Universitat Oberta de Catalunya

uoc.edu

PunkTrail App

Álvaro Manuel Rodríguez Porta
DADM
Àrea de treball final

Pau Dominkovics Coll
Carles Garrigues Olivella

5 de juny de 2019

FITXA DEL TREBALL FINAL

Títol del treball:	*PunkTrail App*
Nom de l'autor:	*Álvaro Manuel Rodríguez Porta*
Nom del consultor/a:	*Pau Dominkovics Coll*
Nom del PRA:	*Carles Garrigues Olivella*
Data de lliurament (mm/aaaa):	*06/2019*
Titulació o programa:	*DADM*
Àrea del Treball Final:	*Treball final de màster DADM aula 2*
Idioma del treball:	*Català*
Paraules clau	

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

PunkTrail App neix amb la finalitat de poder administrar la creació, difusió i gestió de curses de muntanya d'aquesta modalitat fent més fàcil la feina dels organitzadors per una banda (gestió), i les dels corredors (punt comú d'informació i inscripcions).

Una app multi-usuari amb les dades al núvol, client-servidor, amb dos tipus d'usuari: administrador i usuari.

Una app desenvolupada per iOs.

Abstract (in English, 250 words or less):

PunkTrail App was created to manage the creation, dissemination and management of mountain races, making the organizers' work easier on the one hand (management), and those of the runners (common point of information and inscriptions).
A multi-user app with data in the cloud, client-server, with two types of user: administrator and user.
An app developed for iOS.

Índex

Llista de figures

1. Introducció

1.1 Context i justificació del Treball

Existeix un tipus de cursa de muntanya anomenat PunkTrail, són un tipus de curses no competitives, no hi ha classificacions, ni temps, ni premis. És un tipus de cursa solidari, on la inscripció és gratuïta però a canvi el corredor té l'obligació moral de dur alguna cosa al banc d'aliments de la població i/o col·laborar amb la compra de samarreta, buff, guants de la organització...

La implicació és espectacular, i cada cop es recapten més aliments.

Les curses PunkTrails tenen un calendari (Imatge 1) per cada població que s'hi adhereix, sent un èxit ja que les inscripcions sempre s'exhaureixen en menys d'una hora, tot i només comptar amb diferents xarxes socials, algunes no tenen web, algunes utilitzen formulari de Google Docs per les inscripcions, i la informació no queda sempre clara...

Solen tenir un límit d'inscripcions de 400 persones. Aquest any són 9 curses, estaríem parlant d'unes 3600 inscripcions en el que es coneix com "Circuit Punk-Trail".

Il·lustració 1. Circuit PunkTrail 2019

La idea és unificar-ho tot en una App per ajudar als organitzador de les curses a unificar disseny, patrons, i aconseguir una millor gestió per a poder anar adherint poblacions i carreres al calendari PunkTrail.
I també ajudar als corredors a que tinguin un únic punt clar on consultar els detalls de cada carrera i la possible inscripció.

No existeix cap App ara mateix on es gestioni la informació i inscripcions a cap cursa.

Existeix l'App de runedia (plataforma de El Mundo Deportivo), que és un cercador de curses per comunitat autònoma, però on no solen sortir les punktrails, o surt molt poca informació i no es poden gestionar directament les inscripcions.

Fent una recerca es pot trobar una app de carreres de muntanya d'Andorra anomenada "Ultra 2014", os es troba la informació de varies modalitats de carreres, distàncies, mapes, com arribar-hi... però que és del 2014, i no ha renovat contingut.

Tot el relacionat amb el TrailRunning (carreres de muntanya) està orientat a mapes, guies, revistes però no a informació de carreres.

I pel que fa a carreres grans com la Marató de Barcelona, té la seva app on s'informen dels detalls, notícies, seguiment en viu de l'atleta, però que serveix únicament quan el corredor ja està inscrit.

Totes les curses de muntanya es gestionen mitjançant la web (amb més o menys disseny depenen de l'organitzador). Una empresa gran com Klassmark que gestiona 7 carreres té una web principal https://www.klassmark.com/trail i després una web per cada carrera.

Aquesta App també podria ser útil per empreses que volguessin gestionar les seves pròpies carreres.

1.2 Objectius del Treball

La App serà multi-usuari amb les dades al núvol, sent una app client-servidor, per tant requerirà d'una connexió per tal de consultar la base de dades.

Al sistema es podrà loginar el administrador.

Es requerirà una base de dades d'on s'extraurà la informació per mostrar/editar/completar.

La App tindrà dos nivells: nivell usuari i nivell administrador.

A nivell usuari la app tindrà una home amb la informació de cada carrera: Títol i data, on l'usuari polsarà sobre la que necessiti ampliar la

informació. El títol serà el nom de la població, per tal de que l'usuari vegi dues coses claus: Població i data de la cursa.

Un cop seleccionada una localitat/cursa es visualitzarà les dades més importants de la cursa: recorreguts i el programa (hora de sortida) i els apartats: Reglament, Banc d'aliments, Aparcament, Inscripcions, Col·laboradors, Població i Contacte.

A nivell administrador es podrà afegir, editar i eliminar curses, obrir o tancar les inscripcions d'una cursa i extreure el llistat d'inscrits per el dia de la cursa.

En el cas d'afegir una cursa al calendari, s'afegiran tots els camps a la base de dades, tot entrant els ítems necessaris.

En el cas d'edició es mostrarà en el quadre de text el contingut actual, per tal de modificar-ho.

L'administrador podrà extreure el llistat d'inscripcions del dia de la cursa.

1.3 Enfocament i mètode seguit

No hi existeix cap App amb aquesta funcionalitat, per tant serà un desenvolupament de producte nou, per una tasca específica: per tal de l'administrador, gestionar les curses i inscripcions d'una manera senzilla des de un dispositiu mòbil, on no caldrà tenir coneixements en pàgines web per administrar-les.

Per tal de l'usuari, mostrar la informació i el formulari d'inscripcions per enviar les dades.

1.4 Tipus d'aplicació

Serà una aplicació nativa iOS en aquest TFM. Posteriorment es programarà per a Android per tal de cobrir tots dos possibles corredors que tinguin iPhone o Android.

S'ha descartat una app híbrida per tal de treballar-hi amb detall a cadascun dels sistemes operatius, i també ha quedat descartada una aplicació web per tal d'obligar a descarregar la app als dispositius mòbils i així aconseguir més rapidesa alhora de rebre inscripcions o arribar a més públic.

També serà més senzill per als organitzadors poder crear les curses a través del seu mòbil/Tablet.

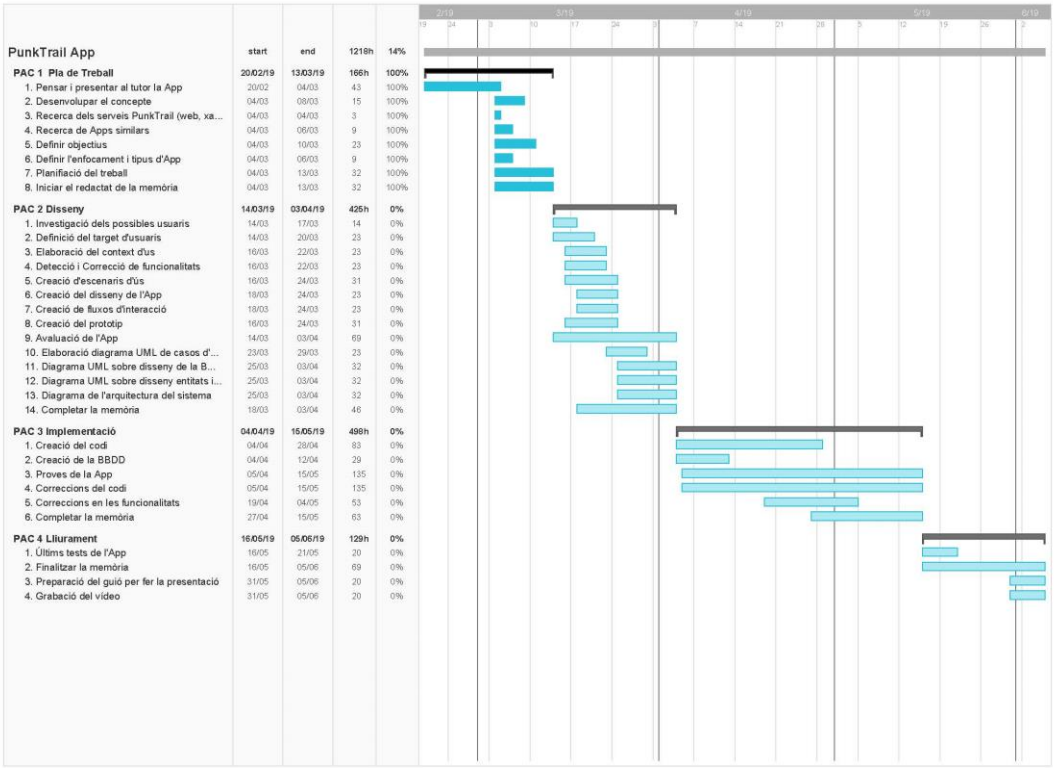
1.5 Requeriments de l'aplicació

Per tal de poder fer funcionar l'aplicació es requerirà connexió a internet per part de l'usuari.

Per el seu funcionament es necessitarà un servidor per la base de dades i allotjament per les fotografies.

1.6 Planificació del Treball

En la creació del diagrama de Gantt (Imatge 2), el càlcul de les hores ha sigut realitzat amb la fórmula 3 hores dia laboral, 4 hores dia festiu.



PunkTrail App	start	end	1218h	14%
PAC 1 Pla de Treball	20/02/19	13/03/19	166h	100%
1. Pensar i presentar al tutor la App	20/02	04/03	43	100%
2. Desenvolupar el concepte	04/03	08/03	15	100%
3. Recerca dels serveis PunkTrail (web, xa...	04/03	04/03	3	100%
4. Recerca de Apps similars	04/03	06/03	9	100%
5. Definir objectius	04/03	10/03	23	100%
6. Definir l'enfocament i tipus d'App	04/03	06/03	9	100%
7. Planificació del treball	04/03	13/03	32	100%
8. Iniciar el redactat de la memòria	04/03	13/03	32	100%
PAC 2 Disseny	14/03/19	03/04/19	425h	0%
1. Investigació dels possibles usuaris	14/03	17/03	14	0%
2. Definició del target d'usuaris	14/03	20/03	23	0%
3. Elaboració del context d'us	16/03	22/03	23	0%
4. Detecció i Correcció de funcionalitats	16/03	22/03	23	0%
5. Creació d'escenaris d'ús	16/03	24/03	31	0%
6. Creació del disseny de l'App	18/03	24/03	23	0%
7. Creació de fluxos d'interacció	18/03	24/03	23	0%
8. Creació del prototip	16/03	24/03	31	0%
9. Avaluació de l'App	14/03	03/04	69	0%
10. Elaboració diagrama UML de casos d'...	23/03	29/03	23	0%
11. Diagrama UML sobre disseny de la B...	25/03	03/04	32	0%
12. Diagrama UML sobre disseny entitats i...	25/03	03/04	32	0%
13. Diagrama de l'arquitectura del sistema	25/03	03/04	32	0%
14. Completar la memòria	18/03	03/04	46	0%
PAC 3 Implementació	04/04/19	15/05/19	498h	0%
1. Creació del codi	04/04	28/04	83	0%
2. Creació de la BBDD	04/04	12/04	29	0%
3. Proves de la App	05/04	15/05	135	0%
4. Correccions del codi	05/04	15/05	135	0%
5. Correccions en les funcionalitats	19/04	04/05	53	0%
6. Completar la memòria	27/04	15/05	63	0%
PAC 4 Lliurament	16/05/19	05/06/19	129h	0%
1. Últims tests de l'App	16/05	21/05	20	0%
2. Finalitzar la memòria	16/05	05/06	69	0%
3. Preparació del guió per fer la presentació	31/05	05/06	20	0%
4. Grabació del vídeo	31/05	05/06	20	0%

Il·lustració 2. Diagrama de Gantt

La planificació serà en mode cascada, amb varies iteracions per tal d'anar corregint amb possibles soluciones i millores fins arribar al producte final, en especial en termes de disseny, escenaris d'ús...

La prioritat serà que l'aplicació executi sense problemes totes les funcions definides, amb el següent ordre de prioritats i estructura de desenvolupament:

1. Crear la base de dades a Firebase Database

2. Crear estructura d'arxius a Firebase Storage
3. Programar la funció per a Mostrar llistat de curses
4. Navegar entre els diferents apartats de la cursa
5. Crear, editar, eliminar cursa
6. Gestió d'inscripcions per part del corredor
7. Extreure llistat d'inscripcions

1.7 Breu sumari de productes obtinguts

A l'acabar el TFM obtindrem:
1. Aplicació funcional
2. La memòria sobre l'aplicació
3. Un manual d'usuari per administrador

1.8 Breu descripció dels altres capítols de la memòria

2. Usuaris i context d'ús: es definirà a quin tipus de target va dirigida l'aplicació i en quines situacions serà d'utilitat fer-la servir.

3. Escenaris d´ús: es mostrarà totes les funcions disponibles que l'aplicació podrà executar.

4. Prototipat: mostra de navegació entre pantalles en mode usuari i administrador, i visualització de disseny de les pantalles més importants.

5. Avaluació: diagrames de casos d'ús, estructura de base de dades, diagrama de classes, tipus de model d'aplicació...

6. Implementació: explicació de la presa de decisions del codi, disseny i descripció del codi de les funcions més importants.

7. Conclusions: recull amb el resultat obtingut, procediments seguits i funcionalitats per afegir.

2. Usuaris i context d'ús

Després d'haver analitzat el públic d'aquestes curses en més de 10 esdeveniments d'aquest tipus, s'extreu l'anàlisi de que els usuaris d'aquesta aplicació tenen les següents característiques:

- Edat entre 22 i 45 anys
- 70% homes, 30% dones
- Principalment procedents de províncies de Barcelona i Manresa
- Poder adquisitiu mig
- Gent extravertida, amb bon humor i solidària
- Esportistes per plaer (no competitius)
- 70% catalanoparlant 30% castellanoparlant

Aquestes característiques faran que l'idioma de l'App sigui en català, amb un disseny intuïtiu i amb la informació justa i necessària per als corredors.

Els contexts d'ús principalment seran:

- Nivell usurari: Corredors que necessitin consultar informació i/o inscriure's a la cursa. Les dades més importants sempre seran la data de la cursa, la localització i la distància a recórrer.
- Nivell administrador: Crear una cursa nova, editar informació d'una cursa, eliminar curses, mostrar/ocultar el formulari d'inscripció, extreure llistat d'inscrits per el dia de la cursa.

Usuari 1. Alberto

Il·lustració 3. Imatge de Alberto

Alberto

Edat: 44
Ocupació: Encarregat d'obra
Educació: Batxillerat
Estat civil: Casat, pare de 2 fills
Viu a: Barcelona
Accés a internet: iPhone
Ordinador: MacBook Pro
Hores online al dia: 5
Coneixements informàtics: avançats
Ús online per a: instagram, prensa digital, twitter, compres, navegació web...

Breu descripció
Alberto era jugador de futbol amateur i quan va retirar-se va provar diferents esports, fins que un amic li va fer sortir córrer a la muntanya. Es va enganxar a un grup d'amics del barri, sortint 2-3 cops per setmana a Collserola.
Alberto gaudeix de cada sortida, acudits dolents, diversió i una foto per pujar a Instagram.
Va descobrir les punktrails de les que comparteix tota la seva essència.

Context d'ús
Alberto necessitarà saber les dates i informació de les pròximes curses d'una manera ràpida, i omplir la inscripció d'una forma còmode
Alberto segueix per les xarxes socials a 4-5 perfils de cada carrera de punkTrail, i aconsegueix trobar el link d'inscripcions quan surten, sempre estan alerta. Diversos amics li demanen que els hi faci la inscripció en el seu nom, perquè ells no troben el link, ja sigui perquè no tenen reds socials, o no troben el link.

PunkTrail App
L'aplicació li farà més fàcil la vida.
Veurà la data quan s'obren les inscripcions i s'incribirà.
No caldrà que enviï el link per whatsapp als amics ni haver d'enregistrar inscripcions en nom de ningú.

Usuari 2. Maria

<table>
<tr>
<td> 
Il·lustració 4. Imatge de Maria</td>
<td>Maria

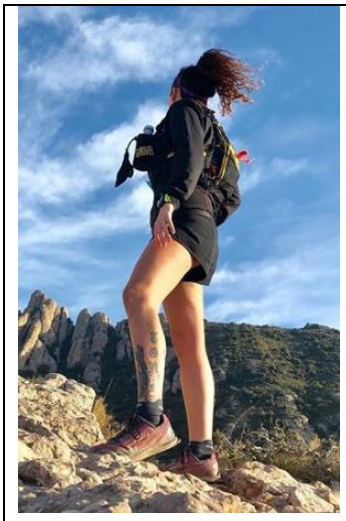
Edat: 39
Ocupació: Administrativa
Educació: Batxillerat
Estat civil: Divorciada, mare d’una filla
Viu a: Terrassa
Accés a internet: iPhone
Ordinador: no
Hores online al dia: 5
Coneixements informàtics: baix
Ús online per a: instagram, prensa digital, navegació web...</td>
</tr>
<tr>
<td colspan="2">Breu descripció
Maria gaudeix de la muntanya, li ajuda a desconnectar dels problemes quotidians.
Amistat, riures, compartir bons moments, sens dubte li encanta l’esperit de les punktrails.</td>
</tr>
<tr>
<td colspan="2">Context d’ús
Maria acostuma a no trobar mai com inscriure’s a les curses.</td>
</tr>
<tr>
<td colspan="2">PunkTrail App
L’aplicació sens dubte serà de gran ajuda per no dependre de ningú i poder inscriure’s sense problemes.</td>
</tr>
</table>

Usuari 3. Sergi

<table>
<tr>
<td> 
Il·lustració 5. Imatge de Sergi</td>
<td>Maria

Edat: 35
Ocupació: Operari
Educació: Secundari
Estat civil: Casat, pare de dos fills
Viu a: Manresa
Accés a internet: iPhone
Ordinador: no
Hores online al dia: 5
Coneixements informàtics: baix
Ús online per a: instagram, navegació web, jocs online...</td>
</tr>
<tr>
<td colspan="2">Breu descripció
Sergi viu a un poble a prop de Manresa, ja fa anys que li agrada el trailrunning i cada cop són més colla alhora de sortir a córrer. Sergi és extravertit, amb un bon humor contagiós, li agrada la vida social al poble. Un dia van pensar crear una cursa de muntanya que ajudés al poble, no econòmicament, sinó al banc d’aliments on molta gent ho agrairia. Així va néixer la PunkTrail.</td>
</tr>
<tr>
<td colspan="2">Context d’ús
Sergi no té molta habilitat al món digital, es defensa a les xarxes socials, però el disseny web no ho controla gens. Crea un perfil de facebook on anirà publicant tota la informació de la cursa i el link d’inscripcions quan toqui.</td>
</tr>
<tr>
<td colspan="2">PunkTrail App
L’aplicació ajudarà a Sergi a crear el contingut i gestionar-ho, d’una forma intuïtiva, només demanant-li que ompli els camps Standard, algun logo i alguna imatge si ho desitja.</td>
</tr>
</table>

Usuari 4. Gemma

<table>
<tr>
<td> 
Il·lustració 6. Imatge de Gemma</td>
<td>Gemma

Edat: 30
Ocupació: Mestre
Educació: Post-universitaria
Estat civil: Soltera
Viu a: Sabadell
Accés a internet: Samsung
Ordinador: Sí, Asus
Hores online al dia: 5
Coneixements informàtics: alts
Ús online per a: navegació web, premsa...</td>
</tr>
<tr>
<td colspan="2">Breu descripció
Gemma corre per desconnectar de la feina. No té xarxes socials perquè té molta vida social, organitza punktrails i anima als pobles veïns a fer-ho.</td>
</tr>
<tr>
<td colspan="2">Context d’ús
Gemma fins ara gastava temps creant webs a wordpress per la punktrail que organitzava.</td>
</tr>
<tr>
<td colspan="2">PunkTrail App
Amb la aplicació tot serà molt més ràpid i fàcil de muntar, cosa que ajudarà a Gemma a ensenyar-ho a pobles del voltant per animar a crear una altra carrera al circuit.</td>
</tr>
</table>

3. Escenaris d´ús

Els diferents escenaris d'ús de l'aplicació seran:

1. Consulta de calendari

Al Alberto li han convidat un dissabte de març a una calçotada familiar a Valls. Ell es va apuntar a una cursa un cap de setmana però no recorda quina data era. Obre la PunkTrail app, veu la data de la cursa, i podrà confirmar la seva assistència a la calçotada.

2. Consulta d'informació d'alguna cursa en concret

Maria no ha tingut temps d'entregar gaire i la cursa ja s'acosta. Decideix consultar el tipus de recorregut per decidir si podrà fer el llarg o el curt.
Obre l'App, selecciona la cursa on està apuntada i obté els detalls de cada recorregut (distància i desnivell).

3. Inscripció a una cursa

Maria al grup de whatsapp ja ha vist que els seus amics s'estan inscrivint a la cursa, ella obre la app, selecciona la cursa, navega a l'apartat "Inscripcions" i omple el formulari. Ja està registrada ella també.

4. Afegir una nova cursa al sistema

Sergi ha de crear la cursa a l'App per primer cop.
L'usuari polsa una icona a la pantalla principal del llistat de curses, es logueja i torna a la pantalla principal de les curses. Veurà un botó afegir cursa i tindrà que omplir els camps:

- Població
- Data
- Programa
- Col·laboradors
- El nostre poble
- Contacte
- Afegir una imatge principal
- Afegir una icona de la cursa
- Afegir una imatge de la població
- Afegir una imatge del col·laborador principal

5. Editar una cursa

L'usuari Sergi ha d'editar la informació sobre el Banc d'Aliments, des d'on l'han demanat que hi publiqui la llista dels productes prioritaris.
Sergi obre l'App, polsa el botó Admin, es logueja. Selecciona la opció Editar Cursa, selecciona la seva cursa i edita el camp "Banc d'aliments". Polsa "Guardar" i ja està la informació actualitzada.

6. Eliminar una cursa

A Sergi li ha trucat el seu amic del poble El Pont de Vilomara, comentant-li que aquest any no podran organitzar la cursa, i li ha demanat que esborri la cursa de l'App.

Sergi obre l'App, polsa el botó Admin, es logueja. Selecciona la opció "Eliminar cursa". Selecciona la cursa a eliminar, confirma l'acció i la cursa queda eliminada del sistema.

7. Mostrar el formulari d'inscripcions

Ha arribat el dia en el que s'obren les inscripcions, i Sergi a les 10:00h del matí obre la App, selecciona la carrera, navega al apartat "Inscripcions" i polsa el botó Mostrar. Ara els corredors es podran inscriure fins un màxim de 400 inscripcions, després el formulari s'ocultarà.

8. Gemma anima a crear una altre carrera a Salelles

Gemma està amb uns amics de Salelles, els hi mostra la App i els hi ensenya la facilitat per crear una carrera al sistema, fet que facilita que s'animin a crear una altra carrera del circuit PunkTrail.

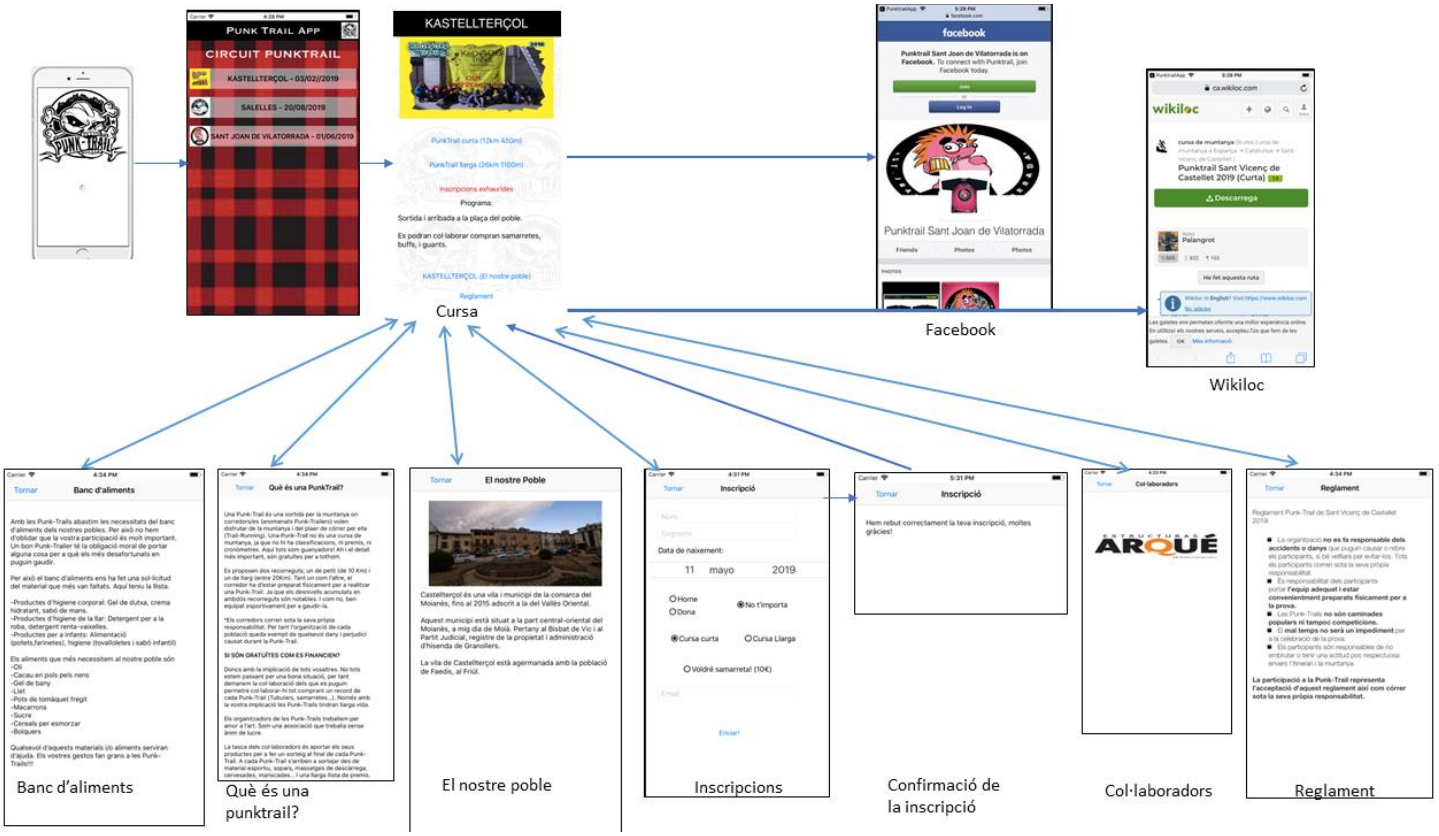
9. Extreure el llistat d'inscripcions

Arriba el dia de la cursa, Sergi necessita el llistat de corredors per passar llista abans de la sortida. Extreu el llistat i l'imprimeix.

En aquest link es podrà descarregar les inscripcions.

4. Prototipat

4.1 Mapa de navegació d'usuari

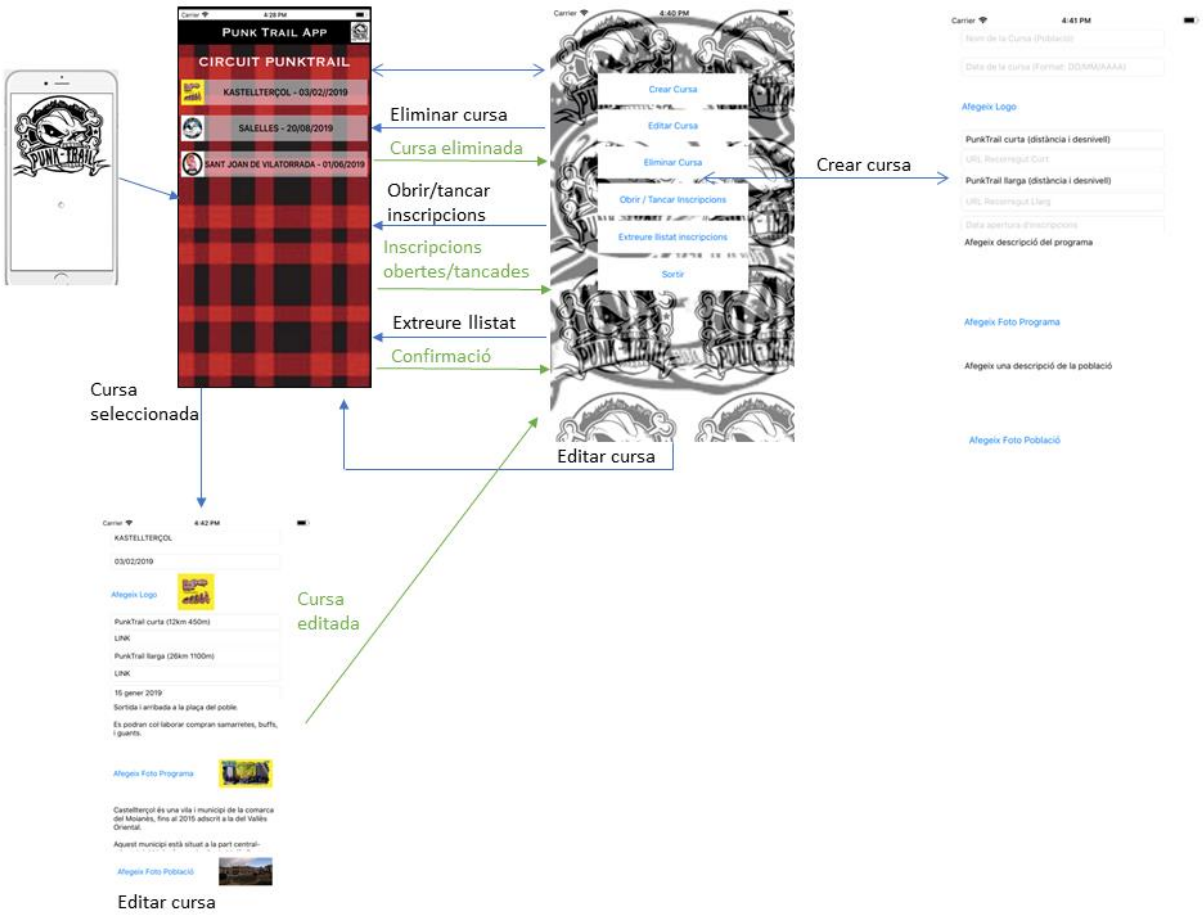


Il·lustració 7. Mapa de navegació d'usuari

En el primer pas es mostra el llistat de curses amb el nom de la població i la data de la cursa.

Un cop seleccionem veurem la informació més destacable de la cursa, i podrem navegar entre els diferents apartats.

4.2 Mapa de navegació d'administrador



Il·lustració 8. Mapa de navegació d'Administrador

A la pantalla principal de selecció de cursa, per entrar en mode "Administrador" es polsarà el botó del logo PunkTrail situat a dalt a la dreta. Un cop fet el login, es podrà escollir una opció de la llista de funcions.

4.3 Prototips

1. Pantalla de càrrega de la App. Imatge amb el logo de les PunkTrails mentre es carrega la pàgina general amb la informació de les curses.

Il·lustració 9. Carrega de App

2. Pantalla general, amb totes les curses. En polsar en la cursa desitjada s'obtindrà tota la informació. També hi tenim el botó "Admin" per accedir a afegir cursa, editar o eliminar cursa.
   Consultats molts corredors, s'ha arribat a la conclusió que el més important per mostrar de les curses són la data i el nom de la cursa que acostuma a ser el nom de la població. Així que aquesta informació es mostrarà acompanyada amb el logo de la cada cursa:

Il·lustració 10. Pantalla principal de l'App

3. Pàgina inicial quan es selecciona una cursa on es mostrarà una imatge principal, 2 links amb el recorregut de les curses.
   Un botó d'inscripcions:
   - Verd si està obert el període d'inscripcions.
   - Blanc amb la data d'apertura d'inscripcions.
   - Vermell si les inscripcions ja estan exhaurides.

   Veurem el programa de la cursa, i després els botons amb més informació sobre el poble, reglament, link a facebook (on es pujaran fotos i vídeos de l'esdeveniment), banc d'aliments, definició de punktrail, col·laboradors i un botó per sortir a la pàgina principal.

   Aquesta pantalla té tot l'essencial per al corredor interessat en inscriure's a la cursa.

   En un principi la idea era fer un menú desplegable per navegar entre les diferents opcions, però aquest disseny és més còmode per a l'usuari, que si polsa sobre "Reglament" pot tornar cap al darrere, evitant tenir que mostrar el menú desplegable i fent-li escollir algun altre ítem.

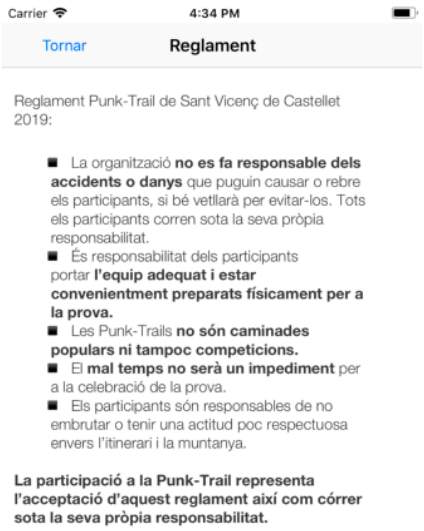
Il·lustració 11. Pantalla Inici Info Cursa

En els següents apartats, només caldrà text, per explicar concretament cada temàtica:

4. Què és una Punktrail?

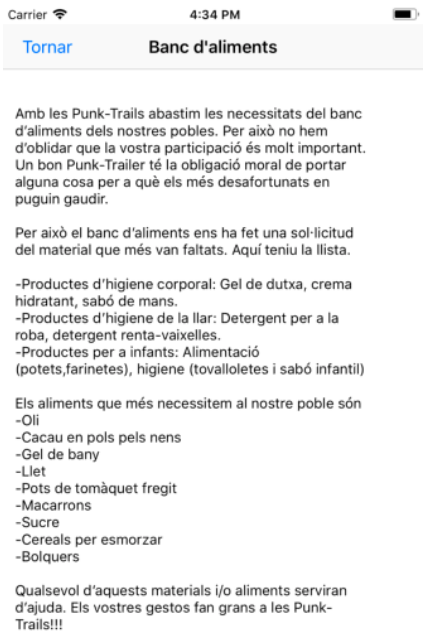
Il·lustració 12. Apartat Què és una punktrail

5. Apartat "Reglament":



Il·lustració 13. Apartat Reglament

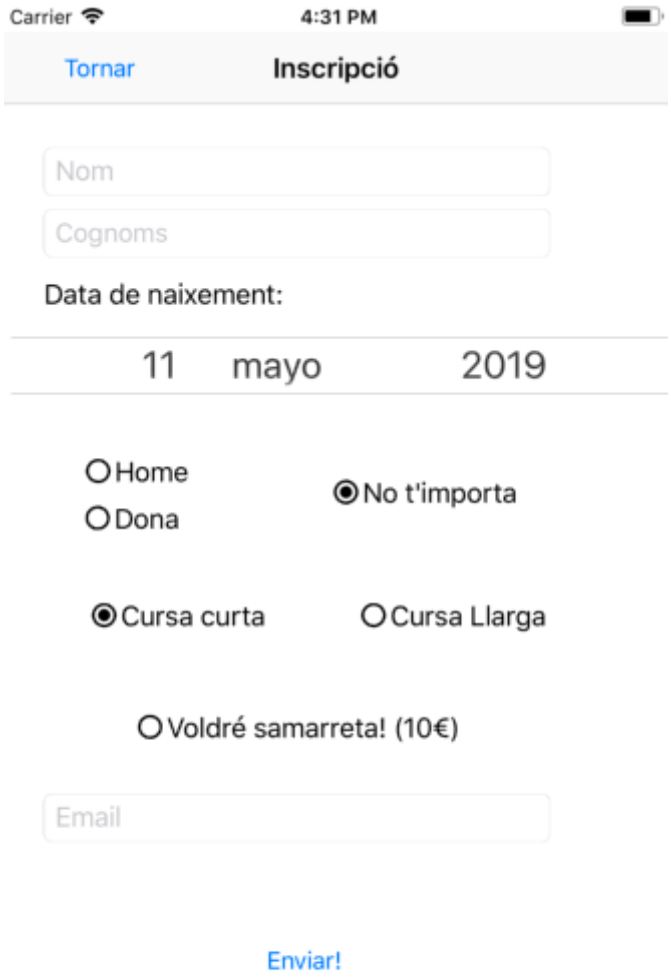
6. Pantalla "Banc d'aliments"



Il·lustració 14. Pantalla Banc d'aliments

S'ha decidit per un text pla descriptiu intentant ser el més breu possible per aquestes pantalles, explicar en cada apartat el més important per si algun corredor ho vol llegir alguna vegada.

7. Apartat d’inscripcions amb els camps editables Nom, cognom, data de naixement, sexe, selecció de cursa curta o llarga, email i el botó de confirmació per poder omplir el formulari en menys d’un minut. Són les dades que en la majoria de curses punktrails estan demanant.



Il·lustració 15. Pantalla Inscripcions

Un cop feta la inscripció rebrem un missatge de confirmació.

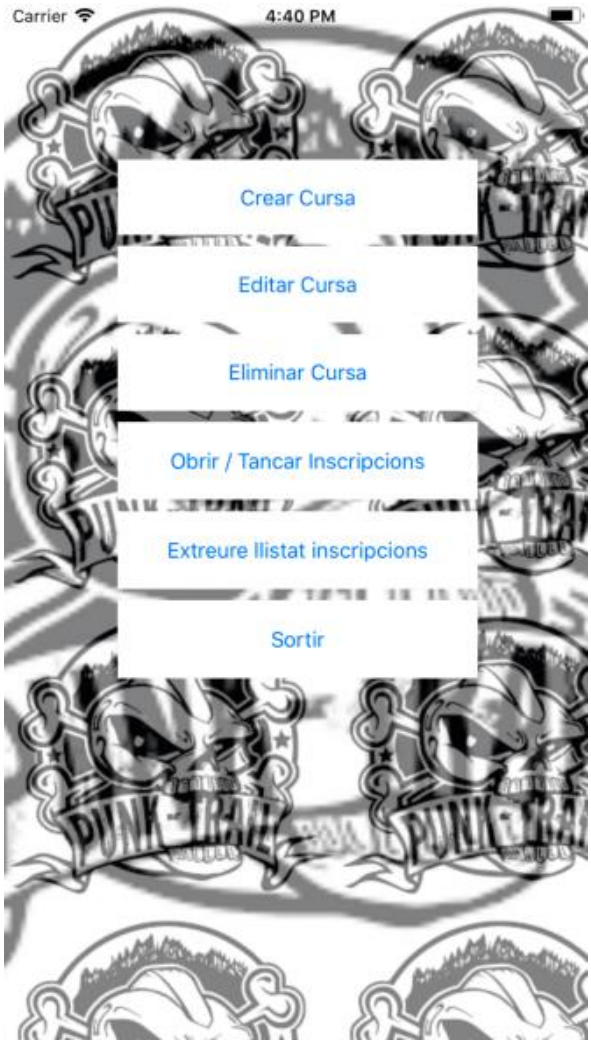
8. Pantalla col·laboradors: reconeixement de les empreses/negocis locals que han col·laborat d'alguna manera amb la cursa amb els logos linkables a les seves webs. És important donar visibilitat i agrair d'alguna manera tota la ajuda, si es donant a conèixer el negoci, l'ajuntament, la empresa de serveis...

Il·lustració 16. Pantalla col·laboradors

9. Pantalla "El nostre poble". Una breu descripció del poble que acull la cursa, acompanyada d'una fotografia. No es voldria omplir de massa text, alguna cosa breu i en llenguatge directe de què és el que trobaran.

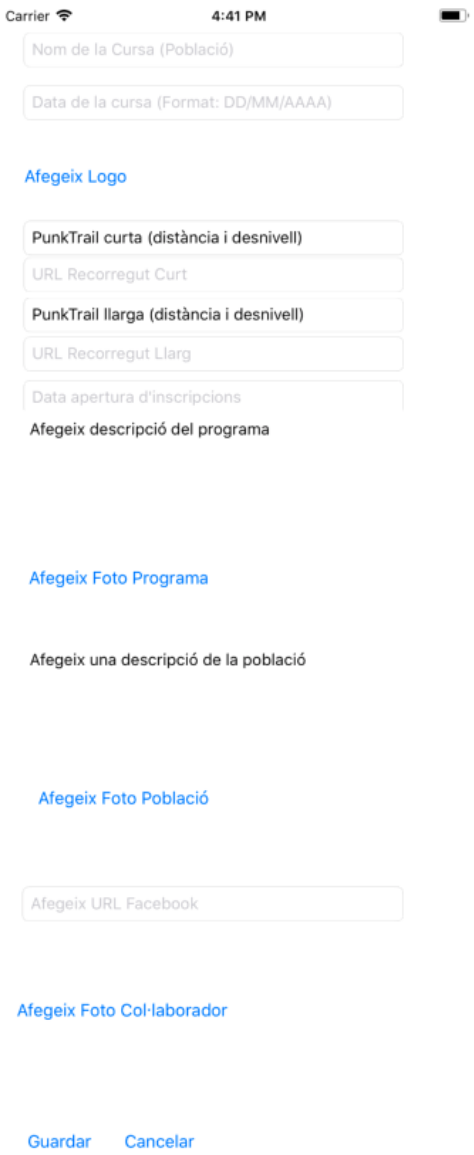
Il·lustració 17. Pantalla "El nostre poble"

10. Pantalla Administrador. En el mode administrador es podrà escollir sis opcions: Afegir cursa, editar cursa, eliminar cursa, obrir/tancar inscripcions, extreure llistat d'inscripcions i sortir. Disseny sense complexitat, intuïtiu per l'usuari, fàcil i ràpid, només ha de polsar el botó de l'acció que desitgi.



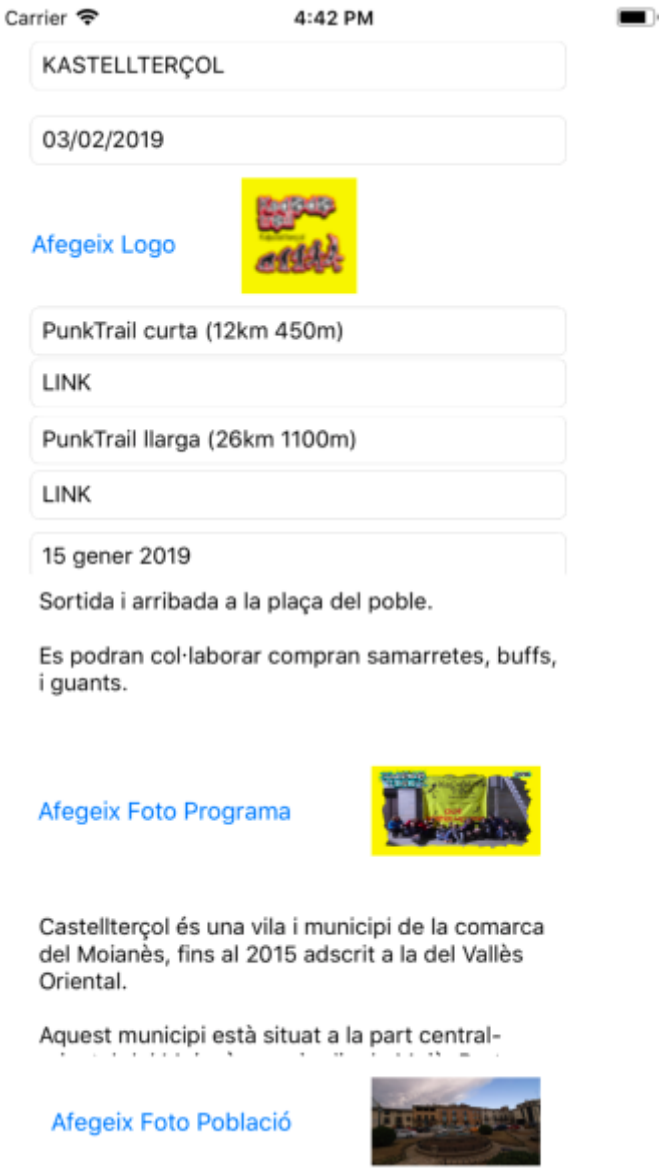
Il·lustració 18. Escollir opció

11. Afegir cursa. L'administrador disposarà dels camps editables per incloure el nom de la cursa, la data, el programa, afegir un logo, imatge per al programa, imatge per a la població, col·laborador, afegir la data d'apertura d'inscripcions.
    Amb uns cinc minuts pot crear una cursa al sistema sense tindre coneixements informàtics.



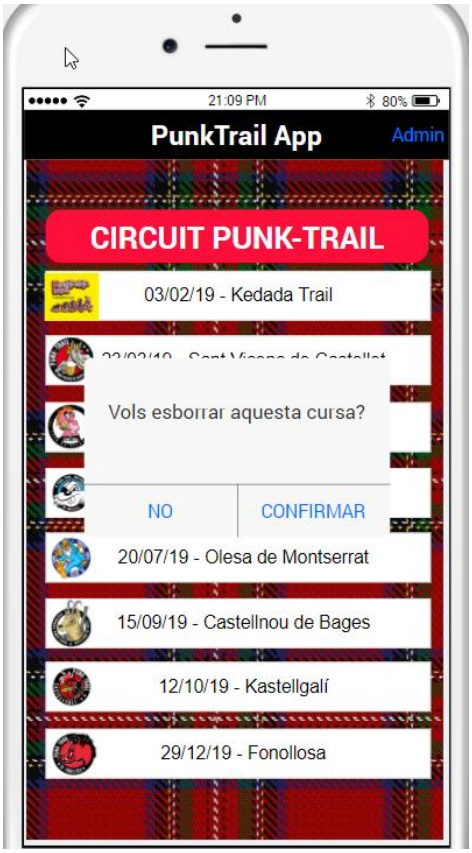
Il·lustració 19. Afegir cursa I

12. Pantalla “Editar cursa”. Es mostra tots els camps editables de la cursa seleccionada, i els botons de Guardar i Cancel·lar.



Il·lustració 20. Pantalla Editar Cursa

13. Si es polsa el botó “Eliminar cursa”, s’anirà a la pantalla inicial, i es seleccionarà una cursa per eliminar. Es demanarà la confirmació abans de esborrar-la.



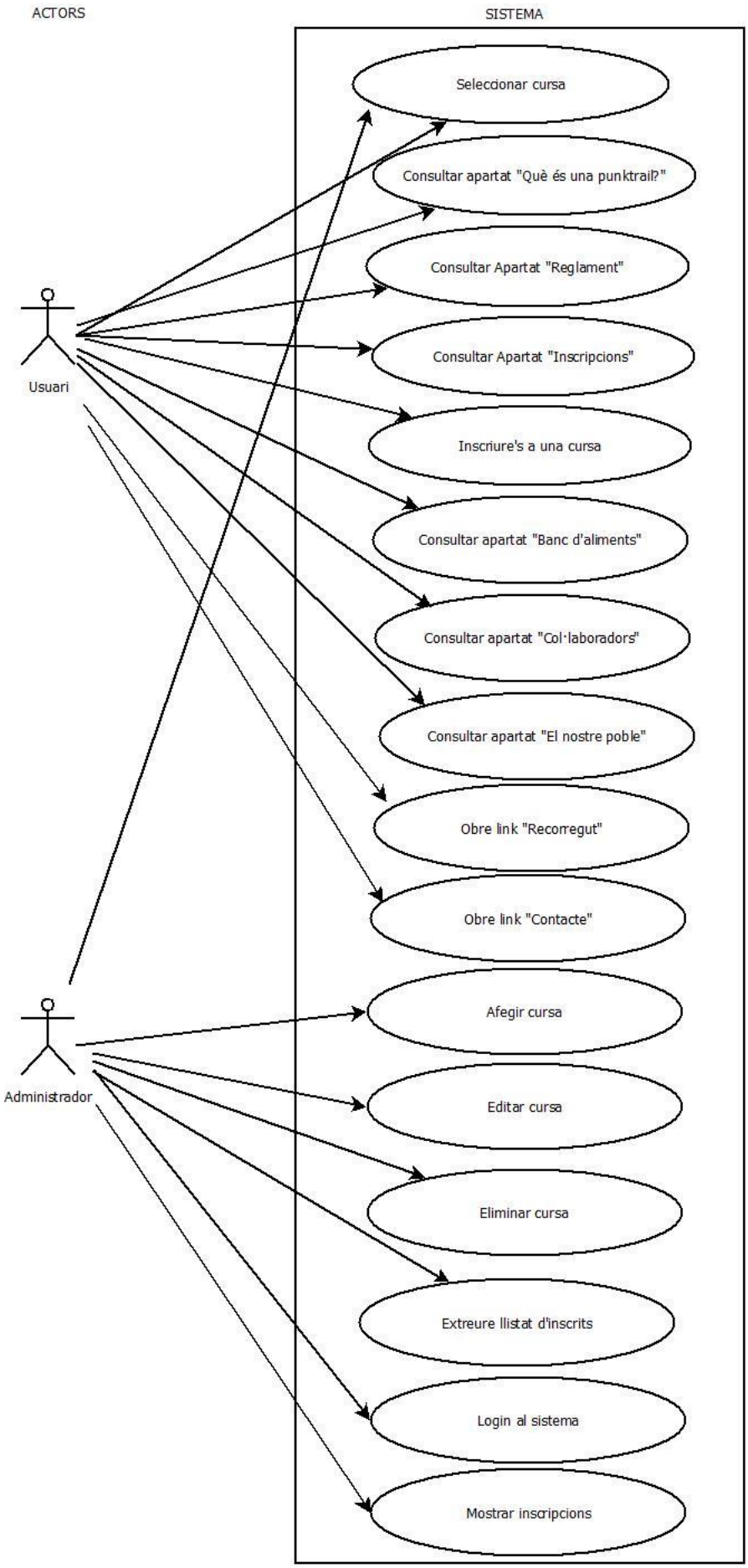
Il·lustració 21. Pantalla Borrar cursa

5. Avaluació

5.1 Diagrama UML de casos d'us

A continuació es mostra el diagrama UML dels casos d'ús. Diferenciem dos actors: usuari i administrador.

L'usuari podrà consultar i inscriure's a la cursa seleccionada.
L'administrador podrà fer login al sistema, afegir, editar i eliminar curses.


ACTORS
SISTEMA
Usuari
Administrador
Seleccionar cursa
Consultar apartat "Què és una punktrail?"
Consultar Apartat "Reglament"
Consultar Apartat "Inscripcions"
Inscriure's a una cursa
Consultar apartat "Banc d'aliments"
Consultar apartat "Col·laboradors"
Consultar apartat "El nostre poble"
Obre link "Recorregut"
Obre link "Contacte"
Afegir cursa
Editar cursa
Eliminar cursa
Extreure llistat d'inscrits
Login al sistema
Mostrar inscripcions

Il·lustració 22. Diagrama UML Casos d´ús

5.2 Llistat de casos d'us

UC1. Seleccionar cursa

Nombre: UC1- Selecciona una cursa del llistat
Actors: Usuari-Sistema
Descripció: L'usuari veu el llistat de curses i selecciona una.
Pre-condicions: Obrir la App i càrrega de llistat de curses
Flux Normal: 1. L'usuari selecciona una cursa del llistat. 2. L'usuari veu la pantalla general amb la informació bàsica de la cursa seleccionada.
Flux Alternatiu: 1a. L'usuari polsa el botó "Admin" 2a. L'usuari no té les credencials per accedir i torna endarrere
Postcondicions: L'usuari va a la pantalla general de la cursa amb la informació bàsica

UC2. Consulta apartat "Què és una punktrail?"

Nombre: UC2- Navega a l'apartat on s'explica què és una punktrail
Actors: Usuari-Sistema
Descripció: L'usuari selecciona una cursa polsa sobre l'ítem "Què és una punktrail?".
Pre-condicions: Haver seleccionat una cursa.
Flux Normal: 1. L'usuari selecciona el ítem "Què és una punktrail?".
Flux Alternatiu: 1a. L'usuari tanca l'App
Postcondicions: L'usuari accedeix a la pantalla "Què és una punktrail?"

UC3. Consulta apartat "Reglament"

Nombre: UC3- Navega a l'apartat on s'explica el reglament
Actors: Usuari-Sistema
Descripció: L'usuari selecciona una cursa i polsa sobre l'ítem "Reglament".
Pre-condicions: Haver seleccionat una cursa.
Flux Normal:

1. L'usuari selecciona el ítem "Reglament".
Flux Alternatiu: 1a. L'usuari tanca l'App
Postcondicions: L'usuari accedeix a la pantalla "Reglament"

UC4. Consulta botó "Contacte"

Nombre: UC4- Botó contacte dins de la cursa
Actors: Usuari-Sistema
Descripció: L'usuari ha seleccionat la cursa i polsa el botó "Contacte"
Pre-condicions: Haver seleccionat una cursa.
Flux Normal: 1. L'usuari clicka el botó Contacte. 2. Sortirà de l'App cap al facebook de la cursa.
Flux Alternatiu: 1a. L'usuari tanca l'App
Postcondicions: L'usuari accedeix a la pantalla Cursa

UC5. Consulta apartat "Inscripcions"

Nombre: UC5- Navega a l'apartat on es pot inscriure's a la cursa.
Actors: Usuari-Sistema
Descripció: L'usuari selecciona una cursa i polsa sobre l'ítem "Inscripcions", veurà les dades necessaris per formular la inscripció.
Pre-condicions: Haver seleccionat una cursa, que estigui obert el període d'inscripcions i que no s'hagin exhaurit.
Flux Normal: 1. L'usuari selecciona el ítem "Inscripcions".
Flux Alternatiu: 1a. L'usuari tanca l'App 1b. L'usuari veu la data d'apertura d'inscripcions. 1c. L'usuari veu que les inscripcions ja s'han exhaurit.
Postcondicions: L'usuari accedeix a la pantalla "Inscripcions"

UC6. Inscriure's a una cursa

Nombre: UC6- Inscriure's a una cursa seleccionada.
Actors: Usuari-Sistema
Descripció: L'usuari omplirà les dades necessàries per formular la inscripció, i

polsarà sobre el botó “Apunta’m”.
Pre-condicions: Haver seleccionat una cursa.
Flux Normal: 1. L’usuari ompirà les dades demanades. 2. L’usuari polsarà el botó “Apunta’m”.
Flux Alternatiu: 1a. L’usuari no polsa el botó “Apunta’m” 2a. L’usuari no omple totes les dades i quan polsa el botó “Apunta’m” rep un error 3a. L’usuari tanca l’App
Postcondicions: L’usuari es registra a la cursa i rep un missatge de confirmació.

UC7. Consulta apartat “Banc d’aliments”

Nombre: UC7- Navega a l’apartat banc d’aliments.
Actors: Usuari-Sistema
Descripció: L’usuari selecciona una cursa i polsa sobre l’ítem “Banc d’aliments”, on veurà quins són els productes més demandats.
Pre-condicions: Haver seleccionat una cursa.
Flux Normal: 1. L’usuari selecciona el ítem “Banc d’aliments”.
Flux Alternatiu: 1a. L’usuari tanca l’App
Postcondicions: L’usuari accedeix a la pantalla “Banc d’aliments”

UC8. Consulta apartat “El nostre poble”

Nombre: UC8- Navega a l’apartat sobre el poble que acull la cursa
Actors: Usuari-Sistema
Descripció: L’usuari obre el menú i polsa sobre l’ítem “El nostre poble”, on tindrà informació sobre el poble que acull la cursa.
Pre-condicions: Haver seleccionat una cursa.
Flux Normal: 1. L’usuari selecciona el ítem “El nostre poble”.
Flux Alternatiu: 1a. L’usuari tanca l’App
Postcondicions: L’usuari accedeix a la pantalla “El nostre poble”

UC9. Consulta apartat “Col·laboradors”

Nombre: UC9- Navega a l'apartat sobre col·laboradors de la cursa
Actors: Usuari-Sistema
Descripció: L'usuari obre el menú i polsa sobre l'ítem "Col·laboradors", on tindrà informació sobre els negocis/empreses que ajuden a fer possible la cursa.
Pre-condicions: Haver seleccionat una cursa.
Flux Normal: 1. L'usuari selecciona el ítem "Col·laboradors".
Flux Alternatiu: 1a. L'usuari polsa sobre l'icona d'un col·laborador, surt de l'app i anirà a la web del col·laborar seleccionat. 2a. L'usuari tanca l'App
Postcondicions: L'usuari accedeix a la pantalla "Col·laboradors"

UC10. Afegir cursa

Nombre: UC11- Afegir una cursa al sistema
Actors: Administrador-Sistema
Descripció: L'administrador omplirà la informació demandada per crear la cursa.
Pre-condicions: Haver introduït les dades d'accés d'administrador.
Flux Normal: 1. L'administrador omple les dades demandades. 2. L'administrador polsa el botó "Guardar" i la cursa s'insereix al sistema.
Flux Alternatiu: 1a. L'administrador dona al botó "Cancelar" 2a. L'administrador no omple correctament les dades i rep un error. 3a. L'administrador tanca l'App.
Postcondicions: S'afegeix la cursa al sistema

UC11. Editar cursa

Nombre: UC12- Editar una cursa seleccionada
Actors: Administrador-Sistema
Descripció: L'administrador seleccionarà la opció Editar cursa, es mostrarà el llistat, seleccionarà una i podrà editar el contingut dels apartats.
Pre-condicions: Haver introduït les dades d'accés d'administrador i haver polsat el botó "Editar cursa".
Flux Normal: 1.L'administrador selecciona una cursa. 2. L'administrador edita els camps dels apartats de la cursa.

2. L’administrador polsa el botó “Guardar” i la cursa s’actualitza al sistema.
Flux Alternatiu: 1a. L’administrador dona al botó “Cancelar” 2a. L’administrador no selecciona cap cursa. 3a. L’administrador tanca l’App.
Postcondicions: S’afegeix la cursa al sistema

UC12. Eliminar cursa

Nombre: UC12- Eliminar una cursa seleccionada
Actors: Administrador-Sistema
Descripció: L’administrador seleccionarà la opció Eliminar cursa, es mostrarà el llistat, seleccionarà una i podrà eliminar la cursa.
Pre-condicions: Haver introduït les dades d’accés d’administrador i haver polsat el botó “Eliminar cursa”.
Flux Normal: 1.L’administrador selecciona una cursa. 2. Confirma que vol eliminar la cursa. 3. La cursa es borra del sistema i es navega al llistat.
Flux Alternatiu: 1a. L’administrador no selecciona cap cursa 2a. L’administrador tanca l’App.
Postcondicions: S’elimina la cursa del sistema

UC13. Login al sistema

Nombre: UC14- Accedir al sistema en mode administrador
Actors: Administrador-Sistema
Descripció: L’administrador pulsa el botó Admin, i introdueix les dades d’accés al sistema per entrar com a administrador.
Pre-condicions: Haver polsat el botó “Admin” de la pantalla inicial.
Flux Normal: 1. L’administrador introdueix username i password 2. L’administrador pulsa “Accedir”.
Flux Alternatiu: 1a. L’administrador no introdueix les dades correctes. 2a. L’administrador tanca l’App.
Postcondicions: S’accedeix al menú per afegir, editar o eliminar cursa.

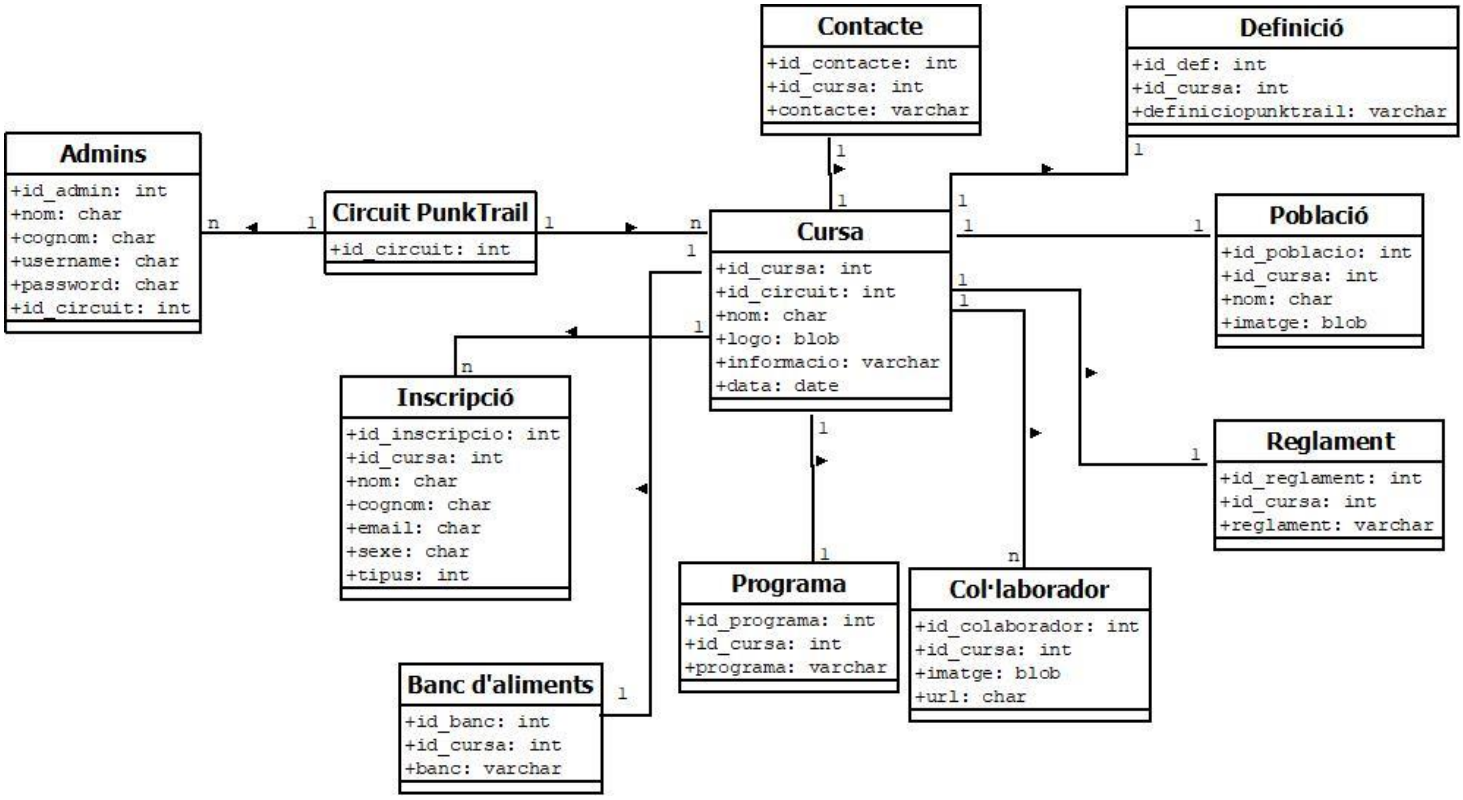
UC14. Extreure inscripcions

Nombre: UC14- Accedir al sistema en mode administrador
Actors: Administrador-Sistema
Descripció: L'administrador pulsa el botó Admin, i introdueix les dades d'accés al sistema per entrar com a administrador.
Pre-condicions: Haver seleccionat una carrera, i haver navegat a l'apartat Inscripcions.
Flux Normal: 1. L'administrador polsa el botó "Extreure llistat inscripcions" 2. Es guardarà un arxiu csv amb el llistat
Flux Alternatiu: 1a. L'administrador tanca l'App.
Postcondicions: Es generarà un arxiu .csv amb el llistat.

UC15. Obrir/tancar període d'inscripcions

Nombre: UC14- Accedir al sistema en mode administrador
Actors: Administrador-Sistema
Descripció: L'administrador pulsa el botó Admin, i introdueix les dades d'accés al sistema per entrar com a administrador.
Pre-condicions: Haver seleccionat una carrera, i haver navegat a l'apartat Inscripcions.
Flux Normal: 1. L'administrador polsa el botó "Obrir/tancar inscripcions" 2. Seleccionarà una cursa
Flux Alternatiu: 1a. L'administrador tanca l'App.
Postcondicions: Els corredors veuran el formulari d'inscripcions.

5.3 Diagrama UML de la base de dades



Il·lustració 23. Diagrama base de dades

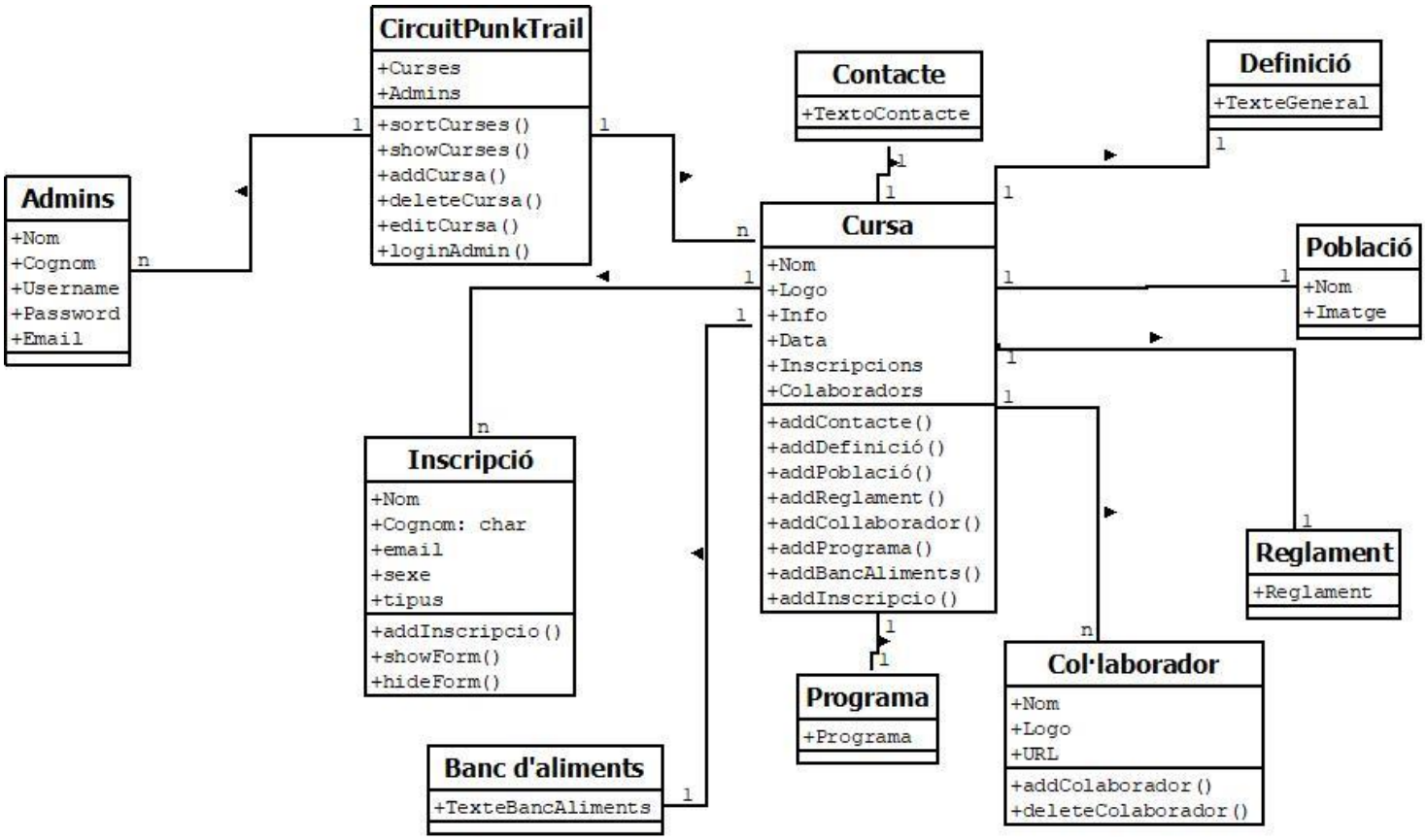
El sistema “Circuit PunkTrail” té una sèrie d’administradors encarregats de crear, editar les curses.
El circuit té n curses.
Cada cursa té 1:

- Contacte. Informació de com contactar-hi
- Programa. És la pàgina principal quan es selecciona la cursa.
- Població. En un circuit anual, només hi ha una cursa per població, per tal de cuidar el medi ambient, i repartir les curses per els altres pobles.
- Reglament. Cada carrera pot tenir les seves particularitats.
- Banc d’aliments.

Cada cursa té N:

- Col·laboradors. Empreses que hi ajuden a fer possible la cursa.
- Inscripcions. Llistat de corredors apuntats.

5.3 Diagrama UML de classes



Il·lustració 24. Diagrama de classes

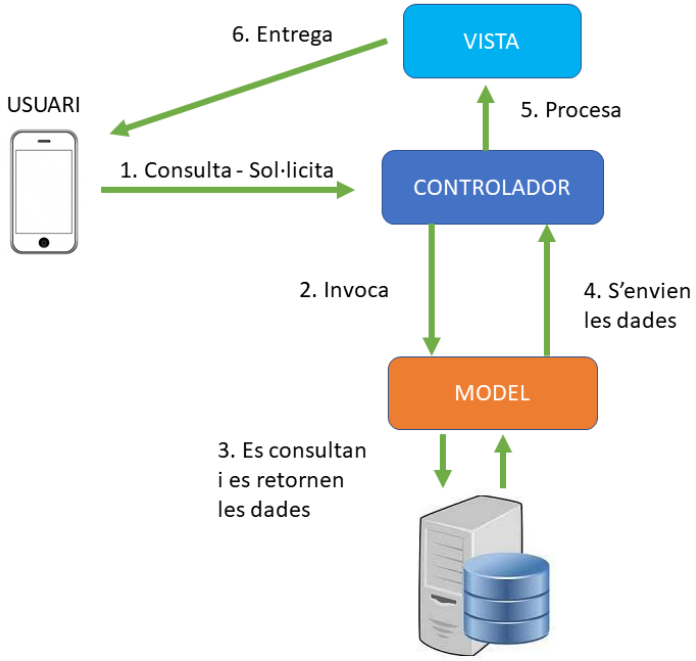
La classe principal es CircuitPunkTrail, que mostrarà les curses ordenades per data. També controlarà l'accès d'administradors al sistema, per afegir, editar i eliminar curses. Tindrà les llistes de Curses i Admins.

La classe Cursa afegirà tots els camps necessaris, i tindrà les llistes de Inscripcions i Col·laboradors.

La classe Inscripció afegirà les inscripcions a la base de dades, i mostrarà/ocultarà el formulari.

La classe Col·laborador tindrà les operacions per afegir col·laborador i per poder esborrar-lo.

5.4 Diagrama MVC



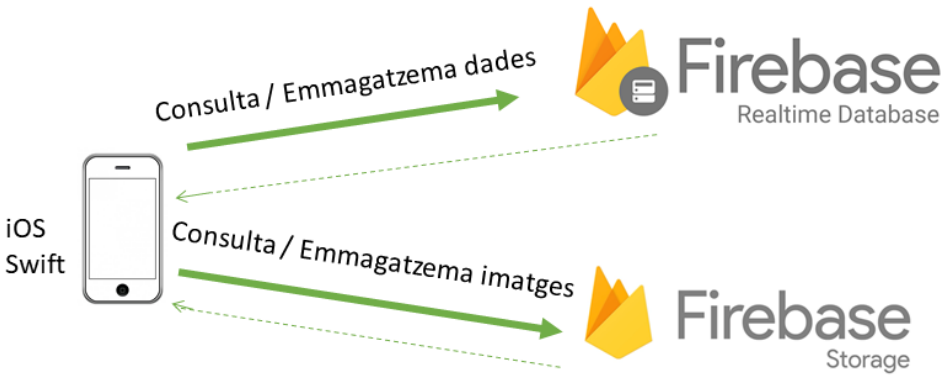
Il·lustració 25. Diagrama arquitectura MVC

L'usuari consulta o sol·licita un servei al controlador, qui genera la consulta al model per obtenir les dades de la base de dades. El controlador rep les dades, crea la vista i la mostra a l'usuari.

L'usuari per exemple, rebrà per pantalla un formulari d'inscripcions, l'omplirà i enviarà, i el controlador guardarà a la base de dades aquesta informació, i confirmarà a l'usuari que la inscripció ha estat completada amb èxit mitjançant una vista.

6. Implementació

6.1 Sistema



Il·lustració 26. Sistema

L'aplicació està programada en Swift 4 connectada al sistema Firebase. Firebase Database és la base de dades d'on s'extraurà i emmagatzemarà tota la informació relacionada amb les curses i dades del corredor en les inscripcions.

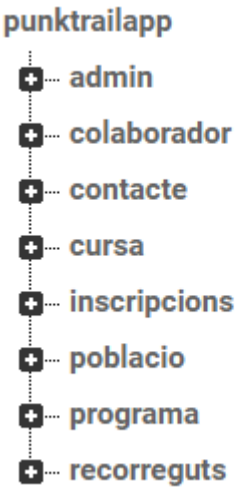
Firebase Storage permetrà emmagatzemar les imatges dels logos de la cursa, població, col·laboradors i programa.

6.2 Estructura Firebase Database

La estructura en Firebase Database alhora de construir la base de dades ha de basar-se en el format JSON. La manera de relacionar els diferents camps i atributs ha sigut mitjançant la creació d'un número que farà d'índex i relacionarà els camps Col·laborador, Contacte, Cursa, Inscripcions, Població, Programa i Recorreguts.

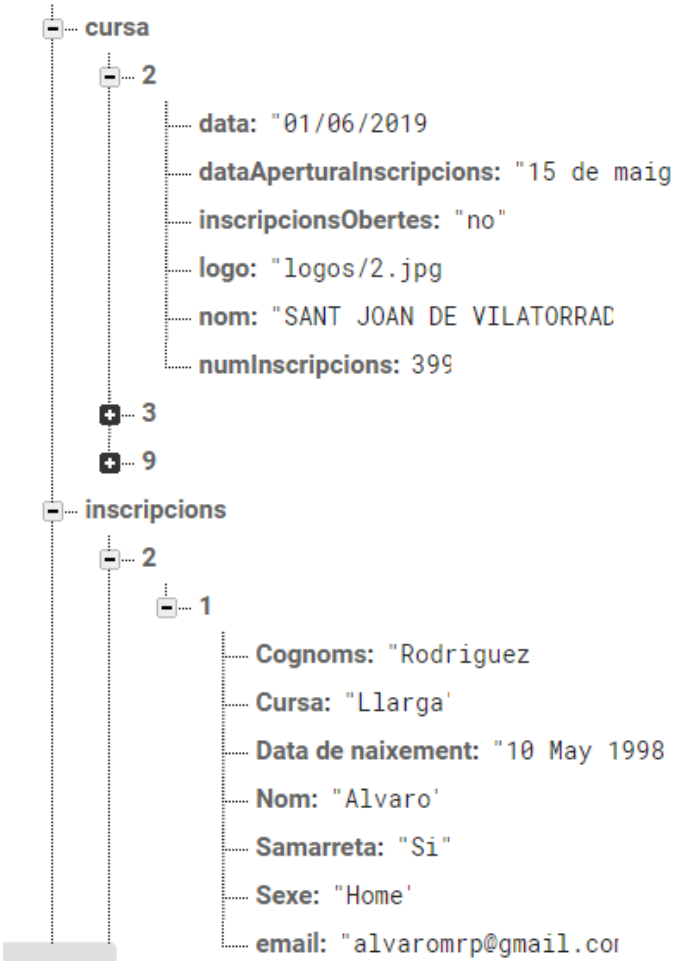
Aquest número serà essencial per identificar per consultar, afegir, editar informació a la cursa.

És clau una bona gestió de consultes a Firebase Database ja que les crides i obtenció de dades es fan de manera asíncrona.



Il·lustració 27. Estructura Firebase Database

Per exemple, tenim la cursa número 2 està relacionada amb inscripcions amb el índex número 2:



Il·lustració 28. Relació cursa-inscripcions JSON

A la taula **Cursa** hi trobem els camps:

Índex a nivell superior: 2

- “data" : "01/06/2019", *indica quan es realitzarà la cursa*
-"dataAperturaInscripcions" : "15 de maig", *La data en la que s’obriran les inscripcions*
-inscripcionsObertes" : "no", *Camp per controlar si estan obertes les inscripcions*
-"logo" : "logos/2.jpg", *Camp on està la direcció on s’emmagatzema el logo dins de Storage*:
- nom": "SANT JOAN DE VILATORRADA", *Camp per al nom de la cursa*
- "numInscripcions" : 399, *Camp al el número d’inscripcions disponibles*

La taula **Inscripcions**, hi trobem per cada cursa, un número d’inscripcions amb les dades del corredor.

Índex a nivell superior: 2 (per relacionar-la amb la cursa)
Índex d’inscripció: 1

"Nom" : "Alvaro",
"Cognoms" : "Rodriguez",
"Cursa" : "Llarga", *indicador si es cursa llarga o curta*
"Data de naixement" : "10 May 1998",
"Samarreta" : "Si", *indicador si el corredor comprarà samarreta*
"Sexe" : "Home",
"email" : alvaromrp@gmail.com

La taula **Recorreguts**, amb índex a nivell superior que indica la cursa, on tenim els links dels recorreguts i els textos de cada tipus de cursa.

```
"recorreguts" : {
   "2" : {
     "texteCurta" : "PunkTrail curta (10km 400m)",
     "texteLlarga" : "PunkTrail llarga (22km 900m)",
     "urlCurta" : "https://ca.wikiloc.com/rutes-cursa-de-muntanya/punktrail-
sant-vicenc-de-castellet-2019-curta-
33136794?fbclid=IwAR3WeabRsAgQ9bgz9Kmt6BlZtVle1QUyg6ErzYW
HSHAwXfwu7RaBjfMjzBs",
    "urlLlarga":  "https://ca.wikiloc.com/rutes-cursa-de-muntanya/punktrail-
sant-vicenc-de-castellet-2019-curta-
33136794?fbclid=IwAR3WeabRsAgQ9bgz9Kmt6BlZtVle1QUyg6ErzYW
HSHAwXfwu7RaBjfMjzBs"
   }
```

La taula **programa**, on estarà la direcció de la imatge principal i el texte de la cursa.


```
"programa" : {
   "2" : {
     "foto" : "programa/2.jpg",
     "programa" : "Sortida a les 9h, 2 avituallaments.\nA l'arribada botifarrada i concert!"
   },
```


Taula **contacte**, amb la url de facebook de la cursa:


```
"contacte" : {
   "2" : {
     "facebook" : "https://www.facebook.com/punktrail.santjoandevilatorrada.7"
   },
```


Taula **col·laborador** amb la direcció de la imatge:


```
"colaborador" : {
   "2" : {
     "foto1" : "colaborador/2.jpg"
   }
```


Taula **població**, un camp per guardar la direcció de la imatge, i un camp amb el texte.

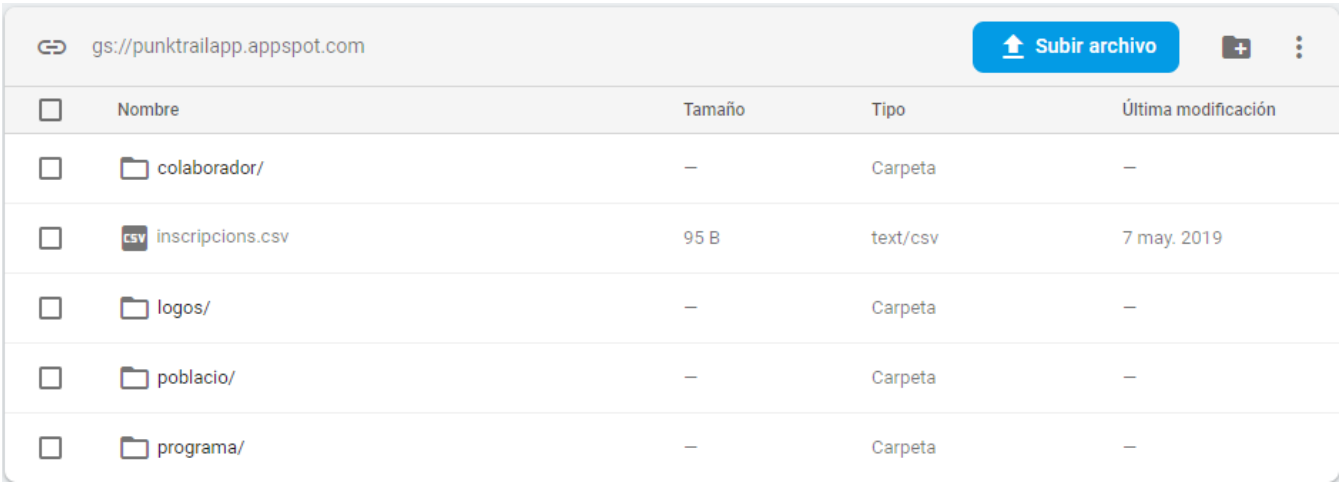

```
"poblacio" : {
   "2" : {
     "foto" : "poblacio/2.jpg",
     "texte" : "Sant Joan de Vilatorrada és una vila, cap del municipi del mateix nom, de la comarca del Bages. El municipi inclou l'entitat municipal descentralitzada de Sant Martí de Torroella. El gener de 2018 la seva població es va situar en els 10.820 habitants i era el segon municipi més poblat de la comarca.\n\nEl nucli originari s'anomenava Sant Martí de Torroella. El 1937 va canviar oficialment a Vilatorrada de Cardener i després de la guerra es va recuperar el nom de Sant Martí de Torroella. El 1962 va canviar a Sant Joan de Torroella, modificat el 1981 per l'actual de Sant Joan de Vilatorrada."
   },
```


Per últim la taula **admin** per emmagatzemar les dades d'usuari i contrasenya dels administradors:


```
  "admin" : {
    "password" : "hola",
```

```
  "username" : "alvaro"
}, {
  "password" : "1234",
  "username" : "admin"
```


6.3 Estructura Firebase Storage



Il·lustració 29. Estructura Firebase Storage

L'arxiu "inscripcions.csv" tindrà les dades de les inscripcions de la cursa de la que s'hagi generat el llistat.
Es guardarà a l'arrel ja que no es necessari guardar un llistat per a cada cursa al ser una necessitat temporal del administrador per al dia de la cursa.

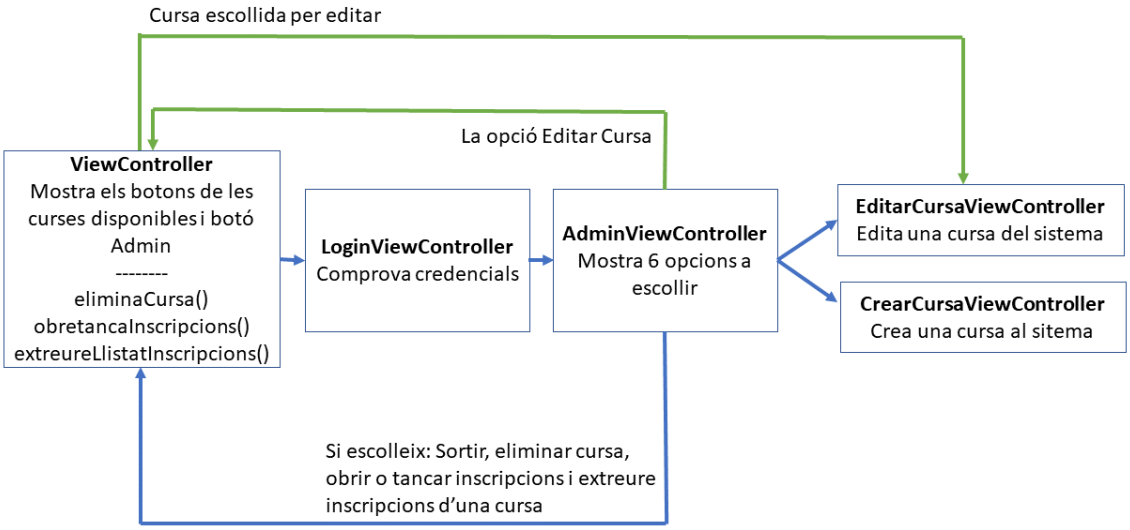
En aquest link estarà disponible l'arxiu per descarregar.

Les imatges de l'apartat col·laborador, logos, població i programa es guardaran a cada carpeta corresponent, sent identificades amb el número de la cursa més ".jpg".

Per exemple les imatges de la cursa número 2 es guardaran de la següent forma:

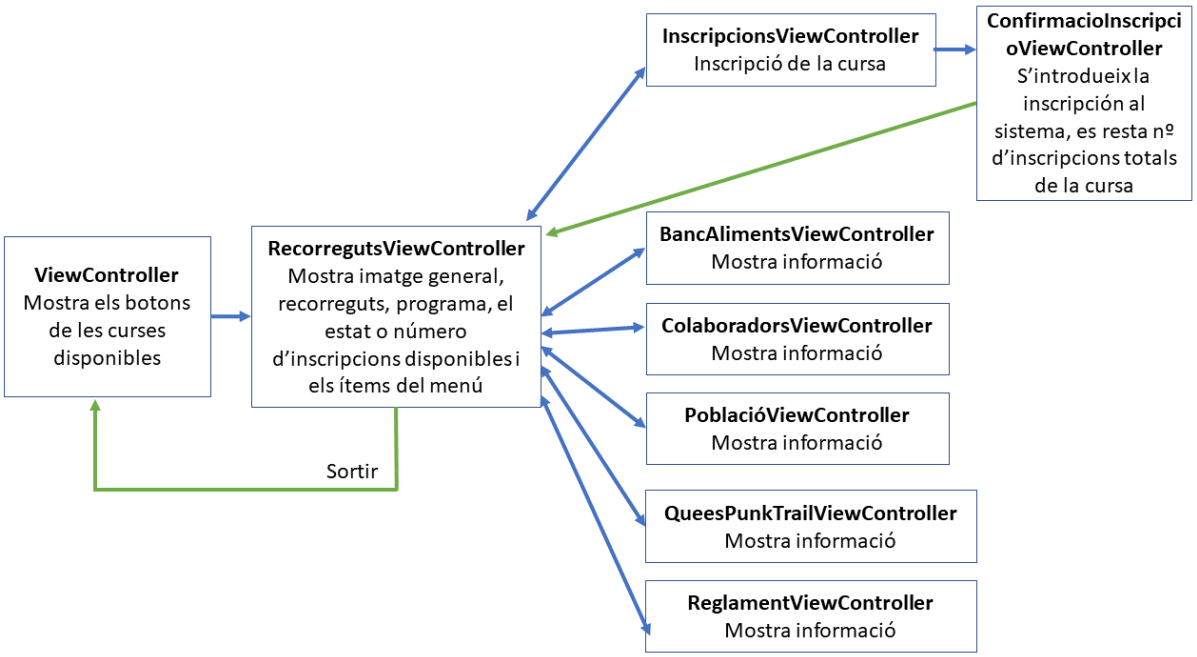
-colaborador/2.jpg
-logos/2.jpg
-poblacio/2.jpg
-programa/2.jpg

6.4 Codi: **estructura del codi part Administrador**



Il·lustració 30. Estructura Codi Administrador

6.5 Codi: **estructura del codi part Usuari**



Il·lustració 31. Estructura Codi Usuari

6.6 Codi: **creació d'una carrera al sistema**
(CrearCursaViewController.swift)

Les passes a seguir per a la creació d'una nova carrera són:

1. Comprovar les dades introduïdes per pantalla

```
//funció que comprova que cap camp estigui buit
func checkDades() -> Bool{

if((self.dataCursaField.text?.isEmpty)!  ||  (self.texteLlarga.text?.isEmpty)!  ||
(self.textePoblacio.text?.isEmpty)!  ||  (self.texteCurta.text?.isEmpty)!  ||
(self.programa.text?.isEmpty)!  ||  (self.urlLlarga.text?.isEmpty)!  ||
(self.urlCurta.text?.isEmpty)!  ||  (self.dataInscripcions.text?.isEmpty)!  ||
(self.nomField.text?.isEmpty)!  ||  (self.imatgePoblacio.image  ==  nil)  ||
(self.fotoPrograma.image == nil) || (self.logo.image == nil) || (self.fotoColaborador.image
== nil)){
        return false
}

 return true

   }
```

Tots els elements ha introduir alhora de crear una cursa són obligatoris, aquesta funció comprova que no estiguin buits, tant els camps de text com les 4 imatges a afegir.

2. Creació/obtenció d'un número índex per a la carrera.

Dins de la filosofia del funcionament de l'aplicació és molt important tenir un número de cursa, i saber que pot ser no consecutiu, ja que poden haver-hi 3 curses creades al sistema però si es borra la cursa número 2, la nova cursa creada heretaria aquest número.

Per això la funció getNumeroCursa retornarà aquest número, recorrent les curses i comprovant si existeix número vacant o si s'ha d'incrementar una unitat a les curses que ja hi han:

```
//funció per aconseguir el nou número de cursa per la nova cursa
   func getNumeroCursa(num: Int, completion:@escaping ((_ numCursa:Int)->Void)){

      var ref: DatabaseReference!
      ref = Database.database().reference()

      //es busca un numero vacant
      for index in 1...num {
         ref.child("cursa").child(String(index)).observeSingleEvent(of:  .value,  with:  {
(snapshot) in
            if(!snapshot.exists()){
               print("S'afegeix la cursa numero ",index)
               completion(index)
```

```
            }


        }) { (error) in
            print(error.localizedDescription)
        }


    }
    //si no hi ha número vacant, es incrementa 1 el número total de curses
    //i aquest serà el nou número
    let numCursa = num + 1
    print("No hi ha cursa saltada, s'afegeix cursa nº ",numCursa)
    completion(numCursa)


}
```

Un cop s'obté el número de cursa s'assigna i es començarà a afegir el contingut amb aquest índex.

```
self.getNumeroCursa(num: self.num){numCursa in
            print("Número cursa obtingut ",numCursa)
            self.num = numCursa
        }
```

3. Afegir les dades a la base de dades

Amb el índex ja només resta crear tots els camps necessaris de la Cursa:

```
//introdueix a la taula Cursa les dades
ref = ref.child("cursa").child(numberString)
ref.child("data").setValue(dataCursa)
ref.child("dataAperturaInscripcions").setValue(self.dataInscripcions.text)
//per defecte les inscripcions estaran tancades
ref.child("inscripcionsObertes").setValue("no")
 //per defecte el número màxim d'inscripcions serà 400 persones
ref.child("numInscripcions").setValue(400)
ref.child("nom").setValue(self.nomField.text)
ref.child("logo").setValue(rutaLogo)
```

La construcció de les referències de les imatges és crearà de la següent forma ("nom de la carpeta/" + "número de la cursa" + ".jpg":

```
let numberString = String(self.num)
let rutaLogo = "logos/" + numberString + ".jpg"
//puja el logo a Storage
self.uploadMedia(number:numberString,ruta:rutaLogo,codi:0) { url in
            guard url != nil else { return }
        }
```

Anar afegint totes les dades als camps corresponents:

```
let rutaPrograma = "programa/" + numberString + ".jpg"
   //puja la foto Programa a Storage
   self.uploadMedia(number:numberString,ruta:rutaPrograma,codi:1) { url in
      guard url != nil else { return }
   }
   //Inserta a la base de dades el programa de la cursa
   ref = Database.database().reference()
   ref = ref.child("programa").child(numberString)
   ref.child("programa").setValue(self.programa.text)
   ref.child("foto").setValue(rutaPrograma)

   //Inserta a la base de dades els recorreguts
   ref = Database.database().reference()
   ref = ref.child("recorreguts").child(numberString)
   ref.child("urlCurta").setValue(self.urlCurta.text)
   ref.child("texteCurta").setValue(self.texteCurta.text)
   ref.child("urlLlarga").setValue(self.urlLlarga.text)
   ref.child("texteLlarga").setValue(self.texteLlarga.text)


   let rutaPoblacio = "poblacio/" + numberString + ".jpg"
   //puja la foto Població a Storage
   self.uploadMedia(number:numberString,ruta:rutaPoblacio,codi:2) { url in
      guard url != nil else { return }
   }
   //Inserta a la base de dades la població
   ref = Database.database().reference()
   ref = ref.child("poblacio").child(numberString)
   ref.child("texte").setValue(self.textePoblacio.text)
   ref.child("foto").setValue(rutaPoblacio)

   //Inserta a la base de dades el contacte
   ref = Database.database().reference()
   ref = ref.child("contacte").child(numberString)
   ref.child("facebook").setValue(self.contacte.text)


   let rutaColaborador = "colaborador/" + numberString + ".jpg"
   //puja la foto Col·laborador a Storage
   self.uploadMedia(number:numberString,ruta:rutaColaborador,codi:3) { url in
      guard url != nil else { return }
   }

   //Inserta a la base de dades el colaborador
   ref = Database.database().reference()
   ref = ref.child("colaborador").child(numberString)
   ref.child("foto1").setValue(rutaColaborador)
```

Un cop acabada la inserció de totes les dades es tornarà a la pantalla d'administrador on s'informarà de que la cursa ha estat creada.

6.7 Codi: **mostrar el llistat de curses**
(ViewController.swift)

És molt important conèixer que per mostrar les curses es tindrà que obtenir fent crides a la base de dades de Firebase Database, que funciona d'una forma asíncrona, per tant haurà d'assegurar-se de la obtenció de la dada abans de treballar amb aquesta.

Això fa que no s'hagi decidit crear els botons manualment per codi i no utilitzar un tableView o stackView.

El primer pas és la obtenció dels índex de cada cursa al sistema. Recordar que els índex de cada cursa són claus, i que poden ser no consecutius.

```
//obtenim les keys de les curses existents a la base de dades
      ref.child("cursa").observeSingleEvent(of: .value, with: { (snapshot) in

         for newcursa in snapshot.children {
            self.keys.append((newcursa as AnyObject).key)
         }
         print(self.keys)
      })
```

Posteriorment obtindrem el número de curses al sistema amb la intenció de crear tants botons com curses existeixin amb la funció creaElsBotonsPerCurses.

```
//consultarem el nº de curses al sistema
      ref.child("cursa").observeSingleEvent(of: .value, with: { (snapshot) in

         let number = snapshot.childrenCount
         self.num = Int(snapshot.childrenCount)

         //Un cop tenim el nº de curses, crearem el mateix nº de botons per cursa
         self.creaElsBotonsPerCurses(numCurses: self.num){botons in
            print("Botons creats ",botons)
         }
```

Si no hi ha carreres al sistema, s'informa al usuari:

```
//si no hi han carreres al sistema es mostra el missatge
         if number == 0 {

            let alertController = UIAlertController(title: "Epp!",message: "Encara no hi han
carreres al circuit!", preferredStyle: .alert)
            let defaultAction = UIAlertAction(title: "OK!", style: .default, handler: nil)
            alertController.addAction(defaultAction)
            self.present(alertController, animated: true, completion: nil)
```

En el cas de que sí existeixin curses, per cada cursa s'obtindran les dades, es crearà la cursa, s'afegirà a la col·lecció, s'obtindrà el logo, i per últim es crearà el botó.

```
//per cada cursa
            for index in 1...number{
                //es crida amb la seva "key", i el numero de cursa per obtindre les dades
                        self.dadesCursa(index: self.keys[Int(index)-1], number :
          Int(number)){cursa in
                    //obtenir la imatge del logo de la cursa i crear el botó
                    self.getImatgeLogo(logo: cursa.logo, index: self.curses.count,nom:
cursa.nom, dataCursa: cursa.data){button in
                        //afegir el botó a la view
                        self.afegeixBoto(button: button){_ in
                            print("boto afegit")
                        }
                    }

                }

            }
```

Al mètode dadesCursa s'envia com a paràmetre el índex guardat de les keys de les curses. Com es recorren les curses amb for index in 1...number i es comença en 1, i en la col·lecció de keys el primer valor és 0, s'envia index-1 per a seleccionar la key apropiada.

La funció dadesCursa, crearà la Cursa obtenint les dades de la base de dades.

```
//funció per obtenir les dades de la base de dades
   func dadesCursa (index: String, number: Int, completion:@escaping ((_
cursaOk:Cursa)->Void)){

        var ref: DatabaseReference!
        ref = Database.database().reference()


        let carrera = index
        //obtenció de les dades indicant en nº de cursa
        ref.child("cursa").child(carrera).observeSingleEvent(of: .value, with: { (snapshot) in
            // Get user value
            let value = snapshot.value as? NSDictionary
            let nom = value?["nom"] as? String ?? ""
            let data = value?["data"] as? String ?? ""
            let logo = value?["logo"] as? String ?? ""
            //creació de Cursa
            let cursa = Cursa(nom: String(nom), logo:logo, data:data)

            print("Cursa numero", index, " es ",nom)
            //afegim la cursa
            self.curses.append(cursa)

            //retornem la cursa
            completion(cursa)
```

Un cop tenim la cursa, necessitem obtenir la imatge del logo, a la funció getImatgeLogo(logo: cursa.logo, index: self.curses.count,nom: cursa.nom, dataCursa: cursa.data) s'envien els paràmetres necessaris per crear el botó.

En primer lloc s'obté la imatge de Storage amb la referència enviada per cursa.logo:

```
//s'obté la referència a storage
    let storage = Storage.storage()
    let storageRef = storage.reference(forURL: "gs://punktrailapp.appspot.com/")
    //es consulta a la direcció obtinguda per "logo"
    let reference1 = storageRef.child(logo)
    reference1.downloadURL(completion: { (url, error) in
      if error != nil {
        print("Failed to download url:", error!)
        return
      } else {

        let data = try? Data(contentsOf: url!)
        //s'obté la imatge
        self.image =  UIImage(data: data!)!

        //s'incrementa la posició en Y
        self.altura = self.altura + 70
```

Es crea el botó amb les mesures, s'afegeix la imatge i el títol tal i com està comentat al codi:

```
//es crea el botó, amb referència index-1, ja que les posicions obtingudes de la
        //base de dades comencen en 1, i la colecció comença en 0
        self.allButtons[index-1].frame = CGRect(x:5, y:self.altura,
width:Int(self.view.frame.size.width - 10), height:50)
        //s'afegeix la imatge
        self.allButtons[index-1].setImage(self.image, for: .normal)
        //color de fons i transparència
        self.allButtons[index-1].backgroundColor =
UIColor.white.withAlphaComponent(0.5)
        //format de la imatge
        self.allButtons[index-1].imageView?.contentMode = .scaleAspectFit
        //posició de la imatge al botó
        self.allButtons[index-1].imageEdgeInsets = UIEdgeInsets(top:0, left:0,
bottom:0, right:self.view.frame.size.width - 10 - 50) //adjust these to have fit right



        //afegim el títol al botó, nom (de la cursa), separem amb guió i la data de la
cursa
        let title = nom + " - " + dataCursa
        self.allButtons[index-1].setTitle(title, for: .normal)
        self.allButtons[index-1].setTitleColor(UIColor.black, for: .normal)
        //tamany del texte
        self.allButtons[index-1].titleLabel?.font = UIFont.boldSystemFont(ofSize: 18)
```

```
            self.allButtons[index-1].titleEdgeInsets = UIEdgeInsets(top:0, left:0, bottom:0,
right:0)
            //controlem que el texte s'adapti al tamany del botó
            self.allButtons[index-1].titleLabel?.adjustsFontSizeToFitWidth = true;
            //li afegim número de tag al botó per controlar el nº de cursa
            self.allButtons[index-1].tag=Int(self.keys[Int(index)-1])!
            //afegim acció per a quan es premi el botó
            self.allButtons[index-1].addTarget(self,   action:   #selector(self.buttonClicked),
for: .touchUpInside)
            self.allButtons[index-1].semanticContentAttribute = .forceLeftToRight

            //es completa la creació del botó
            completion(self.allButtons[index-1])
```

Un cop el botó s'ha creat, la funció afegeixBoto(button: button) afegirà aquest botó a la vista:

```
//funció per afegir el botó a la vista
   func afegeixBoto(button: UIButton, completion: @escaping ((Bool))->Void){

      self.content.addSubview(button)


      completion(true)
   }
```

6.8 Codi: **mostrar info de la cursa** (ViewController.swift i RecorregutViewController.swift)

Un cop l'usuari té el llistat de curses disponible, si polsa sobre una s'obrirà el RecorregutViewController amb la informació de la cursa seleccionada.

Desde ViewController s'enviarà per Userdefaults el valor de la cursa que s'envia a través del tag creat i assignat a cada botó per cada cursa.

```
//si no està en mode Admin, es carregarà la info de la cursa seleccionada
      } else {


            let defaults = UserDefaults.standard
            defaults.set(sender.tag, forKey: "Cursa")
            let recorregutVC = UIStoryboard(name: "Main", bundle:
nil).instantiateViewController(withIdentifier: "RecorregutsViewController") as!
RecorregutsViewController
            recorregutVC.cursa = sender.tag
            self.present(recorregutVC, animated: true, completion: nil)
```

Un cop carregat l'arxiu RecorregutViewController s'obté la cursa seleccionada:

```
self.cursa = UserDefaults.standard.integer(forKey: "Cursa")
```

I es comença per posar el títol del nom de la cursa a la barra de navegació

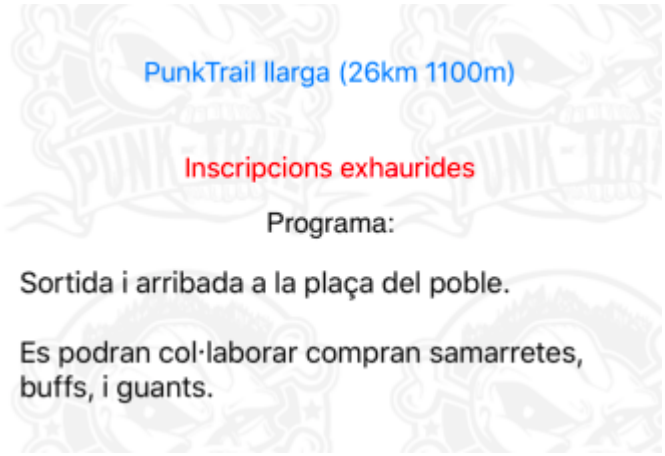
```
//Afegim el titol
      ref.child("cursa").child(carrera).observeSingleEvent(of: .value, with: { (snapshot) in
         // Get user value
         let value = snapshot.value as? NSDictionary
         self.titol.text = value?["nom"] as? String ?? ""
```

Es continua observant l'estat de les inscripcions, si s'han exhaurit es mostrarà el texte en vermell, si encara no estan obertes es mostrarà el texte en blanc, i si estan obertes i no s'han exhaurit botó verd amb selector de funció:

```
let inscripcions = value?["inscripcionsObertes"] as? String ?? ""
         let numInscripcions:Int = (value?["numInscripcions"] as? Int)!
         //s'obté el número d'inscripcions
         //si les inscripcions estan exhaurides es posarà al botó
         if (numInscripcions == 0){
            self.buttonInscripcions.setTitle("Inscripcions exhaurides", for: .normal)
            //color vermell del texte
            self.buttonInscripcions.setTitleColor(UIColor.red, for: .normal)
         }


else if (inscripcions == "no") {
            //si les inscripcions estan tancades, es posarà al botó la data d'apertura
            let texteButtonInscripcions = value?["dataAperturaInscripcions"] as? String ??
""
            let texteFinalButtonInscripcions = "Les inscripcions s'obriran el " +
texteButtonInscripcions
            self.buttonInscripcions.setTitle(texteFinalButtonInscripcions, for: .normal)
            //color blanc del texte
            self.buttonInscripcions.setTitleColor(UIColor.white, for: .normal)
         }

else{
            //si queden inscripcions i estan obertes, titol del texte
            self.buttonInscripcions.setTitle("Inscriu-te aquí! Resten " +
String(numInscripcions) + " inscripcions", for: .normal)
            self.buttonInscripcions.addTarget(self, action: #selector(self.buttonClicked),
for: .touchUpInside)
            //color del texte verd
            self.buttonInscripcions.setTitleColor(UIColor.green, for: .normal)
         }
```



Il·lustració 32. Exemple Inscripcions exhaurides

Il·lustració 33. Exemple Inscripcions obertes

Il·lustració 34. Exemple inscripcions encara no obertes

Es descarrega la imatge del programa que estarà a la capçalera en portada:

```
//Descarreguem la imatge del programa
      ref.child("programa").child(carrera).observeSingleEvent(of:      .value,      with:      {
(snapshot) in

          let value = snapshot.value as? NSDictionary
          let imatgeRecorregut =  value?["foto"] as? String ?? ""
          //amb el nom de la ruta es carregara la imatge al uiimageview
          self.carregaImatge(imatgeRecorregut: imatgeRecorregut)

          self.texteGeneral.text = value?["programa"] as? String ?? ""

          //self.getCurses(curses: self.curses, number: number)
          // ...
      }) { (error) in
          print(error.localizedDescription)
      }
```

La funció carrega Imatge serà l'encarregada de mostrar la imatge al UIImageView extraient la imatge de Firebase Storage amb la referència de la imatge:

```
//Carrega la imatge al uiimageview
   func carregaImatge(imatgeRecorregut:String){
      let storage = Storage.storage()
      let storageRef = storage.reference(forURL: "gs://punktrailapp.appspot.com/")
      let reference1 = storageRef.child(imatgeRecorregut)
      reference1.downloadURL(completion: { (url, error) in
          if error != nil {
              print("Failed to download url:", error!)
              return
          } else {
              //Do something with url
              //self.logo.sd_setImage(with: url, placeholderImage: placeholderimage)
              let data = try? Data(contentsOf: url!)
              self.imatge.image =  UIImage(data: data!)!
              self.imatge.contentMode = .scaleAspectFit
          }
      })

   }
```

Es creen els botons amb els textos dels recorreguts, i les URLs dels links a Wikiloc on estan els recorreguts. S'assigna un tag que serà utilitzat a la funció buttonClicked per diferenciar el botó polsat:

```
//es carreguen els textos dels recorreguts i els links
      ref.child("recorreguts").child(carrera).observeSingleEvent(of:      .value,      with:      {
(snapshot) in

          let value = snapshot.value as? NSDictionary
```

```
            let texteCurta = value?["texteCurta"] as? String ?? ""
            self.buttonCurta.setTitle(texteCurta, for: .normal)
            self.buttonCurta.addTarget(self, action: #selector(self.buttonClicked), for:
.touchUpInside)
            let urlCurtaString = value?["urlCurta"] as? String ?? ""
            self.urlCurta = URL(string: urlCurtaString)
            let texteLlarga = value?["texteLlarga"] as? String ?? ""
            self.buttonLlarga.setTitle(texteLlarga, for: .normal)
            self.buttonLlarga.addTarget(self, action: #selector(self.buttonClicked), for:
.touchUpInside)

            let urlLlargaString = value?["urlLlarga"] as? String ?? ""
            self.urlLlarga = URL(string: urlLlargaString)
```

Mateix procediment per el link de Facebook de Contacte:

```
        //s'obté el link del contacte
      ref.child("contacte").child(carrera).observeSingleEvent(of: .value, with: { (snapshot)
in

            let value = snapshot.value as? NSDictionary

            let urlContacteString = value?["facebook"] as? String ?? ""
            self.urlContacte = URL(string: urlContacteString)
            self.buttonContacte.addTarget(self, action: #selector(self.buttonClicked), for:
.touchUpInside)
```

Il·lustració 35. Links disponibles

Si es polsa sobre els botons: Població, Reglament, Banc d'aliments, Què és una PunkTrail, Col·laboradors o Tornar al circuit PunkTrail, es navegarà mitjançant segues.

Si es polsa sobre els botons Inscripcions, Visita'ns a Facebook, Cursa curta o Cursa llarga, es navegarà mitjançant el mètode buttonClicked

```
    //s'obren els links, per el tag diferenciem entre els links de cursa curta, llarga, o contace
    @objc func buttonClicked(sender: UIButton){

        var url: URL!
        if (sender.tag < 3){
            if (sender.tag == 0){

                url = self.urlCurta

            }else if (sender.tag == 1){
                url = self.urlLlarga
            }else if (sender.tag == 2){
                url = self.urlContacte
            }

            if UIApplication.shared.canOpenURL(url) {
                if #available(iOS 10.0, *) {
                    UIApplication.shared.open(url, options: [:], completionHandler: nil)
                } else {
                    UIApplication.shared.openURL(url)
                }
            }


        }else if (sender.tag == 5){
            //inscripcions
            let inscripcionsVC = UIStoryboard(name: "Main", bundle: nil).instantiateViewController(withIdentifier: "InscripcionsViewController") as! InscripcionsViewController

            self.present(inscripcionsVC, animated: true, completion: nil)


        }


    }
```

6.9 Codi: **Editar cursa** (AdminViewController.swift, ViewController.swift i EditarCursaViewController.swift)

Desde el AdminViewController es controla el botó polsat i s'envia a ViewController per a mostrar les curses i seleccionar una. El paràmetre enviat és UserDefaults.standard.set(true, forKey: "AdminEditar") per tal que en ViewController es reconegui que estem en mode administrador i volem editar la cursa seleccionada.

A ViewController qual es polsi el botó de la cursa, es detectarà el mode adminEditar i s'obrirà el EditarCursaViewController.

```
else if (UserDefaults.standard.bool(forKey: "AdminEditar")){
        UserDefaults.standard.set(false, forKey: "AdminEditar")
        //s'envia la cursa a editar
```

```
        let defaults = UserDefaults.standard
        defaults.set(sender.tag, forKey: "Cursa")
        //es carrega el viewcontroller per editar cursa
        let recorregutVC = UIStoryboard(name: "Main", bundle:
nil).instantiateViewController(withIdentifier: "EditarCursaViewController") as!
EditarCursaViewController

        self.present(recorregutVC, animated: true, completion: nil)
```

En EditarCursaViewController el funcionament és el mateix que en CrearCursaViewController amb la diferència que aquí els camps es mostren ja omplerts amb la informació actual.

6.10 Codi: **Eliminar cursa**
(AdminViewController i ViewController)

Mateix funcionament que Editar Cursa, desde la pantalla Admin s'activa el mode AdminEliminar:

```
//botó eliminar cursa
   @IBAction func eliminarCursa(_ sender: Any) {

      UserDefaults.standard.set(true, forKey: "AdminEliminar")
      let adminVC = UIStoryboard(name: "Main", bundle:
nil).instantiateViewController(withIdentifier: "ViewController") as! ViewController
      self.present(adminVC, animated: true, completion: nil)
   }
```

Un cop al ViewController qual es polsi la cursa, es controlarà i eliminarà del sistema:

```
if (UserDefaults.standard.bool(forKey: "AdminEliminar")){
        UserDefaults.standard.set(false, forKey: "AdminEliminar")
        var ref: DatabaseReference!
        ref = Database.database().reference()
```

Per esborrar la cursa, eliminarem totes les dades de la base de dades i Storage.

6.11 Codi: **Obrir/Tancar inscripcions**
AdminViewController i ViewController

Quan l'administrador vulgui canviar l'estat a obrir o tancar les inscripcions, i polsi el botó s'activarà a aquest mode.

```
//botó obrir/tancar inscripcions
```

```
    @IBAction func obrirTancarInscripcions(_ sender: Any) {
        UserDefaults.standard.set(true, forKey: "ObrirTancarInscripcions")
        let adminVC = UIStoryboard(name: "Main", bundle: nil).instantiateViewController(withIdentifier: "ViewController") as! ViewController
        self.present(adminVC, animated: true, completion: nil)
    }
```

Al ViewController un cop seleccionada la cursa, el primer que fem és posar l'estat a fals, un cop s'ha accedit al mode:

```
if (UserDefaults.standard.bool(forKey: "ObrirTancarInscripcions")){
            UserDefaults.standard.set(false, forKey: "ObrirTancarInscripcions")
```

Posteriorment mirarem l'estat de les inscripcions, si estan ja obertes o tancades, per procedir a l'acció contraria.

```
ref.child("cursa").child(String(sender.tag)).observeSingleEvent(of: .value, with: { (snapshot) in
            // Get user value
            let value = snapshot.value as? NSDictionary
            let estatInscripcions = value?["inscripcionsObertes"] as? String ?? ""
            var dialogMessage = UIAlertController(title: "Confirmació", message: "Confirmes?", preferredStyle: .alert)
            //si l'estat es que ja estan obertes, es consulta si es volen tancar
            if (estatInscripcions == "si"){
                dialogMessage = UIAlertController(title: "Confirmació", message: "Confirmes que vols tancar les inscripcions?", preferredStyle: .alert)
            //si estaven tancades, es consulta si es volen obrir
            }else{
                dialogMessage = UIAlertController(title: "Confirmació", message: "Confirmes que vols obrir les inscripcions?", preferredStyle: .alert)
            }
```

I si el administrador confirma, canviem el valor a la base de dades:

```
// Create OK button with action handler
            let ok = UIAlertAction(title: "Confirmar", style: .default, handler: { (action) -> Void in
                print("Ok button tapped")

                //es canvia la referència a la base de dades
                if(estatInscripcions == "si"){

ref.child("cursa").child(String(sender.tag)).child("inscripcionsObertes").setValue("no")
                }else{

ref.child("cursa").child(String(sender.tag)).child("inscripcionsObertes").setValue("si")
                }
```

6.12 Codi: **Extreure llistat d'inscrits**
(AdminViewController, ViewController)

Al AdminViewController s'activa aquest mode un cop polsat el botó:

```
//botó crear llistat d'inscripcions
   @IBAction func extreureInscripcions(_ sender: Any) {

      UserDefaults.standard.set(true, forKey: "ExtreureInscripcions")
      let adminVC = UIStoryboard(name: "Main", bundle:
nil).instantiateViewController(withIdentifier: "ViewController") as! ViewController
      self.present(adminVC, animated: true, completion: nil)
```

Al ViewController un cop seleccionada la cursa la que es vol extreure el llistat, el primer que es fa es comprovar si existeixen inscripcions. Si no existeixen s'informarà al administrador.

```
if (UserDefaults.standard.bool(forKey: "ExtreureInscripcions")){
         UserDefaults.standard.set(false, forKey: "ExtreureInscripcions")
         var ref: DatabaseReference!
         ref = Database.database().reference()
         //es consulta a la taula inscripcions, amb el numero de cursa
         ref.child("inscripcions").child(String(sender.tag)).observeSingleEvent(of:    .value,
with: { (snapshot) in
            // Get user value
            let numInscripcions = Int(snapshot.childrenCount)
            print("numero inscripcions ",numInscripcions)
if(numInscripcions == 0){
               UserDefaults.standard.set(true, forKey: "ExtreureInscripcions")
               let alertController = UIAlertController(title: "Epp!",message: "Encara no hi
han inscripcions en aquesta cursa!", preferredStyle: .alert)
               let defaultAction = UIAlertAction(title: "OK!", style: .default, handler: nil)
               alertController.addAction(defaultAction)
               self.present(alertController, animated: true, completion: nil)
```

Si hi existeixen inscripcions, s'aniran recorren i creant l'arxiu .csv. La funció getInscripció obtindrà la inscripció i la funció creaCSV crearà l'arxiu "inscripcions.csv".

```
//es recorre per cada nº de inscripcio
               for index in 1...numInscripcions{
                  //s'obtindrà la inscripció i s'afegirà al CSV
                  self.getInscripcio(index: index, cursa: String(sender.tag)) {inscripcions in
                     self.creaCsv(inscripcions: inscripcions) {_ in
                        print("Inscripcions creades")
```

La funció getInscripció recupera les dades fonamentals per passar llista el dia de al carrera: Nom, Cognoms, Tipus de carrera, i si comprarà samarreta.

```
//funció que retorna la inscripció solicitada per cursa, i nº d'inscripció
```

```
   func getInscripcio(index: Int, cursa:String, completion: @escaping ((_ inscripcions: [Inscripcio])->Void)){
          var ref: DatabaseReference!
          ref = Database.database().reference()
       ref.child("inscripcions").child(cursa).child(String(index)).observeSingleEvent(of: .value, with: { (snapshot) in

             let value = snapshot.value as? NSDictionary
             let nom = value?["Nom"] as? String ?? ""
             let cognoms = value?["Cognoms"] as? String ?? ""
             let cursa = value?["Cursa"] as? String ?? ""
             let samarreta = value?["Samarreta"] as? String ?? ""

             let inscripcio:Inscripcio = Inscripcio(nom:nom, cognoms:cognoms, cursa:cursa, samarreta:samarreta)

             print("afegeix cursa ",nom)
             self.inscripcions.append(inscripcio)

             completion(self.inscripcions)
```

Un cop es té la inscripció es completa a l'arxiu CSV amb la funció creaCSV

```
//funció que crea el llistat d'inscripcions a l'arxiu CSV
   func creaCsv(inscripcions: [Inscripcio], completion: @escaping ((Bool))->Void){

      let fileName = "Inscripcions.csv"
      let path = NSURL(fileURLWithPath: NSTemporaryDirectory()).appendingPathComponent(fileName)
      //capçalera de l'arxiu csv
      var csvText = "Nom,Cognoms,Cursa,Samarreta\n"

      //per cada inscripció es crea la línea
      for inscripcio in inscripcions {
         let newLine = "\(inscripcio.nom),\(inscripcio.cognoms),\(inscripcio.cursa),\(inscripcio.samarreta)\n"
         //csvText.appendContentsOf(newLine)
         csvText.append(contentsOf: newLine)
      }

      do {
         try csvText.write(to: path!, atomically: true, encoding: String.Encoding.utf8)
      } catch {
         print("Failed to create file")
         print("\(error)")
      }

      //es guarda l'arxiu a l'arrel de Storage
      let storageRef = Storage.storage().reference(forURL: "gs://punktrailapp.appspot.com/").child("inscripcions.csv")
      storageRef.putFile(from: path!)

      //let vc = UIActivityViewController(activityItems: [path!], applicationActivities: [])
      //self.present(vc, animated: true, completion: nil)

      //es carrega viewcontroller admin
```

```
        let adminVC = UIStoryboard(name: "Main", bundle: nil).instantiateViewController(withIdentifier: "AdminViewController") as! AdminViewController
        UserDefaults.standard.set(true, forKey: "LlistatInscripcionsCreades")

        self.present(adminVC, animated: true, completion: nil)


    }
```

L'arxiu queda emmagatzemat a l'arrel de Storage, disponible en aquest link per descarregar.

6.13 Codi: **Crear una inscripció**

(RecorregusViewController, InscripcionsViewController, ConfirmacióInscripcioViewController)

L'usuari polsa el botó Inscripcions a RecorregutsViewController, identifiquem el botó amb el tag 5:

```
else if (sender.tag == 5){
            //inscripcions
            let inscripcionsVC = UIStoryboard(name: "Main", bundle: nil).instantiateViewController(withIdentifier: "InscripcionsViewController") as! InscripcionsViewController

            self.present(inscripcionsVC, animated: true, completion: nil)
```

Utilitzarem botons del tipus DLLRadioButton per tal d'escollir entre opcions Sexe i Tipus de cursa. Un DataPicker per introduir la data de naixement i camps de texte.
Un cop carregat el InscripcionsViewController, l'usuari introduirà les dades demandades, i si polsa "Enviar" el primer que es farà es comprovar que totes les dades demanades hagin estat introduïdes amb la funció checkDades:

```
    //funció que comprova que els camps no estiguin buits
    func checkDades() -> Bool{

        if((self.FieldNom.text?.isEmpty)! || (self.FieldCognoms.text?.isEmpty)! || (self.FieldEmail.text?.isEmpty)! ){
            return false
        }

        return true

    }
```

Posteriorment es comprova que l'email tingui el format vàlid d'un email amb la funció:

```
//funció que comprova que el camp email tingui format email
```

```
func isValidEmail(testStr:String) -> Bool {
    let emailRegEx = "[A-Z0-9a-z._%+-]+@[A-Za-z0-9.-]+\\.[A-Za-z]{2,64}"

    let emailTest = NSPredicate(format:"SELF MATCHES %@", emailRegEx)
    return emailTest.evaluate(with: testStr)
}
```

Si no fossin correctes s'informa a l'usuari. En cas contrari s'introdueix la inscripció al sistema a la base de dades:

```
var ref: DatabaseReference!

        //S'obté la referència de la nova inscripció i s'afegeixen els valor
        ref = Database.database().reference().child("inscripcions").child(String(self.cursa)).child(String(self.numeroInscripcio))

        ref.child("Nom").setValue(self.FieldNom.text)
        ref.child("Cognoms").setValue(self.FieldCognoms.text)

        let dateFormatr = DateFormatter()
        dateFormatr.dateFormat = "dd MMMM yyyy"
        let strDate = dateFormatr.string(from: (self.dataNaixement.date))
        ref.child("Data de naixement").setValue(strDate)

        if (self.buttonHome.isSelected){
            ref.child("Sexe").setValue("Home")
        }else if (self.buttonDona.isSelected){
            ref.child("Sexe").setValue("Dona")
        }else if (self.buttonNoimporta.isSelected) {
            ref.child("Sexe").setValue("NC")
        }

        if (self.btncursaCurta.isSelected){
            ref.child("Cursa").setValue("Curta")
        }else if (self.btncursaLlarga.isSelected){
            ref.child("Cursa").setValue("Llarga")
        }

        if (self.btnSamarreta.isSelected){
            ref.child("Samarreta").setValue("Si")
        }else{
            ref.child("Samarreta").setValue("No")
        }

        ref.child("email").setValue(self.FieldEmail.text)
```

Quan s'ha introduït la inscripció al sistema es crida al viewcontroller de Confirmació:

```
        //es navega al viewcontroller de confirmació

            let recorregutVC = UIStoryboard(name: "Main", bundle: nil).instantiateViewController(withIdentifier: "ConfirmacioInscripcioViewController") as! ConfirmacioInscripcioViewController
```

```
            self.present(recorregutVC, animated: true, completion: nil)
```

Un cop aquí es confirma a l'usuari que s'ha rebut la inscripció y es resta una unitat a les inscripcions disponibles:

```
self.cursa = UserDefaults.standard.integer(forKey: "Cursa")
    var ref: DatabaseReference!
    ref = Database.database().reference()

    //s'obté el número d'inscripcions de la cursa
    ref.child("cursa").child(String(self.cursa)).observeSingleEvent(of: .value, with: {
(snapshot) in
        // Get user value
        let value = snapshot.value as? NSDictionary
        var inscripcions = value?["numInscripcions"] as? Int

        //es resta el número de inscripcions i s'actualitza el contador a la BBDD
        inscripcions = inscripcions! - 1


ref.child("cursa").child(String(self.cursa)).child("numInscripcions").setValue(inscripcions)



    }) { (error) in
        print(error.localizedDescription)
    }
```

6.14 Codi: **Login d'administrador** (LoginViewController)

Primer desde el viewDidLoad es guardaran els administradors de la base de dades a la col·lecció admins.

```
func getAdmins(){

    var ref: DatabaseReference!
    ref = Database.database().reference()


    ref.child("admin").observeSingleEvent(of: .value, with: { (snapshot) in
        // Get user value
        let numberAdmins = Int(snapshot.childrenCount)

        for index in 1...numberAdmins{

            ref.child("admin").child(String(index)).observeSingleEvent(of: .value, with: {
(snapshot) in
                let value = snapshot.value as? NSDictionary

                let usuariDB = value?["username"] as? String ?? ""
                let passwordDB = value?["password"] as? String ?? ""
                let admin = Administrador(username: usuariDB, password: passwordDB)
                self.admins.append(admin)
```

Posteriorment recorrerem comparant les dades introduïdes per cadascun dels administrador que existeixen al sistema:

```
//es recorren tots els administradors del sistema comparant els valors introduïts
      for admin in self.admins{

         if(admin.username == username && admin.password == pass){

            let adminVC = UIStoryboard(name: "Main", bundle: nil).instantiateViewController(withIdentifier: "AdminViewController") as! AdminViewController
            self.present(adminVC, animated: true, completion: nil)

         }
      }
      //si no s'han trobat les dades es carrega la funció incorrecte
      self.loginIncorrecte()
```

6.15 Codi: **Navegació entre pantalles**

El tipus de navegació emprat entre el canvi entre les diferents pantalles és diferent depenent de la pantalla, botó, o paràmetres a enviar.

Si es navega depenent d'alguna variable o validació es carrega el viewcontroller de destí i es carregarà la view al Storyboard:

```
Let adminVC = UIStoryboard(name: "Main", bundle: nil).instantiateViewController(withIdentifier: "AdminViewController") as! AdminViewController
self.present(adminVC, animated: true, completion: nil)
```

Si es un botó de navegació sense condicions es navegarà amb la funció Segue indicant-li la destinació.

7. Conclusions

7.1 Coneixements aplicats

En aquesta aplicació s'ha après:

- Treballar gestionant informació des de una base de dades
- Emmagatzemar o esborrar arxius del servidor
- Gestionar imatges (pujar, esborrar, actualitzar)
- Crear fitxers csv des de la aplicació
- Entendre la navegació entre les diferents pantalles
- El pas d'arguments entre pantalles
- Construcció, enviament i recepció de dades dels formularis

7.2 Assoliment d'objectius

Totes les funcions plantejades i requerides per cobrir les necessitats d'usuaris i administradors han sigut assolides, totes estan disponibles i funcionant correctament.

7.3 Planificació i metodologia

La planificació i metodologia s'han seguit segons el que estava previst, i seguint les prioritats de les funcionalitats de l'aplicació per tal d'assegurar i garantir l'èxit en el treball i l'aplicació, però penalitzant en el disseny del codi el que seria més correcte o les formes més standards de realitzar segons quines operacions.

7.4 Millores a introduir en el futur

- Revisió i reestructuració del codi, creació del llistat de les curses amb una TableView
- Poder afegir el número de col·laboradors que cada cursa necessiti
- Geolocalització de cada cursa per conèixer com arribar-hi
- Crear nous administradors al sistema
- Ordenar les curses per data

8. Glossari

- Punk-trail: tipus de curses no competitives, no hi ha classificacions, ni temps, ni premis. És un tipus de cursa solidari, on la inscripció és gratuïta però a canvi el corredor té l'obligació moral de dur alguna cosa al banc d'aliments.

- Circuit Punk-Trail: conjunt de curses del tipus Punk-Trail

- Trailrunning: cursa de muntanya

9. Bibliografia

Apps
App Store

Webs
https://punktrails.wordpress.com/ 14 de febrer 2019
https://punktrailsantvicencdecastellet.wordpress.com/ 14 de febrer 2019
https://www.facebook.com/punktraildesantvicencdecastellet 14 de febrer 2019
https://www.klassmark.com/trail 12 de març 2019
https://runedia.mundodeportivo.com 13 de març 2019
https://www.facebook.com/punkysolesa/ 29 de març 2019
https://www.facebook.com/punktrail.desalelles/ 29 de març 2019
https://www.facebook.com/PunkKastellgali 29 de març 2019
https://www.facebook.com/punktrail.santjoandevilatorrada.7 29 de març 2019
https://www.facebook.com/punk.devilomara 29 de març 2019
https://es.wikipedia.org/wiki/Modelo%E2%80%93vista%E2%80%93controlador 2 d'abril 2019
https://firebase.google.com 12 d'abril 2019
https://firebase.google.com/docs/ios/setup?authuser=0 12 d'abril 2019
https://github.com/DavydLiu/DLRadioButton 4 de maig 2019

10. Annexos

Manual d’usuari per organitzadors de curses.

Polsar el botó de la icona PunkTrail situat a la barra superior a la dreta, i fer el login amb les credencials d’administrador.

Seleccionar opció desitjada.

1.Crear cursa

Polsar el botó crear cursa, i omplir tots els camps demanats, tant de text com les imatges.
A tenir en compte:

- Nom de la cursa: s’haurà d’escriure en majúscules el nom de la població.
- Data de la cursa: format DD/MM/AAAA
- Afegir un logo: imatge clara
- Punktrail Curta / Punktrail Llarga: Editar el text entre parèntesis per afegir la distància i el desnivell.
- URL Punktrail curta / llarga: URL sencera
- Data apertura d’inscripcions: Escriure la data en format exemple: 2 de maig de 2019
- Escriure text del programa (hora de sortida, lloc específic de sortida, activitats a l’arribada...)
- Afegir imatge del programa: imatge general destacada de la cursa
- Decripció de la població: explicar breument les característiques més importants del poble.
- Afegir imatge de la població: una imatge clara de la vila
- Afegir URL Facebook: URL sencera
- Afegeix foto Col·laborador: una imatge del patrocinador principal

Polsar el botó Guardar.

2. Editar cursa

Polsar el botó editar cursa i canviar els texts i/o imatges per actualitzar.
Polsar el botó Guardar

3. Eliminar cursa

Polsar el botó Eliminar cursa, se'ns mostrarà el llistat de curses. Seleccionarem la que volem eliminar, confirmem i ja estarà eliminada.

4. Obrir / Tancar inscripcions.

Polsar el botó Obrir / Tancar inscripcions, se'ns mostrarà el llistat de curses. Seleccionarem la que volem canviar l'estat de les inscripcions, confirmem i ja estarà oberta o tancada depenent del estat anterior que se'ns informarà.

5. Extreure la llista d'inscripcions

Polsar el botó Extreure llistat d'inscripcions, se'ns mostrarà el llistat de curses. Seleccionarem de la que volem extreure el llistat i se'ns informarà de la creació de l'arxiu que per defecte es guardarà a aquest link.

6. Sortir

Polsar el botó Sortir, ens farà desconnectar-nos del mode administrador i tornar en mode usuari a la pantalla principal on es mostren les curses.