Memòria - PFC

Comptador de clicks *(clickerMaster)*

Aplicació per a dispositius mòbils usant HTML5, CSS i *Javascript*

Autor: Roberto Ruiz Moreno

Consultor: Roman Roset Mayals

PFC - 2n cicle Enginyeria Informàtica

2013-2014 (1)

Índex

Índex de figures

1. Introducció

Actualment en el món de la informàtica estan apareixent noves tecnologies i nous paradigmes pel que fa a la programació d'aplicacions. A més, cada cop més aquestes aplicacions estan pensades per executar-se en els, tan de moda, dispositius mòbils (*smartphones*) i les tauletes.

Degut a aquest mon en constant canvi i evolució, els enginyers informàtics ens hem d'adaptar contínuament a aquelles tecnologies noves que van sorgint, i fruit d'aquesta necessitat, va sorgir la idea de realitzar el PFC en aquesta àrea, per tal d'investigar, aprendre i saber aplicar aquelles tecnologies punteres en el moment actual, i més concretament, aquelles que es centren en els dispositius mòbils.

Una aplicació senzilla, però alhora útil i funcional, i amb la qual es poguessin aplicar les noves tecnologies basades en HTML5, és el millor escenari per posar-les en pràctica.

Per resumir-ho en una frase, el present PFC tracta de realitzar un comptador de clics per a dispositius mòbils.

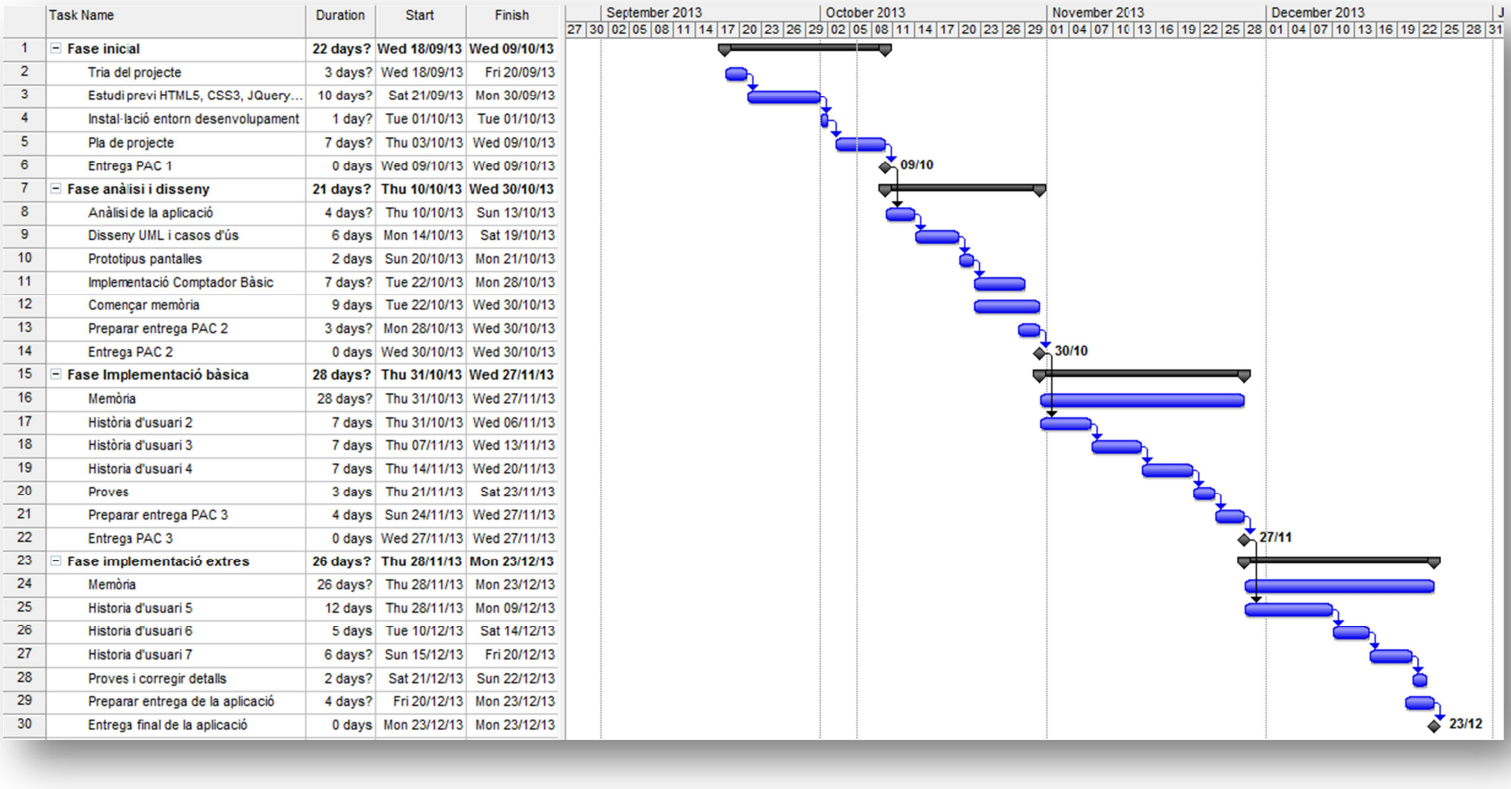
Entrant més en detall, es tracta d'una aplicació que permetrà a l'usuari incrementar un comptador cada vegada que premi un botó que apareixerà per la pantalla del seu dispositiu. Evidentment, serà l'usuari qui decideixi el que està comptant en aquell moment, ja siguin persones que entren en un local, cigarretes que fumo durant el dia, trucades que faig durant el mes, etc...

Aquesta eina pot ser útil per dur aquells comptes quotidians que molta gent necessita fer, i que no vol apuntar-s'ho en un paper o recordar-se'n de memòria, sinó que vol fer-ho fent servir el seu telèfon intel·ligent.

A més, el comptador aportarà d'altres funcionalitats que el pot convertir en una eina molt interessant en el dia a dia.

2. Planificació

Durant el decurs del quadrimestre, s'han dut a terme entregues parcials, afegint en cada entrega alguna novetat més en la aplicació. El calendari que s'ha assolit, fins arribar a l'entrega final, es mostra en aquest diagrama de Gantt.



	Task Name	Duration	Start	Finish
1	**Fase inicial**	**22 days?**	**Wed 18/09/13**	**Wed 09/10/13**
2	Tria del projecte	3 days?	Wed 18/09/13	Fri 20/09/13
3	Estudi previ HTML5, CSS3, JQuery...	10 days?	Sat 21/09/13	Mon 30/09/13
4	Instal·lació entorn desenvolupament	1 day?	Tue 01/10/13	Tue 01/10/13
5	Pla de projecte	7 days?	Thu 03/10/13	Wed 09/10/13
6	Entrega PAC 1	0 days	Wed 09/10/13	Wed 09/10/13
7	**Fase anàlisi i disseny**	**21 days?**	**Thu 10/10/13**	**Wed 30/10/13**
8	Anàlisi de la aplicació	4 days?	Thu 10/10/13	Sun 13/10/13
9	Disseny UML i casos d'ús	6 days	Mon 14/10/13	Sat 19/10/13
10	Prototipus pantalles	2 days	Sun 20/10/13	Mon 21/10/13
11	Implementació Comptador Bàsic	7 days?	Tue 22/10/13	Mon 28/10/13
12	Començar memòria	9 days	Tue 22/10/13	Wed 30/10/13
13	Preparar entrega PAC 2	3 days?	Mon 28/10/13	Wed 30/10/13
14	Entrega PAC 2	0 days	Wed 30/10/13	Wed 30/10/13
15	**Fase Implementació bàsica**	**28 days?**	**Thu 31/10/13**	**Wed 27/11/13**
16	Memòria	28 days?	Thu 31/10/13	Wed 27/11/13
17	Història d'usuari 2	7 days	Thu 31/10/13	Wed 06/11/13
18	Història d'usuari 3	7 days	Thu 07/11/13	Wed 13/11/13
19	Historia d'usuari 4	7 days	Thu 14/11/13	Wed 20/11/13
20	Proves	3 days	Thu 21/11/13	Sat 23/11/13
21	Preparar entrega PAC 3	4 days	Sun 24/11/13	Wed 27/11/13
22	Entrega PAC 3	0 days	Wed 27/11/13	Wed 27/11/13
23	**Fase implementació extres**	**26 days?**	**Thu 28/11/13**	**Mon 23/12/13**
24	Memòria	26 days?	Thu 28/11/13	Mon 23/12/13
25	Historia d'usuari 5	12 days	Thu 28/11/13	Mon 09/12/13
26	Historia d'usuari 6	5 days	Tue 10/12/13	Sat 14/12/13
27	Historia d'usuari 7	6 days?	Sun 15/12/13	Fri 20/12/13
28	Proves i corregir detalls	2 days?	Sat 21/12/13	Sun 22/12/13
29	Preparar entrega de la aplicació	4 days?	Fri 20/12/13	Mon 23/12/13
30	Entrega final de la aplicació	0 days	Mon 23/12/13	Mon 23/12/13

Fig. 1 - Planificació temporal

3. Estat de l'art

Actualment, hi ha moltíssimes aplicacions en el mercat d'aplicacions per a *smartphones*, que tenen la funció de comptador de clics (en anglès, als comptadors de clics se'ls anomena *Tally Counter*).

No podríem mencionar-les totes, perquè seria massa extens i repetitiu, però es farà un petit recull de les dues aplicacions més rellevants, tant per a dispositius *Android* com per a *iOS*, i se'n citaran aquelles característiques més importants i destacades.

T-Counter	
Plataforma	Android
Descàrregues	De 100.000 a 500.000
Tamany	484 Kb
Tipus	Gratis i de pagament (1.52€)
Funcionalitats destacades	- Múltiples comptadors. - Botó de volum per incrementar el comptador. - Decrementar comptador. - Enviar per email les dades d'un comptador

Captures

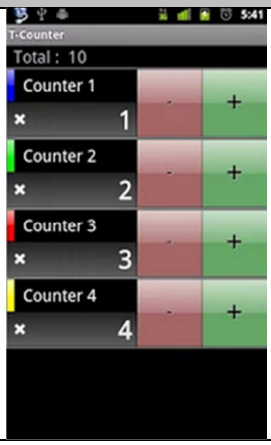


Fig. 2 - T-Counter

Click Counter

Plataforma	Android
Descàrregues	De 100.000 a 500.000
Tamany	652 Kb
Tipus	Gratis i de pagament (1€)
Funcionalitats destacades	- Un únic comptador. - Botó de volum per incrementar el comptador. - Decrementar comptador. - So al prémer el comptador. - Avís cada X clics.

Captures

Fig. 3 - Click Counter

Tally Counter

Plataforma	iOS
Descàrregues	---
Tamany	1.8 Mb
Tipus	Gratis i de pagament (1.79€)
Funcionalitats destacades	- Un comptador (múltiples la versió de pagament). - So al clicar. - Decrementar comptador. - Límit configurable. - Reset automàtic configurable.

Captures

Fig. 4 - Tally Counter

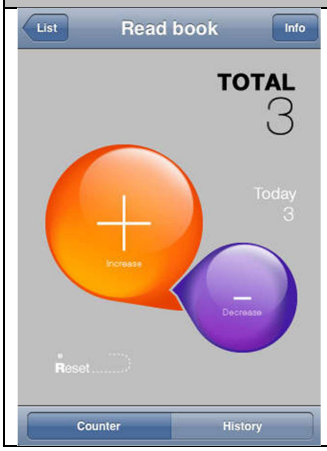
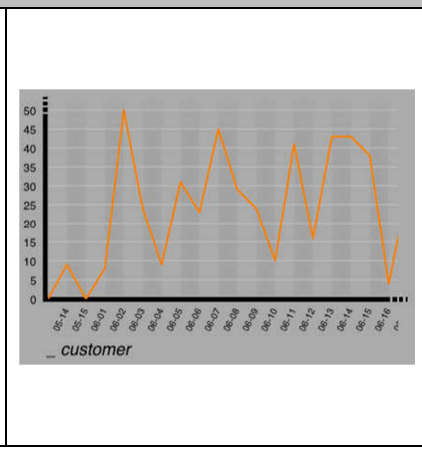
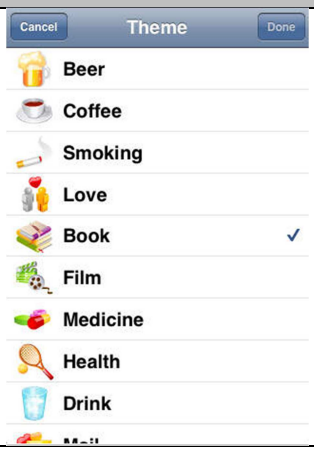
Smart Tally Counter		
	**Plataforma**	iOS
	Descàrregues	---
	Tamany	200 Kb
	Tipus	De pagament (13$)
	Funcionalitats destacades	- Múltiples comptadors. - Informació històrica i visualització gràfica. - Decrementar comptador. - Llista preconfigurada de comptadors. - *Auto-save* al sortir.
Captures		
			

Fig. 5 - Smart Tally Counter

Com s'ha dit al principi, aquestes aplicacions són només una petita mostra del que actualment hi ha al mercat.

Moltes de les aplicacions analitzades, presenten funcionalitats que posteriorment el nostre comptador "*The Clicker Master*" portarà incorporades, com per exemple alarma, històric, múltiples comptadors, etc...

D'altres funcionalitats, seria molt interesant incloure-les, com per exemple fer augmentar/decrementar el comptador amb el botó de volum, o tenir una llista de comptadors predefinida, però si ho sumem tot, amb el que volem fer, són moltes funcionalitats, i en principi, escaparà de l'abast del projecte i es deixarà per a futures millores.

4. Tecnologies utilitzades

Durant la realització del PFC, es faran servir una sèrie de tecnologies i *frameworks* de treball, amb els quals es construirà la aplicació final. Tot seguit es donarà cinc cèntims de quines es faran servir i què és el que fa cadascuna d'elles.

backbone.js [http://backbonejs.org/]

Backbone.js és un *framework Javascript* que permet implementar el paradigma MVC en el navegador client.

Fer servir aquesta llibreria, ens permetrà implementar el model del projecte en el client (el diagrama de classes), i al mateix temps controlar les vistes de la aplicació (pantalles), i les transicions que hi haurà entre elles, és a dir, la navegabilitat de la aplicació.

És una tecnologia relativament nova, que últimament està agafant cada cop més i més adeptes, i que aplicar-la en aquest PFC ens donarà molta flexibilitat a l'hora de programar i ens facilitarà les coses.

underscore.js [http://underscorejs.org/]

Underscore.js és una llibreria *Javascript* que el *framework* anterior, *Backbone.js*, fa servir de forma nativa.

Es tracta d'una llibreria d'utilitats (més de 80) que, pel simple fet d'importar-la al nostre projecte (en la plana web), ens donarà tota una sèrie de funcionalitats ja fetes, com per exemple utilitzar plantilles HTML (*templates*), treballar amb col·leccions d'objectes de forma senzilla, o clonar objectes del model, entre moltes altres.

jQuery [http://jquery.com/]

Qui no coneix avui dia la llibreria *jQuery*. S'està convertint en tot un estàndard per treballar amb el DOM (*Document Object Model*) dels navegadors, d'una forma senzilla i amigable, de tal forma que ens

assegura que tot les funcionalitats que farem amb *jQuery* funcionin en la majoria de navegadors actuals.

Tal com diu el seu eslògan: *write less, do more*.

jQueryMobile ***[http://jquerymobile.com/]***

La llibreria *jQueryMobile* ens proporciona components basats en HTML5, per fer la maquetació i programació de les pantalles que realitzem per a la aplicació. Està basat en *jQuery*, i per aquest motiu ens garanteix que la aplicació es visualitzarà correctament en la majoria de dispositius mòbils.

A més, és molt personalitzable i les pàgines tenen un aspecte molt polit i professional. Durant el projecte, es farà servir per realitzar les vistes de la aplicació.

Parse.com ***[https://www.parse.com/]***

Parse.com és una plataforma online que ens proporcionarà la opció de tenir les dades que la aplicació necessiti, allotjades al núvol.

Es tracta també d'una llibreria *Javascript*, que inclourem en la nostra aplicació, i que amb senzilles crides ens donarà aquesta funcionalitat tant interessant, i així ens oferirà un ampli ventall de possibilitats per a dotar a la aplicació de funcionalitats online.

Apache Cordova ***[http://cordova.apache.org/]***

La llibreria *Apache Cordova* ens ajudarà a comunicar-nos amb les funcionalitats pròpies del mateix dispositius, com per exemple emetre vibració, emetre sons, etc...

L'avantatge de fer servir aquesta llibreria és que podrem cridar a aquestes funcionalitats des del mateix navegador, amb crides senzilles *Javascript*.

* PhoneGap [http://phonegap.com/]

PhoneGap és el que uneix tot plegat. Totes les llibreries mencionades anteriorment serveixen per construir aplicacions web, que poden funcionar en qualsevol navegador, ja sigui mòbil o no. Com que estem construint aplicacions per a dispositius mòbils, aquesta llibreria embolcallarà la aplicació web que construirem, i la convertirà en una aplicació hibrida, que funcionarà i s'instal·larà com si fos una aplicació nativa en el dispositiu. D'aquesta manera, ens estalviarem de posar-nos a treballar amb els SDK de cada plataforma mòbil, i de programar una aplicació per cada una d'elles.

** Donat que per l'entrega final s'entregarà la aplicació només per Android, aquesta tecnologia es menciona, però no s'utilitzarà en l'aplicació que s'entrega. Es menciona aquí, perquè s'hauria de fer servir si es volgués crear l'executable, també per a iOS i d'altres sistemes operatius mòbils.*

5. Anàlisi i disseny

5.1 Requeriments funcionals

Com s'ha comentat, el plantejament inicial del projecte és realitzar una aplicació no gaire enrevessada, però que permeti treballar amb les noves tecnologies per a mòbils, i a més que sigui funcional.

L'objectiu principal és desenvolupar un comptador de clics per a dispositius mòbils. A cada clic (un toc amb el dit) de l'usuari en la pantalla, s'incrementarà el número del comptador.

S'han de poder crear tants comptadors com es desitgi, i poder triar quin comptador utilitzar en cada moment.

Un altre punt important es donar a l'usuari la possibilitat de configurar cada comptador. S'ha de poder configurar el color del comptador, el color de fons, quant s'incrementarà a cada clic, així com poder configurar alarmes que saltin quan s'arribi a un número concret de *clics*.

L'usuari també ha de poder bloquejar el comptador en la pantalla, de tal manera que si està bloquejat, no funcionin els clics que es facin. També s'ha de poder posar un comptador a 0 (o el que és el mateix, fer un *reset*).

L'usuari ha de poder veure l'evolució (historial) dels seus clics, en funció del comptador. És a dir, per cada comptador hi haurà períodes que començaran en el moment del primer clic, i acabaran el moment que es faci *reset*. També se li ha de poder posar un nom a aquests cicles.

Encara que d'entrada pugui semblar una aplicació que no requereixi autentificar-se, s'ha pensat en donar la possibilitat als usuaris de poder registrar-se en la aplicació, i de fer *login*. Un cop fet això, els usuaris podran accedir a les funcionalitats online. Per exemple, han de tenir la possibilitat de pujar un comptador al núvol, i d'aquesta manera poder compartir-lo amb d'altres usuaris coneguts per ell.

També, des de la aplicació s'ha de poder enviar la informació d'un comptador en concret amb usuaris via e-mail.

La aplicació ha de permetre també eliminar un comptador definitivament del sistema.

Cal dir que la aplicació a desenvolupar s'anomenarà *clickerMaster*, donat que és un nom que encara no està agafat en els mercats d'aplicacions (*Google Play*, *iTunes*), i perquè crec que és un nom bastant comercial.

5.2 Requeriments no funcionals

A banda dels requeriments funcionals, hi haurà una sèrie de requeriments no funcionals que la aplicació ha de complir. Són els següents:

- S'ha de poder utilitzar la aplicació, sense tenir un usuari creat en l'aplicació.

- La aplicació ha de respondre ràpidament als clics de l'usuari, sense "*delays*".

- L'aplicació ha d'avisar en tot moment a l'usuari, en cas de que hi hagi algun error.

- Tota la informació dels comptadors ha d'estar sempre disponible en el dispositiu, i s'ha de mantenir viva entre diferents execucions de l'aplicació, tot això de forma transparent per l'usuari.

- La pujada/baixada de les dades al núvol ha de ser transparent per l'usuari.

5.3 Casos d'ús

El diagrama de casos d'ús, que englobarà els requeriments funcionals de la aplicació i satisfarà les històries d'usuari anteriorment descrites, és el següent:

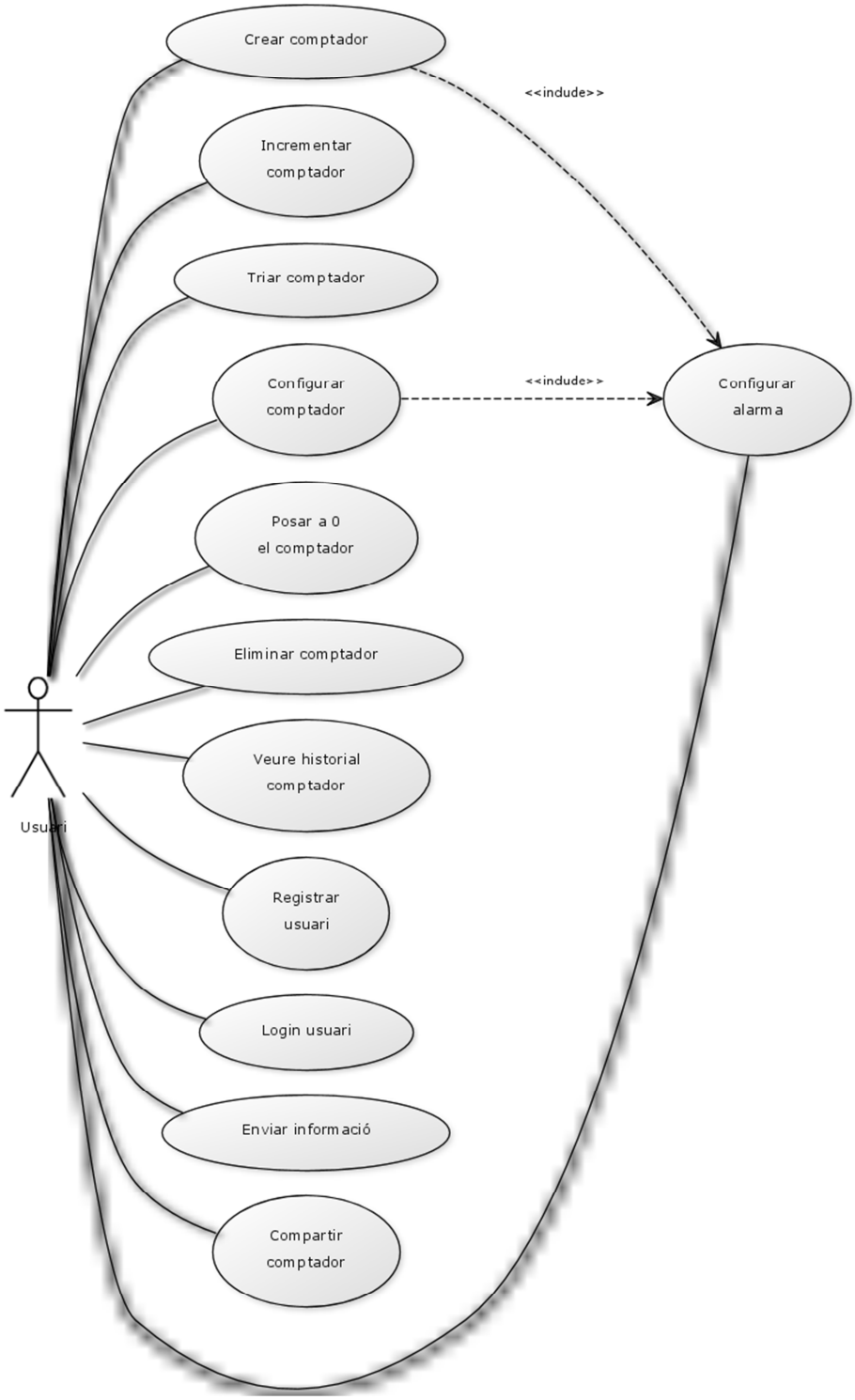


Fig. 6 - Diagrama de casos d'ús

Si mirem cada cas d'ús en detall, tenim el següent:

1. **Cas d'ús: Crear comptador**

Cas d'ús	Crear Comptador
Descripció	Crear un nou comptador al sistema
Actors	Usuari
Pre-condició	Cap
Flux principal	**1.** L'usuari prem el boto 'Crear nou comptador'. **2.** El sistema li mostra un formulari a omplir, amb la configuració inicial que tindrà. **3.** L'usuari introdueix tots els paràmetres. **4.** El sistema comprova si tot es correcte. **5.** Es crea el comptador amb la configuració triada.
Flux alternatiu	**4a**. El sistema detecta un error en les dades introduïdes i torna a demanar les dades. **4b**. Torna al pas 3
Post-condicions	Es crea un comptador al sistema

2. **Cas d'ús: Incrementar comptador**

Cas d'ús	Incrementar comptador
Descripció	Incrementar el comptador que apareix per pantalla.
Actors	Usuari
Pre-condició	Haver creat el comptador
Flux principal	**1.** L'usuari clica en la rodona del comptador. **2.** El sistema incrementarà el comptador per cada toc de l'usuari a la rodona.
Flux alternatiu	
Post-condicions	S'incrementa el comptador

3. Cas d'ús: Triar comptador

Cas d'ús	Triar comptador
Descripció	Triar un comptador dels que hi ha disponibles en la aplicació.
Actors	Usuari
Pre-condició	Haver creat algun comptador.
Flux principal	**1.** L'usuari tria un comptador de la llista de comptadors disponibles. **2.** El sistema mostrarà el comptador triat.
Flux alternatiu	
Post-condicions	Es mostra per pantalla el comptador triat.

4. Cas d'ús: Configurar comptador

Cas d'ús	Configurar comptador
Descripció	L'usuari vol configurar el comptador que està veient per pantalla.
Actors	Usuari
Pre-condició	Haver creat algun comptador i estar en la pantalla del mateix.
Flux principal	**1.** L'usuari tria l'opció de configurar el comptador. **2.** El sistema mostra un formulari amb els atributs disponibles a configurar. **3.** L'usuari fa els canvis. **4.** El sistema comprova si tot es correcte. **5.** Es mostra el comptador amb la nova configuració.
Flux alternatiu	**4a**. El sistema detecta un error en les dades introduïdes i torna a demanar les dades. **4b**. Torna al pas 2.
Post-condicions	Es mostra per pantalla el comptador amb la nova configuració.

5. Cas d'ús: Configurar/crear alarma

Cas d'ús	Configurar/crear alarma
Descripció	L'usuari vol configurar (o crear) l'alarma d'un comptador.
Actors	Usuari
Pre-condició	Estar en la pantalla de configuració o de creació d'un comptador.
Flux principal	**1.** L'usuari vol crear/configurar una alarma. **2.** El sistema mostra un botó per afegir una nova alarma, en cas de que el comptador no en tingui, o mostra l'alarma del comptador, en cas de que s'estigui editant una. **3.** L'usuari introdueix les dades de l'alarma. **4.** El sistema comprova les dades. **5.** S'enregistra l'alarma.
Flux alternatiu	**4a**. El sistema detecta un error en les dades introduïdes i torna a demanar les dades. **4b**. Torna al pas 2.
Post-condicions	Es configura l'alarma per a aquell comptador.

6. Cas d'ús: Posar a 0 el comptador

Cas d'ús	Posar a 0 el comptador
Descripció	L'usuari vol posar a 0 un comptador.
Actors	Usuari
Pre-condició	Haver creat algun comptador i estar en la pantalla del mateix.
Flux principal	**1.** L'usuari clica en el boto per posar a 0. **2.** El sistema pregunta si vol guardar-ne la informació. **3.** L'usuari introdueix el nom amb el que es guardarà aquest cicle. **4.** El sistema registra la informació. **5.** El sistema posa a 0 el comptador i el torna a mostrar per pantalla.
Flux alternatiu	**3a.** L'usuari no desitja guardar la informació del cicle, i escull posar a 0, sense guardar.
Post-condicions	Es mostra el comptador a 0.

7. Cas d'ús: Eliminar comptador

Cas d'ús	Eliminar el comptador
Descripció	L'usuari vol eliminar un comptador definitivament.
Actors	Usuari
Pre-condició	Haver creat algun comptador i estar en la pantalla del mateix.
Flux principal	**1.** L'usuari selecciona que vol eliminar el comptador. **2.** El sistema preguntarà si esta segur. **3.** Si esta segur, el sistema eliminarà el comptador, i tota la informació associada a ell (alarmes i cicles).
Flux alternatiu	
Post-condicions	S'elimina el comptador, així com les alarmes i cicles associats a ell.

8. Cas d'ús: Veure l'historial d'un comptador

Cas d'ús	Veure l'historial d'un comptador
Descripció	L'usuari vol veure l'historial d'un comptador.
Actors	Usuari
Pre-condició	Haver creat algun comptador i estar en la pantalla del mateix.
Flux principal	**1.** L'usuari tria la opció de veure l'historial d'un comptador. **2.** El sistema mostra per pantalla un llistat amb les dades històriques (cicles guardats) d'aquell comptador.
Flux alternatiu	
Post-condicions	Es mostra l'historial del comptador per pantalla.

9. Cas d'ús: Registrar usuari

Cas d'ús	Registrar usuari
Descripció	L'usuari vol registrar-se com a usuari en la aplicació.
Actors	Usuari
Pre-condició	No estar registrat prèviament i tenir connexió a Internet.
Flux principal	**1.** L'usuari selecciona l'opció de registrar-se a l'aplicació. **2.** El sistema mostra un formulari amb els camps per registrar-se. **3.** L'usuari els introdueix. **4.** El sistema comprova que els camps són correctes i que l'usuari no existeix ja en el sistema. **5.** El sistema registra al nou usuari al sistema i fa el *login*.
Flux alternatiu	**4a**. El sistema detecta un error en el formulari **4b**. Torna al pas 2.
Post-condicions	L'usuari queda registrat en el sistema.

10. Cas d'ús: *Login* usuari

Cas d'ús	*Login* usuari
Descripció	L'usuari vol fer *login* a l'aplicació.
Actors	Usuari
Pre-condició	Estar registrat prèviament, no tenir *login* fet i tenir connexió a Internet.
Flux principal	**1.** L'usuari selecciona l'opció de fer *login* a l'aplicació. **2.** El sistema mostra un formulari amb els camps usuari i *password*. **3.** L'usuari els introdueix. **4.** El sistema comprova que el *login* es correcte. **5.** El sistema anuncia el *login* satisfactori i mostra a l'usuari la pantalla de Home.
Flux alternatiu	**4a**. El sistema detecta un que el *login* es incorrecte. **4b**. Torna al pas 2.
Post-condicions	L'usuari queda autenticat en el sistema.

11. Cas d'ús: Enviar informació

Cas d'ús	Enviar informació
Descripció	L'usuari vol enviar informació referent a un comptador a una persona.
Actors	Usuari
Pre-condició	Haver creat algun comptador, estar en la pantalla del mateix, estar autentificat en la aplicació i tenir connexió a internet.
Flux principal	**1.** L'usuari selecciona l'opció enviar informació. **2.** El sistema mostra un formulari amb els camps text i *email*, i d'altres opcions. **3.** L'usuari els introdueix. **4.** El sistema comprova que la informació introduïda és correcte. **5.** S'envia per *email* la informació desitjada.
Flux alternatiu	**4a.** El sistema detecta que hi ha errors en la informació introduïda. **4b.** Torna al pas 2
Post-condicions	S'envia per *email* la informació desitjada.

12. Cas d'ús: Compartir comptador

Cas d'ús	Compartir comptador
Descripció	L'usuari vol compartir un comptador amb un altre usuari.
Actors	Usuari
Pre-condició	Haver creat algun comptador, estar en la pantalla del mateix, estar autentificat en la aplicació i tenir connexió a internet.
Flux principal	**1.** L'usuari selecciona la opció de compartir comptador. **2.** El sistema mostra un camp buit per tal que l'usuari introdueixi l'*email* del destinatari. **3.** L'usuari escriu l'*email* del destinatari. **4.** Es comprova que el destinatari existeix, i que tot està correcte. **5.** Quan es connecti, el destinatari veurà el comptador disponible en la seva llista de comptadors disponibles.
Flux alternatiu	**3a**. Hi ha algun error amb l'*email* del destinatari.

	3b. S'informa de l'error.
Post-condicions	L'usuari destí rep el comptador.

5.4 Diagrama de classes

El diagrama de classes de la aplicació és molt simple, donat que no estem parlant d'una aplicació amb una complicada i rebuscada lògica de negoci, sinó que es centra més en l'aspecte visual i, per les funcionalitats i l'abast que es vol dotar al projecte, n'hi ha més que prou amb el següent:

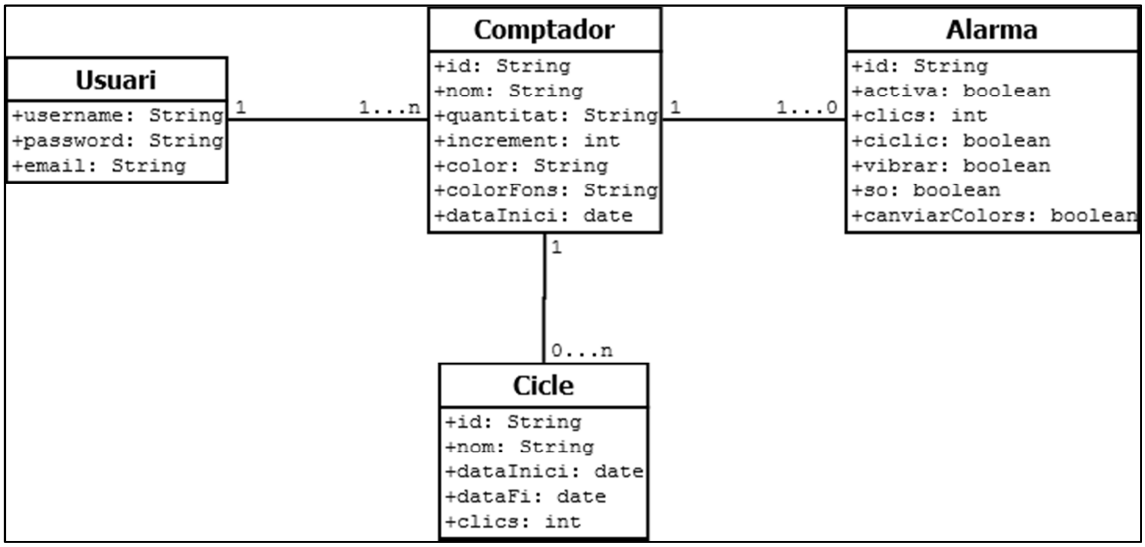


Fig. 7 - Diagrama de classes

Per entendre el diagrama de classes de la figura anterior, primer cal explicar quins serà el funcionament de l'aplicació.

Tal i com està pensat, l'usuari de l'aplicació podrà crear comptadors (tants com vulgui), als quals hi podrà accedir des de la plana principal de l'aplicació. A més, cada un dels comptadors tindrà la seva configuració concreta (nom, increment, colors, etc...). Un cop l'usuari hagi triat un comptador dels que hi ha creats, i el vulgui incrementar, el que estarà incrementant realment és una instància de l'entitat "Comptador". Després, si en algun moment l'usuari posa a 0 el comptador en curs, i en vulgui desar la informació del cicle, aquesta es guardarà en l'entitat "Cicle", que actuarà com a històric d'informació, associada a cada comptador.

A més, cada comptador podrà tenir associada "Alarmes" (una alarma per cada comptador). Les alarmes es podran crear en el moment de crear el comptador, i/o posteriorment editar-les en els comptadors ja creats. També seran molt configurables, podent triar a quants clics saltarà, si serà cíclica o no (si saltarà a cada X clics, o

només el primer cop), i també es podran configurar el tipus d'avís (vibració, so i colors).

Pel que fa a l'usuari, necessitarem que aquest faci *login* en l'aplicació només per realitzar les funcionalitats "online", que seran "Enviar informació" i "Compartir comptador". Òbviament per fer aquestes operacions el dispositiu mòbil ha de tenir connexió a Internet.

Un cop explicat el funcionament general, explicarem en detall cada classe.

5.4.1 Usuari

Aquesta classe entrarà en joc quan l'usuari es vulgui registrar o fer *login*. Aquesta classe en concret no la tindrem físicament en el dispositiu mòbil, sinó que es tindrà a Parse.com, com s'explicarà més endavant. Tindrà els següents atributs:

- *Usuari:* El nom d'usuari amb el que l'usuari es registrarà.
- *Password*: Es desarà codificat
- *Email*: Ha de ser un *email* en format vàlid.

5.4.2 Comptador

La classe comptador serà la pedra angular del model de dades. A ella se l'associaran les altres tres classes, "Cicle", "Alarma" i "Usuari". Els atributs de la classe Comptador són aquests:

- Id: Identificador del comptador
- *Nom*: El nom del comptador; el que comptarà el comptador.
- *Quantitat*: Quantitat inicial del comptador (per defecte 0).
- *Increment*: Quin serà l'increment cada cop que es faci clic (per defecte 1).
- *Color*: El color de la rodona que farà de comptador (per defecte serà negre).
- *ColorFons*: El color de fons del comptador (per defecte serà blanc perla)
- *dataInici*: La data de creació del comptador.

5.4.3 Alarma

Com ja s'ha explicat, opcionalment un comptador pot tenir una alarma associada, que s'activarà quan s'arribi a un nombre de clics pre-definit per l'usuari, i avisant de la forma que ell també determini. Els camps d'aquesta classe són els següents:

- Id: Identificador de la alarma
- *Activa*: Camp booleà que determina si l'alarma està activa o no. Si no està activa no actuarà (per defecte *false*).
- *Clics*: Número de clics que activaran l'alarma (per defecte 0).
- *Ciclic*: Camp booleà que determinarà si l'alarma serà cíclica, és a dir, saltarà cada X clics o només el primer cop (per defecte *false*).
- *Vibrar*: Si està actiu, el telèfon mòbil vibrarà 2 segons quan s'activi l'alarma (per defecte *false*).
- *So*: Si està actiu, el telèfon mòbil emetrà un so quan s'activi l'alarma (per defecte *false*).
- *canviarColors*: Si està actiu, els colors del comptador i el de fons, faran "pampallugues" durant breus instants (per defecte *false*).

5.4.4 Cicle

Els cicle guardaran la informació històrica dels comptadors. Es guardarà un cicle sempre que l'usuari posi el comptador a 0 i desitgi guardar-ne la informació. Els cicles poden durar tant de temps com l'usuari vulgui, segons, minuts, hores, dies, setmanes... Això sempre quedarà a elecció de l'usuari. Els camps seran els següents:

- Id: Identificador del cicle
- *Nom*: Nom amb el qual l'usuari identificarà aquest cicle.
- *dataInici*: La data en que es va iniciar el cicle.
- *dataFi*: La data de finalització del cicle.
- *Clics*: Nombre de clics que tenia el comptador quan es va finalitzar el cicle.

Com es pot veure, és un diagrama de classes senzill i entenedor, acord a les característiques del projecte i funcionalitats i abast que es vol assolir.

5.5 Prototipus pantalles

Per tal de tenir una primera idea de com podrà ser la aplicació, a nivell de visualització de les pantalles, és una molt bona idea fer primer el disseny de pantalles prototipus, per després poder tenir una idea visual de com volem més o menys el disseny.

Òbviament, al ser prototipus, pot diferir l'aspecte que tindrà al final l'aplicació real, del disseny d'aquest, però al menys ens servirà com a referència per saber a on volem arribar, quines pantalles necessitarem, quina pinta han de tenir i quina informació volem que hi aparegui en cada una.

Per realitzar el prototipatge de les pantalles, ens hem decantat per utilitzar un programari específic per aquest efecte, anomenat *Balsamiq Mockups*. Aquest software ens permet de forma molt i molt intuïtiva, i molt fàcilment, dibuixar amb elements ja pre-definits a l'eina, com volem que siguin les nostres pantalles.

A continuació es mostren les pantalles resultants de fer el prototipatge:

Fig. 8 - Prototipus de pantalles

6. Arquitectura

Un cop definides les funcionalitats i característiques que haurà de tenir el projecte, i dissenyades com hauran de ser, més o menys, les pantalles, el següent pas haurà de ser definir quina serà la arquitectura que tindrà la aplicació mòbil, tant a nivell d'arquitectura física (servidors, components físics, etc...), com d'arquitectura lògica, a nivell de software.

6.1 Arquitectura física

En essència, s'ha utilitzat una arquitectura híbrida entre el codi de la aplicació mòbil, que s'executarà i farà servir el propi dispositiu, i per a algunes funcionalitats, el servidor en el núvol "**Parse.com**", que l'utilitzarem com a magatzem per a guardar dades de la aplicació, i també com a proveïdor de codi remot.

També farem servir unes funcionalitats pròpies de l'HTML5, anomenades ***localStorage*** i ***sessionStorage***, que ens serviran com a magatzem de dades en el dispositiu. D'aquesta forma, les dades de la aplicació seran persistents entre diferents execucions de la aplicació.

Per tant, tenim que a nivell físic, en la aplicació trobarem els següents components:

- **Dispositiu mòbil**
  - Serà el dispositiu mòbil o s'executarà la aplicació. D'aquí farem servir el mateix dispositiu per a guardar dades, per les funcions *offline*, fent servir *localStorage* i *sessionStorage* d'HTML5.

- **Internet**
  - És el canal de comunicació per on accedirem a Parse.com. També és necessari per a accedir a alguns recursos imprescindibles per a carregar la aplicació, com per exemple les llibreries *JQueryMobile*.

- **Servidor Parse.com**
  - És el servidor en el núvol, en el qual ens recolzarem per a assolir les funcionalitats online, així com el control d'usuaris.

Si ho posem tot en un esquema, tindríem el següent:

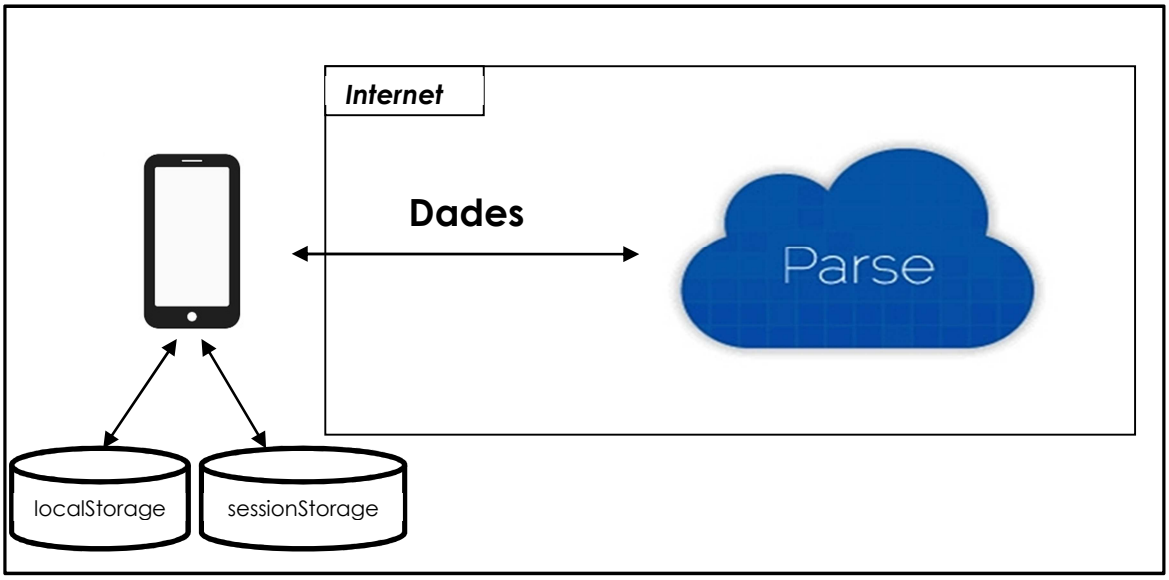


Fig. 9 - Esquema arquitectura física

6.2 Arquitectura lògica

D'altra banda, l'arquitectura de la aplicació a nivell lògic, o de software, ve marcada pel fet de fer servir el *framework* de desenvolupament ***Backbone.js***. El fet d'utilitzar *Backbone.js*, ens marca que haurem d'estructurar l'aplicació per capes, en una estructura MVC (Model-Vista-Controlador).

Model

Per una banda, tenim el model de l'aplicació. El model el definirem en un fitxer *Javascript*, mitjançant una notació JSON i seguint la sintaxi que demana *Backbone.js*. En el nostre cas, necessitarem definir 3 entitats del model en el nostre *Javascript*: *Comptador*, *Alarma*, *Cicle*.

L'entitat *Usuari* no la definirem en el fitxer *Javascript*, ja que *Parse.com* té un mòdul encarregat de gestionar la informació i les sessions dels usuaris, i per tant, no necessitem tenir aquesta informació en local.

Vista

D'altra banda, tenim les vistes. Les vistes també es defineixen en els fitxers *Javascript*, juntament amb una plantilla HTML.

És a dir, cada vista, com les utilitzarem en la aplicació, estarà associada a un pàgina en forma de plantilla en format HTML, però sempre fent servir el *framework JQueryMobile*. Per exemple, la vista que utilitzarem per a incrementar un comptador, estarà associada a una plantilla genèrica, a la qual li passarem un comptador o un altre (una instància de model, o un altre).

Controlador (Router)

Els controladors són aquells artefactes que recolliran les URL's i les mapejaran amb una vista.

Per exemple, quan s'introdueixi la URL acabada en "*#addCounter*", el *router* agafarà aquesta petició, i carregarà la vista "*addCounter*" (que al seu temps està associada a la plantilla HTML per aquest fet).

Collections

Tot i que no forma part de la arquitectura pròpiament dita, aquest element l'introdueix *Backbone.js* per a emmagatzemar en col·leccions les instàncies dels models que es vagin creant en la aplicació.

Per tant, durant tota la aplicació, cada cop que es creï un nou comptador, alarma o cicle, aquests es guardaran en col·leccions creades per a guardar aquests models.

Un recull de tot plegat, el podem veure en la següent imatge:

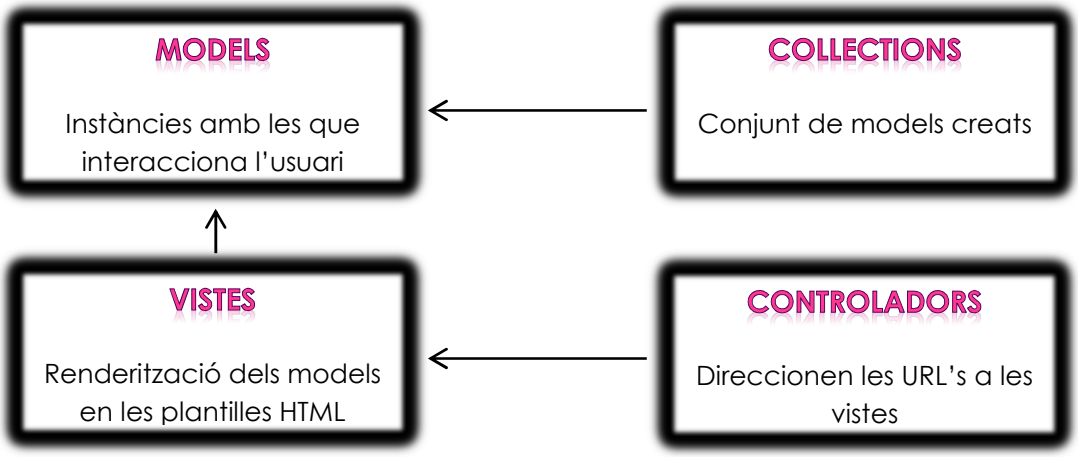


Fig. 10 - Esquema Backbone.js

7. Implementació i pantalles

Un cop ja s'ha especificat quines seran les funcionalitats de l'aplicació mòbil, tant funcionals com no funcionals, s'ha fet l'anàlisi, el disseny, el prototipatge de les pantalles i s'ha definit quina arquitectura i tecnologies s'utilitzaran, només queda especificar i detallar quin ha estat el resultat de fer la implementació de tot plegat, ja mostrant directament les pantalles de l'aplicació i mostrant el seu funcionament.

7.1 Estructuració de fitxers

L'aplicació s'ha construït basant-se principalment en fitxers HTML, CSS i sobretot, fitxers *Javascript*.

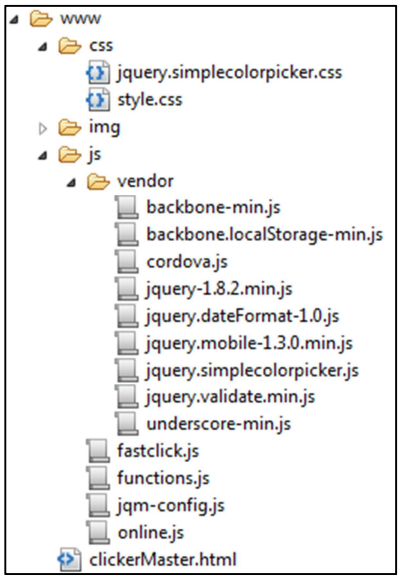


Fig. 11 - Estructura de fitxers

HTML (clickerMaster.html)

Només hi haurà un únic fitxer HTML, el qual bàsicament tindrà les crides als fitxers CSS i als *Javascript* que necessiti, i a més a més també les definicions de les plantilles (*templates*), que definiran les pantalles que la aplicació tindrà.

CSS

La aplicació tindrà dos fitxers CSS, *style.css* i *jquery.simplecolorpicker.css*. El primer, és el que s'ha creat a mà per a definir els estils de la aplicació, i el segon forma part d'un plugin, que és necessari per triar els colors que tindrà un comptador, de forma senzilla, amb el mètode "*colorPicker*".

Javascript

Per una part ens hem descarregat tots els fitxers *Javascript* possibles, i els hem inclòs dins la carpeta "*vendor*". Aquests fitxers són les llibreries que la aplicació farà servir. Ho hem fet així per evitar que el navegador es

descarregui cada vegada els fitxers necessaris per a funcionar, amb la conseqüent lentitud de la aplicació i l'augment de la tarifa de dades. Hi ha fitxers, però, que no s'ha pogut descarregar, i per tant es van a buscar a la xarxa.

D'altra banda estan els fitxers propis *Javascript*, que són 4:

- **fastclick.js**: Script trobat per internet, que permet fer clics ràpids en un dispositiu mòbil. Sense aquest script, hi havia un *delay* important entre clics en el comptadors, de tal manera que entre clic i clic s'havia d'esperar 1 segon per que funcionés. Amb aquest fitxer, el *delay* entre clic i clic és quasi inexistent (300 ms).

- **functions.js**: És on està el cor de la aplicació. Configuració dels models, vistes, *routers* i *collections*, fruit de fer servir *Backbone.js*. Tota la lògica de la aplicació *offline* està aquí.

- **jqm-config.js**: Per entendre aquest fitxer, cal explicar primer que fer servir *Backbone.js* i *JQueryMobile* en una mateixa aplicació, és redundant pel que fa a la navegabilitat entre pàgines. *JQueryMobile* té el seu mètode de redireccionament entre pàgines, i *Backbone.js* fa servir el seu. Buscant per internet, s'ha vist que la tendència actual és deshabilitar la funció de controlador de *JQueryMobile*, i deixar que sigui la de *Backbone.js* la que faci la feina. Per això existeix aquest fitxer, són unes poques línies que deshabiliten la funció de controlador de *JQueryMobile*.

- **online.js**: És el codi que gestionarà la part online de la aplicació, comunicant-se amb Parse.com i fent tasques com *login*, *logout*, enviament de correus, etc...

7.2 *Offline* i *Online*

Tot i que ja s'ha mencionat de passada amb anterioritat, abans d'entrar en detall amb la descripció i implementació de cada pantalla, cal insistir en que la aplicació comptarà amb dos mòduls, el mòdul *offline* i el *online*.

Les funcionalitats que estan sota el mòdul *offline*, es refereixen a aquelles que no cal estar autentificat en la aplicació per a fer-les servir.

Per contra, un cop es fa *login* satisfactòriament en la aplicació, l'usuari ja podrà accedir a les funcions *online*, que són dues: "Enviar informació" i "Compartir comptador".

7.3 Estructura d'una pantalla

A nivell de codi, totes les pàgines que es presentaran, tenen una estructura gairebé idèntica, a nivell de components. Per no fer repetitives les posteriors explicacions de les pantalles, amb codi similar en totes, en aquest punt s'explicarà quins són els passos per on passa l'execució al carregar i processar una pàgina, i d'aquesta mateixa manera s'hauran fet la resta de pàgines.

Agafarem l'exemple de la pantalla de Home, on es llisten tots els comptadors que hi ha guardats en el dispositiu, en el *localStorage*.

Introduir una URL

Al introduir qualsevol URL en el navegador (que és l'equivalent a canviar de plana en la aplicació), s'activarà el *router* del *Backbone.js* que hem definit. Si el que introduïm és una URL acabada en **/clickerMaster.html#** (o sense el coixinet), s'activarà la funció associada a una crida sense paràmetres:

```
var AppRouter = Backbone.Router.extend({

    routes:{
            "":"homeView",
            "addCounter":"addCounterView",
            "counter?id=:id":"counterView"
            "editar?id=:id":"editarCounterView",
            "historial?id=:id":"historialCounterView",
            "enviar?id=:id":"enviarView",
            "compartir?id=:id":"compartirView",
            "registre":"registrePage"
    },

    initialize:function () {
        // Handle back button throughout the application
        $('#back').live('click', function(event) {
            //event.preventDefault();
           window.history.back();
            return false;
        });
        this.firstPage = true;
    },
```

Al cridar a una URL sense paràmetres ("") , es cridarà al mètode **homeView**

```
    homeView:function () {
          this.changePage(new HomeView({collection: elsMeusComptadors}));

    },

……
```

Funció homeView en el *router*. Aquesta funció crea una nova instància de la vista **HomeView**, i li passa com a col·lecció, el magatzem genèric de comptadors que tenim, anomenat "*elsMeusComptadors*". La vista ja s'encarregarà de dibuixar la pàgina.

Com es pot veure, cada URL que es crida en la aplicació, té la corresponent funció en el *router*, i aquesta al seu temps carrega la vista que calgui.

La vista

```
window.HomeView = Backbone.View.extend({

      events: {
       'click #logout': 'logout',
       'click #login': 'openPopUp',
       'click #loginSubmit': 'loginSubmit'
    },

      loginSubmit: function(e) {
            …
      },
      logout :function (e) {
            …
       },

    template:_.template($('#home').html()),

    render:function (eventName) {
            if(isLogged()){
                  obtenirComptadorsPendents();
                  this.collection = elsMeusComptadors;
            }

        $(this.el).html(this.template({comptadors: this.
isLogged: isLogged(), username: getUserInfo("username")}));
        return this;
    },

       openPopUp: function(e) {
      …
      }
});
```

Aquí és defineixen les accions en la pantalla, que tindran una funció *Javascript* associada.

El template (plantilla HTML), que carregarà la vista. En aquest cas, carregarà el template amb id "home"

La funció render s'executa automàticament al crear-se una vista. El que fem es carregar el template anteriorment definit

Les demés funcions de la vista no s'han indicat a fons (hi ha punts suspensius), ja que cada vista tindrà la seva lògica de negoci i les seves funcionalitats. N'hi haurà que tenen un formulari, i caldrà comprovar que tot estigui correcte, n'hi haurà que hagin de fer operacions, al fer algun clic en pantalla, etc...

El template HTML

Els *templates* estan definits en el *<head>* de la plana web clickerMaster.html (podrien estar en un fitxer *Javascript* a part, però s'han posat aquí). El codi del *template* "home" és el següent:

```
<script type="text/template" id="home">
	<div data-role="header">
		<h1>The Clicker Master</h1>
		...
	</div>
	<div data-role="content">
		<a href="#addCounter" data-role="button" data-icon="plus" data-iconpos="bottom" id="addCounterSub">Crear nou
comptador</a>
	<br/>
	<div style="width: 100%; margin: 0 auto;">
	<ul>
		<% _.each(comptadors, function(comptador) { %>
			<li class="resum" style="background-color: <%= comptador.colorFons %>; cursor: pointer; <%
if(comptador.hasAlarm) { %>  background-image: url('img/alarm.png'); background-position: left bottom; background-repeat: no-
repeat; background-size: 22% 17%;<% } %> ">
				<a href="#counter?id=<%=comptador.id%>" style="display:inline-block; width: 100%; height:
100%;">
					<div class="llistaComptadors" >
						<span><b><%=comptador.nom%></b></span>
						<br></br>
						<div class="circle_summary" style="background-color: <%= comptador.color
%>"><%= comptador.quantitat %>

						</div>
					</div>
				</a>
			</li>
		<%});%>
	</ul >
	  <div>
	</div>
	<div data-role="footer" data-position="fixed">

	</div>

</script
```

Inici del template amb id "home"

El *template* esta dividit en tres parts, ja que fem servir el *framework JQueryMobile*. El *header*, el content i el *footer* (que el tenim buit sempre).

En el *header* tindrem sempre el nom de la plana on estem, així com informació addicional, i ocasionalment, algun *popup* amagat. En el *content* tindrem el cos de la plana, és a dir, el contingut principal.

Pel que fa als models i col·leccions, a la vista li passem un o altre objecte en funció de la necessitat de la pantalla. En el cas de l'exemple, li passem la col·lecció amb tots els comptadors creats, i en el *template* el dibuixem per pantalla.

Cal dir que, en tot el codi, tindrem tres col·leccions diferents, que ens faran de magatzem local per a tota la aplicació: **elsMeusComptadors**, **lesMevesAlarmes** i **elsMeusCicles.**

7.4. Pantalla Home

La pantalla de Home, serà la primera que es mostrarà al obrir la aplicació. En ella, l'usuari podrà realitzar les següents operacions:

- Crear un nou comptador.
- Triar un comptador, ja creat anteriorment.
- Fer *login* en l'aplicació, i registrar-se.

Fig. 12 - Pantalla Home

Com es pot veure en la captura de pantalla de la Home, els comptadors que ja estan creats es mostren en format miniatura (no es poden incrementar aquí, només es mostren en quin estat estan). Cadascun mostra el seu nom, la seva quantitat de clics que porta en la actualitat, així com la seva configuració de colors, i també es marca amb una icona si té una alarma configurada o no.

Per pintar per pantalla la llista de comptadors, s'ha fet servir, en el *template* HTML, la llibreria underscore.js (funció *_.each*), que iterarà per cada objecte JSON que se li passa des de la vista, en *Backbone.js*.

```
<ul>
      <% _.each(comptadors, function(comptador) { %>

…… //dibuixem la llista amb HTML i CSS

      <%});%
</ul >
```

7.5 Crear nou comptador

En aquesta pantalla es pot definir un nou comptador en la aplicació. En ella apareix un formulari on es podran definir:

- Un nou comptador amb les opcions:
  - Nom
  - Increment al fer clic
  - Quantitat inicial amb la qual es crearà
  - Color del comptador
  - Color de fons

- Una alarma pel comptador, amb les opcions:
  - Si l'alarma està activa o no (si no està activa, l'alarma no saltarà)
  - Clics als que s'activarà l'alarma
  - Si aquesta és cíclica o no (sí saltarà cada X clics, o només el primer cop)
  - El tipus d'alerta : Per so, per vibració, per fer intermitents els colors (es poden combinar totes les alertes)

Fig. 13 - Creació d'un comptador

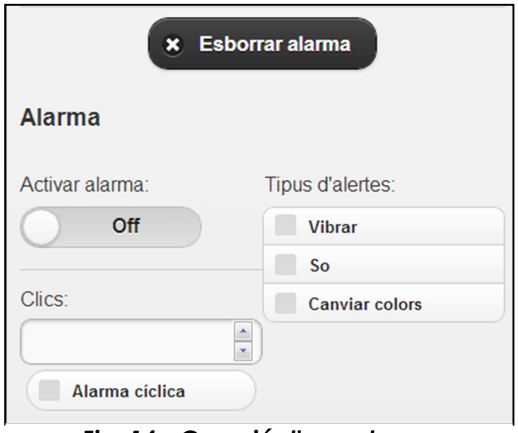


Fig. 14 - Creació d'una alarma

Respecte les alarmes, és important remarcar la importància de marcar els camps "Activar alarma" i definir algun "tipus d'alerta". Si no es fa això, tot i que la alarma estigui definida per un comptador, no saltarà al arribar al número de clics configurat.

Un altre punt a mencionar referent a la alarma, és la necessitat de que el número de clics de l'alarma coincideixi amb el número de clics del comptador, per que aquesta s'activi. Per exemple, si definim un increment de comptador de 5 en 5, i configurem la alarma per que salti en el clic 43 (no múltiple de 5), aquesta no saltarà mai.

Respecte al formulari de creació del comptador, els camps obligatoris es marquen amb un asterisc (*). Quan s'envia el formulari, es fa una

validació, en el mateix client, de que els paràmetres que s'han enviat siguin correctes. Aquesta validació es fa mitjançant la llibreria **jquery-validate.js**, la qual ens permetrà definir quins camps són obligatoris, així com d'altres regles. En cas que algun camp estigui mal informat, apareixerà un missatge d'error en la aplicació, indicant l'error.

Un cop s'hagi validat que tots els camps del comptador estan correctes, es crearà el comptador en i apareixerà en la llista de comptadors, en la Home.

7.6 Comptador X

Aquesta es la pantalla principal del comptador, que s'arriba quan es selecciona un comptador a la Home.

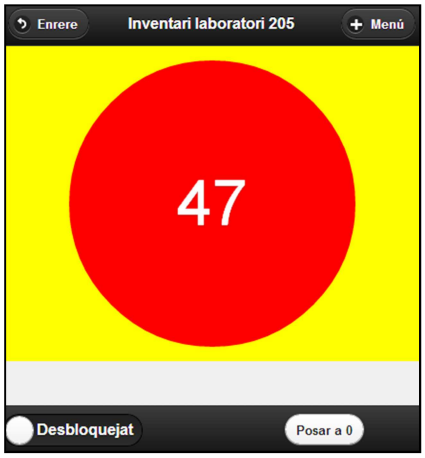


Fig. 15 - Pantalla d'un comptador

Es poden fer les següents funcions:

- Incrementar el comptador. Això s'aconsegueix tocant amb el dit la rodona central.
- Bloquejar el comptador (si està bloquejat no es podrà incrementar)
- Posar a 0 el comptador, guardant la informació del cicle, o sense guardar-la.

- Accedir al menú del comptador, on es podrà fer (en mode *offline*):
  - Editar el comptador
  - Veure l'historial del comptador (els cicles guardats del comptador).
  - Eliminar el comptador, i tota la seva informació associada (alarmes i cicles).

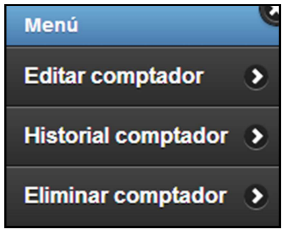


Fig. 16 - Menú d'opcions

Una de les particularitats a nivell de codi que trobem en aquesta pantalla, és la de que els increments que es fan en el comptador, no cal guardar-los explícitament, sinó que un cop s'incrementa un comptador, automàticament es guarda la nova

quantitat. Aquesta actualització és immediata, gràcies a *Backbone.js*, i el fet de fer servir els models.

Dins de la funció d'incrementar el comptador, es comprova sempre si la nova quantitat incrementada farà saltar la alarma del comptador. Si la fa saltar, s'activa una funció *Javascript*, preparada per a activar la alarma, en funció del que hi hagi configurat:

```
function activarAlarma(vibrar, so, canviarColors) {

	if(vibrar) navigator.notification.vibrate(2000);
	if(so) navigator.notification.beep(1);
	if(canviarColors){
		for (var i = 0; i < 5; i++) {
			$("#counterFons").fadeOut(200).fadeIn(200);
			$("#fastclick").fadeOut(200).fadeIn(200);
		}
	}
}
```

La funció anterior rep 3 paràmetres booleans, corresponent a les tres alarmes que hi ha. Per a fer vibrar i emetre sons, s'ha fet servir la llibreria cordova.js. Aquesta llibreria, ens permet manipular certes accions en els dispositius mòbils amb una simple sentència, com per exemple vibrar (*navigator.notificacion.vibrate*), i emetre sons (*navigator.notificacion.beep*). En concret, les vibracions duraran 2 segons, i s'emetrà un so de notificació (aquell que el terminal tingui configurat per defecte).

L'altre alarma, la de canviar colors, es serveix de les funcions *fadeIn* i *fadeOut* de *jQuery*.

Com s'ha mencionat abans, les alarmes es guardaran en la col·lecció *lesMevesAlarmes*, i s'associaran amb els comptadors, mitjançant un paràmetre "comptador", que guardarem en la instància d'alarma, i que tindrà com a valor l'identificador (id) del comptador associat. D'aquesta manera fem la relació entre comptadors i alarmes.

7.7 Editar comptador

En aquesta pantalla, es podran editar els diferents paràmetres que es van definir a l'hora de crear un comptador, incloent l'alarma. S'arriba fent clic en la opció del menú corresponent, en la pantalla d'un comptador.

La implementació d'aquesta pantalla no té cap misteri. És, bàsicament, la mateixa pantalla de la creació d'un comptador, però en aquest cas, hi ha els camps omplerts amb la configuració del mateix, així com de la seva alarma (si la tenia), i es poden modificar tots els valors que es desitgin.

7.8 Historial comptador

En aquesta pantalla es podrà veure l'historial de cicles que l'usuari ha guardat amb anterioritat, cada cop que ha posat a 0 el comptador en qüestió (i ha triat desar-ne la informació històrica).

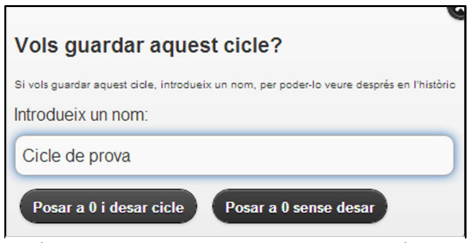


Fig. 17 - Popup per guardar un cicle

El concepte de cicle és el següent. En el moment que es crea un comptador, a aquest se li dona una data (i hora) d'inici. L'usuari anirà fent clics (durant el temps que ell vulgui), fins que arribarà un moment que el posarà a 0, per fer un reset. Llavors, la aplicació li preguntarà si vol desar el cicle en qüestió, i en cas afirmatiu, li haurà de posar un nom. També se li donarà la opció de posar el comptador a 0, sense guardar el cicle.

Quan l'usuari guardi el cicle, se li posarà una data de fi al cicle, i es desarà una nova instància de l'entitat cicle, definida en el model *Backbone.js*, i es guardarà en la col·lecció elsMeusCicles. Igual que passava amb les alarmes, els cicles s'associaran amb els comptadors mitjançant un paràmetre comptador, en la instància cicle, i que portarà el identificador (id) del comptador amb el que s'associa.

Fig. 18 - Historial d'un comptador

Després, quan s'accedeixi a la opció el menú "Historial comptador", es veurà una taula amb tots els cicles guardats per aquell comptador (si en té).

7.9 Eliminar comptador

Si es tria aquesta opció, s'eliminarà el comptador seleccionat, a més a més de tota les alarmes associades, i tots els cicles guardats.

El que es fa internament en el codi és eliminar el comptador en sí, així com tots els cicles que es trobin associats al comptador (es fa una cerca a la col·lecció *elsMeusCicles*, i es busquen aquells associats al comptador), i de la mateixa manera es fa amb la alarma (si el comptador en té).

Com a particularitat, dir que aquesta funcionalitat no té cap *template* HTML associat, ja que no s'ha considerat necessari, i per tant tot es fa en *Javascript*. El que sí que apareix per pantalla és un missatge de confirmació (ja que es tracta d'una operació sensible), i en cas de que tot hagi anat bé al esborrar, es mostra un avís per pantalla.

Fins ara s'ha descrit la part offline de la aplicació, o dit d'un altre manera, la part que per fer-la anar no es necessita estar autenticat en la aplicació. Les següents funcionalitats es refereixen a aquelles que requereixen tenir usuari vàlid en la aplicació, i està autenticat (part online). Primer, però, caldrà explicar el procés de registrar-se i fer login.

7.10 Login i registre

Abans, però, caldrà explicar el procés de registre i *login* en la aplicació.

Es pot fer *login* des de el botó que apareix en la capçalera de la Home. Al fer clic en "*Login*", apareixerà un pop-up, per introduir usuari i contrasenya, i un enllaç a la pantalla de "Registre", per si l'usuari es vol registrar.

La part d'usuaris de la aplicació, s'ha delegat en un mòdul existent a Parse.com, que s'encarrega de guardar aquesta informació per nosaltres.

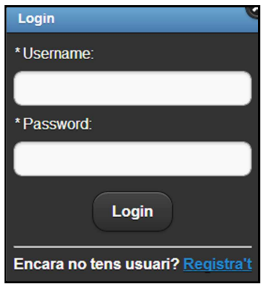


Fig. 19 - Pantalla de login

Per tant, en el client no es guarda cap base de dades d'Usuaris, ni es té cap model creat d'ells. Tant el registre com el *login* en la aplicació es fa mitjançant crides REST a la API de Parse.com.

De fet, al fil del que s'acaba de dir, ja es pot mencionar que quasi tota la comunicació que es fa amb Parse.com (excepte l'enviament de emails, que veurem més endavant), es fa mitjançant crides REST a la seva API. A més, quasi totes les crides són asíncronyes, ja que moltes vegades ens interessa esperar que s'acabi la comunicació entre el client i Parse.com, per a continuar l'execució normal del programa.

Per posar un exemple d'una crida REST, en el fitxer **online.js** podrem trobar la funció següent:

```
function registre() {

	var userData = { "username": $('#username').val(), "password": $('#password').val(),
"email": $('#email').val() };

	// Convert data object to JSON string:
	var data = JSON.stringify(userData);
	var ret;
	$.ajax({
		"async": false,
		"type":"POST",
		"url":"https://api.parse.com/1/users",
		"dataType":"json",
		"data":data,
		"contentType":"application/json",
		"headers":headers,
		success:function(data, status, xhr) {
			ret = true;
		},
		error: function(xhr, error){
			ret = false;
		}
	});
	return ret;
}
```

En el codi anterior es pot veure com, primer es preparen las dades a passar-li a Parse.com, que en aquest cas és una crida per POST a la URL acabada en */users*, i ell ens retorna si l'usuari està validat o no, i si tot ha anat bé, es crida al mètode de *callback* corresponent, i en funció d'això, la funció retorna *true* o *false*.

Un altre aspecte a destacar del procés de registre, és que, per tal de que el registre sigui vàlid, caldrà fer registrar-se ***amb un email existent i real***. Parse.com enviarà un correu a la direcció d'email introduïda, amb un link, que l'usuari haurà de clicar per tal de validar la direcció de correu.

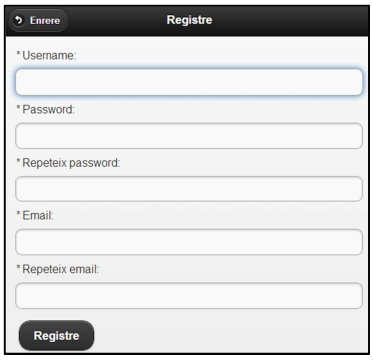


Fig. 20 - Pantalla de registre

Aquesta decisió en el disseny pot semblar un fre pels usuaris a registrar-se, però s'ha escollit, per tal de dotar a la aplicació de robustesa pel que fa als usuaris, ja que sino es pot arribar a l'escenari de tenir una aplicació on hi ha molts usuaris fantasmes (amb emails inventats), cosa que no es desitja, ja que les funcions online envien correus electrònics.

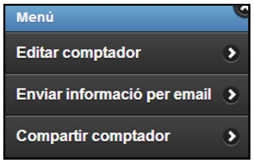


Fig. 21 - Menú amb opcions online

Un cop validat l'email quan l'usuari faci el *login*, ja serà un usuari vàlid i podrà fer servir les funcions online, que apareixeran en el menú del comptador, un cop faci *login*.

Les dades de l'usuari que fa *login* correctament en la aplicació, sí que es guarden en el dispositiu, concretament en el ***sessionStorage***. En aquesta informació que es guarda, a part de les dades propies de l'usuari, també trobem un sessionToken (string molt llarg, proporcionat per Parse.com), que farem servir posteriorment per a saber si l'usuari actual està autenticat en el sistema o no.

Quan l'usuari faci logout de la aplicació, símplement el que es farpa es netejar el *sessionStorage* amb el mètode següent:

```
function logouton() {
	sessionStorage.clear();
	if(sessionStorage.length==0) return true;
	return false;
}
```

7.11 Enviar informació per email

Una de les funcionalitats online és la d'enviar informació d'un comptador, per email, a un altre persona. Aquesta eina pot ser útil per donar a conèixer de forma fàcil, i des de la mateixa aplicació, informació rellevant per algú (com per exemple, progressos al deixar de fumar).

Fig. 22 - Pantalla enviament informació per email

Per tant, tenim que en aquesta pantalla es pot enviar un email a la persona que es desitgi (pot ser qualsevol email, no cal que estigui

registrat en el sistema), amb un assumpte, un text, i adjuntant, si es vol, la informació històrica del comptador.

La informació històrica a incloure podrà ser la del cicle actual (que encara no té data de fi), o la dels cicles guardats amb anterioritat.

Un altre característica d'aquesta pantalla és que es podrà triar la direcció amb la que es vol fer de remitent del correu. L'usuari podrà triar entre la direcció real de l'usuari, i una direcció fictícia i inexistent, amb el format:

<nomusuari>@theclickermaster.com

El motiu de donar aquestes dues opcions és, bàsicament, per si l'usuari vol mantenir la privacitat del correu electrònic amb el que es va donar d'alta. Dit d'un altre manera, és per si algú no vol donar a conèixer la seva adreça real.

Si entrem a nivell de codi, la funcionalitat d'enviament de correus s'ha construït fent servir un mòdul de Parse.com, anomenat **Cloud Code** (o codi en el núvol).

Parse.com dona la possibilitat de pujar codi en el seu servidor, i des de la aplicació, fer crides amb la API de Parse.com a aquest codi (en aquest cas, no són crides REST, sinó per *Ajax*, normal). En el nostre cas, hem creat un codi que fa servir el servei d'enviament de correus "Mailgun". Per fer-lo servir, ens hem hagut de crear un compte a Mailgun, i configurar-lo correctament.

El cloud code que hem pujat al servidor de Parse.com és aquest:

```
var Mailgun = require('mailgun');
Mailgun.initialize('theclickermaster.com', 'key-4a7wqc8bxlopt75fddbjkak96xlvXXXX');

Parse.Cloud.define("sendMail", function(request, response) {
    Mailgun.sendEmail({
      to: request.params.to,
      from: request.params.from,
      subject: request.params.assumpte,
      html: request.params.html
    }, {
      success: function(httpResponse) {
        console.log(httpResponse);
        response.success("Email enviat!");
      },
      error: function(httpResponse) {
```

```
        console.error(httpResponse);
        response.error("Error! Alguna cosa ha anat malament...");
      }
    });
});
```

Com es pot veure, la funció espera una *request*, i dins aquesta *request* li passem el remitent, el destinatari, assumpte, i el cos del missatge (que serà en format HTML, ja que moltes vegades el que enviarem serà una taula HTML amb la informació dels cicles).

```
var resposta;
Parse.Cloud.run('sendMail', {assumpte: assumpteP, to: toP, from: fromP, html: htmlP},
{
       success: function(result) {
              alert("Email enviat correctament!");
              window.history.back();
       },
       error: function(error) {
              alert("Error! No s'ha pogut enviar el correu.");
       }
});
return resposta;
```

En el codi anterior fem servir la funció *Parse.Cloud.run (nom-de-la-funcio-remota, parametres)*, per cridar al codi en el núvol en Parse.com.

El motiu de fer servir el servidor Parse.com per enviar correus, és que no és possible enviar correus des de un dispositiu client. És necessari un servidor de correu, i fent servir el *cloud code* de Parse.com i el servei d'enviament de correus en el núvol *Mailgun*, s'ha aconseguit aquesta funcionalitat.

7.12 Compartir comptador

La darrera funcionalitat online que té la aplicació és la de compartir (enviar, literalment) un comptador a un altre usuari de la aplicació. Això s'aconseguirà simplement introduint el correu del destinatari, és a dir, introduint la mateixa

Fig. 23 - Pantalla compartir comptador

direcció de correu amb la que el destinatari es va registrar en la aplicació.

Per motius de privacitat, no es suggerirà cap direcció de correu, ni apareixerà cap llistat amb els usuaris disponibles, sinó que haurà de ser el remitent qui conegui anticipadament la direcció de correu del destinatari.

Un cop estigui fet l'enviament, quan el destinatari faci *login* satisfactòriament en l'aplicació, li apareixerà en la seva Home el nou comptador. Cal dir que l'únic que s'envia és el comptador en sí; no s'envien ni alarmes, ni cicles. Això sí, s'envia el comptador en el mateix estat i configuració que el tenia l'usuari origen.

Aquest comportament s'aconsegueix fent servir com a canal d'informació el servidor Parse.com. El que ocorre és que quan l'usuari introdueix un *email* per a compartir, es fan les comprovacions pertinents (que el correu introduït no sigui el de ell mateix; que el correu introduït sigui el d'un usuari del sistema), i seguidament es puja el comptador a Parse.com, especificant en el camp ACL el "id" del usuari destinatari.

El camp ACL és un camp que proporciona Parse.com a tots els objectes, en el qual hi podem definir els permisos sobre quins usuaris podran escriure o llegir la informació d'aquell objecte. Especificant en el camp ACL l'identificador de l'usuari destí (un cop comprovat que aquest existeix), ens assegurem que aquell, i només aquell usuari rebrà el comptador.

La funció *Javascript* que pujarà el comptador a Parse.com és la següent:

```
function pujarComptador(dataP, to, userDesti) {

	var data = dataP.toJSON();

	delete data.objectId;
	delete data.id;
	delete data.createdAt;
	delete data.updatedAt;

	var ObjectId = userDesti[0]["objectId"];
	var ACL = {};
	ACL[ObjectId] = { "read": true,"write": true};
	data.ACL = ACL;//concateno el Id del usuari al comptador

	data.dataInici = { "__type": "Date", "iso": data.dataInici};

	data = JSON.stringify(data);
	// Send data:
	$.ajax({
		"type":"POST",
```

```
		"url":"https://api.parse.com/1/classes/Comptador",
		"data":data,
		"contentType":"application/json",
		"dataType":"json",
		"headers":headers,
		success:function(data, status, xhr) {

			if(status === "success") {

				alert("El comptador s'ha compartit correctament");

			} else {
				alert("Hi ha hagut un error al compartir el comptador");

			}
		},
		error:function(dataFail, status, xhr) {
			alert("Hi ha hagut un error al compartir el comptador");
		}
	});
}
```

Com s'ha dit, la funció anterior, *pujarComptador*, es crida automàticament al fer clic en "Compartir", i un cop fetes les comprovacions pertinents, de forma transparent a l'usuari.

L'altra part de la funcionalitat, la de rebre el comptador, es farà també automàticament i de forma transparent, quan l'usuari destinatari faci *login* satisfactòriament en l'aplicació, i es mirarà a Parse.com si l'usuari té algun comptador pendent de descarregar-se. Si és així, s'afegirà com a un comptador més dels que té en local, i s'esborrarà el comptador de Parse.com.

```
function obtenirComptadorsPendents() {

	//passem el token de l'usuari per que s'identifiqui
	var token = sessionStorage.getItem("token");
	headers["X-Parse-Session-Token"] = token;

	$.ajax({
		"async": false,
		"type":"GET",
		"url":"https://api.parse.com/1/classes/Comptador",
		"dataType":"json",
		"contentType":"application/json",
		"headers":headers,
		success:function(data, status, xhr) {
			for(var i = 0; i < data.results.length; i++) {
				var comptador = data.results[i];
				comptador.id = comptador.objectId;
				comptador.dataInici = comptador.dataInici.iso;
				elsMeusComptadors.add(comptador);
				deleteCounter(comptador.objectId);
			}
		}
	});
	delete headers["X-Parse-Session-Token"];
```

```
}
```

En la funció anterior, es fa una crida REST per obtenir els comptadors pendents, però en aquest cas, se li passa en la capçalera de la *request* REST, el *sessionToken* que al fer *login* s'ha guardat en el *sessionStorage* i que identifica a l'usuari en Parse.com. Si la crida és correcte i l'usuari té algun comptador pendent, mirant la ACL dels comptadors), es descarrega el comptador i s'esborrarà, mitjançant un altre funció que fa un altre crida REST a Parse.com (*deleteCounter*).

Cal dir que un cop s'ha rebut el comptador, aquest sempre actuarà com un comptador més, dels definits de forma "tradicional", i en ell es podran fer les mateixes accions que en qualsevol altre.

8. Línies de futur

En un primer moment, al començament del projecte, és va fer una llista de requeriments que la aplicació podria suportar. Es van prioritzar, de més senzilles, a més complexes, i finalment, la llista definitiva de funcionalitats s'ha quedat com s'ha pogut veure en la documentació.

En aquesta apartat, tot seguit es llistaran totes aquelles funcionalitat que s'han quedat fora de la versió entregada. No seria mala idea, revisar-les i incloure-les mica en mica, en futures versions de la aplicació.

- Afegir opció de decrementar un comptador.
- Afegir opció de botons físics per incrementar/decrementar comptador (volum *up* i *down*).
- Afegir una llista predeterminada de comptadors a escollir, quan es crea un de nou.
- Afegir gràfics per veure l'evolució d'un comptador.
- Fer que soni algun so al incrementar el comptador (opcionalment).
- Afegir la opció de posar un límit de clics al comptador.
- Afegir la possibilitat de fer un *backup* dels comptadors en Parse.com. Aquesta funcionalitat es va aconseguir, però no s'ha inclòs en l'entrega final ja que no tenia sentit sense la opció de obtenir les dades de *backup* de tornada al dispositiu. Al no poder-se fer aquest últim per manca de temps, s'ha tret també la opció de pujar-ho.
- Fer que quan s'engegui l'aplicació, aparegui com a primera pantalla la del darrer comptador seleccionat, en comptes de la Home.
- Afegir una pantalla d'edició de perfil d'usuari i de configuració general de la aplicació.

- Avisar d'alguna manera (visualment, amb vibració, etc...) que s'ha afegit un nou comptador provinent d'un altre usuari del sistema. Actualment, quan l'usuari que rep el comptador fa *login* en la aplicació i té comptadors pendents, simplement aquests apareixen en la seva Home, però sense cap distintiu que els diferenciï dels creats per ell. No s'ha pogut afegir cap distintiu a aquests nous comptadors, per manca de temps.

- Fer que la aplicació no necessiti internet per a funcionar. Actualment necessita internet per a descarregar-se els fitxers font de *JQueryMobile*, per muntar la pàgina.

Com es pot veure, hi ha moltes millores a fer. Aquesta és una primera versió, i com a tal, ha servit per a començar un projecte nou, i prometedor, però principalment ha servit per aprendre a utilitzar les tecnologies web actuals. Si el temps i les circumstàncies ho permeten, s'aniran afegint en un futur.

9. Conclusions

Un cop finalitzat el projecte, són moltes les conclusions que es poden extreure de tot plegat.

El primer de tots, i crec que el que més em farà servei en la meva vida professional, és que la informàtica avança rapidíssimament, i que cal mantenir-se al dia pel que fa a les noves tecnologies.

Abans de començar aquest projecte, jo no sabia res de les noves tecnologies web, ni HTML5, ni CSS3, ni sabia *Backbone.js*, ni *JQueryMobile*... M'ho vaig plantejar com un repte, i com a una molt bona oportunitat d'aprendre i de posar-me al dia en tecnologies web.

Al final del camí, i després de moltíssimes hores de llegir documentació, buscar per internet, tenir problemes... veig que la aplicació que n'ha resultat de tot plegat és una aplicació amb cara i ulls, i que compleix correctament amb els requisits que es van plantejar de bon principi i que es van pactar amb el tutor.

Si entrem més en detall, abans de començar el projecte, i com ja he dit, sense coneixement de les tecnologies que anava a tractar, pensava que *JQueryMobile* seria la "pedra angular" del projecte, és a dir, la tecnologia amb la que hauria de lidiar amb més intensitat. Doncs ha resultat que la tecnologia amb la que més he hagut de treballar no ha estat aquesta, sinó que ha estat *Backbone.js*. Com bé diu el seu nom, és la columna vertebral de la aplicació. *Backbone.js* s'ha convertit en el vertader esperit de la aplicació, i el que ha fet que finalment s'hagi aconseguit una aplicació tan fluida i ràpida, i que respongui tan bé als esdeveniments que ocorren.

JQueryMobile, ha esdevingut simplement l'eina amb la que s'han maquetat les pàgines, i gràcies a ell s'han dissenyat interfícies adaptables i components atractius i que segueixen les directrius del, tan de moda "*responive design*". Però un cop feta una plantilla, i ja conegut els components, no ha ocasionat gaire més problema.

Un altre dels descobriments d'aquest projecte ha estat el d'utilitzar el servei que ofereix Parse.com. És increïble les possibilitats que dona, i tot el que es pot arribar a fer i lo senzill que és d'utilitzar.

Un dels altres descobriments que he fet, ha estat el de fer crides REST a serveis ja existents. Fent servir *JQuery*, fer aquestes operacions es tornen molt senzilles.

El fet de fer plantejar-ho tot per a fer servir *PhoneGap* també ha estat un gran descobriment. Aquesta manera de desenvolupar aplicacions per a mòbils, és una gran avantatge, tant de temps, com (en una organització) de diners, ja que es desenvolupa un únic cop la plana web, en HTML5, i s'empaqueta, gràcies a *PhoneGap*, en diferents formats. Tot i que, com s'ha dit al començament de la documentació, en el projecte, només s'ha creat la aplicació per a dispositius *Android*, ja que per a fer-ho per a *iPhone* calia donar-se d'alta en un servei de pagament.

A nivell del programa obtingut respecte a les expectatives (requeriments) inicials, cal dir que s'ha complert tot el que es va plantejar a l'inici, excepte alguna cosa, com el *backup/reload* de les dades a Parse.com, per falta de temps, així com algunes altres millores visuals. Amb tot plegat, la aplicació funciona bé i de forma molt fluida en els dispositius *Android*.

En definitiva, estic molt content amb el resultat obtingut. La aplicació funciona i ho fa correctament, i amb un aspecte visual, en la meva humil opinió, bastant atractiu. A més a més, n'he extret un coneixement i una experiència que, de ben segur, em servirà i m'ajudarà en la meva vida professional.

10. Referències

[1] Backbone.js
http://backbonejs.org/ [en línia]

[2] Underscore.js
http://underscorejs.org/ [en línia]

[3] Jquery.js
http://jquery.com/ [en línia]

[4] JQueryMobile
http://jquerymobile.com/ [en línia]

[5] Parse.com
https://www.parse.com/ [en línia]

[6] Phonegap
http://phonegap.com/ [en línia]

[7] Apache Cordova
http://cordova.apache.org/ [en línia]

[8] Backbonetutorials
http://backbonetutorials.com [en línia]

[9] Dive into HTML5
http://diveintohtml5.info/ [en línia]

[10] Stackoverflow
http://stackoverflow.com/ [en línia]

[11] World Wide Web Consortium (W3C)
www.w3.org/ [en línia]

[12] Creador de diagrames de Casos d'ús
http://yuml.me/diagram/usecase/draw [en línia]

[13] Convertir backbone a parse
https://www.parse.com/docs/js_guide#convert [en línia]

[14] Fast click buttons de Google
https://developers.google.com/mobile/articles/fast_buttons [en línia]