignades a el *Sprint* actual, detallades en tasques més concretes i organitzades en tres columnes (pendents, en curs o finalitzades).

D'aquesta manera, al reprendre la feina, sabem en que estàvem treballant i quina feina ens queda de fer. També portem un control aproximat del temps dedicat a cada tasca.

Un cop finalitzat el *Sprint*, ens reunirem amb el client i el tutor, per fer una demostració del producte. Cal destacar que una de les premisses de *Scrum*, és tenir en tot moment un versió funcional del producte.

En aquestes reunions de demostració, el client ens guiarà cap a on està interessat que evolucioni el projecte. I això farà que apareguin noves històries o funcionalitats que afegirem al *Product Backlog*.

Anirem fent iteracions de Sprint fins a finalitzar totes les històries del *Product Backlog*.

5 Planificació i requisits del sistema

La primera reunió amb el *Product Owner* ens deixa una llista de requeriments, que formen el *Product Backlog*.

Per a cada requisit, realitzem una estimació del seu valor dins la totalitat del projecte i del temps necessari per desenvolupar-la.

Num.	Història d'usuari	Prioritat	Temps
1	Identificar usuari al Quiosc	40	30 h.
2	Descarregar portades publicacions del dia	100	20 h.
3	Mostrar portades publicacions del dia	100	12 h.
4	Descarregar una publicació concreta	80	20 h.
5	Comprovar si usuari pot descarregar publicació	40	12 h.
6	Descarregar una publicació antiga	40	30 h.
7	Comprovar si hi ha publicacions descarregades	30	6 h.
8	Eliminar publicacions descarregades	30	10 h.
9	Mostrar una publicació descarregada	80	26 h.
10	Navegació per seccions de publicació	20	20 h.

Aquesta llista inicial, no conté detallats tots els requeriments, simplement ens permet identificar-los de manera superficial. Els detalls de cada requeriment van apareixent a mesura que són seleccionats per el següent *Sprint*.

Els temps estimats per a cada funcionalitat, no inclouen el temps d'aprenentatge necessari per la seva implementació.

Tot i haver estudiat les tecnologies necessàries per començar a treballar en el desenvolupament del projecte, la manca d'experiència amb aquestes fan que les estimacions de temps per a cada història d'usuari siguin poc realistes.

6 Estudis i decisions

6.1 Protocol XML-RPC

Una de les decisions importants que hem pres durant el desenvolupament del projecte, ha estat la de abandonar les simples peticions HTTP que fèiem servir per obtenir els XML amb els continguts necessaris per el funcionament de l'aplicació.

En substitució, escollim el protocol XML-RPC. És un protocol de crida a procediments remots (RPC) que fa servir XML per codificar les crides i HTTP com a mecanisme de transport.

L'adopció del protocol XML-RPC, tot i que ens obliga a desenvolupar *plugins* al web per poder obtenir la informació, ens proporciona una manera molt més còmode i segura de comunicar-nos.

Tot i això, decidim protegir les crides a XML-RPC amb autenticació via HTTP.

6.2 Lliberies externes

The Cocoa XML RPC Framework

Aquest framework simple i lleuger, escrit en Objective-C i compatible amb Mac i iOS, ens permet desenvolupar un client XML-RPC i comunicar-nos fàcilment en aquest protocol.

Font: <https://github.com/eczarny/xmlrpc>

OpenUDID

Aquesta llibreria ens permet generar un identificador únic per el dispositiu on l'estem executant. Es basa en l'obtenció de l'adreça mac del dispositiu per fer-ho. El SDK de iOS disposa d'una funció similar però l'ha deixat obsoleta en les darreres actualitzacions.

Font: <https://github.com/ylechelle/OpenUDID>

SSZipArchive

La llibreria *SSZipArchive* ens facilita la gestió de fitxers ZIP. Ens permetrà tant comprimir com descomprimir aquest tipus de fitxers.

Font: <https://github.com/samsoffes/ssziparchive>

Reachability

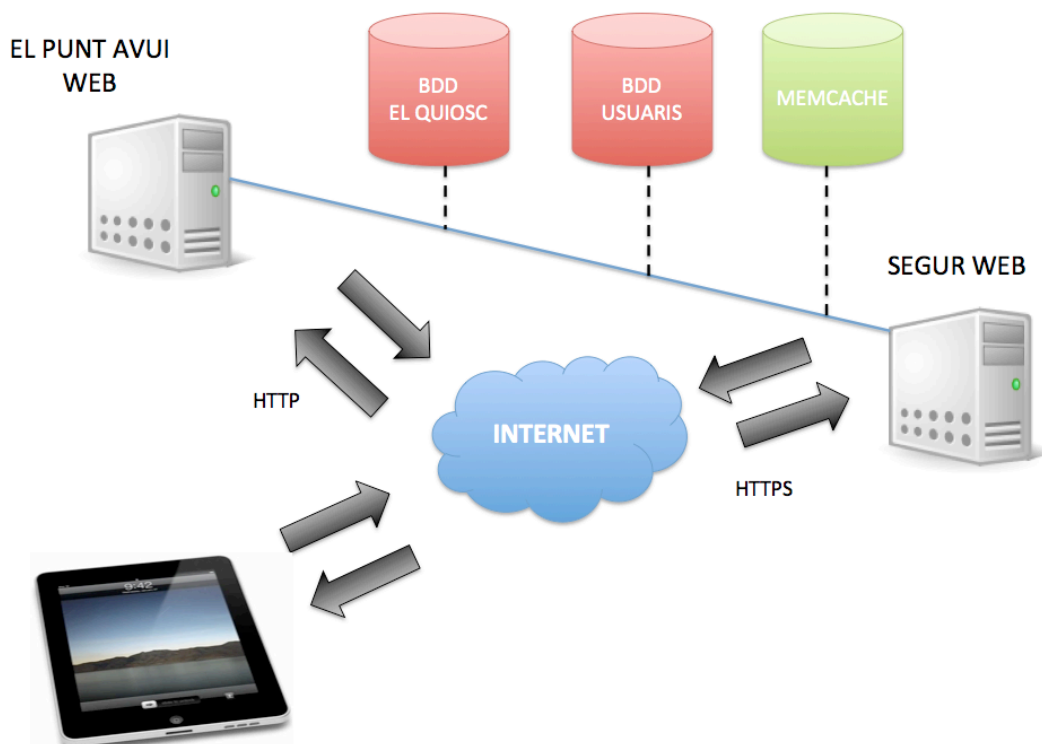
Aquesta llibreria, ens permetrà controlar de manera més senzilla que amb les llibreries que ens proporciona Apple, si disposem o no de connexió a internet. També ens permetrà diferenciar entre la connexió 3G i la WiFi.

Font: <https://github.com/tonymillion/Reachability>

7 Anàlisi i disseny del sistema

Degut a la metodologia utilitzada, la major part de l'anàlisi i del disseny s'ha anat duent a terme en el transcurs dels diferents *Sprints*.

7.1 Esquema de connexions



Disposem de dos servidors, que corren un *Apache* amb el *Joomla* que té instal·lats els dos *plugins* que atendran les peticions XML-RPC.

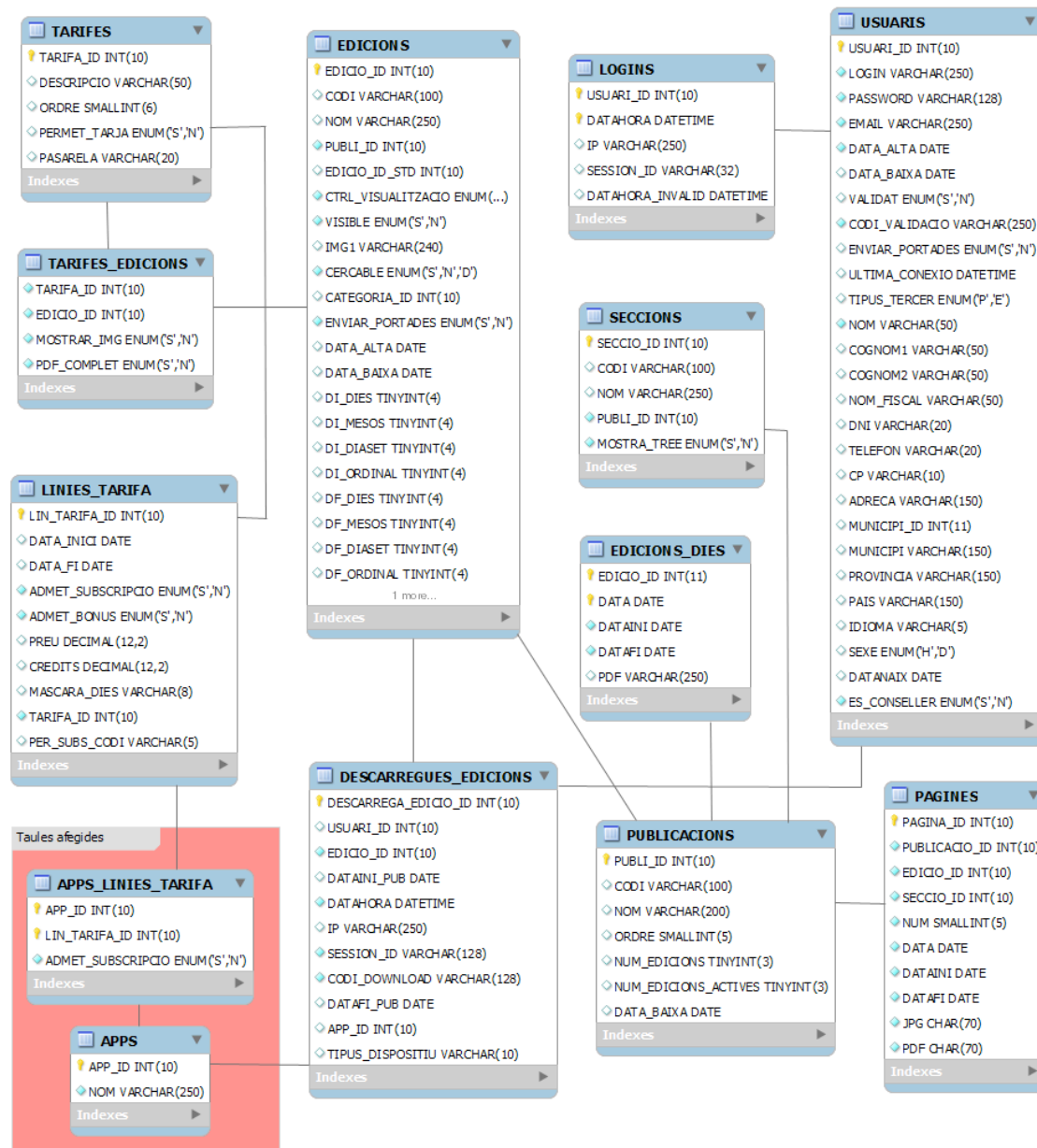
Les peticions que fem a *SEGUR*, que són totes les relacionades amb el tema d'usuaris i identificació, seran sempre amb *HTTPS* per millorar-ne la seguretat.

Observem que els dos servidors tenen accés a les dades de la base de dades de El Quiosc.

El sistema de *Memcache*, ens permetrà comunicar-nos entre les dues màquines amb la generació de fitxers a cache, per tal de garantir la seguretat entre les peticions que fem a una màquina i l'altre.

7.2 Esquema d'entitats BDD El Quiosc

A continuació observem l'esquema de la base de dades de El Quiosc.

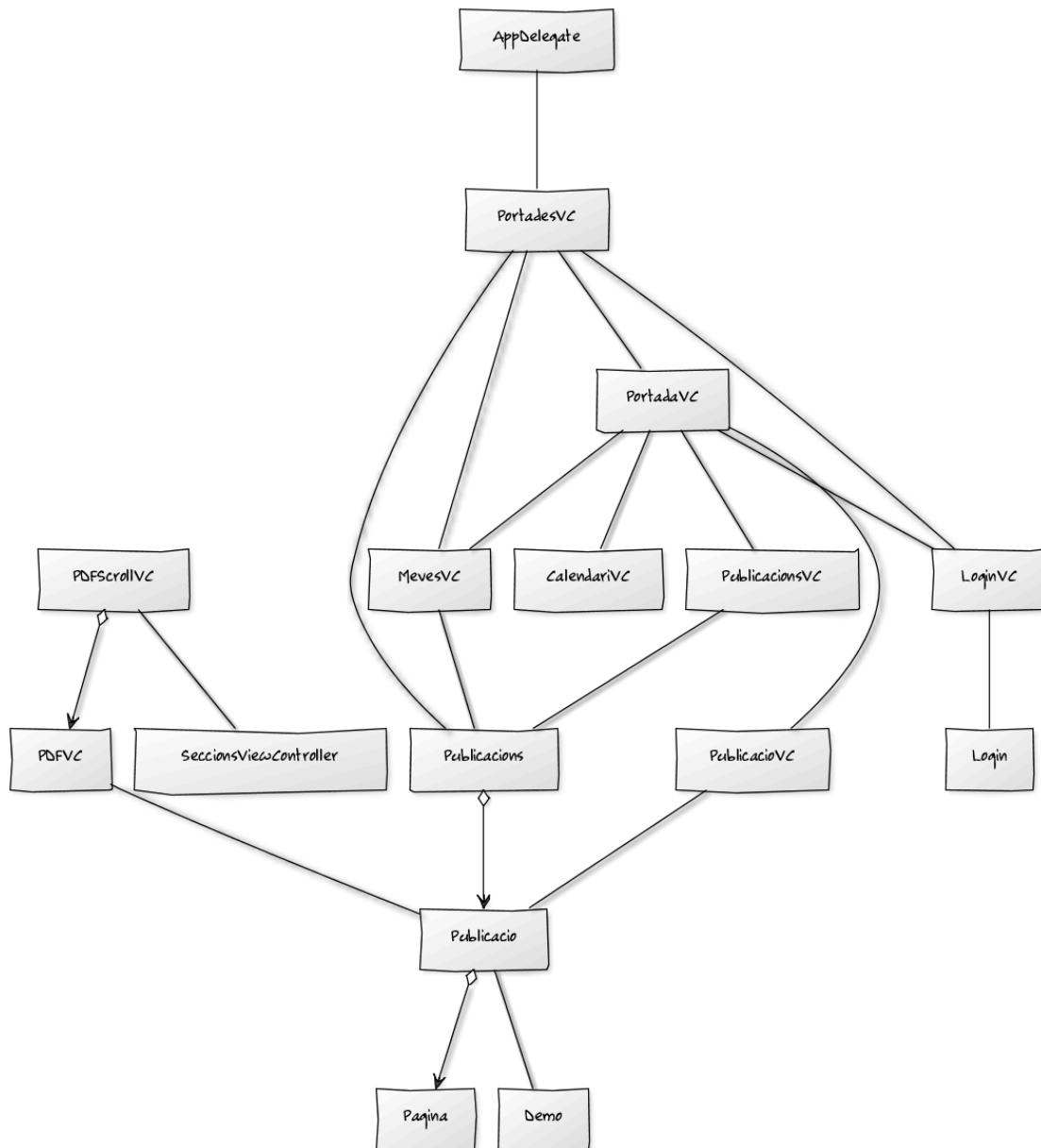


Durant el desenvolupament del projecte, hem afegit les taules que fan referència a la APP ID (APPS i APPS_LINIES_TARIFA).

Les noves entitats ens permeten obtenir les PUBLICACIONS, en funció de l'aplicació i usuari donats.

Això ens permet carregar només les publicacions necessàries per l'aplicació o en el cas de la identificació d'usuari, conèixer les publicacions a les que ha de tenir accés.

7.3 Diagrama de classes de l'aplicació



Cada pantalla de l'aplicació és controlada per un o més controladors (classes acabades amb VC).

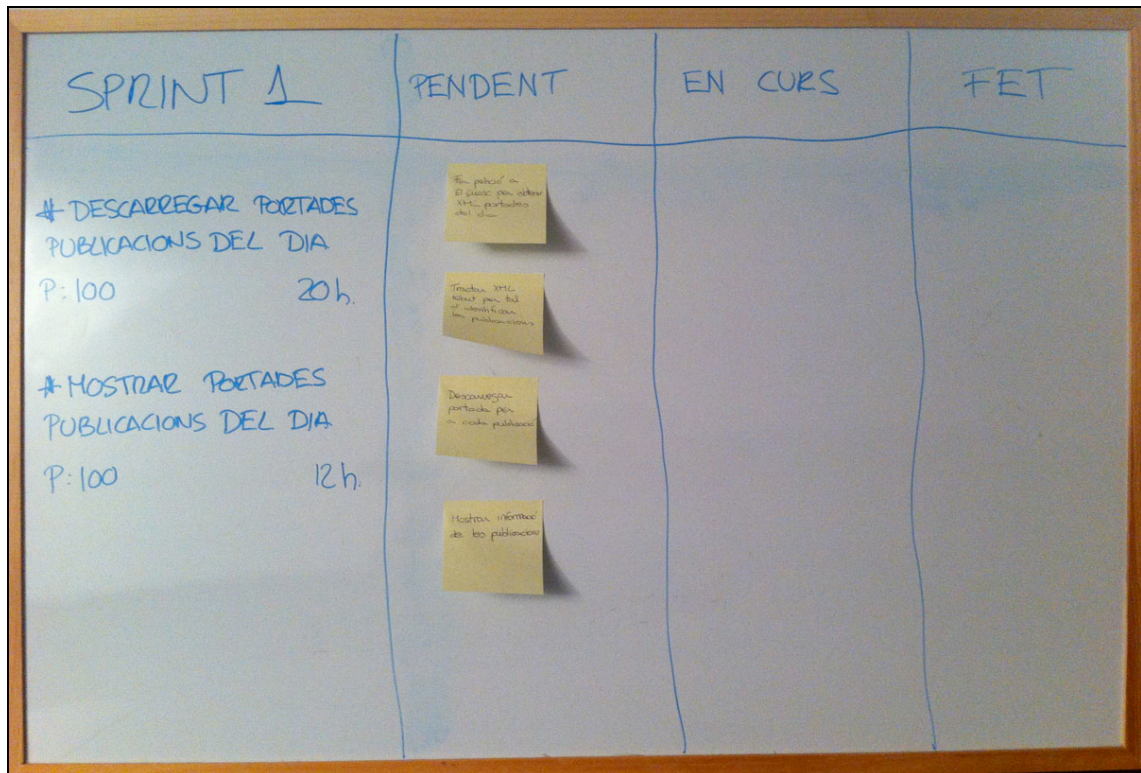
La pantalla amb el llistat de publicacions del dia, està controlada per *PortadesVC*, mentre que la pantalla amb les darreres publicacions d'un mateix tipus, està controlada per *PortadaVC*, que utilitza dos sub-controladors *PublicacioVC* i *PublicacionsVC*.

Els controladors de vista, utilitzen models, que representen per exemple una publicació o una pàgina.

8 Implementació i proves

8.1 Sprint 1

- Descarregar portades publicacions del dia
- Mostrar portades publicacions del dia



Imatge pissarra inici Sprint 1

8.1.1 Històries d'usuari detallades

- Fer petició a “El Quiosc” per obtenir XML portades del dia
- Tractar XML rebut per tal d'identificar les publicacions
- Per cada publicació descarregar la seva portada
- Mostrar portades “El Quiosc”
- El Quiosc: XML format correcte

Tot i que el temps estimat en les dues histories d'usuari seleccionades suposava menys temps del dedicat a el *Sprint*, vam trobar convenient no sobrecarregar-lo, per poder assolir els objectius amb certa tranquil·litat.

8.1.2 Descarregar portades publicacions del dia

Consisteix en obtenir del web *El Quiosc*, la informació referent a les publicacions actuals i les seves corresponents portades.

Per obtenir la informació, fem una petició HTTP al web *El Quiosc*. La resposta obtinguda és un XML amb la informació referent a les edicions vigents en el dia d'avui.

Adreça on fem la petició per obtenir les publicacions vigents:

<http://quiosc.elpunt.cat/Quiosc.php?do=edicionsXML>

El *Foundation framework* ens proporciona la classes *NSURLConnection*, que ens permet fer la petició i llegir-ne la seva resposta. I la classe *NSXMLParser* que ens permet interpretar la resposta en format XML obtinguda.

Un exemple del XML obtingut:

```
<?xml version="1.0" encoding="UTF-8"?>

<nodo label="El Punt Avui" icona="1">

    <nodo label="Nacional" icona="2"
dirurl="/&amp;edicio_id=30&amp;data=2012-05-31"/>

    <nodo label="Comarques Gironines" icona="2"
dirurl="/&amp;edicio_id=29&amp;data=2012-05-31"/>

</nodo>

<nodo label="El 9 Esportiu" icona="1">

    <nodo label="El 9 Esportiu" icona="2"
dirurl="/&amp;edicio_id=31&amp;data=2012-05-31"/>

</nodo>
```

```

<nodo label="Presència" icona="1"
dirurl="/&edicio_id=15&data=2012-05-25"/>

<nodo label="L'econòmic" icona="1"
dirurl="/&edicio_id=22&data=2012-05-26"/>

<nodo label="Sortim" icona="1"
dirurl="/&edicio_id=19&data=2012-05-25"/>

<nodo label="Catalonia Today" icona="1"
dirurl="/&edicio_id=16&data=2012-05-31"/>

<nodo label="Cultura" icona="1"
dirurl="/&edicio_id=28&data=2012-05-31"/>

<nodo label="B30" icona="1"
dirurl="/&edicio_id=27&data=2012-05-04"/>

```

Una vegada obtingut el XML, el tractem per obtenir la informació de les publicacions.

Per cada “node” obtenim el títol, d’una publicació i una adreça URL, que tractem per obtenir una edició ID i una data de publicació.

Exemple de “node”:

```

<nodo label="Sortim" icona="1"
dirurl="/&edicio_id=19&data=2012-05-25"/>

```

Alguns dels nodes, requereixen un tractament especial ja que “encapsulen” diferents edicions de la mateixa publicació. Aquestes edicions són tractades com a publicacions diferents.

Els títols de les diferents edicions dins una publicació, seran tractats com a descripcions d’aquestes “noves” publicacions generades.

Això ens ha generat una classe “Publicacions”, que conté una col·lecció d’objectes “Publicació”. Aquesta classe “Publicació” disposarà dels atributs obtinguts del XML.

Un cop obtenim totes les publicacions, realitzem una segona petició HTTP a “El Quiosc” per cada una d’elles, per descarregar la imatge corresponent a la seva portada.

La adreça on fem la petició la generem a partir de l’edició ID i la data obtingudes del tractament anterior i ens retornarà una imatge JPEG.

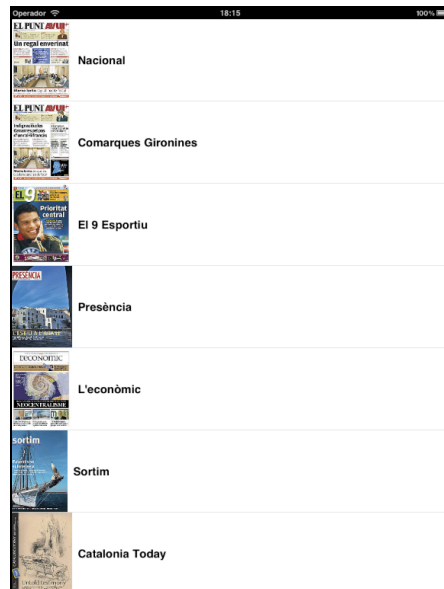
Exemple de URL de portada:

```
http://quiosc.elpunt.cat/Quiosc.php?do=portadesDelDia&forma  
t=jpga&edicio_id=31&data_pub=2012-04-12
```

8.1.3 Mostrar portades publicacions del dia

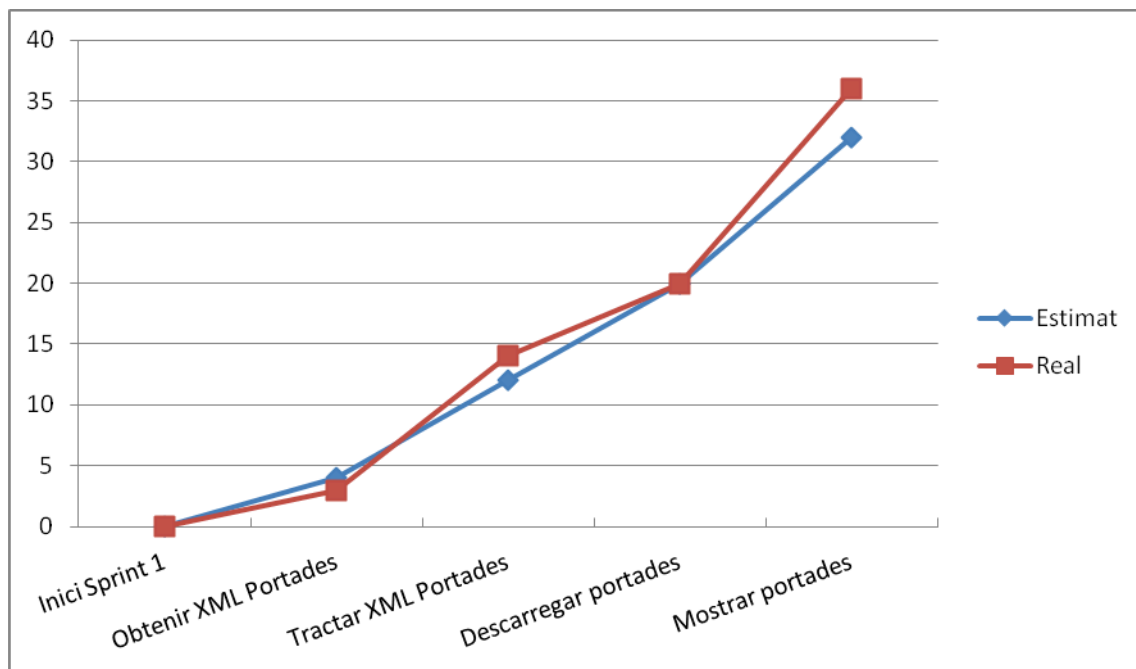
Consisteix en que una vegada tenim les publicacions en memòria, mostrar-les en el dispositiu en forma de llistat detallat amb tota la informació disponible.

Per mostrar la informació, utilitzem una de les classes que ens proporciona el *UIKit Framework*, concretament un *UITableViewController*. Aquesta classe és bàsicament un controlador per gestionar *UITableViews* que són un tipus de vistes genèriques per mostrar informació tabulada. Aquesta vista *UITableView* al mateix temps està formada per unes sub-vistes anomenades *UITableViewCell*s, que representen les cel·les de la taula.



Captura llista portades

8.1.4 Diagrama d'evolució del Sprint



8.1.5 Incidències destacables

Durant el tractament del XML de les publicacions, ens trobem en que el *NSXMLParser* no pot tractar les dades correctament degut a que el XML obtingut no té el format correcta. Per poder continuar amb la interpretació de les dades, hem de realitzar una modificació a “El Quiosc” per tal de corregir el problema. Ho solucionem, creant un nou *template* per l’acció “portadesXML” que ens donarà un XML el format correcta, al rebre el paràmetre “&format=xml”.

8.1.6 Reunió final Sprint

Al finalitzar aquest primer *Sprint* hem aconseguit els objectius marcats, tot i que no estem del tot satisfets amb els resultats obtinguts. Detallem alguns dels problemes detectats:

- Temps de càrrega lent
- Repetim la petició de les dades cada vegada que cal mostrar-les
- Falta informació sobre les publicacions
- Estètica simple

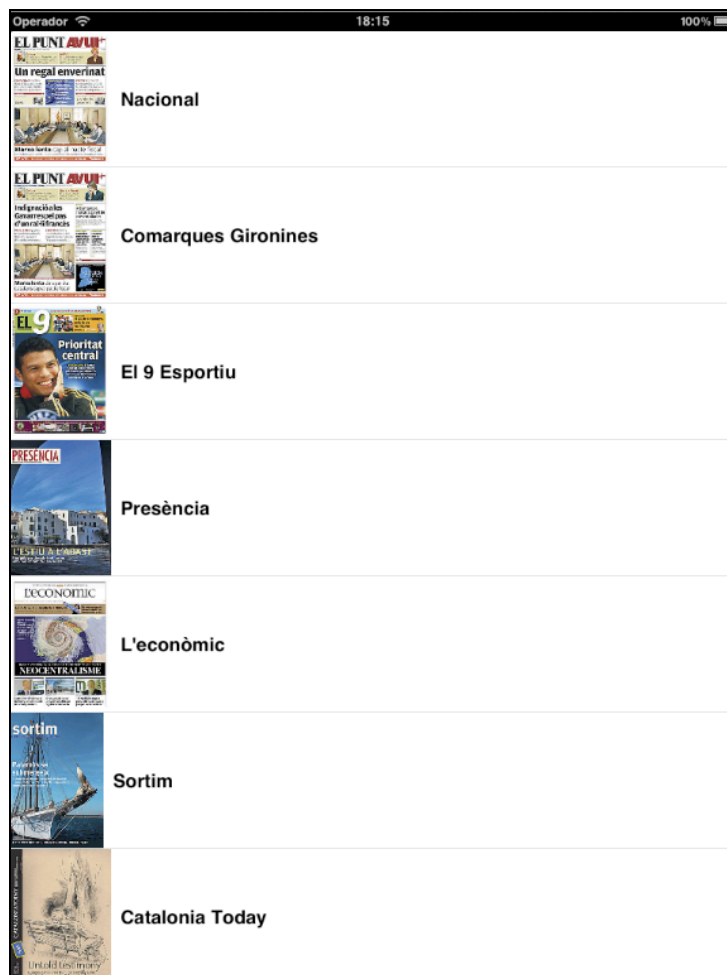
Analitzant aquests resultats ens apareixen noves històries d'usuari que caldrà tractar en els següents *Sprints*.

SPRINT 1	PENDENT	EN CURS	FET
#DESCARREGAR PORTADES PUBLICACIONS DEL DIA P: 100 20h #MOSTRAR PORTADES PUBLICACIONS DEL DIA P: 100 12h			Realitzar un diagrama per visualitzar els dades Realitzar un diagrama per visualitzar els dades Realitzar un diagrama per visualitzar els dades Realitzar un diagrama per visualitzar els dades Realitzar un diagrama per visualitzar els dades

Imatge pissarra fi Sprint 1

8.1.7 Demostració del producte al client

A la captura veiem la llista de publicacions que hem obtingut de El Quiosc i que mostrem en una taula.

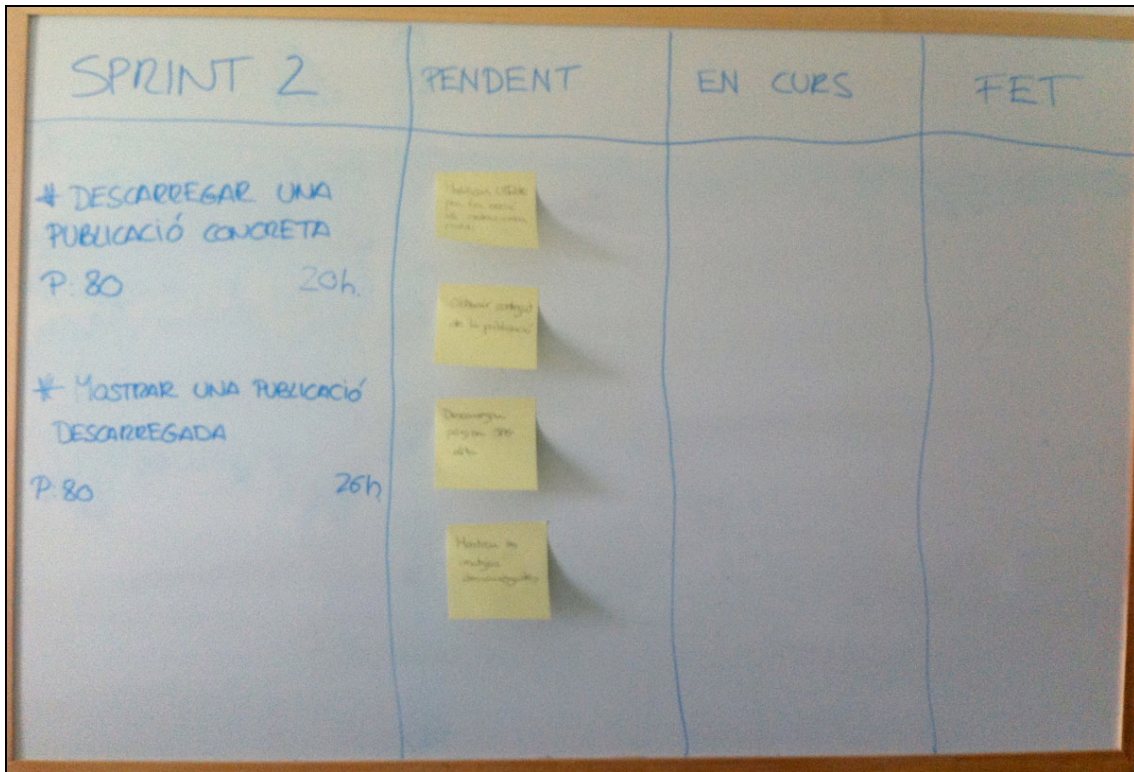


	Nacional
	Comarques Gironines
	El 9 Esportiu
	Presència
	L'econòmic
	Sortim
	Catalonia Today

Captura de la versió lliurada al client al finalitzar el 1r Sprint

8.2 Sprint 2

- Descarregar una publicació concreta
- Mostrar una publicació descarregada



Imatge pissarra inici Sprint 2

El temps estimat de les dues histories seleccionades per aquest *Sprint*, era de 46 hores, superant el temps de duració del *Sprint*. Tot hi així degut a la importància de les dues històries i seguint l'evolució lògica del projecte decidim tirar-les endavant.

8.2.1 Histories d'usuari detallades

- Modificar *UITableView* per fer una acció al seleccionar una cel·la
- Obtenir informació del contingut de la publicació
- Descarregar pàgina JPG d'alta resolució
- Mostrar imatges descarregada

8.2.2 Descarregar una publicació concreta

El primer pas consisteix en modificar la *UITableViewController* que utilitzem per mostrar les publicacions, per tal que al seleccionar una publicació en concret, faci una petició al *El Quiosc* per obtenir la informació sobre el contingut d'aquesta.

Dins el nostre controlador de *Publicacions UITableViewController*, implementem la funció *didSelectRowAtIndexPath*, que ens permet definir el comportament de les files al ser seleccionades.

Al seleccionar una publicació, fem una petició *HTTP* amb *NSURLConnection* a *El Quiosc* amb el seu identificador i data i obtenim un XML amb la informació del contingut.

Exemple adreça on fem la petició per obtenir el contingut:

```
http://www.elquiosc.cat/Quiosc.php?do=edicioXML&edicio_id=29&format=alta
```

Un exemple del XML obtingut:

```
<?xml version="1.0" encoding="UTF-8"?>
<edicio>
  <pagina>
    <numero/>
    <seccio/>
    <urljpg>nula.jpg</urljpg>
    <urlpdf/>
  </pagina>
  <pagina>
    <numero>1</numero>
    <seccio>Portada</seccio>
    <urljpg>http://quiosc.elpunt.net/jpga/5ea37d34091e7b11b2c9e8a56b5473ce/4fc8cae0/2012/06/01/DGIR/01DGIR_001A.JPG</urljpg>
```

```

<urlpdf>/Quiosc.php?do=mostrarPDF&pagina_id=734675</urlpdf>

    <urlce></urlce>

</pagina>

<pagina>

    <numero>2</numero>

    <seccio>Països Catalans</seccio>

<urljpg>http://quiosc.elpunt.net/jpga/9f18a347408398ffed46835944
29fc09/4fc8cae0/2012/06/01/DGIR/01DGIR_002A.JPG</urljpg>

<urlpdf>/Quiosc.php?do=mostrarPDF&pagina_id=734676</urlpdf>

    <urlce></urlce>

</pagina>

<pagina>

    <numero>3</numero>

    <seccio>Països Catalans</seccio>

<urljpg>http://quiosc.elpunt.net/jpga/084624ac3043d46de8285f1a12
08fd51/4fc8cae0/2012/06/01/DGIR/01DGIR_003A.JPG</urljpg>

<urlpdf>/Quiosc.php?do=mostrarPDF&pagina_id=734677</urlpdf>

    <urlce></urlce>

</pagina>

....

<pagina>

    <numero/>

    <seccio/>

    <urljpg>nula.jpg</urljpg>

    <urlpdf/>

</pagina>

</edicio>

```

Un cop obtingut el XML, el tractem amb *NSXMLParser* i n'obtenim el numero de pàgines de la publicació, el nom de les seccions de la publicació, a quina secció pertany cada pàgina i la URL per poder obtenir la pàgina en format JPG d'alta resolució.

Això ens genera una nova classe en el nostre codi, la classe "Pagina". Que tindrà atributs com el numero de pàgina, la secció a la que pertany, i la adreça on obtenir la pàgina. La classe publicació, passa a tenir una col·lecció de "Páginas".

Per a cada pàgina, ens cal descarregar la imatge corresponent. Ho fem a través de *NSURLRequest* que ens permet crear una imatge, representada per la classe *UIImage* amb el contingut de la resposta.

8.2.3 Mostrar una publicació descarregada

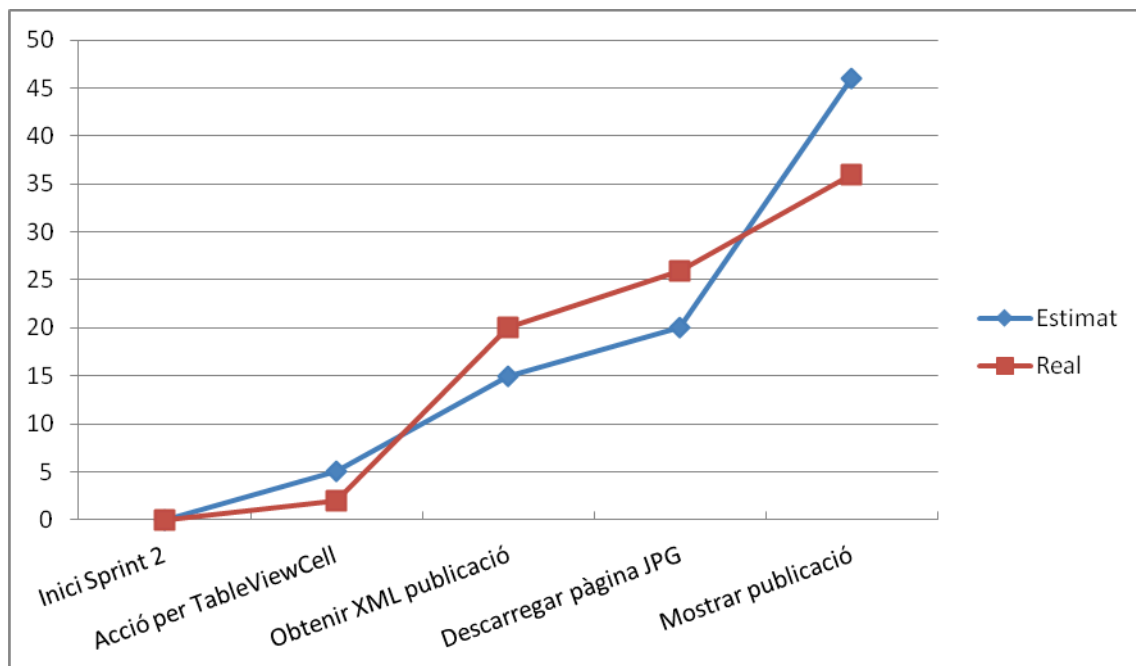
Degut a el temps que hem dedicat a el desenvolupament de la historia anterior i a la necessitat de tenir una versió presentable a el client, durant aquest Sprint aquesta funcionalitat simplement mostrarà una pàgina de totes les descarregades.

Gràcies a la classe *UINavigationController*, podem navegar entre diferents vistes, sense perdre les referències a les *UIViews* que no són visibles.

El *UINavigationController*, ens permet definir un controlador principal o *rootViewController* que en el nostre cas serà el controlador de *Publicacions desenvolupat en el primer Sprint (UITableViewController)*.

Un cop seleccionada la publicació, empilem un nou controlador a el controlador de navegació. Aquest nou controlador, que anomenem *Publicacio*, serà un simple *UIViewController* que iniciarà el procés de descarrega explicat en la història anterior. Un cop finalitzada la descarrega de totes les pàgines, mostra en una vista que permet carregar imatges (*UIImageView*) una de les pàgines.

8.2.4 Diagrama d'evolució del Sprint



8.2.5 Incidències destacables

Com en a l'Sprint anterior, els problemes principals els hem tingut al tractar les dades del XML amb el *NSXMLParser*, degut a la codificació utilitzada a *El Quiosc* (ISO-8859-1) i que la capçalera del XML indica que les dades estan codificades en UTF-8, el *parser* no era capaç de llegir correctament les paraules accentuades. Deixant els noms de moltes seccions a incomplets.

Exemple de dades:

```
<seccio>PolÃ-tica</seccio>
```

Aquesta secció que correspondria a “*Política*”, era interpretada com “*Pol*”.

Ho solucionem aplicant una conversió amb *UTF8String* de la classe *NSString*:

```
NSString * seccio = [NSString stringWithUTF8String:(const char *) [[elem valueForKey:@"seccio"] UTF8String]];
```

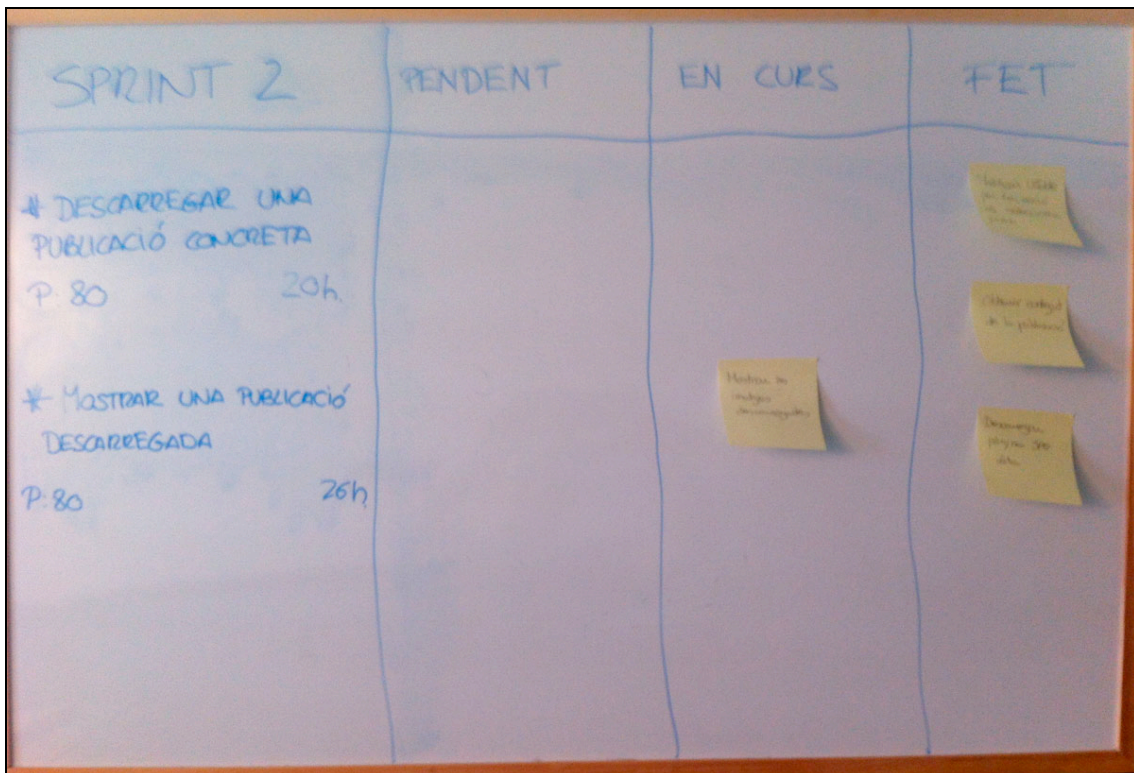
8.2.6 Reunió final Sprint

Al finalitzar aquest segon Sprint, no hem pogut assolir els objectius fixats inicialment. Tot i que en la planificació de el *Sprint*, ja vàrem veure que les dues històries seleccionades superaven en temps estimat, a la durada fixada del *Sprint*.

Hem resolt la història de la descàrrega d'una publicació concreta i hem aconseguit mostrar una pàgina d'aquesta.

Aquest *Sprint* ens deixa noves funcionalitat a afegir a el *Product Backlog*:

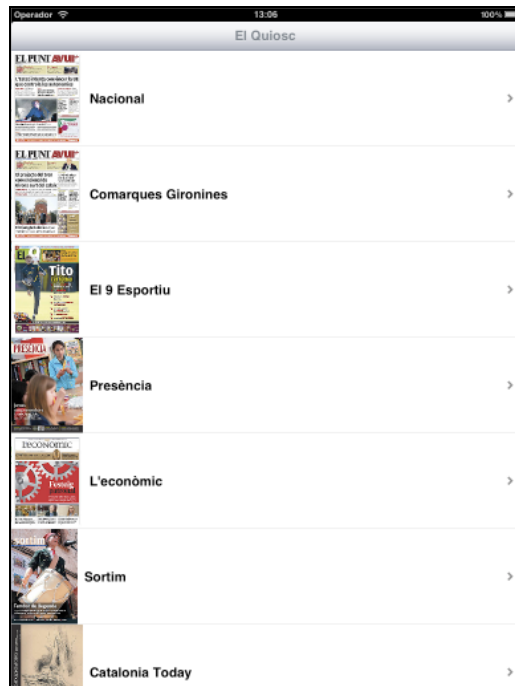
- Pagar les imatges descarregades de les diferents pàgines
- Guardar la informació descarregada per a properes execucions
- Deslligar el procés de descàrrega del de lectura de la publicació



Imatge pissarra fi Sprint 2

8.2.7 Demostració del producte al client

A la primera captura veiem la vista amb les publicacions vigents que hem modificat. Una vegada seleccionada una publicació, en aquest cas l'edició *comarques gironines* del *Punt Avui*, s'inicia el procés de descàrrega i finalitza amb la visualització de la primera pàgina (captura 2).



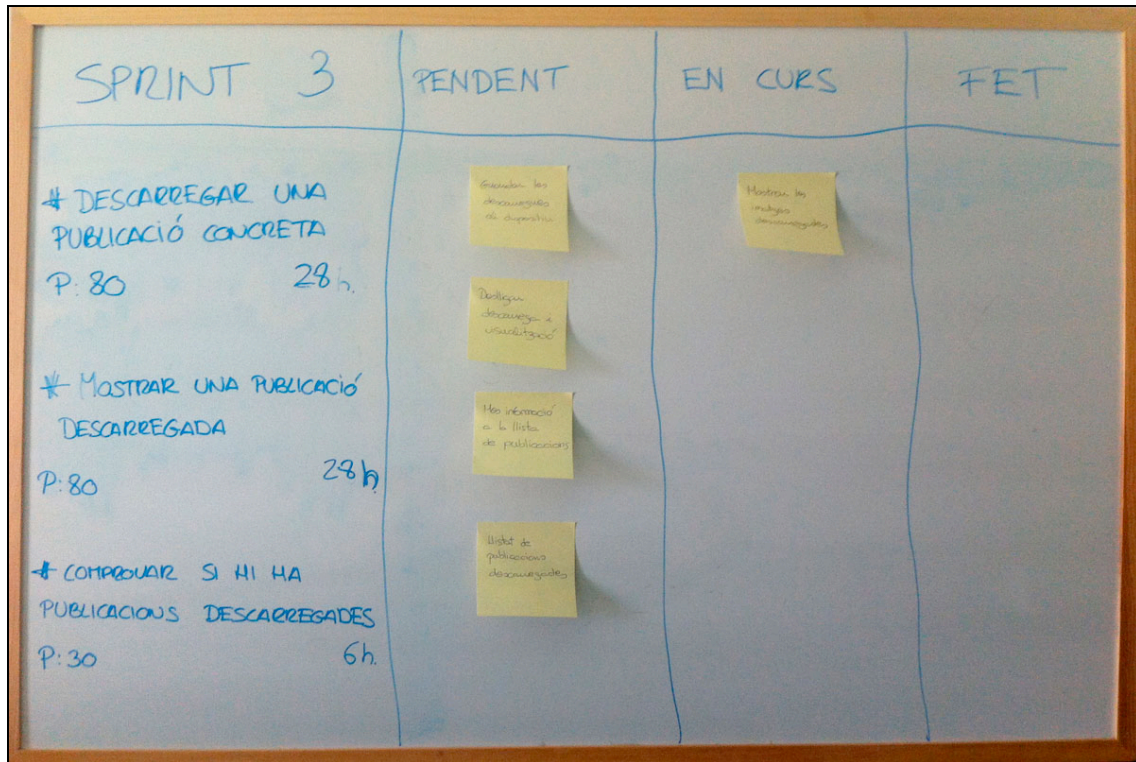
Captura 1



Captura 2

8.3 Sprint 3

- Descarregar una publicació concreta
- Mostrar una publicació descarregada
- Comprovar si hi ha publicacions descarregades



Imatge pissarra inici Sprint 3

8.3.1 Històries d'usuari detallades

- Més informació a la llista de publicacions
- Deslligar el procés de descarrega i visualització
- Guardar les descarregues en el dispositiu
- Mostrar una publicació descarregada (paginada)
- Llistat publicacions descarregades

8.3.2 Més informació a la llista de publicacions

Afegim una descripció per les publicacions que tinguin més d'una edició i la data en que ha estat publicada. També millorem una mica l'aspecte de la llista de publicacions.



Captura de la publicació amb la informació addicional

8.3.3 Deslligar el procés de descàrrega i visualització

Fins al moment el procés de descàrrega s'iniciava al carregar el controlador, que un cop finalitzada la descàrrega ens mostrava la publicació. Per tal de deslligar aquests dos processos, afegim un botó per a cada element de la llista de publicacions que iniciarà la descàrrega.



Captura durant la descàrrega d'una publicació

Per crear el botó utilitzem la classe *UIButton* del *framework UIKit*. Un cop iniciada la descàrrega, amaguem el botó i mostrem l'evolució d'aquesta en una barra de progrés *UIProgressBar*. La barra de progrés creixerà en funció del numero de pàgines descarregades. Al finalitzar la descàrrega, canviem la barra per un botó que ens mostrarà la publicació.



Captura amb la descàrrega de la publicació finalitzada

8.3.4 Guardar les descarregues en el dispositiu

Fins al moment, les imatges que descarreguem tant en el llistat de publicacions com a la visualització d'una publicació, són guardades en memòria. Per tant, al reiniciar l'aplicació ja no hi teníem accés i les hem de tornar a demanar.

Per millorar això, decidim guardar les imatges descarregades a disc. La classe *NSFileManager*, ens permet interactuar amb les carpetes i fitxers accessibles des de l'aplicació. Muntem una estructura de directoris per tal de tenir les imatges ben ordenades.

Les imatges descarregades en el llistat de publicacions, són guardades al directori *portades*. Cada imatge rebrà per nom *portada-{ID_PUBLICACIO}.jpg*. On *ID_PUBLICACIO* és diferent per cada una de les portades visibles al llistat.

Les imatges que representen les pàgines de les diferents publicacions. Són guardades en una estructura de directoris amb el criteri següent:

```
{DATA_PUBLICACIO}/{ID_PUBLICACIO}/pag-{ID_PUBLICACIO}-  
{NUM_PAGINA}.jpg
```

D'aquesta manera garantim un accés àgil a les imatges i l'avantatge de poder comprovar fàcilment si disposem o no d'un fitxer, per estalviar-nos la seva descarrega.

Un cop solucionat el tema de les imatges, ens caldrà conservar d'alguna manera la informació referent a les publicacions que hem descarregat.

Tot i que podríem controlar-ho comprovant l'existència o no de les imatges corresponents, decidim guardar-nos aquesta informació en una taula de la base de dades.

El sistema *iOS* disposa de llibreries *SQLite*. Aquesta potent llibreria, ens permet crear i gestionar les taules i executar sentències SQL sobre elles.

Creem una taula *PUBLICACIONS*, on guardarem tota la informació de les publicacions que anem descarregant i que ens permetrà conèixer en qualsevol moment de quines publicacions disposem.

Adaptem el *PublicacionsViewController* per llegir la taula i reconèixer si ha de marcar una publicació com a descarregada o com a pendent de descarregar.

8.3.5 Mostrar una publicació descarregada (paginada)

Continuem amb el *PublicacioViewController* que havíem deixat a mitges al finalitzar el 2n *Sprint*.

El controlador passa a implementar *UIPageViewController*, aquesta classe disponible des de la versió 5.0 de iOS dins el *framework UIKit*, ens permet navegar entre diferents *ViewControllers* amb les transicions i gestos que definim. Imitant l'efecte que de passar les pàgines com ho faríem amb una publicació real.

Ens caldrà un nou controlador que serà l'encarregat de mostrar realment la pàgina. Aquest controlador que anomenem *PaginaViewController*, carrega la pagina en una vista que disposa d'un element *UIImageView*, que ens permet mostrar la pàgina a pantalla completa i fer-hi zoom, per poder veure el contingut amb més detall.

Per tant, tindrem un *UIPageViewController* que controlarà un conjunt de controladors *PaginaViewController*.

8.3.6 Llistat publicacions descarregades

Per poder accedir directament a les publicacions que hem descarregat anteriorment i que ja no són visibles a la vista de publicacions actuals, creem un *UITableViewController*, que anomenem *MevesViewController*, que llegirà les publicacions de la taula PUBLICACIONS, i les mostrarà en una taula ordenades per data de descàrrega.



Captura de la taula de publicacions descarregades

Al seleccionar una publicació la mostrarem amb el "paginador", comentat en el punt anterior.

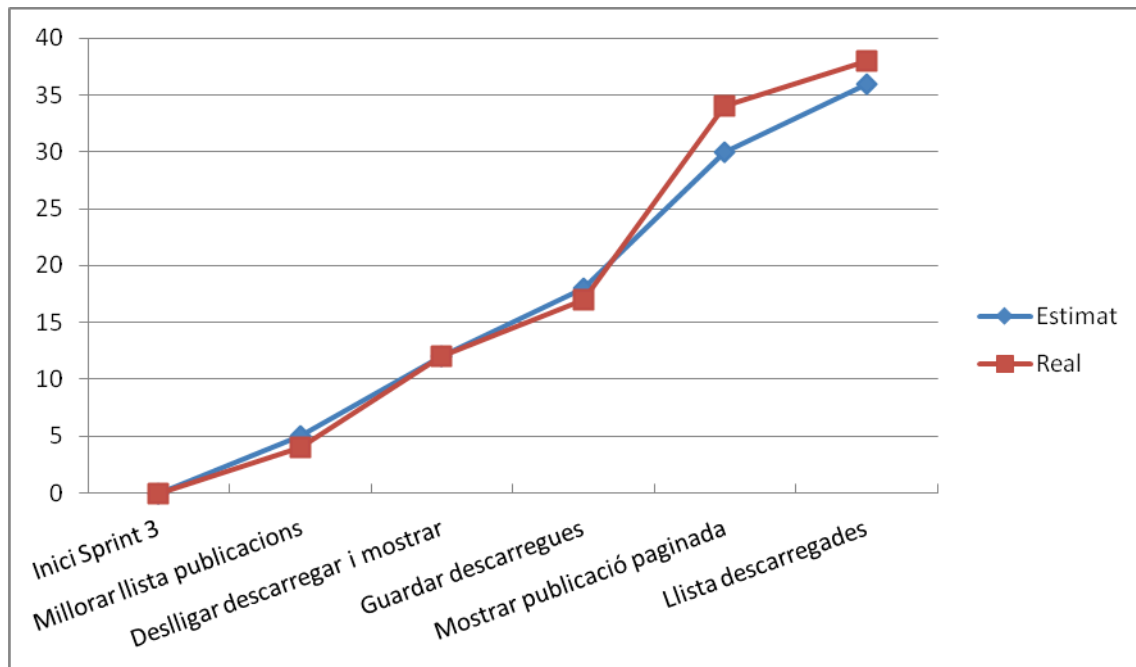
Per tal de poder accedir a aquesta vista, canviem l'estructura principal de l'aplicació. El controlador principal passa a ser un *UITabViewController*, que no és res més un conjunt de pestanyes associades cada una a un controlador diferent.

La pestanya "*recents*", mostra la pantalla amb les publicacions vigents i la pestanya "*downloads*" la pantalla amb les publicacions descarregades.



Captura de la barra de pestanyes "TabBar"

8.3.7 Diagrama d'evolució del Sprint



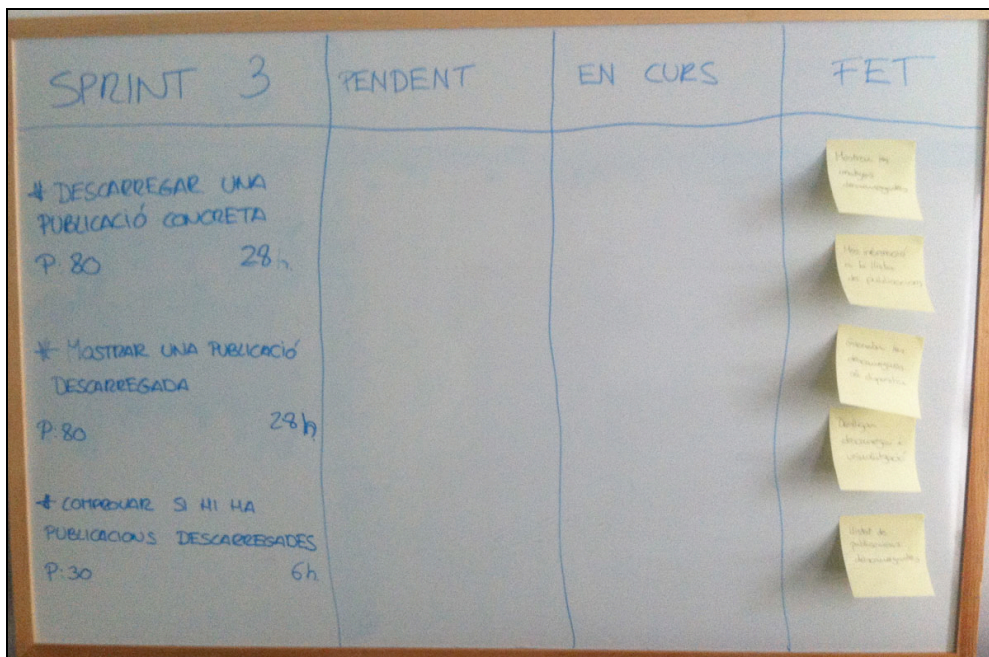
8.3.8 Incidències destacables

Durant aquest *Sprint* no hem trobat incidències que hagin de ser comentades.

8.3.9 Reunió final Sprint

En aquest *Sprint*, hem assolit tots els objectius marcats inicialment. Millorant moltes de les funcionalitats desenvolupades anteriorment i acabant la visualització de la publicació que havíem deixat a mitges a l'anterior *Sprint*.

Arribem a la conclusió que utilitzant els fitxers *PDF* que tenim a *El Quiosc* de les publicacions, podrem tenir més definició per la lectura que amb les imatges en alta resolució que utilitzem actualment. Això implicarà modificar els processos de descarrega i visualització per poder treballar amb aquest tipus de fitxers.



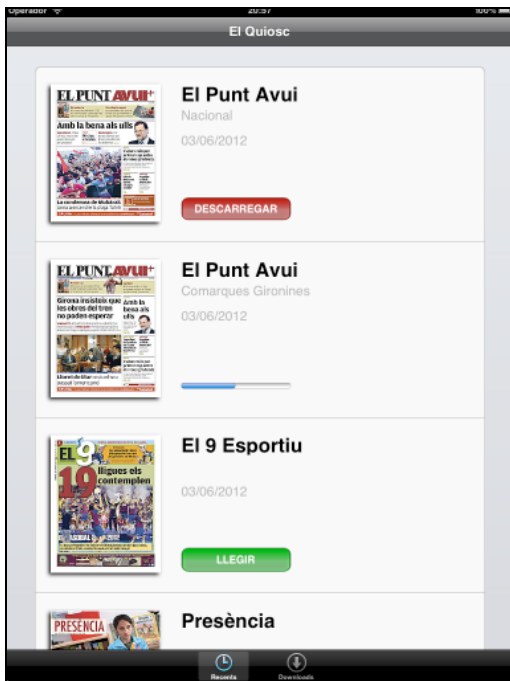
Imatge pissarra fi Sprint 3

Al finalitzar aquest Sprint, l'empresa em proposa incrementar el temps de dedicació en el projecte, permetent-me dedicar-li la totalitat de la jornada laboral.

Això ens porta a replantejar alguns dels requeriments ja assolits i a plantejar-ne de nous. També implica modificar els temps marcats per cada Sprint.

8.3.10 Demostració del producte al client

A la primera captura veiem la vista amb les publicacions vigents, amb una publicació descarregada i una altre en curs. Al prémer el botó “llegir”, carreguem la publicació (captura 2). A la captura 3 podem veure la pantalla amb les publicacions descarregues.



Captura 1



Captura 2



8.4 Sprint 4

- Navegació per seccions de la publicació
- Eliminar publicacions
- Veure darreres publicacions d'una publicació
- Adaptar vistes al nou disseny
- Descàrrega i visualització publicació PDF

SPRINT 4	PENDENT	EN CURS	FET
# Navegació per seccions P:40 20h.	Eliminar publicacions BDD Eliminar fitxers de les publicacions		
# Eliminar publicacions P:60 10h.	Refactoritzar accés a BDD Taula PublicacioPagina		
# Veure darreres publicacions d'una publicació P:80 38h.	Distribuir de manera de cada pagina Adaptar vista publicacions darreres		
# Adaptar vistes al nou disseny P:80 38h.	Obtenir l'últim publicacions del sistema Crear un nou publicacions autors		
# Descàrrega i visualització PDF P:100 2h.	Adaptar vista publicacions PDF		

Imatge pissarra inici Sprint 4

8.4.1 Històries d'usuari detallades

- Eliminar publicacions de la BDD
- Eliminar fitxers de les publicacions
- Refactoritzar accés BDD Publicacions
- Creació taula "PublicacioPagina"

- Llista seccions a vista publicació
- Adaptar vista publicacions vigents
- Adaptar vista publicacions descarregades
- Crear vista darreres publicacions

Després d'acordar amb l'empresa intensificar la dedicació en el projecte i ampliar i millorar-ne les funcionalitats. Escollim les històries d'usuari que volem assolir durant aquest *Sprint*.

8.4.2 Navegació per seccions de publicació

Per tal de facilitar la navegació entre les pàgines de la publicació, afegim un desplegable amb totes les seccions que conformen la publicació. Al seleccionar-ne una mostrarem a l'usuari la pàgina que l'encapçala.

Per mostrar el llistat de seccions sobre la pàgina que estem visualitzant, utilitzem una *UIPopoverView* que ens permet obrir una vista sobre una altre, amb un controlador independent (*UIPopoverViewController*). Aquest controlador, que anomenem *SeccionsViewController*, és l'encarregat de carregar les seccions de la publicació i mostrar-nos el llistat.



Captura exemple *UIPopoverView* amb les seccions

Al escollir una de les seccions, *SeccionsViewController* llança una notificació (*NSNotification*) que serà interpretada per *UIPageViewController* (controlador que ens mostra les pàgines) i que carregarà la primera pàgina de la secció rebuda.

Creem una nova taula *PUBLICACIO_PAGINA*, per poder guardar informació sobre les pàgines d'una publicació determinada. Com per exemple a quina secció pertany cada una.

8.4.3 Canvi imatges JPG alta resolució per PDF

Modifiquem els processos que teníem tant de descàrrega com de visualització de pàgines per treballar amb fitxers PDF. La informació per obtenir el fitxer PDF de cada pàgina ja la teníem ja que està inclosa en el XML que descarreguem amb tot el contingut de la publicació.

Per mostrar la pàgina completa en el visualitzador, a partir d'aquest moment *PDFViewController*, utilitzem un *UIWebView* en substitució del *UIImageView*. Aquest element *UIWebView* ens permet visualitzar el PDF igual que ho fèiem amb les imatges, desplaçar-nos i fer-hi zoom.

8.4.4 Adaptar vistes al nou disseny

Modifiquem les vistes que tenim fins al moment per adaptar-les al disseny que hem consensuat amb l'empresa.

A *PortadesViewController*, que controla la vista amb les publicacions del dia, hem de fer-hi varies modificacions.

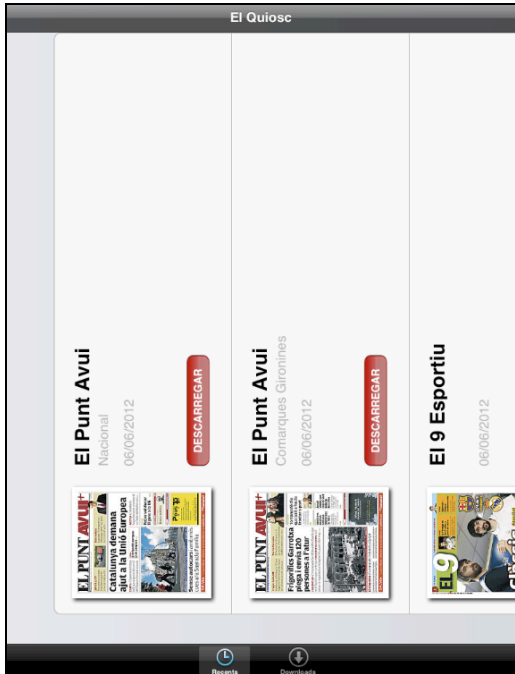
A la part superior mostrarem el logotip de l'aplicació i la data de la darrera vegada que hem pogut actualitzar el contingut.



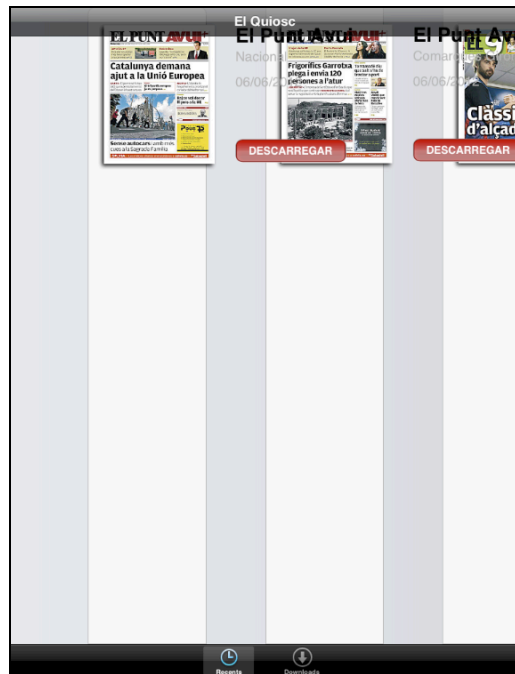
Logotip i data de darrera actualització

Afegim un element *UIBarButtonItem* a la barra de navegació per “actualitzar” la llista de publicacions. Simplement repeteix la petició a El Quiosc per obtenir les publicacions i es refresca la taula amb les noves dades.

Referent a la llista de publicacions, observem que fins al moment teníem un desplaçament vertical de la taula i en el nou disseny el desplaçament ha de ser lateral. Per aconseguir-ho, rotem 90° la taula cap a l'esquerra i el contingut de les cel·les 90° a la dreta.



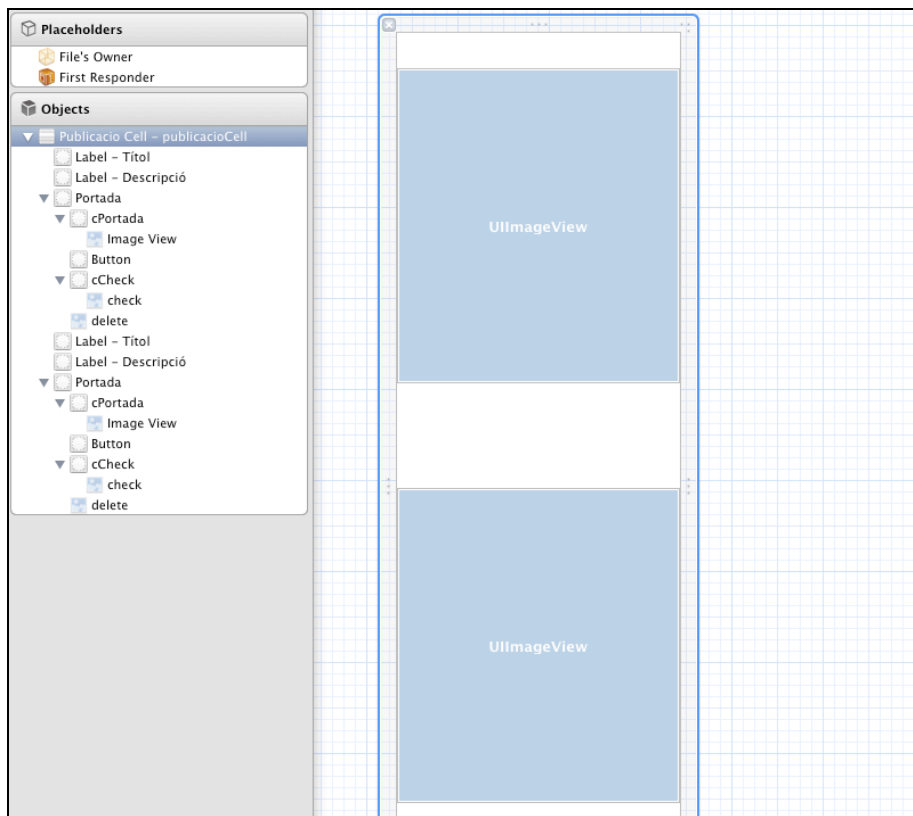
UITableView rotació 90°



UITableView i UITableViewCell rotació 90°

Per adaptar-nos al disseny proposat, ens cal pintar dues publicacions a cada cel·la de la taula. Els elements disponibles en una cel·la estàndard com *UITableViewCell* no ens ho faciliten. La solució és “extendre” la classe *UITableViewCell* i crear un nou tipus de cel·la que anomenem *PublicacioCell*.

Amb l'ajuda de *Interface Builder*, dissenyem la distribució dels elements que tindrà aquesta nova cel·la.



Captura IB, estructura PublicacioCell

Com ja hem esmentat, si és necessària, ha de poder mostrar la informació de dues publicacions. Una imatge de portada (*UIImageView*), el títol d'aquesta (*UILabel*) i la data de publicació (*UILabel*).

Amb el nou disseny, “desapareixen” els botons de descàrrega i visualització. Clicant sobre una de les imatges de portada de les publicacions, carreguem el *PortadaViewController*, que és la pantalla on ara es gestionen les descàrregues.

També apareix una nova icona amb un petit “check”, que ens indicarà les publicacions que ja hem descarregat. Al clicar sobre el “check” podrem accedir directament a la visualització de la publicació.

A la *UITableView* que conté les publicacions, activem el *Paging Mode*, que modifica el comportament del “scroll” i simula un “scroll paginat” de les cel·les.

Per indicar a l'usuari que hi ha més publicacions disponibles, afegim un control a la part inferior que ens indica el nombre de pàgines disponibles i a quina ens trobem (Page Control).



Nova aparença del llistat de publicacions vigents

La llista de publicacions descarregades *MevesViewController*, igual que la de seccions la mostrarem en un *UIPopoverView*. Això ens permet tenir-la accessible des de totes les vistes de l'aplicació.

A nivell d'aparença, a cada publicació i afegim el nom de l'edició si en té i la imatge de la portada.



UIPopoverView amb la llista de publicacions descarregades

8.4.5 Eliminar publicacions

Ens cal poder eliminar una publicació prèviament descarregada. Per fer-ho, despleguem la vista de publicacions descarregades i al prémer “Editar”, ens apareixen els elements típics de la interfície de iOS per eliminar un element de la taula.



Captures del procés d'eliminació d'una publicació

Per poder utilitzar aquesta forma d'eliminar elements d'una taula que ens proporciona iOS, haurem d'implementar la funció `commitEditingStyle` que ens permet definir quines accions s'executaran al confirmar l'eliminació.

L'eliminació completa de la publicació la dividim en dues funcionalitats, eliminar els fitxers relacionats amb la publicació (portada JPG i pàgines PDF) i eliminar

les files corresponents a les taules *PUBLICACIO* i *PUBLICACIO_PAGINA*.

L'eliminació dels fitxers relacionats amb la publicació, igual que la creació, la gestionem amb la classe *NSFileManager*.

Per eliminar les dades a la taula *PUBLICACIONS* i *PUBLICACIO_PAGINA*, executarem les sentències SQL pertinents gràcies a la llibreria *SQLite*.

Exemple de les sentències SQL per eliminar una publicació:

```
DELETE FROM 'PUBLICACIO_PAGINA' WHERE PUBLI_ID=?
```

```
DELETE FROM 'PUBLICACIONS' WHERE ID=?
```

Un cop eliminada la publicació, actualitzem les dades de la llista de publicacions descarregades (*MevesViewController*).

8.4.6 Veure darreres publicacions d'una publicació

Per obtenir les darreres publicacions ens cal adaptar un *action* nou a El Quiosc, degut a que amb les existents no obtenim les dades requerides.

Aquesta nova *action*, ens retornarà les dades de les N darreres publicacions, on N serà un paràmetre de la crida. Aquestes dades les obtindrem en format XML, igual que en les peticions anteriors.

Un cop aplicats els canvis necessaris a El Quiosc, anem a desenvolupar el controlador que gestionarà aquestes publicacions anteriors.

Exemple URL per obtenir les darreres publicacions:

```
http://www.elquiosc.cat/Quiosc.php?do=darrersEdicioXML&edicio_id=29&limit=7
```

Exemple del XML obtingut:

```
<?xml version="1.0" encoding="UTF-8"?>

<nodos>

  <nodo label="El Punt Avui" icona="1">

    <nodo          label="Comarques          Gironines"          icona="2"
    dirurl="/&amp;edicio_id=29&amp;data=2012-06-06" obert="S"/>

  </nodo>

  <nodo label="El Punt Avui" icona="1">

    <nodo          label="Comarques          Gironines"          icona="2"
    dirurl="/&amp;edicio_id=29&amp;data=2012-06-05" obert="S"/>

  </nodo>

  <nodo label="El Punt Avui" icona="1">

    <nodo          label="Comarques          Gironines"          icona="2"
    dirurl="/&amp;edicio_id=29&amp;data=2012-06-04" obert="S"/>
```

```

</nodo>

<nodo label="El Punt Avui" icona="1">

    <nodo          label="Comarques          Gironines"          icona="2"
dirurl="/&edicio_id=29&data=2012-06-03" obert="S"/>

</nodo>

...

<nodo label="El Punt Avui" icona="1">

    <nodo          label="Comarques          Gironines"          icona="2"
dirurl="/&edicio_id=29&data=2012-05-31" obert="S"/>

</nodo>

<nodo label="El Punt Avui" icona="1">

    <nodo          label="Comarques          Gironines"          icona="2"
dirurl="/&edicio_id=29&data=2012-05-30" obert="S"/>

</nodo>

</nodos>

```

El nou controlador que anomenem *PortadaViewController*, ens permet dividir la pantalla en dues zones, on cada zona és governada per un altre controlador.

Una zona que anomenem *PublicacioViewController*, on veiem ampliada la portada de la darrera publicació, la data de publicació i el nom.



Captura de la vista principal de PublicacioViewController

És un UIViewController senzill, amb una *UIImageView* per mostrar la portada i diferents *UILabels* que contenen el nom i la data de publicació. També mostrarà la icona amb el “check” en cas d’haver-la descarregat prèviament.

Si la publicació no ha estat descarregada, al prémer sobre la portada iniciarà el procés de descàrrega. En el cas que ja hagi estat descarregada la carreguem i visualitzem amb *PDFScrollViewController*.



Captures del procés de descàrrega

La publicació que es mostra al carregar el controlador inicialment, ens la serveix el controlador anterior (*PortadesViewController*). El que ens permet

mostrar-la sense haver d'esperar la resposta de la petició que fa *PortadaViewController* per obtenir les darreres publicacions.

L'altre zona, que anomenem *PublicacionsViewController*, ens mostra les portades en miniatura de la resta de publicacions ordenades cronològicament.

Semblant a la *UITableView* del controlador *PortadesViewController*, és una taula amb una rotació de 90° a l'esquerra que ens proporciona un desplaçament lateral de les cel·les.

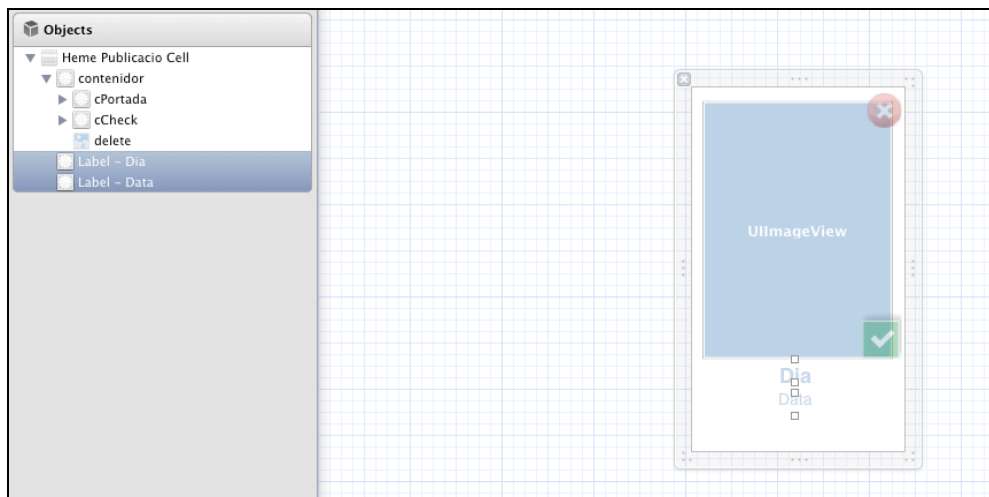


Captura de la vista principal de *PublicacionsViewController*

Aquestes cel·les, contenen la portada, el dia i data de publicació i si es dona el cas, la icona amb el "check" que indica que la publicació ha sigut descarrega.

PortadaViewController, rep la resposta amb les darreres publicacions, tracta el XML i carrega les dades obtingudes a el controlador *PublicacionsVController*.

Com en el cas de les *PublicacioCell*, creem un nou tipus de classe cel·la, *HemePublicacioCell*.



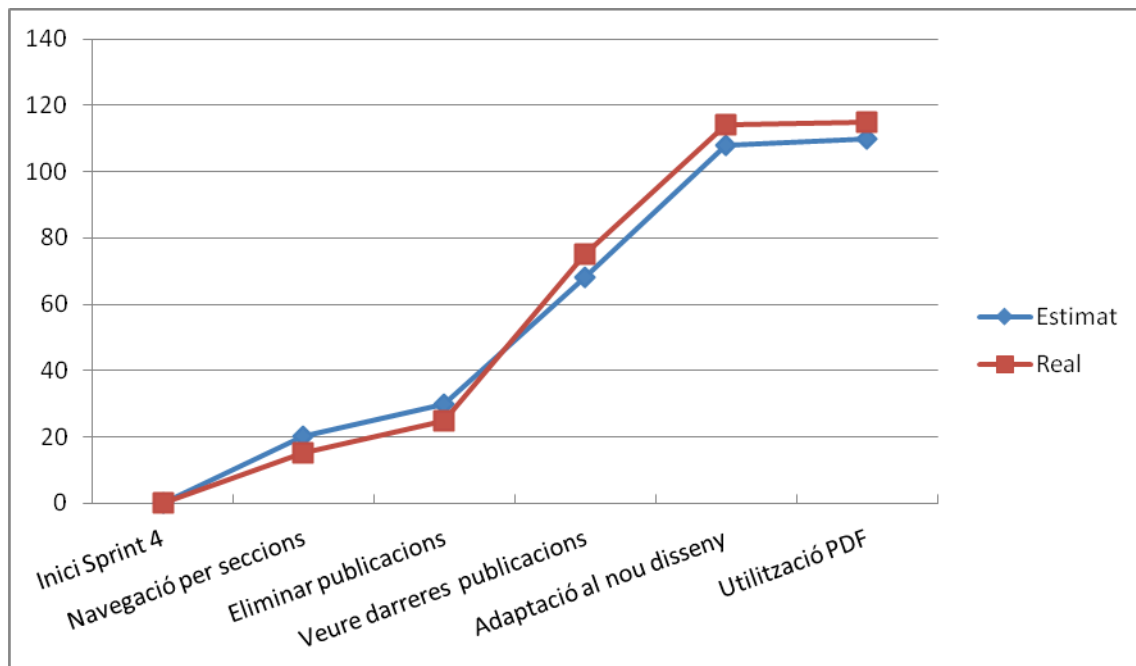
Captura IB, estructura HemePublicacioCell

Al seleccionar una de les portades en miniatura, aquesta és carrega a el controlador *PublicacioViewController* i es mostra ampliada a la part superior.



Captura vista PortadaViewController

8.4.7 Diagrama d'evolució del Sprint



8.4.8 Incidències destacables

Durant el desenvolupament de la història d'usuari **Veure darreres publicacions d'una publicació**, ens hem trobat amb el problema de no disposar a *El Quiosc* de la crida necessària per poder obtenir la informació requerida. Ho hem solucionat afegint un punt més a el *Sprint Backlog* per tal de desenvolupar aquesta funcionalitat a *El Quiosc* i per poder continuar treballant.

Un altre problema que volem mencionar, l'hem trobat al desenvolupar el controlador *PortadaViewController*. Inicialment, preteníem utilitzar com a base d'aquest controlador un *UISplitViewController*, un controlador pensat per gestionar dos controladors dins seu i implementar una interfície "master-detall".

El problema que ens hem trobat és que no ens donava la flexibilitat necessària per adaptar la interfície als nostres requeriments i hem hagut de desenvolupar el nostre propi "split-view".

8.4.9 Reunió final Sprint

En aquest *Sprint*, hem assolit tots els objectius marcats inicialment. Adaptant les funcionalitats que ja teníem a el nou plantejament i desenvolupant-ne de noves.

Durant aquesta reunió es planteja la voluntat de l'empresa de deslligar el funcionament de l'aplicació del de *El Quiosc*.

Tot i que seguirem utilitzant les seves dades, requerim més seguretat en les peticions per el desenvolupament dels requeriments d'identificació.

Per fer-ho, decidim substituir les crides *HTTP* que fem a *El Quiosc*, per peticions amb protocol XML-RPC.

Per tant, apareixen nous requeriments al *Product Backlog*.

SPRINT 4	PENDENT	EN CURS	FET
# Navegació per seccions P:40 20h. # Eliminar publicacions P:60 10h. # Veure diverses publicacions d'una publicació P:80 38h. # Adaptar vistas al nou disseny P:80 38h. # Disseny i visualització PDF P:100 2h.			- Eliminar dades de les publicacions - Eliminar pàgines 200 - Eliminar publicacions 300 - Treure publicacions eliminades - Tractar de trobar la manera de canviar programari - Canviar codi de disseny per a autors - Obtenir l'estat dels autors del Quiosc - Adaptar les pàgines de publicacions - Adaptar les pàgines de publicacions

Imatge pissarra fi Sprint 4

8.4.10 Demostració del producte al client

A la primera captura veiem el resultat del “redisseny” que hem fet de *PortadesVC*. Al seleccionar “El Punt Avui - Nacional”, carreguem PortadaVC, que podem veure a la segona captura.



Captura 1



Captura 2

En aquesta pantalla, naveguem per el llistat de publicacions anteriors i descarreguem la publicació d'avui (captures 3).



Captura 3



Captura 4

Amb la publicació descarrega (captura 4), podem accedir al seu contingut (captura 5) o eliminar-la des de la llista de publicacions descarregades (captura 6).



Captura 5



Captura 6

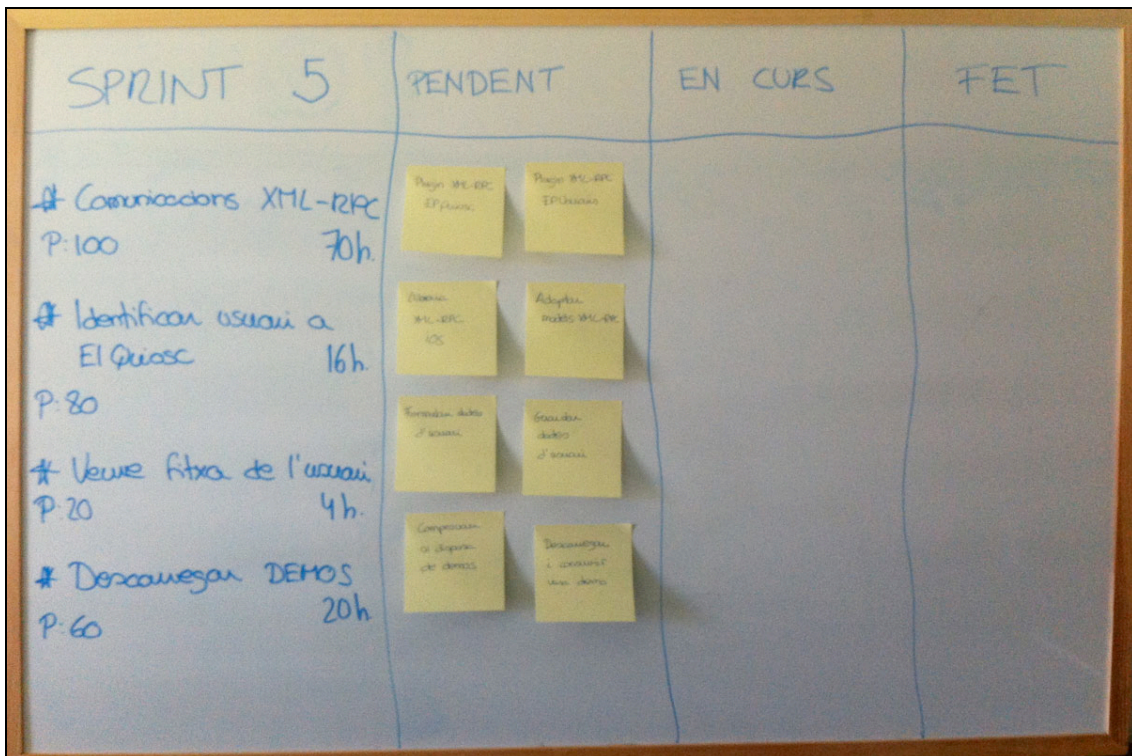
A la captura numero 7 observem el llistat de seccions de la publicació, que ens permet accedir directament a les pàgines d'una secció determinada.



Captura 7

8.5 Sprint 5

- Comunicacions amb XML-RPC
- Identificar usuari a El Quiosc
- Veure fitxa de l'usuari
- Descarregar publicacions gratuïtes



Imatge pissarra inici Sprint 5

8.5.1 Històries d'usuari detallades

- Plugin XML-RPC EPQuiosc
- Plugin XML-RPC EPUusuaris
- Llibreria XML-RPC en iOS
- Adaptar models a XML-RPC
- Formulari dades d'usuari

- Comprovar identificació usuari
- Guardar dades d'usuari
- Comprovar si pot descarregar una "demo"
- Descarregar una publicació de "demo"

Durant aquest *Sprint* tenim una quantitat important de funcionalitats que es desenvoluparan fora del que és l'aplicació en si.

8.5.2 Comunicacions amb XML-RPC

Com vàrem acordar al finalitzar el darrer *Sprint*, pretenem deixar d'utilitzar les crides *HTTP* que fem a *El Quiosc* per obtenir les dades necessàries per el funcionament de l'aplicació.

En substitució, desenvoluparem dos *plugins* *EPUsuaris* i *EPQuiosc* que instal·larem a el web *El Punt Avui* i que ens serviran les dades necessàries mitjançant el protocol *XML-RPC*.

El web de *El Punt Avui* està basat en el CMS *Joomla*, que suporta la instal·lació de *plugins* i disposa de llibreries per implementar el protocol *XML-RPC*.

Els *plugins* igual que el web, són programats en PHP. La instal·lació dins el web ens permetrà utilitzar els *models* i *helpers* dels que disposa, per facilitant-nos l'accés a les dades necessàries.

Com ja hem comentat, separem les crides en dos *plugins* diferents.

8.5.3 Plugin EPQuiosc

Un primer *plugin* que anomenem *EPQuiosc*, que serveix la informació referent a les publicacions i el seu contingut.

Disposa de les següents crides:

- *getEdicions*
- *getDarrersEdicio*

- *getEdicio*

epquiosc.getEdicions

La crida *getEdicions*, rep com a paràmetres un identificador d'aplicació (APPKEY) i un identificador únic de dispositiu i aplicació (IDENKEY).

```
<?php function getEdicions($appkey, $idenkey) { ... } ?>
```

Carrega de la BDD de *El Quiosc* les publicacions que hi ha lligades a la APPKEY rebuda i retorna un *array* amb les dades de les publicacions (edicions) obtingudes.

Dades retornades per cada edició:

```
$edicio = new xmlrpcval(array(

    'id' => new xmlrpcval($item->EDICIO_ID, $xmlrpcString),

    'titol' => new xmlrpcval($item->PUBLICACIO_NOM,
$xmlrpcString),

    'edicio' => new xmlrpcval($item->EDICIO_NOM,
$xmlrpcString),

    'data' => new xmlrpcval($item->MAX_DATAINI,
$xmlrpcString),

    'data_alta' => new xmlrpcval($item->DATA_ALTA,
$xmlrpcString),

    'ordre' => new xmlrpcval($item->SUM_ORDRE, $xmlrpcString)

), 'struct');
```

epquiosc.getDarrersEdicio

La crida *getDarrersEdicio*, rep com a paràmetres un identificador d'aplicació (APPKEY), un identificador únic de dispositiu i aplicació (IDENKEY), l'identificador de l'edició que estem consultant (EDICIO_ID), la data inicial a partir de la qual volem obtenir les publicacions (DATA), el numero d'elements màxim que volem obtenir (NUM) i l'ordre en que les volem (DIRECCIO).

```
<?php function getDarrersEdicio($appkey, $idenkey, $edicio_id,
$data, $num, $direccio){ ... } ?>
```

Retorna un *array* amb les NUM edicions existents del tipus EDICIO_ID a partir de la DATA ordenats seguint la DIRECCIO rebuda, si aquesta EDICIO_ID pertany a la APPKEY.

Dades retornades per cada element:

```
$edicio = new xmlrpcval(array(

    'id'    => new xmlrpcval($item->EDICIO_ID, $xmlrpcString),

    'titol'    => new xmlrpcval($item->PUBLICACIO_NOM,
$xmlrpcString),

    'edicio' => new xmlrpcval($item->EDICIO_NOM,
$xmlrpcString),

    'data'    => new xmlrpcval($item->MAX_DATAINI,
$xmlrpcString),

    'data_alta' => new xmlrpcval($item->DATA_ALTA,
$xmlrpcString)

), 'struct');
```

epquiosc.getEdicio

La crida *getEdicio*, rep com a paràmetres un identificador d'aplicació (APPKEY), un identificador únic de dispositiu i aplicació (IDENKEY), l'identificador de l'edició que volem descarregar (EDICIO_ID), la data de publicació de l'edició (DATA), un identificador d'usuari si està identificat (USUARI_ID) i un identificador del sistema operatiu del dispositiu que fa la petició (OS).

```
<?php function getEdicio($appkey, $idenkey, $edicio_id, $data,
$usuari_id, $os){ ... } ?>
```

Carrega la informació de l'edició amb EDICIO_ID i DATA. Comptabilitza la descarrega d'aquesta edició a la taula de control de descarregues de *E/*

Quiosc, per una edició determinada (EDICIO_ID i DATA) i per USUARI_ID APPKEY i OS.

Ens retorna un *array* amb totes les pàgines de l'edició i per a cada pàgina, la informació de la secció i els enllaços temporals per descarregar les imatges i PDF.

Dades retornades per cada pàgina:

```
$pagina = new xmlrpcval(array(

    'id' => new xmlrpcval($item->PAGINA_ID, $xmlrpcString),

    'seccio_id' => new xmlrpcval($item->SECCIO_ID,
$xmlrpcString),

    'seccio' => new xmlrpcval($item->SECCIO_NOM,
$xmlrpcString),

    'numero' => new xmlrpcval($item->NUM, $xmlrpcString),

    'jpg' => new xmlrpcval(EPQuioscHelper::staticUrlJPG($item-
>JPG), $xmlrpcString),

    'jpgA' => new xmlrpcval(EPQuioscHelper::staticUrlJPG($item-
>JPG,'alta'), $xmlrpcString),

    'pdf' => new xmlrpcval(EPQuioscHelper::staticUrlPDF($item-
>PDF), $xmlrpcString)

), 'struct');
```

Un cop tenim preparat el *plugin* l'instal·lem a el web El Punt Avui i configurem Joomla perquè escolti les peticions que rebi en XML-RPC.

Per millorar la seguretat, decidim que les peticions requeriran autenticació amb usuari i contrasenya.

8.5.4 Plugin EPUssuaris

El segon *plugin* que anomenem *EPUssuaris*, tractarà les dades més personals de l'aplicació.

Disposa de les següents crides:

- *registrar*
- *consumirDemo*
- *hashQuiosc*

epusuaris.registrar

La crida *registrar*, rep com a paràmetres un identificador d'aplicació (APPKEY) , un identificador únic de dispositiu (DISP_KEY), un identificador d'usuari i un password, només si està identificat (USUARI_ID i PASSWORD), un *array* amb dades sobre el dispositiu (DISP_INFO) i un identificador del sistema operatiu del dispositiu que fa la petició (OS).

```
<?php    function    registrar($app_key,    $disp_key,    $username,
$password, $disp_info,$os){ ... } ?>
```

Primer de tot, comprova si ha rebut identificador d'usuari i password, en cas afirmatiu intenta identificar-lo. Un cop identificat l'usuari, carreguem els identificadors de les publicacions a les quals està subscrit.

A continuació, generem un identificador únic, que identificarà les peticions que fem en tot moment. Aquest identificador es genera a partir del APP_KEY, el DISP_KEY que recordem és un identificador únic del dispositiu i d'un identificador d'usuari en cas d'haver estat identificat com a tal o el valor 0 en cas contrari.

Guarda a la taula USERS_MOBILS a la BDD de El Quiosc, per cada IDEN_KEY generat, la APP_KEY, DISP_KEY, l'identificador d'usuari si en teníem, les dades del dispositiu (DISP_INFO) i el numero de "demos" que hem consumit. En cas que el IDEN_KEY generat no existeixi a la taula el nombre de "demos" serà 0.

Aquest identificador únic, també ens permetrà identificar un dispositiu encara que l'usuari reinstal·li l'aplicació. Evitant que es tornin a consumir les descarregues gratuïtes que proporcionem.

Finalment ens retornarà les dades de l'usuari (si l'ha pogut identificar), les publicacions a les quals tenim accés, l'identificador IDEN_KEY i el numero de “demos” consumides per aquest IDEN_KEY.

Dades retornades:

```
$struct = new xmlrpcval(

    array(

        'username' => new xmlrpcval($user->get('username')),

        'name' => new xmlrpcval($user->get('name')),

        'email' => new xmlrpcval($user->get('email')),

        'usuari_id' => new xmlrpcval($user->id),

        'demos' => new xmlrpcval($userMobil['demos']),

        'edicions' => new xmlrpcval($edicionsAux,
        $xmlrpcStruct),

        'iden_key' => new xmlrpcval($iden_key)

    )

, $xmlrpcStruct);
```

epusuaris.consumirDemo

La crida *potConsumirDemo*, rep com a paràmetres un identificador d'aplicació (APPKEY) , un identificador de dispositiu (DISP_KEY), un identificador únic (IDEN_KEY), un *array* amb dades sobre el dispositiu (DISP_INFO), un identificador del sistema operatiu del dispositiu que fa la petició (OS) i el màxim de “demos” que permet l'aplicació.

```
<?php function potConsumirDemo($app_key, $disp_key, $iden_key,
$disp_info, $os, $max_demos){ ... } ?>
```

En cas de no rebre un IDEN_KEY, el generarem amb la resta de dades obtingudes. En aquest cas l'identificador d'usuari sempre serà “0”, ja que si s'hagués identificat, ja tindria un IDEN_KEY vàlid.

Carrega les dades de la taula USERS_MOBILS per el IDEN_KEY corresponent. Si el IDEN_KEY es correspon a el d'un usuari registrat, carrega les seves dades i les publicacions a les que té accés.

Comprovem que el nombre de “demos” descarregues sigui inferior al màxim permès. Si no hem superat el màxim, incrementem el comptador de “demos” descarregades.

I finalment retornem les dades d'usuari, el IDEN_KEY, un “flag” que indica si hem de permetre la descàrrega i un camp de text que en cas d'error contindrà el missatge corresponent.

```
$struct = new xmlrpcval(
    array(
        'username' => new xmlrpcval($user-
>get('username')),
        'name' => new xmlrpcval($user->get('name')),
```

```

        'email'      =>      new      xmlrpcval($user-
>get('email')),

        'usuari_id'   => new xmlrpcval($user->id),

        'demos'      =>                                     new
xmlrpcval($userMobil['demos']),

        'edicions'    =>      new      xmlrpcval($edicionsAux,
$xmlrpcStruct),

        'iden_key' => new xmlrpcval($iden_key),

        'servir_demo' => new xmlrpcval($servirDemo),

        'error_msg' => new xmlrpcval($errorMsg)

    ), $xmlrpcStruct);

```


epusuaris.hashQuiosc

La crida *hashQuiosc*, rep com a paràmetres l'identificador d'aplicació (APPKEY) , un identificador d'usuari i un password (USERNAME i PASSWORD), només si està identificat, i una constant que indica una secció de El Quiosc (SECCIOQ).

Aquesta funció ens permetrà obrir en el navegador el web de El Quiosc a l'usuari i mostrar-li les seves dades sense haver-lo de tornar a autenticar.

```
<?php    function    hashQuiosc($appkey,    $username,    $password,
    $seccioQ){ ... } ?>
```

Amb l'identificador d'usuari i el password, generem un codi *hash* que determinarà l'usuari i amb la el valor de la secció generem la URL que hem d'entregar.

Finalment retornem un *array* amb la URL base de El Quiosc, el *hash* d'usuari i la URL composta per la secció i el *hash*.

Un cop tenim preparat el *plugin* l'instal·lem a el web que anomenem *Segur*, també basat en *Joomla* i habilitem les comunicacions amb *XML-RPC*.

En aquest cas les peticions també requeriran autenticació amb usuari i contrasenya.

8.5.5 Adaptar models per treballar amb XML-RPC

Un cop tenim els *plugins* en funcionament, ens cal adaptar l'aplicació per poder treballar amb el protocol XML-RPC.

El *SDK* de *iOS*, no disposa d'una llibreria nativa per treballar amb el protocol XML-RPC.

Depres de fer una mica de recerca i analitzar possibles alternatives, escollim “*The Cocoa XML-RPC Framework*”. Un framework simple i lleuger escrit en Objective-C, que ens permetrà treballar amb crides *XML-RPC* en *iOS*.

Modifiquem les funcions que utilitzaven peticions amb *NSURLConnection* a *El Quiosc* per les noves crides amb *XMLRPCConnection*.

Les funcions modificades són:

- demanarPublicacions (Classe Publicacions)
- demanarDarreresPublicacio (Classe Publicacions)
- getInfoPagine (Classe Publicacio)

Exemple d'utilització XMLRPC:

```
NSURL *URL = [NSURL URLWithString: kXMLRPC];

XMLRPCRequest *request = [[XMLRPCRequest alloc] initWithURL:
URL];

XMLRPCConnectionManager *manager = [XMLRPCConnectionManager
sharedManager];

NSArray *params = [NSArray arrayWithObjects:kAppId, dispId, nil];

[request setMethod: @"epquiosc.getEdicions" withParameters:
params];

[manager spawnConnectionWithXMLRPCRequest: request delegate:
self];
```

Definim la URL a la que hem de fer la petició “kXMLRPC”. Declarem una *XMLRPCRequest* inicialitzada amb la URL anterior.

Declarem una instància de *XMLRPCConnectionManager*, que és la classe encarregada de gestionar les connexions.

Creem un *array* amb els paràmetres necessaris per la petició. En aquest cas “epquiosc.getEdicions”. Assignem el mètode i els paràmetres a la petició.

I llancem la petició a través del “connection manager”. Observem també que podem definir qui serà el “delegate”, en aquest cas la mateixa classe.

El “delegate”, és la classe que implementarà les funcions necessàries per atendre els possibles missatges de retorn de la petició.

Les funcions bàsiques a implementar són:

- (void) request: (XMLRPCRequest*) request didReceiveResponse: (XMLRPCResponse*) response
- (void) request: (XMLRPCRequest*) request didFailWithError: (NSError *) error

La primera funció s’executarà en cas de rebre una resposta a la petició i la segona en cas de que la petició falli.

Un altre mètode que hem implementat, tot i que ho hem fet a nivell de la *XMLRPCConnection* per no haver-ho de fer cada vegada que llancem una petició, és:

- (void)connection: (NSURLConnection *)connection didReceiveAuthenticationChallenge: (NSURLAuthenticationChallenge *)challenge

Aquesta funció s’executarà en el moment que el servidor respon que la petició requereix ser autenticada amb l’usuari i password.

8.5.6 Identificar usuari a El Quiosc

Per poder gestionar la informació de l'usuari ens servirem de la classe *NSUserDefaults*. Funciona com un diccionari de dades, que ens permet guardar parelles de clau i valor.

La informació és guardada en una base de dades a disc, de manera que no la perdrem en cas de sortir de l'aplicació. Tot i que manté una còpia de les dades en cache per accedir-hi amb rapidesa.

Amb la declaració següent podrem accedir a les dades d'usuari en qualsevol punt de l'aplicació.

```
NSUserDefaults * settings = [NSUserDefaults  
standardUserDefaults];
```

Hi guardarem dades com, l'identificador únic IDEN_KEY, el nombre de “demos” consumides, la data de darrera verificació i les dades d'usuari en cas d'identificar-se.

En el moment d'intentar descarregar una publicació, comprovarem si disposem d'aquestes dades d'usuari.

En cas afirmatiu i de que l'edició que es vol descarregar sigui una de les edicions a les que està subscrit, s'iniciarà la descarrega.

En cas contrari, es presentarà un formulari sol·licitant les dades a l'usuari.

Aquest formulari també permetrà a l'usuari, veure la quantitat de descarregues gratuïtes que té disponibles i consumir-les.



Esquema conceptual pantalla identificació

Per gestionar aquesta nova pantalla amb les dades d'usuari creem un nou controlador que anomenem *LoginViewController*.

El controlador *LoginViewController* es basa en un *UITableViewController*, amb estil *UITableViewStyleGrouped*.

Implementem la funció *cellForRowAtIndexPath*, per definir a cada fila quina informació a de contenir.

A les dues primeres files dos *UITextField*, que permeten a l'usuari entrar el seu usuari i contrasenya.

La resta de cel·les les utilitzem com a botons.

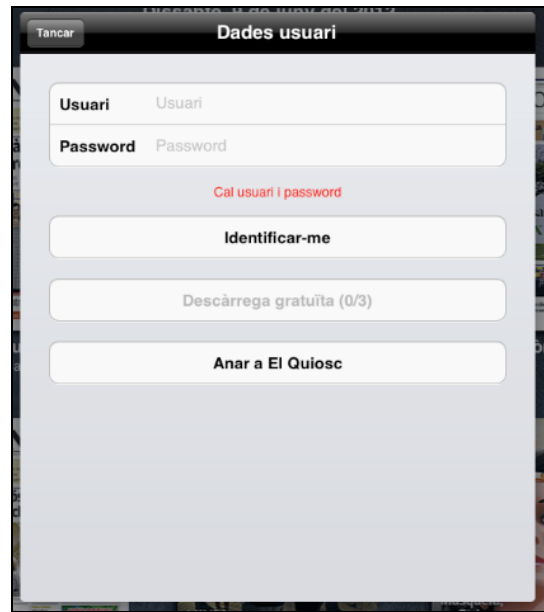
El primer, executarà el procés d'identificació en cas de no està identificat i el desconnectar-se en cas d'estar-ho.

El segon, només estarà actiu si el formulari s'ha mostrat al intentar descarregar una publicació i encara queden “demos” per consumir, permetrà fer una descarrega gratuïta o “demo”.

El tercer, obrirà el navegador amb el web amb informació sobre l'aplicació.

This is a screenshot of a mobile application's login screen. The title bar at the top is black with a white 'Tancar' button on the left and the title 'Dades usuari' in the center. The main content area has a light gray background. It contains two text input fields: the first is labeled 'Usuari' and the second is labeled 'Password'. Below these fields are three buttons: 'Identificar-me', 'Descàrrega gratuïta (0/3)', and 'Anar a El Quiosc'.

Captura pantalla identificació

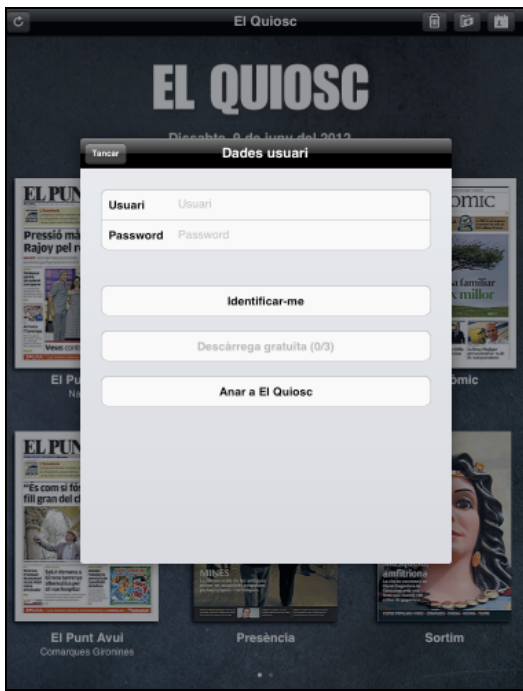
This is a screenshot of the same mobile application's login screen, but it displays an error message. The title bar and form fields are identical to the previous screenshot. However, a red error message 'Cal usuari i password' is displayed in the center of the screen, above the 'Identificar-me' button.

Captura pantalla identificació amb error

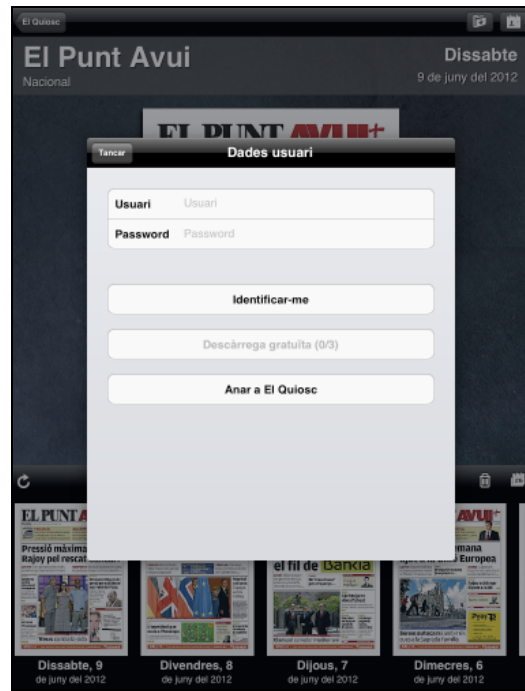
Controlem alguns errors, com intentar identificar-se sense haver introduït usuari i contrasenya. Els possibles errors que es produeixin durant les peticions d'identificació també es descriuran en aquesta pantalla.

Amb l'ajuda de Interface Builder, definim que la vista on contenim el formulari d'identificació i la resta de controls té una mida estàndard “*form sheet*”.

Al donar-li aquesta mida el podem presentar com una “modal view” sobre qualsevol altre vista i en qualsevol moment.



Exemple ModalView sobre PortadesVC



Exemple ModalView sobre PortadaVC

Afegim un botó a la part dreta de la barra de navegació per poder accedir el formulari amb comoditat.

8.5.7 Identificar-me

El procés d'identificació consisteix en realitzar una petició asíncrona a el servidor que anomenem *Segur* (<https://segur.elpunt.cat>), amb l'usuari, contrasenya i la resta de paràmetres necessaris per la funció *registrar* de *EPUsuaris*.

Si la resposta és positiva, guardem amb *NSUsersDefault* les dades de l'usuari i les edicions a les que tenim accés.

En cas contrari mostrarem el missatge d'error en el formulari d'identificació.



Captura pantalla identificació amb usuari identificat

8.5.8 Desconnectar-me

El procés de desconnectar-me o de logout, no requereix de cap petició externa. Simplement esborra les dades d'usuari del sistema amb *NSUsersDefault*.

8.5.9 Descarrega gratuïta

El procés de descarrega gratuïta o “demo”, consisteix en una petició asíncrona a *Segur*, per la funció *consumirDemo* amb els paràmetres requerits. La petició com ja hem vist, ens retorna si podem o no permetre la descàrrega. Igual que en el procés d'identificació, guardem les dades d'usuari de la resposta al sistema i comencem la descarrega o mostrem l'error.

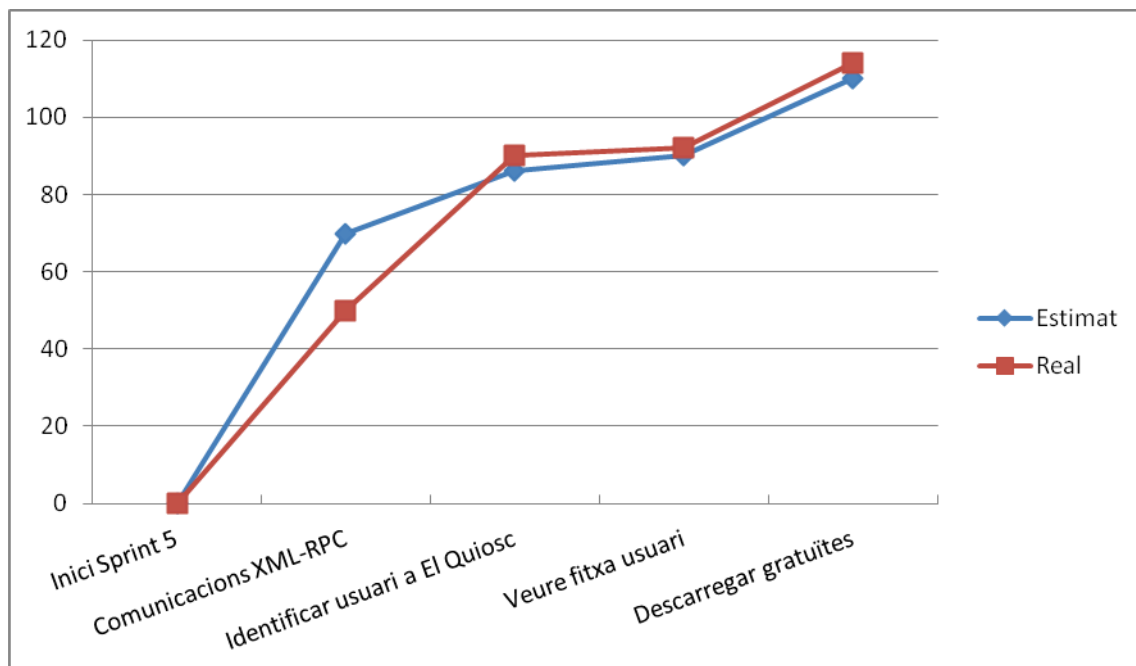
8.5.10 Anar a ...

El procés d'anar a *El Quiosc* (en aquest cas), consisteix en una petició asíncrona a *Segur*, per la funció *hashQuiosc* amb els paràmetres requerits. En funció de si estem identificats o no, del APP_ID i de la secció, ens retornarà una URL o una altre. Un cop rebuda la resposta, obrim el navegador amb aquesta adreça obtinguda. Si l'usuari està presentat a l'aplicació, també ho estarà al web que obrim al navegador.

8.5.11 Veure fitxa d'usuari

El procés per veure la fitxa d'usuari, és el mateix que el de *anar a El Quiosc*. Amb la icona d'informació que mostrem al costat del nom d'usuari un cop identificat, s'executa el procés. Només variarà el paràmetre que indica la secció que es vol veure.

8.5.12 Diagrama d'evolució del Sprint



8.5.13 Incidències destacables

Durant el desenvolupament dels processos d'identificació i de consum de “demos”, hem indicat que necessitem un identificador únic de dispositiu.

El SDK de iOS, disposa de la classe *UIDevice* amb el mètode *uniqueIdentifier*, que ens retorna un identificador UUID d'aquest tipus. El problema bé que a partir de la versió 5 de iOS, aquesta funció està obsoleta.

Això podria suposar que al intentar publicar la aplicació, podria ser refusada per Apple.

Ho hem resolt amb una llibreria externa, OpenUDID. Aquesta llibreria ens facilitarà la generació d'un identificador únic mitjançant l'adreça MAC del dispositiu.

També ens hem trobat amb problemes a l'hora de compilar el framework XMLRPC. Aquest framework està preparat per la versió 4 de iOS i no per la versió 5 que utilitzem nosaltres.

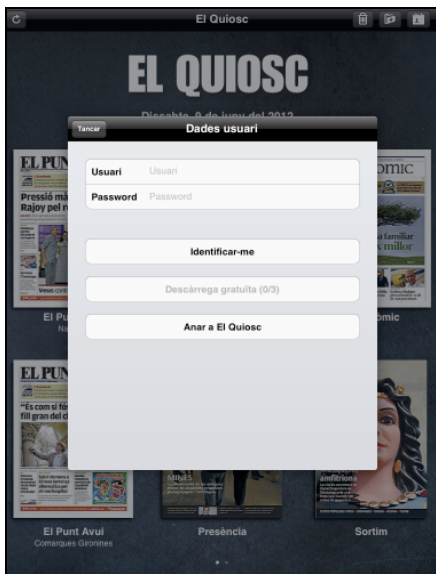
El problema és degut a la gestió automàtica de la memòria que fa iOS 5.

Ho hem solucionat afegint per a cada fitxer del framework, el paràmetre “-fno-objc-arc” a la llista de “*Compile Sources*” de la pestanya “*Build Phases*” del projecte.

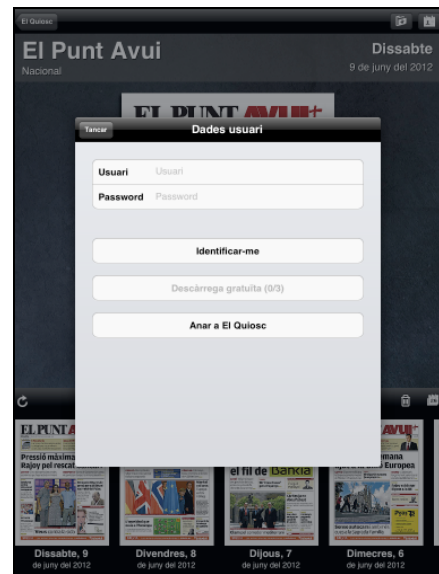
8.5.15 Demostració del producte al client

Expliquem al client que molts dels canvis aplicats durant aquest *Sprint* no són apreciables a simple vista.

Mostrem la pantalla amb les dades d'usuari sobre *PortadaVC* (captura 1) i *PortadesVC* (captura 2). Hi accedim des de la icona que hem afegit a la part dreta de la barra de navegació.



Captura 1



Captura 2

Mostrem el procés d'identificació d'usuari.

Usuari	driera@elpunt.com
Password	••••••
Identificar-me	

Captura abans d'identificar-se

Usuari	driera@elpunt.com
Password	••••••••
Desconnectar-me	

Captura després d'identificar-se

Ens desconnectem i procedim a consumir una de les descàrregues gratuïtes de les que disposem.

Què és

El Punt Avui

Nacional

Dijous 7 de juny del 2012

EL PUNT AVUI

Tancar

Dades usuari

Usuari

Password

Identificar-me

Descàrrega gratuïta (0/3)

Anar a El Quilosc

La fiscalia el fil de l'escàndol

Dijous, 7 de juny del 2012

Dimecres, 6 de juny del 2012

Dimarts, 5 de juny del 2012

Dilluns, 4 de juny del 2012

El Punt Avui

Dimecres
6 de juny del 2012

EL PUNT AVUI+

Dades usuari

Usuari

Password

Identificar-me

Descàrrega gratuïta (1/3)

Anar a El Quiosc

EL PUNTA

La fiscalia el fill de Bantia

Avui

El quiosc d'Avui

Dijous, 7 de juny del 2012

Dimecres, 6 de juny del 2012

Dimarts, 5 de juny del 2012

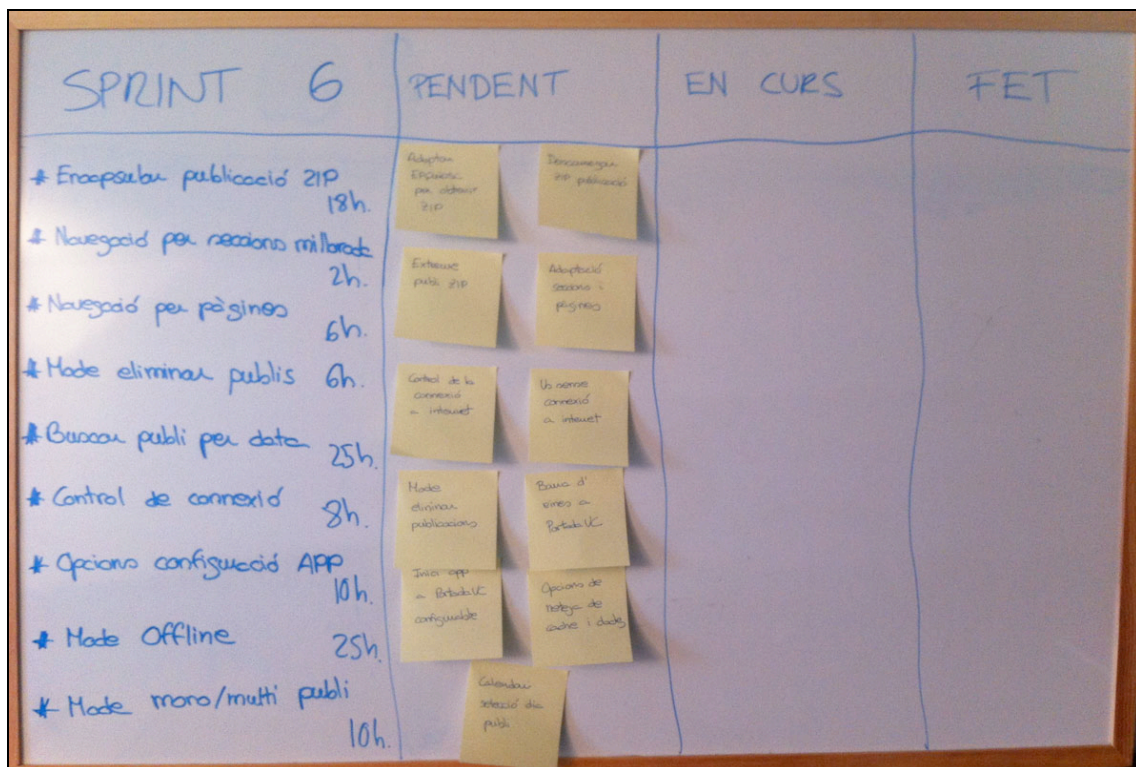
Dilluns, 4 de juny del 2012

[illegible]

104

8.6 Sprint 6

- Encapsular publicació ZIP
- Navegació per seccions
- Navegació per pàgines
- Mode eliminar publicacions descarregades
- Buscar publicació per data
- Control disponibilitat de connexió
- Opcions de configuració de l'aplicació
- Utilització de la aplicació offline
- Mode Mono-Multi publicació



Imatge pissarra inici Sprint 6

8.6.1 Històries d'usuari detallades

- Adaptar plugin EPQuiosc per obtenir URL del ZIP

- Descarregar el ZIP amb el contingut de la publicació
- Llibreria per treballar amb fitxers ZIP
- Extreure el contingut del ZIP i guardar-lo al seu lloc
- Adaptació llista de seccions i pàgines
- Controlar disponibilitat de la connexió
- Adaptar funcionalitats a ús sense connexió
- Mode eliminar publicacions descarregades
- Barra d'eines a PortadaVC
- Adaptar aplicació per treballar amb inici a PortadaVC
- Opció eliminar publicacions anteriors
- Opció netejar cache
- Opció esborrar totes les dades

8.6.2 Encapsular publicació ZIP

Durant aquesta història desenvolupem tots els canvis necessaris per descarregar la publicació com un únic fitxer ZIP i extreure'n el seu contingut.

Modifiquem el plugin *EPQuiosc*, perquè la petició a *epquiosc.getEdicio* ens retorni la URL on obtenir el fitxer ZIP de la publicació. La mateixa funció, comprovarà si existeix un ZIP amb la informació de la publicació que estem sol·licitant i en cas de no tenir-lo el generarà.

Adaptem la classe *Publicacio* perquè un cop rebí l'adreça URL del ZIP n'iniciï la descarrega.

Per treballar amb fitxers ZIP, utilitzarem la llibreria externa *SSZipArchive*.

Els fitxers continguts en el ZIP estan organitzats seguint el criteri que hem marcat per el correcte funcionament de l'aplicació. I no caldrà reordenar-los ni "reanomenar-los".

Observem que amb la compressió dels fitxers reduïm aproximadament un 10% el volum total de transferència. La compressió de la dades, i sobretot el fet de tenir una sola connexió en comptes d'una per a cada pàgina i imatge, millora notablement la velocitat de descarrega.

Una vegada hem extret el contingut del ZIP, l'eliminem per no duplicar l'espai ocupat per cada publicació.

8.6.3 Opcions de configuració de l'aplicació

Creem un *bundle* amb un llistat amb algunes opcions de configuració, que s'executaran automàticament al inici de l'aplicació.

La primera opció consisteix seleccionar un llindar de dies a partir del qual s'eliminaran les publicacions no utilitzades. Al iniciar l'aplicació, es comprovaran les publicacions descarregades i s'eliminaran les que la seva data d'utilització superi aquest valor. El valor per defecte és de 15 dies.

La segona opció, consisteix en eliminar les imatges que es van descarregant amb l'ús de l'aplicació i que no formen part de la publicació en si. Al seleccionar l'opció i arrancar l'aplicació, s'eliminaran els fitxers i es desactivarà l'opció altre vegada.

La tercera opció, té un comportament igual que l'anterior, però eliminant totes les dades de l'aplicació. Elimina dades d'usuari, publicacions descarregades, imatges descarregades i la base de dades de l'aplicació.

Les diferents accions que podem configurar en el menú d'opcions del dispositiu, s'executaran al iniciar l'aplicació. El control de si cal executar o no alguna de les accions disponibles, es realitzarà a la classe *AppDelegate*. Concretament en la implementació de la seva funció *didFinishLaunchingWithOptions*.

8.6.4 Navegació per pàgines i seccions

Per cada pàgina de la publicació, hem obtingut un petit JPG que anomenarem miniatura. Aquestes pàgines en miniatura ens permetran desplegar un índex per facilitar a l'usuari la navegació entre les diferents pàgines de la publicació i creant una visió general d'aquesta.

Amb l'ajuda de *Interface Builder*, creem un *UIView* que contindrà dues *UITableViews* horitzontals, amb les seccions i les pàgines en miniatura de la publicació.



Captura dels llistats de seccions i miniatures de pàgina

Igual que les anteriors vegades que hem adaptat la *UITableView* com a taules horitzontals, hem d'utilitzar les nostres pròpies cel·les. En aquest cas creem *MiniPaginaCell* i *SeccionsCell*.

Gràcies a *UITabGestureRecognizer*, amb un toc al centre de la pantalla mostrarem i amagarem els controls de seccions i miniatures igual que fem amb la barra de navegació.



Captura controls amagats



Captura controls visibles

Al escollir una miniatura o secció carregarem a *PDFScrollViewController* la pàgina corresponent.

8.6.5 Mode eliminar publicacions descarregades

Aquesta funcionalitat ens ha de permetre eliminar les publicacions descarregades sense haver d'accedir a la llista de publicacions descarregades.

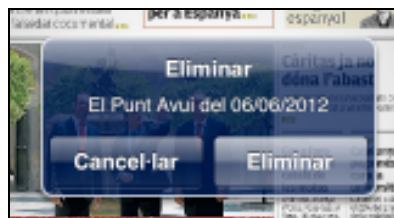
Afegim un botó eliminar a la interfície de *PortadaVC* i *PortadesVC*, que activarà el “mode eliminar”. El moviment de les portades ens indicarà que estem en aquest mode i una petita icona vermella sobre les publicacions descarregades, ens indicarà les que poden ser eliminades.



Captura “mode eliminar”

El moviment de les portades, similar al del *Springboard* de *iOS* quan volem eliminar una aplicació, el generem amb *CABasicAnimation*.

Al prémer sobre la publicació descarrega que volem eliminar, demanem la confirmació de l'usuari i l'eliminem.



Missatge confirmació eliminació

Per sortir del “mode eliminar”, tornarem a prémer el botó amb la icona de la paperera o en una zona sense publicacions.

8.6.6 Buscar publicació per data

Per accedir d'una manera més ràpida i precisa a la publicació d'una data determinada, afegim un calendari que ens permeti seleccionar una data.

La data màxima disponible en el calendari, l'obtenim de la primera petició a *epquiosc.getDarrersEdicio*.

El calendari és un control de la classe *UIDatePicker*, contingut en un controlador *CalendariViewController* que mostrarem sobre un *UIPopoverView*.



Captura del calendari sobre el popoverview

Un cop seleccionada la data, repetim la crida a *epquiosc.getDarrersEdicio* amb la nova data d'inici, que ens retornarà les publicacions a partir de la data seleccionada. Si la data escollida és la que estem mostrant, no farem la petició.

Per tornar a les darreres publicacions, les refrescarem amb el boto de la part dreta de la barra d'eines que hem afegit sobre les miniatures de portades a la part inferior.

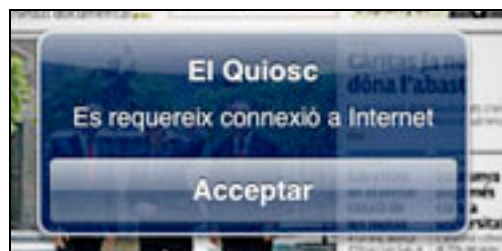
8.6.7 Control disponibilitat de connexió

Implementem un control per evitar fer peticions quan no disposem de connexió a internet.

Per conèixer l'estat de la connexió, utilitzem una llibreria externa anomenada *Reachability*. Ens permet detectar si tenim connexió via WiFi, si tenim connexió via 3G o si no tenim connexió a internet.

Adaptem les funcionalitats que hem anat desenvolupant i que requereixen de connexió a internet, per comprovar que el dispositiu disposa de connexió abans de fer qualsevol petició.

Si la connexió està disponible s'executa la petició, altrament mostrem un missatge informant que per realitzar aquesta acció es requereix de connexió a internet.



Captura missatge

Afegim a les opcions de l'aplicació, la possibilitat de autoritzar només les peticions en cas de disposar de connexió WiFi, evitant així l'ús de la xarxa 3G.

8.6.8 Utilització de la aplicació offline

Tot i no disposar de connexió per obtenir noves publicacions, ens interessa que l'usuari pugui consultar les publicacions que hagi descarregat amb anterioritat.

Per això, guardem a la BDD la informació de les publicacions que obtenim quan disposem de connexió, per poder-les recuperar quan no ens sigui possible obtenir-les de les peticions al servidor.

El canvi ens implica afegir un camp a la BDD de PUBLICACIONS per indicar si ha sigut descarregada o no. Fins el moment la taula només contenia les PUBLICACIONS descarregades.

També implica la creació d'una nova taula PORTADES que contindrà les darreres publicacions mostrades a *PortadesVC*. La data que mostrem a la part superior, mostrarà la data de la darrera vegada que hem obtingut portades.

D'aquesta manera podrem navegar per les publicacions de l'aplicació sense necessitat de connexió. Nomes amb la limitació d'algunes funcionalitats com la cerca per una data concreta o carregar publicacions que no havíem visualitzat prèviament.

8.6.9 Mode mono/multi publicació

Un altre dels canvis importants acordats durant el replantejament de l'aplicació, era el de tenir una aplicació que es pogués adaptar a els diferents productes del grup ja fos de manera conjunta o individual.

El que ens permetrà tenir una app per cada producte (publicació) o conjunt d'ells, segons ens interressi.

A nivell de desenvolupament això ens suposa, adaptar el controlador *PortadaVC*, per funcionar com a controlador principal de l'aplicació igual que ho fa *PortadesVC*. En un fitxer de paràmetres de l'aplicació definirem els valors que puguin ser canviants en les diferents versions, com puguin ser el nom de l'aplicació o la APPID.

Com ja hem vist en el desenvolupament dels plugins que ens serviran els continguts, la APPID serà el que decidirà quin contingut visualitzem a l'aplicació.

Finalment caldrà personalitzar en cada cas l'aparença de les vistes amb *Interface Builder*.

Per facilitar aquest procés de personalització, agrupem els fitxers que caldrà tocar en les diferents versions dins una carpeta "Especifics".

Definir el punt d'entrada de l'aplicació a *PortadaVC* o *PortadesVC*, determinarà si l'aplicació és mono-publicació, és a dir només ens permet consultar un sol tipus de publicació o multi-publicació.



Captures versió El Quiosc

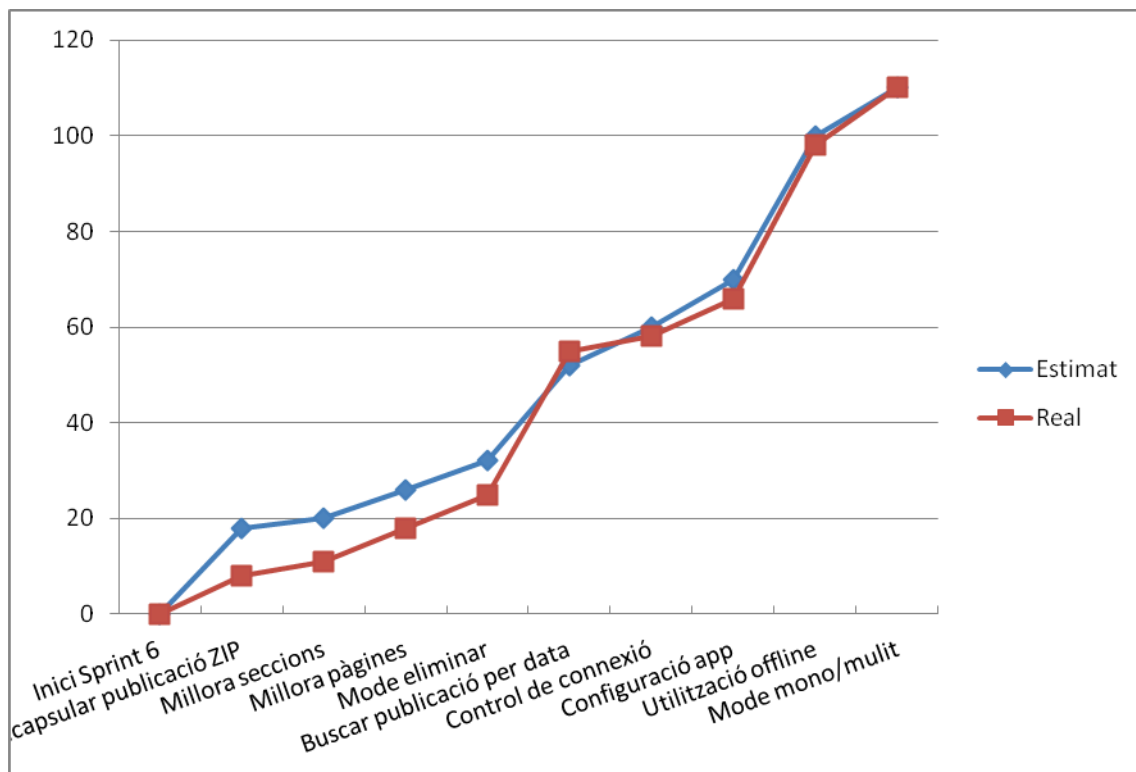


Captures versió El 9



Captures versió Presència

8.6.10 Diagrama d'evolució del Sprint



8.6.11 Incidències destacables

Al compilar les llibreries *SSZipArchive*, igual que amb el framework XML-RPC, ens hem trobat amb algun problema de compatibilitat per culpa de la versió de iOS utilitzada.

Ho hem solucionat afegint per a cada fitxer de la llibreria, el paràmetre “*-fno-objc-arc*” a la llista de “*Compile Sources*” de la pestanya “*Build Phases*” del projecte.

8.6.12 Reunió final Sprint

Al finalitzar el *Sprint* hem assolit els objectius marcats inicialment i gairebé la totalitat de les funcionalitats recollides en el *Product Backlog*.

Destaquem la millora de temps en la descàrrega de les publicacions, gràcies a la compressió i encapsulament de les dades.

Durant la reunió es proposa, configurar una versió definitiva de l'aplicació per el producte Presència. Que ens servirà per veure com funciona el procediment de validació de l'aplicació per part d'Apple i un cop aprovada, per veure com és rebuda per part dels subscriptors i usuaris.

SPRINT 6	PENDENT	EN CURS	FET
* Encapsula publicació ZIP 18h.			Adaptar l'aplicació per a iOS
* Navegació per seccions mil·lbrats 2h.			Encapsulament de publicacions
* Navegació per pàgines 6h.			Exhibició de publicacions
* Mode eliminar publicis 6h.			Adaptació de l'aplicació a iOS
* Buscador públic per data 25h.			Control de la connexió a internet
* Control de connexió 8h.			Mode offline
* Opcions configuració APP 10h.			Mode d'usuari
* Mode Offline 25h.			Mode d'usuari
* Mode mono/multi públic 10h.			Mode d'usuari

Imatge pissarra fi Sprint 6

8.6.13 Demostració del producte al client

Mostrem al client la possibilitat de configurar l'aplicació com a mono-publicació o multi-publicació.



Captura multi-publicacio



Captura mono-publicacio

Observem que a la captura mono-publicació (*PortadaVC*), no tenim la possibilitat de canviar de tipus de publicació.

Mostrem la funcionalitat de demanar una publicació per una data determinada (captures 1 i 2).



Captura 1



Captura 2

Descarreguem la publicació (captura 3) i mostrem la nova navegació per pàgines i seccions (captura 4).



Captura 3



Captura 4

Inhabilitem la connexió a internet des del menú d'opcions del dispositiu i comprovem que podem navegar per les publicacions que ja havíem consultat.

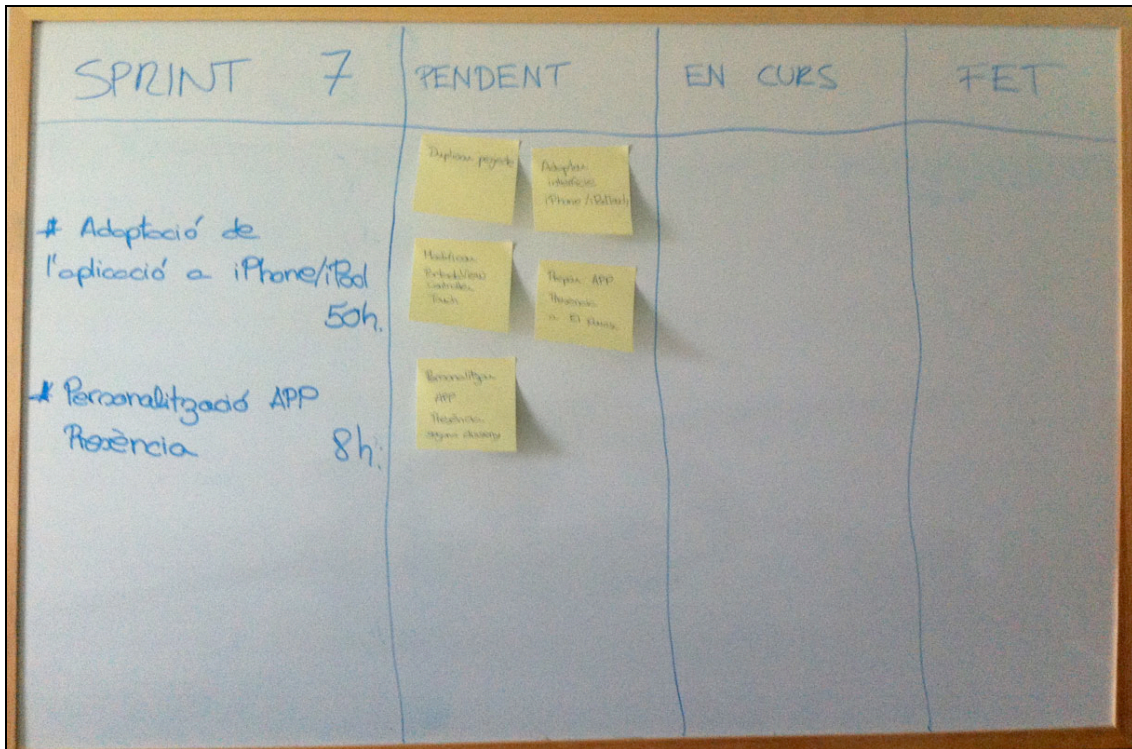
A la captura 5, observem el missatge al intentar descarregar una publicació sense tenir connexió.



Captura 5

8.7 Sprint 7

- Adaptació de l'aplicació a iPhone/iPod Touch
- Personalització de l'aplicació per Presència



Imatge pissarra inici Sprint 7

8.7.1 Històries d'usuari detallades

- Duplicar el projecte
- Adaptar interfície a iPhone/iPod Touch
- Modificar *PortadaViewController*
- Preparar APP Presència a El Quiosc
- Personalitzar APP Presència segons disseny

Adaptació de l'aplicació a iPhone/iPod Touch

Decidim duplicar el projecte i adaptar la interfície a els dispositius iPhone i iPod Touch.

En alguns cassos tant sols haurem d'adaptar la mida dels elements, però en algun altre haurem de replantejar la disposició d'alguns elements per adaptar-nos a les noves mides de la pantalla.

Seguint l'ordre lògic de l'aplicació, primer de tot adaptarem el controlador *PortadesViewController*.

L'únic canvi significatiu, és la variació del nombre de publicacions visibles per "pàgina", passant a mostrar 4 publicacions per les 6 que veiem a la versió per *iPad*.



Captura PortadesVC



Captura MevesVC

La llista de publicacions descarregades, gestionada per *MevesViewController*, que a la versió per iPad veiem sobre un *PopoverView*, passem a visualitzar-la a pantalla completa.

La pantalla d'identificació i dades d'usuari, igual que en el cas anterior passem a visualitzar-la a pantalla completa.

El controlador *PortadaViewController*, és el que més canvis presentarà respecte la versió per iPad. Les dimensions més reduïdes de la pantalla, no ens permeten mostrar completament les "subvistes" *PublicacioVC* i *PublicacionsVC* al mateix temps.

Per defecte amaguem la llista de portades de publicacions anteriors, que l'usuari podrà desplegar quan ho necessiti, mitjançant un botó de la barra d'eines.



Captura pantalla identificació



PortaVC amb llistat desplegat

A la visualització de la publicació, gestionada per *PDFScrollViewController*, simplement adaptarem les vistes a les noves mides de pantalla.

8.7.2 Personalització de l'aplicació per Presència

Afegim les dades necessàries a les taules de El Quiosc, per crear una APPID que ens retornarà únicament la publicació de Presència.

Configurem l'aplicació com a mono-publicació i perquè utilitzi la APPID de Presència.

```
#ifndef ElQuiosc_constants_h
#define ElQuiosc_constants_h

#define kAppId @"101"
#define kTitolApp @"Presència"
#define kURLApp @"http://www.elquiosc.cat/Quiosc.php?do=login"
#define kOS @"ios"
#define en @"26/10/2008"

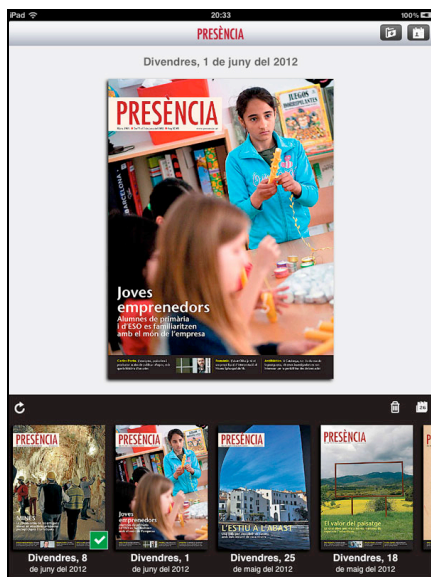
// #define kXMLRPC @"http://elektra.elpunt.local/elpunt_web/xmlrpc/"
#define kXMLRPC @"http://admin.elpunt.cat/xmlrpc/"
// #define kXMLRPCSegur @"http://elektra.elpunt.local/segur/xmlrpc/"
#define kXMLRPCSegur @"https://segur.elpunt.cat/xmlrpc/"

#define kURLQuiosc @"http://quiosc.elpunt.net"
#define kURLPortadesDia @"http://quiosc.elpunt.cat/Quiosc.php?do=portadesDelDia"

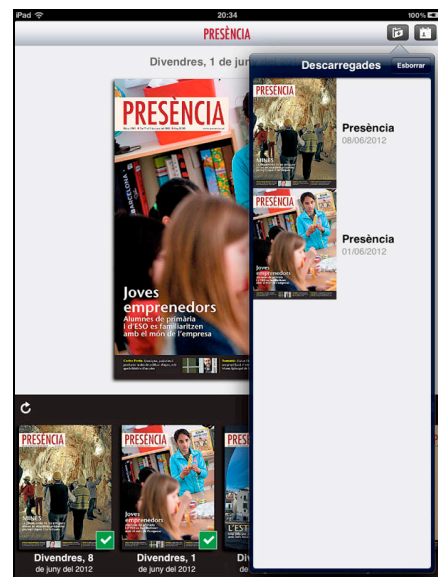
#define kNumDarreres @"25"
```

Captura dades de configuració

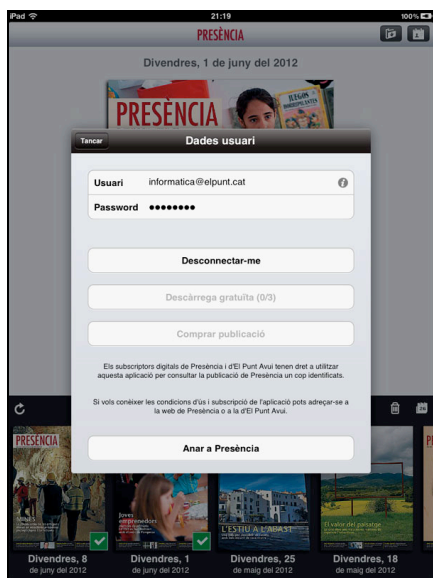
Personalitzem l'aplicació amb els colors i logotip característic de la publicació Presència.



Captura PortadaVC



Captura publicacions descarregades

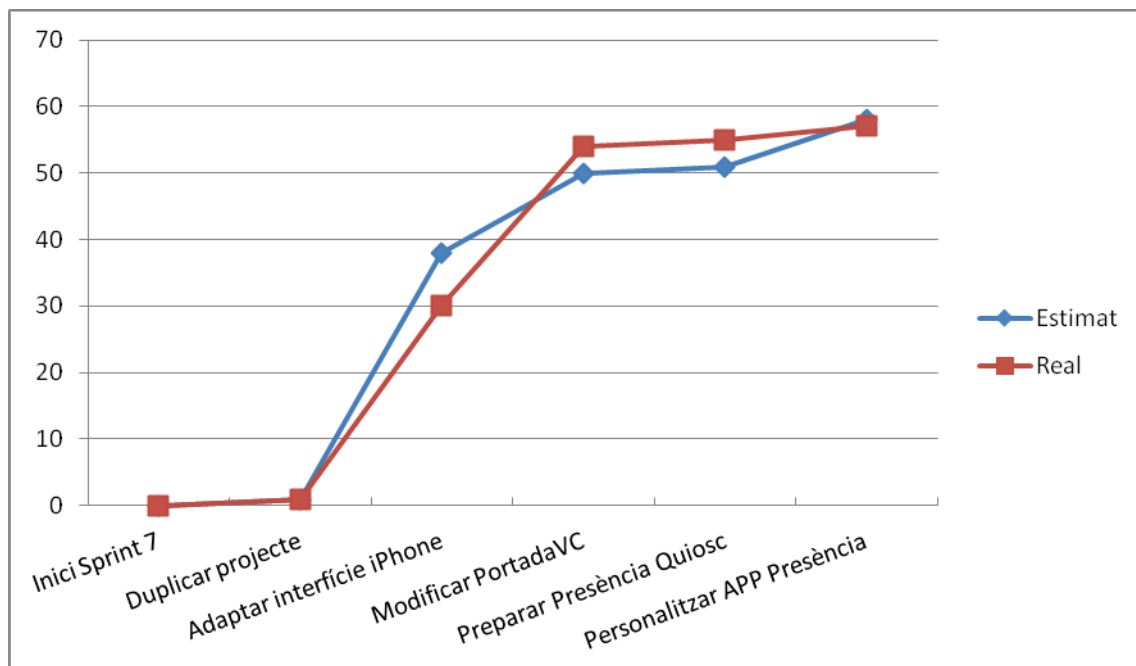


Captura presentació usuari



Captura visualització publicació

8.7.3 Diagrama d'evolució del Sprint



8.7.4 Incidències destacables

Al personalitzar alguns elements, ens hem trobat amb el problema que algunes propietats depenien del codi de l'aplicació. Com que volem evitar mantenir el codi per cada APP, extraiem els valors en un fitxer que anomenem "recursos.h".

Aquest fitxer contindrà tots els valors que puguin variar d'una configuració a una altre.

```
#ifndef ElQuiosc_constants_h
#define ElQuiosc_constants_h

#define kAppId @"101"
#define kTitolApp @"Presència"
#define kURLApp @"http://www.elquiosc.cat/Quiosc.php?do=login"
#define kOS @"ios"
#define en @"26/10/2008"

// #define kXMLRPC @"http://elektra.elpunt.local/elpunt_web/xmlrpc/"
#define kXMLRPC @"http://admin.elpunt.cat/xmlrpc/"
// #define kXMLRPCSegur @"http://elektra.elpunt.local/segur/xmlrpc/"
#define kXMLRPCSegur @"https://segur.elpunt.cat/xmlrpc/"

#define kURLQuiosc @"http://quiosc.elpunt.net"
#define kURLPortadesDia @"http://quiosc.elpunt.cat/Quiosc.php?do=portadesDelDia"

#define kNumDarreres @"25"
```

Captura fitxer de recursos

8.7.5 Demostració del producte al client

Mostrem al client les diferents funcionalitats de l'aplicació sobre un dispositiu iPhone. Navegació entre publicacions (captures 1 i 2).



Captura 1



Captura 2

Descarrega (captura 3 i 4), llistat de publicacions (captura 5) i visualització d'una publicació (captura 6).



Captura 3



Captura 4



Captura 5



Captura 6

I finalment, el calendari per escollir una data determinada (captura 7) i la pantalla amb les dades d'usuari (captura 8).

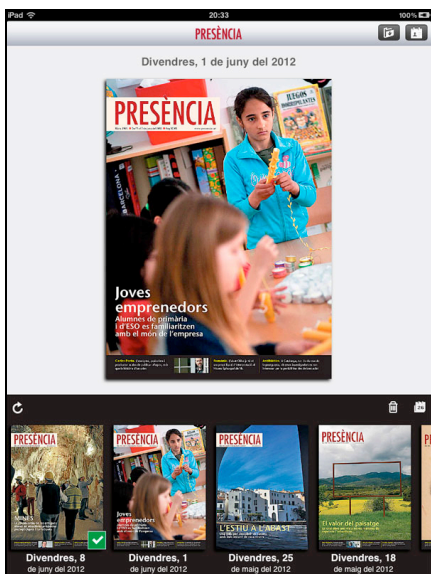


Captura 7

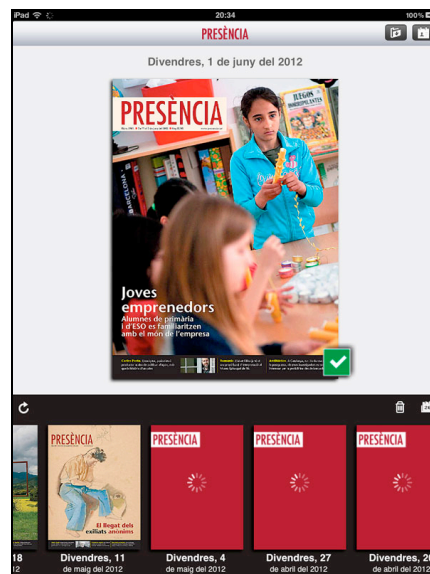


Captura 8

Mostrem al client el funcionament de l'aplicació personalitzada per el producte Presència. Igual que en la presentació anterior, navegació entre publicacions (captures 1 i 2).

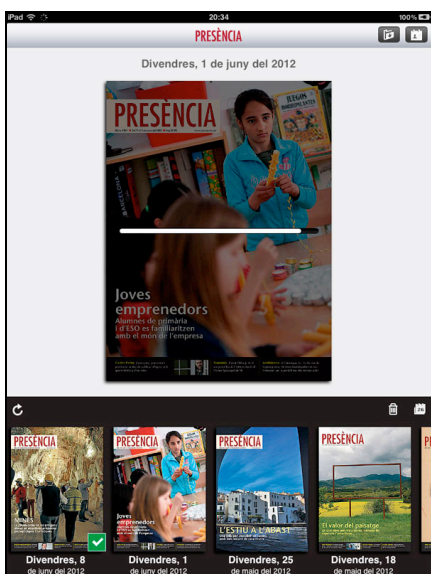


Captura 1

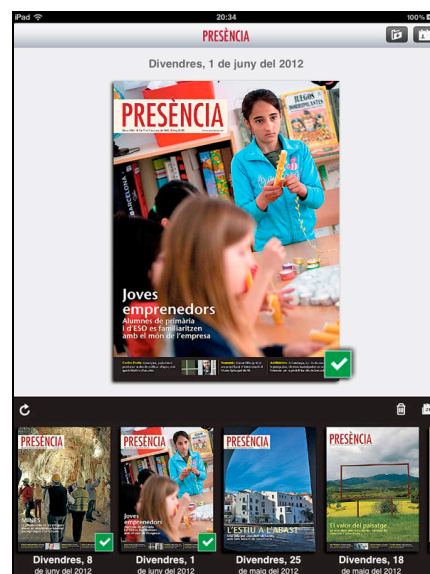


Captura 2

Descarrega (captura 3 i 4), llistat de publicacions (captura 6) i visualització d'una publicació (captura 5).



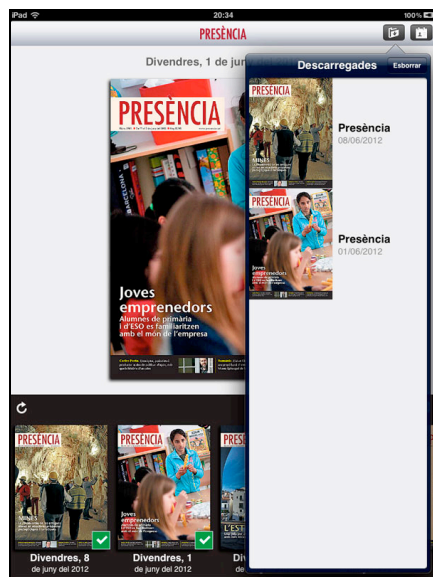
Captura 3



Captura 4



Captura 5

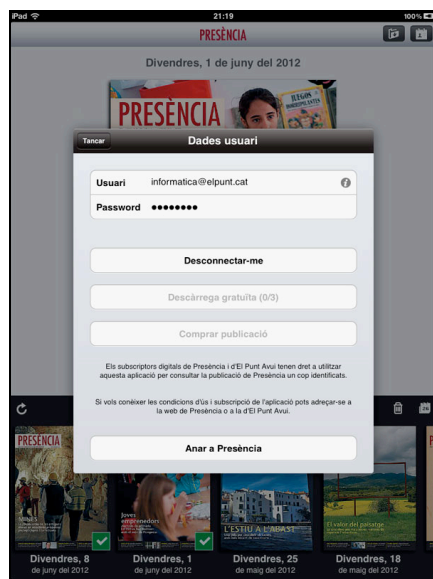


Captura 6

I finalment, el calendari per escollir una data determinada (captura 7) i la pantalla amb les dades d'usuari (captura 8).



Captura 7



Captura 8

9 Implantació

Per tal de distribuir la nostra APP als clients, l'hem de publicar a la botiga d'aplicacions d'Apple.

Per poder fer efectiva la publicació de l'aplicació a la *App Store* hem de complir una sèrie de requisits.

El primer de tots és disposar d'una llicència de desenvolupador. Aquesta llicència vinculada a un *AppleID*, ens permet accedir a la web per desenvolupadors d'Apple.

Des de la web, podem accedir a una secció on podem gestionar els nostres propis certificats digitals.

Hi ha diferents tipus de certificats, un tipus per desenvolupadors autoritzats i dos més que Apple anomena de *Provisioning*, un per desenvolupament i un altre per distribució.

Durant el desenvolupament de l'aplicació hem utilitzat el certificat per a desenvolupament per tal de poder provar el codi en dispositius reals.

Per tal de firmar l'aplicació i pujar-la als servidors d'Apple, generem un certificat de "*Provisioning for distribution*".

Des del web de desenvolupadors d'Apple, accedim a *iTunesConnect*, on podrem administrar totes les dades referents a les nostres apps.

Accedim a la secció "*Manage Your Applications*". On procedim a donar d'alta la nostre nova app.

Omplim el formulari amb les dades de la aplicació, nom, *SKU* (codi intern per l'aplicació), *Bundle ID* (identificador per l'aplicació que ja hem creat al generar el certificat), la versió i l'idioma per defecte.



Captura iTunes Connect

Especifiquem el contingut de la nostre aplicació per determinar l'edat recomanada.

Pugem les imatges sol·licitades per mostrar a la *App Store*, captures de l'aplicació en funcionament i la icona de la app amb alta resolució.

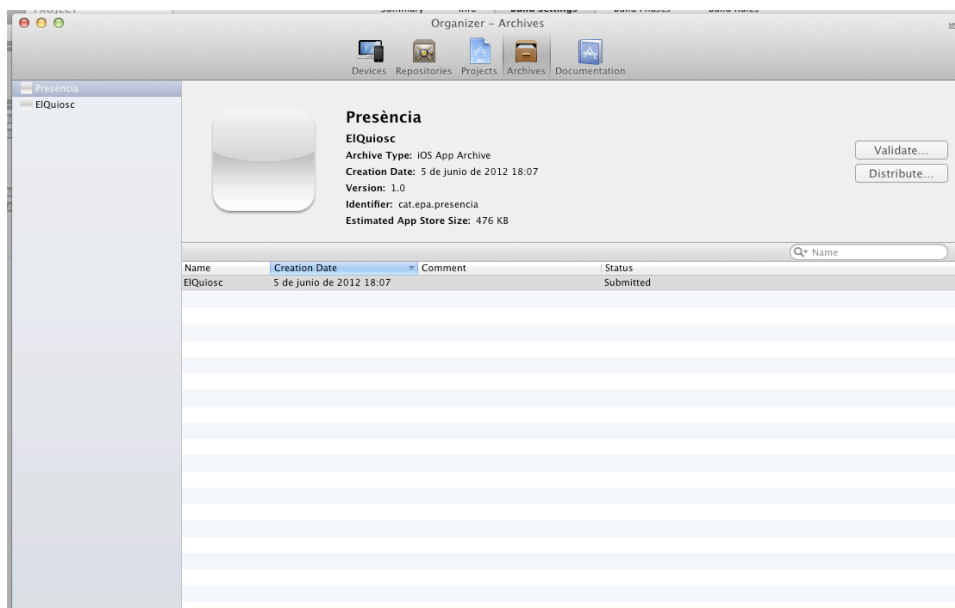
Instal·lem el certificat al XCode de la nostre màquina, seleccionem el nou certificat en les opcions de compilació del nostre projecte, especifiquem el *Bundle ID* i compilem amb l'opció "Archive".



Captura opcions del projecte

Des de la pantalla “Organizer” a la pestanya “Archives” procedim a enviar l’aplicació als servidors d’Apple.

Sol·licitem la validació del projecte, amb el botó “Validate”. Un assistent ens sol·licitarà les nostres dades i comprovarà que el projecte compleix tots els requisits per ser enviat.

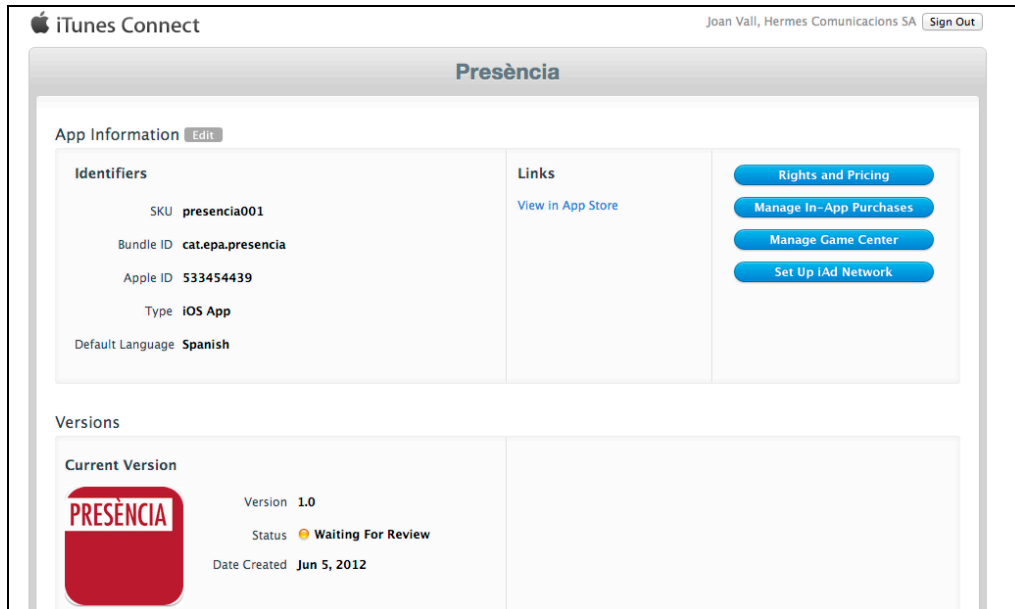


Captura de Organizer

Un cop l’assistent ens confirma que el projecte és vàlid, procedim a enviar la aplicació amb el botó “Distribute”.

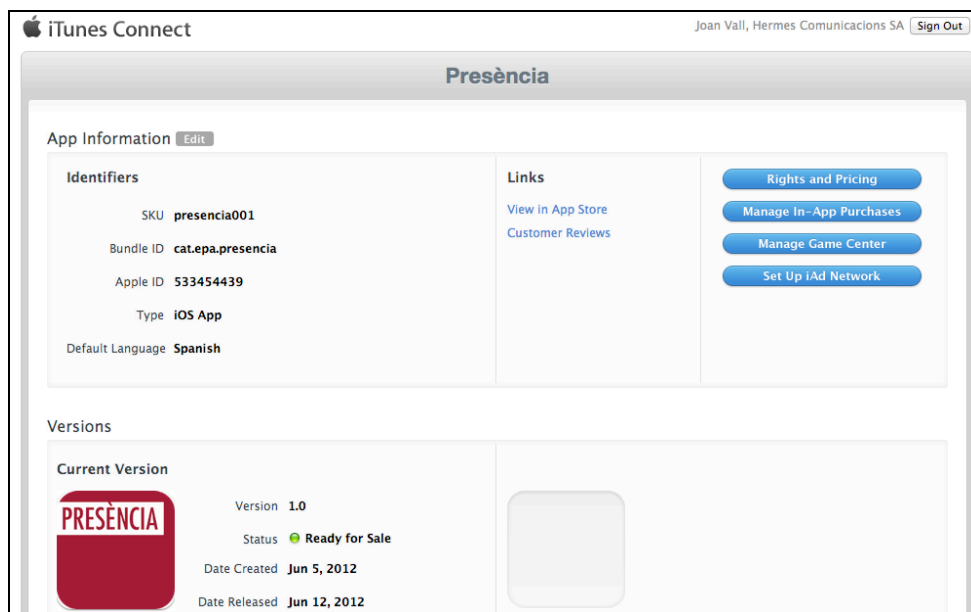
Al finalitzar el procés, rebrem la notificació que el projecte ha estat enviat correctament i que des del web de desenvolupador d’Apple, podrem fer el seguiment dels estats en que es troba.

A partir d'aquest moment, l'aplicació està en espera per ser revisada per Apple. Durant aquest procés es decidirà si l'aplicació és publicada o si cal revisar algun aspecte.



Captura iTunes Connect – Waiting for review

Un cop la app hagi estat aprovada, serà accessible per els usuaris des de la App Store.

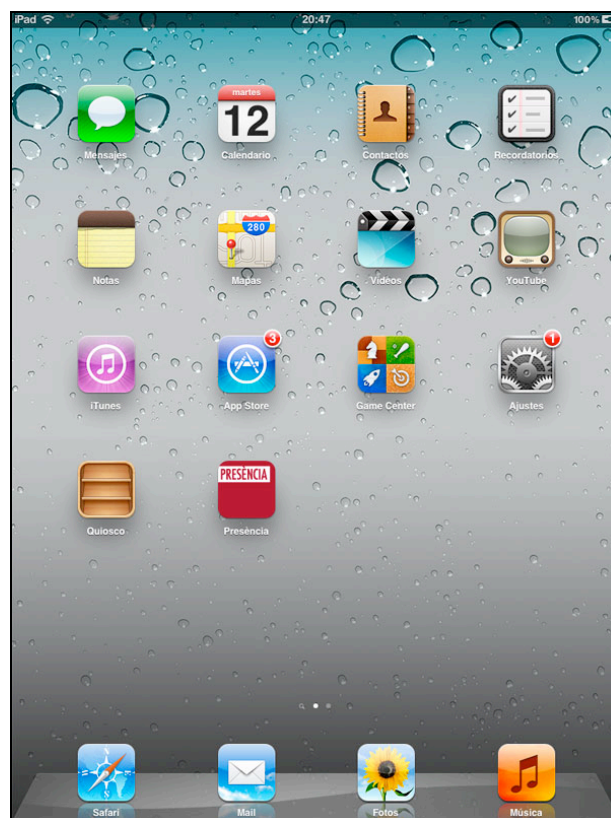


Captura iTunes Connect – Ready for sale

Finalment, veiem la app aprovada i llesta per ser descarregada des de la App Store.



Captura de la app de Presència a la App Store



Captura app de Presència en un iPad

10 Conclusions

Aquest projecte, ha acomplert amb èxit els objectius marcats inicialment, tant a nivell personal com de l'empresa.

Hem desenvolupat una aplicació compatible amb dispositius iOS. Totalment adaptable tant a nivell visual com de continguts als diferents productes de l'empresa i amb la possibilitat de mostrar-los en conjunt o de manera individual.

Hem aconseguit una interfície senzilla i clara que al mateix temps permet una gestió i una lectura còmodes de les publicacions de l'usuari.

Hem desenvolupat un sistema d'identificació que donarà accés als subscriptors a les seves publicacions. I permetrà a usuaris que encara no ho són provar alguns dels nostres productes.

Hem adaptat la plataforma que ens serveix els continguts per treballar d'una manera més eficient i segura, però pensant en tot moment amb la compatibilitat amb la resta de tecnologies. Implantant un nou protocol de comunicació entre l'aplicació i la plataforma de distribució.

Hem acabat amb la personalització i publicació de la APP de Presència.

A nivell personal, he obtingut coneixements i experiència en la utilització del SDK de iOS. Que m'han permès desenvolupar des del primer pas fins a l'últim l'aplicació desitjada. Acabant finalment amb el procés de validació i publicació a centre de distribució d'aplicacions d'Apple (App Store).

11 Treball futur

A part de la personalització i implantació de diferents apps per la resta de productes del grup, hi ha diferents funcionalitats que poden ser desenvolupades en un futur.

Compres In-App de publicacions

Aquesta funcionalitat permetria a l'usuari adquirir una publicació sense necessitat de ser subscriptor i un cop esgotades les descarregues gratuïtes. Vendríem a l'usuari una o varies descarregues, mitjançant el sistema de compra In-App que ens proporciona Apple.

Navegació per titulars

Una futura funcionalitat a desenvolupar, seria la possibilitat de navegar per la publicació mitjançant una llistat de titulars, semblant a la navegació per seccions ja implementada.

Cerques dins la publicació

Una futura funcionalitat, podria ser la cerca d'una paraula o paraules determinades dins la publicació.

Recomanació de notícies

Una altre funcionalitat, podria ser la de poder recomanar o compartir via xarxes socials o correu electrònic una notícia.

Reconeixement de notícies dins la publicació

El reconeixement de notícies, consistiria en identificar les notícies d'una pàgina determinada. Això ens obriria un altre ventall molt ampli de futures funcionalitats a desenvolupar com per exemple, continguts multimèdia associats a una notícia determinada o la possibilitat d'oferir un sistema de lectura en format pla.

12 Bibliografia

http://ca.wikipedia.org/wiki/Metodologia_%C3%A0gil

<http://www.proyectosagiles.org/que-es-scrum>