



FACULTADE DE INFORMÁTICA  
DEPARTAMENTO DE TECNOLOXÍAS DA INFORMACIÓN E DAS COMUNICACIÓNS

PROXECTO DE FIN DE CARRERA  
ENXEÑERÍA INFORMÁTICA

***Aplicación web e móvil para a xestión electrónica de actas deportivas.***

**Autor/a:** Pablo Castro Valiño  
**Director/a:** Santiago Saavedra López  
**Tutor/a:** Fernando Bellas Permuy

*A Coruña, a 14 de xaneiro de 2016.*



*DEDICATORIA*



## Agradecimentos

...



## **Resumen**

Pese aos avances nas Tecnoloxías da Información e das Comunicacións, e na Enxeñería do Software, a xestión de competicións deportivas continúa a atoparse extremadamente atrasada tecnolóxicamente e as federacións e asociacións deportivas invirten gran cantidade de tempo en centos de trámites que teñen que facer de forma manual, e entre os que se atopa a redacción, distribución, revisión e finalmente publicación das actas cos datos estadísticos dos encontros da súas competicións.

Con este proxecto introdúcese unha aplicación móvil para que os árbitros poidan cubrir ditas actas directamente no seu teléfono, permitindo manter actualizados os resultados e as estadísticas dos mesmos en tempo real e mesmo traballar de forma offline.

Actualmente dende a iniciativa empresarial VACmatch, estamos impulsando un sistema de xestión de competicións co fin de darrle aos xestores de federacións unha ferramenta na que realizar o seu traballo diario de forma electrónica. Este proxecto, VACmatch Mobile, intégrase nesta ferramenta.

Decidíuse empregar tecnoloxías web na implementación deste desenvolvemento co fin de facilitar a súa utilización en calquera plataforma móvil ou web así como pola versatilidade que aportan.



**Palabras clave:**

- ✓ VACmatch
- ✓ Aplicación híbrida.
- ✓ Reactjs.
- ✓ Javascript.
- ✓ PouchDB.
- ✓ CouchDB.
- ✓ Deporte.
- ✓ Gestión de competiciones.



# Índice general

	Página
<b>1. Introducción</b>	<b>1</b>
1.1. O deporte amateur e o avance tecnolóxico . . . . .	1
1.2. A problemática . . . . .	2
1.3. VACmatch . . . . .	2
1.4. Resumo do proxecto . . . . .	3
1.4.1. VACmatch Mobile . . . . .	3
1.4.2. VACmatch Web . . . . .	4
1.4.3. Estrutura da memoria . . . . .	4
<b>2. Estado da arte</b>	<b>5</b>
2.1. A xestión de competicións e actas deportivas . . . . .	5
2.2. Competidores no mercado . . . . .	7
2.2.1. Follas de cálculo . . . . .	7
2.2.2. Ferramentas tradicionais . . . . .	8
2.2.2.1. Novanet . . . . .	8
2.2.2.2. Federatio . . . . .	9
2.2.3. Ferramentas na nube . . . . .	10
2.2.3.1. miLeyenda . . . . .	10
2.2.3.2. Esportics . . . . .	11
2.2.3.3. Sportngin . . . . .	11
2.2.4. Outras plataformas . . . . .	12
2.3. Aplicacións libres no mercado . . . . .	13
2.4. Solución aberta e adaptable . . . . .	13
<b>3. Metodoloxía</b>	<b>14</b>

3.1. Lean Startup . . . . .	15
3.2. eXtreme Programming . . . . .	15
3.3. Scrum . . . . .	15
3.4. Adaptación da metodoloxía . . . . .	16
3.4.1. Desenvolvemento orientado ao cliente . . . . .	16
3.4.2. Sprints con backlog adaptable . . . . .	16
3.4.3. Reunións semanais . . . . .	16
3.4.4. Reunións diarias . . . . .	17
3.4.5. Releases . . . . .	17
3.4.6. Simplicidade . . . . .	17
3.4.7. Tests . . . . .	17
3.4.8. Fluxo de contribución ao proxecto . . . . .	18
<b>4. Análise de requisitos globais</b>	<b>19</b>
4.1. Consultas a xestores de federacións . . . . .	19
4.2. Peticións obtidas . . . . .	20
4.3. Requisitos finais . . . . .	21
4.3.1. Usuarios . . . . .	21
4.3.2. Listar actas . . . . .	21
4.3.3. Visualizar actas . . . . .	21
4.3.4. Xeración de actas offline . . . . .	22
4.3.5. Modificación de actas . . . . .	22
<b>5. Planificación e seguimento</b>	<b>23</b>
5.1. Validación de negocio (Xullo 2015 – Novembro 2015) . . . . .	24
5.1.1. Prototipo visual . . . . .	24
5.1.1.1. Planificación e definición da iteración . . . . .	24
5.1.1.2. Revisión e feedback . . . . .	24
5.1.1.3. Tarefas e seguimento . . . . .	24
5.1.2. MVP funcional . . . . .	25
5.1.2.1. Planificación e definición da iteración . . . . .	25
5.1.2.2. Revisión e feedback . . . . .	26
5.1.2.3. Tarefas e seguimento . . . . .	26
5.1.2.4. I Torneo VACmatch . . . . .	27

---

5.2. Desenvolvemento de produto (Novembro 2015 – Xaneiro 2016) . . . . .	27
5.2.1. 1 <sup>a</sup> e 2 <sup>o</sup> iteración. Creación do proxecto e xestión de actas . . . . .	28
5.2.1.1. Planificación e definición da iteración . . . . .	28
5.2.1.2. Revisión e feedback . . . . .	28
5.2.1.3. Tarefas e seguimento . . . . .	29
5.2.1.4. Participación na I Lonxa de Financiamento Responsable . . . . .	30
5.2.2. 3 <sup>a</sup> iteración. Eventos . . . . .	30
5.2.2.1. Planificación e definición da iteración . . . . .	30
5.2.2.2. Revisión e feedback . . . . .	31
5.2.2.3. Tarefas e seguimento . . . . .	31
5.2.3. 4 <sup>a</sup> iteración. Xestión de usuarios e creación offline de actas . . . . .	32
5.2.3.1. Planificación e definición da iteración . . . . .	32
5.2.3.2. Revisión e feedback . . . . .	32
5.2.3.3. Tarefas e seguimento . . . . .	33
5.2.4. 5 <sup>a</sup> iteración. Sinaturas . . . . .	34
5.2.4.1. Planificación e definición da iteración . . . . .	34
5.2.4.2. Revisión e feedback . . . . .	34
5.2.4.3. Tarefas e seguimento . . . . .	34
5.3. De empresa a comunidade (Xaneiro 2016 – Maio 2016) . . . . .	34
5.3.1. 6 <sup>a</sup> e 7 <sup>a</sup> iteración. Optimización e melloras . . . . .	35
5.3.1.1. Planificación e definición da iteración . . . . .	35
5.3.1.2. Revisión e feedback . . . . .	35
5.3.1.3. Tarefas e seguimento . . . . .	36
5.3.2. 8 <sup>a</sup> iteración. Testing e integración continua . . . . .	36
5.3.2.1. Planificación e definición da iteración . . . . .	36
5.3.2.2. Revisión e feedback . . . . .	36
5.3.2.3. Tarefas e seguimento . . . . .	37
5.3.3. 9 <sup>a</sup> e 10 <sup>a</sup> iteración. Inxección de dependencias . . . . .	37
5.3.3.1. Planificación e definición da iteración . . . . .	37
5.3.3.2. Revisión e feedback . . . . .	38
5.3.3.3. Tarefas e seguimento . . . . .	38
5.3.4. Release 0.2.0: Usabilidade en menús . . . . .	38
5.3.4.1. Planificación temporal . . . . .	38

---

5.3.4.2. Definición da iteración . . . . .	39
5.3.4.3. Concurso Universitario de Software Libre . . . . .	39
5.3.4.4. Revisión e feedback . . . . .	39
5.3.4.5. Tarefas e seguimento . . . . .	39
5.3.5. Release 0.2.1: I18n e app híbrida . . . . .	40
5.3.5.1. Planificación temporal . . . . .	40
5.3.5.2. Definición da iteración . . . . .	40
5.3.5.3. Revisión e feedback . . . . .	40
5.3.5.4. Tarefas e seguimento . . . . .	40
5.3.6. Release 0.2.2: Imáxe corporativa e revisión de erros . . . . .	40
5.3.6.1. Planificación temporal . . . . .	41
5.3.6.2. Definición da iteración . . . . .	41
5.3.6.3. Revisión e feedback . . . . .	41
5.3.6.4. Tarefas e seguimento . . . . .	41
<b>6. Fundamentos tecnolóxicos</b>	<b>42</b>
6.1. Linguaxes e frameworks empregados . . . . .	42
6.2. Bases de datos . . . . .	43
6.3. Estándares de comunicación . . . . .	43
6.4. Repositorios de código . . . . .	44
6.5. Ferramentas de xestión . . . . .	44
6.6. Ferramentas documentais . . . . .	45
<b>7. Deseño e implementación</b>	<b>46</b>
7.1. ReactJS e Flux . . . . .	47
7.1.1. Introdución e elección da tecnoloxía . . . . .	47
7.1.2. Elementos básicos . . . . .	48
7.1.2.1. Compoñentes de React . . . . .	48
7.1.2.2. Arquitectura Flux . . . . .	48
7.1.2.3. Implementación de Flux. Reflux. . . . .	49
7.1.3. Estructura da aplicación . . . . .	50
7.2. Bases de datos e funcionamiento offline . . . . .	51
7.2.1. PouchDB . . . . .	52
7.2.2. CouchDB . . . . .	52

---

7.2.3. Sincronización e xestión de conflictos . . . . .	52
7.3. App híbrida con Apache Cordova . . . . .	52
7.4. Interface gráfica e usabilidade . . . . .	54
7.4.1. Elementos comúns . . . . .	54
7.4.1.1. Menú lateral esquierdo . . . . .	54
7.4.1.2. Enlaces do menu superior dereito . . . . .	54
7.4.1.3. Información e axustes . . . . .	55
7.4.1.4. Barra de notificacións . . . . .	56
7.4.1.5. Autenticación . . . . .	57
7.4.1.6. Lista de pestanas . . . . .	58
7.4.2. Iniciar sesión . . . . .	59
7.4.3. Listado de actas . . . . .	59
7.4.4. Acta . . . . .	60
7.4.4.1. Convocar xogadores . . . . .	61
7.4.4.2. Inicio e fin do partido . . . . .	62
7.4.5. Finalización do encontro . . . . .	63
7.4.6. Modificación de estilos . . . . .	64
7.5. Multideporte . . . . .	65
7.5.1. Deporte . . . . .	65
7.5.2. Roles de usuarios . . . . .	66
7.5.3. Eventos . . . . .	66
7.6. Deseño da DB . . . . .	68
7.7. I18n . . . . .	69
7.8. Inxección de dependencias . . . . .	69
7.9. Testing e Integración continua . . . . .	70
7.9.1. TDD e BDD . . . . .	71
7.9.2. Jest . . . . .	71
7.9.3. Travis CI . . . . .	74
<b>8. Conclusóns e traballo futuro</b>	<b>75</b>
8.1. Recoñecementos . . . . .	75
8.1.1. Finalista no Certamen de Proyectos Libres da UGR . . . . .	75
8.1.2. Premio Universitario de Software Libre . . . . .	76

---

8.2. Traballo futuro . . . . .	76
8.2.1. Melloras de desenvolvemento . . . . .	76
8.2.2. Creación de comunidade . . . . .	77
8.3. Conclusóns . . . . .	77
<b>A. Configurar a aplicación.</b>	<b>80</b>
A.1. Base de datos remota. . . . .	80
A.2. Aplicación. . . . .	80
<b>B. Executar a aplicación.</b>	<b>81</b>
B.1. Compilación e execución web. . . . .	81
B.2. Compilación para móvil. . . . .	81
<b>C. Glosario de acrónimos</b>	<b>82</b>
<b>D. Glosario de términos</b>	<b>83</b>
<b>Bibliografía</b>	<b>85</b>

# Índice de figuras

<b>Figura</b>	<b>Página</b>
2.1. Exemplo de acta deportiva . . . . .	6
2.2. Folla de cálculo . . . . .	7
2.3. Aplicación web de Novanet . . . . .	8
2.4. Web da FGVB co sistema Federatio . . . . .	9
2.5. APP móvil de MiLeyenda . . . . .	10
2.6. Torneo de tenis na web de Esportics . . . . .	11
2.7. APP móvil de Sportngin . . . . .	12
5.1. Web do I Torneo VACmatch . . . . .	27
5.2. Diagrama de Gant do sprint 1 . . . . .	29
5.3. Diagrama de Gant do sprint 2 . . . . .	30
5.4. I Lonxa de Financiamento Responsable . . . . .	30
5.5. Diagrama de Gant do sprint 3 . . . . .	32
5.6. Diagrama de Gant do sprint 4 . . . . .	33
5.7. Diagrama de Gant do sprint 5 . . . . .	35
5.8. Diagrama de Gant do sprint 7 . . . . .	36
5.9. Diagrama de Gant do sprint 8 . . . . .	37
5.10. Diagramas de Gant do sprints 9 e 10 . . . . .	39
5.11. Finalistas CUSL . . . . .	40
7.1. Esquema da arquitectura Flux . . . . .	48
7.2. Arquitectura de unha aplicación con Apache Cordova . . . . .	53
7.3. Estructura da aplicación gráficamente. . . . .	54
7.4. Menú lateral esquierdo. . . . .	55

7.5. Menú desplegable dereito.	56
7.6. Ventás de información e axustes.	56
7.7. Barra de notificacións.	57
7.8. Vista de inicio de sesión.	59
7.9. Listado de actas.	60
7.10. Vista principal de un acta.	61
7.11. Seleccionar xogadores presentes no encontro.	62
7.12. Fin de encontro.	63
7.13. Sinatura de un acta.	64
7.14. Patron estratexia para o deporte.	65
7.15. Diagrama parcial da vista dos eventos.	67
7.16. Diagrama de clases para a xestión de eventos.	68
7.17. Esquema base de datos de Person.	68

# Índice de cuadros

Tabla	Página
-------	--------



# Capítulo 1

## Introducción

### Índice general

---

<a href="#">1.1. O deporte amateur e o avance tecnolóxico</a> . . . . .	<a href="#">1</a>
<a href="#">1.2. A problemática</a> . . . . .	<a href="#">2</a>
<a href="#">1.3. VACmatch</a> . . . . .	<a href="#">2</a>
<a href="#">1.4. Resumo do proxecto</a> . . . . .	<a href="#">3</a>
<a href="#">1.4.1. VACmatch Mobile</a> . . . . .	<a href="#">3</a>
<a href="#">1.4.2. VACmatch Web</a> . . . . .	<a href="#">4</a>
<a href="#">1.4.3. Estrutura da memoria</a> . . . . .	<a href="#">4</a>

---

NESTE capítulo trataranse os aspectos básicos para comprender o proxecto así como os motivos que levaron ao seu desenvolvemento e a estrutura da presente memoria.

Falaremos do estado do deporte na actualidade e da xestión deportiva en concreto, co fin de mostrar a necesidade de impulsar un proxecto dentro de este campo para, posteriormente introducir a iniciativa dentro da que este proxecto xurdíu e rematar con un resumo da problemática que resolve.

### 1.1. O deporte amateur e o avance tecnolóxico

Actualmente o deporte é fundamental na vida das persoas, durante os últimos anos o número de españois que realizan algunha actividade física medrou enormemente así como o número de competicións amateur, que permiten a estos deportistas, competir por un custe moito más asequible que as federacións oficiais.

Se embargo, este crecemento do número de deportistas non veu acompañado tamén dunha renovación tecnolóxica das competicións polo que gran parte dos seus xestores seguen a invertir un tempo elevado nas súas competicións e non teñen apenas relación dixital coas persoas que compiten nas mesma.

## 1.2. A problemática

Actualmente os organizadores de competicións deben realizar unha serie de tarefas que se describen a continuación e que na súa meirande parte, realizan de forma manual ou axudados de follas de cálculo, ao non dispor das ferramentas tecnolóxicas axeitadas a un prezo accesible.

**Inscricións** Na maior parte das competicións, os xogadores seguen a ter que levar cuberta a súa ficha cos seus datos persoais en papel, fotocopia do DNI, fotografía, etc para que a federación garde esos datos nunha folla de cálculo.

**Aplazamentos de partidos** Moitas esixenllas aos equipos, unha vez postos de acordo, enviar unha confirmación en papel, por correo ordinario ou fax.

**Notificacións** Deben avisar aos sancionados, os cambios no calendario, etc por correo electrónico cando menos.

**Revisión de sancionados** A federación debe comprobar que un xogador sancionado non xogou un partido que non debía.

**Loxística das actas dos encontros** O árbitro do encontro debe recoller as actas na asociación e volver a traelas cubertas despois dos encontros.

**Publicación de resultados** A federación debe recopilar tódolos datos das actas para publicalos, ben sexa nunha web ou por email aos participantes.

**Publicación de clasificacións e estadísticas** A federación debe calcular a clasificación e recopilar as estadísticas para publicalas posteriormente.

O proxecto desenvolto trata de resolver os últimos apartados mencionados no punto anterior, *a xestión das actas dos encontros, a súa loxística e a automatización da publicación de resultados e clasificacións*.

Para iso decidíuse crear unha aplicación móvil que permita que os árbitros xestionen as súas actas directamente dende o seu teléfono móvil ou tableta, nunha aplicación multidispositivo baseada en tecnoloxías web, permitindo incluso realizar ditas actas sen conexión a internet, algo que hoxe en día ningunha aplicación ofrece no mercado nacional.

Esta aplicación móvil integrarase tamén con un sistema de xestión de competicións co fin de que os árbitros poidan cubrir as actas e publicar as estadísticas e resultados directamente na web da federación, a través do seu sistema de xestión.

## 1.3. VACmatch

VACmatch é unha iniciativa empresarial xurdida na Universidade da Coruña para mellorar a xestión de competicións deportivas a través dunha serie de aplicacións entre as que se atopa este proxecto.

---

A iniciativa recibiu o pasado ano a calificación de *Iniciativa Empresarial de Base Tecnolóxica (IEBT)* reconecendo o seu grao de innovación así como participou en diversos programas de apoio a ideas emprendedoras como *Yuzz*<sup>1</sup> ou *Telefónica Galicia OpenFuture*<sup>2</sup> e durante case un ano, conviviu con outras iniciativas empresariais no *Viveiro de empresas da Universidade da Coruña*.

Ano e medio despois do seu comezo decidíuse abandonar o proxecto como iniciativa empresarial pero VACmatch continúa como comunidade baseada nun proxecto de software libre.

## 1.4. Resumo do proxecto

O proxecto desenvolto compõe de dúas partes diferenciadas que permiten a xestión das actas dos encontros por parte das federacións deportivas.

### 1.4.1. VACmatch Mobile

É unha aplicación móvil híbrida realizada con tecnoloxías web co fin de poder utilizala en calquera plataforma, tanto a través da web como nun móvil Android, IOS, FirefoxOS... e que permitirá aos árbitros das competicións realizar todas as xestións coas actas dos encontros dende o seu teléfono.

**Lista de actas** Esta aplicación permite que os árbitros poidan dispoñer no seu teléfono das actas dos partidos que teñen que dirixir, coa localización e a data dos mesmos e que se actualizan de forma automática cando se reasigan ou se cancelan.

**Convocatoria de xogadores** Unha vez o árbitro chega ao encontro pode seleccionar na aplicación os xogadores que asistiron ao mesmo únicamente con un click, introducir a algúin novo se o deseja ou editar datos como o dorsal dun xogador.

**Xestión de actas** Unha vez começado o encontro, a aplicación permitirá introducir os diversos eventos que ocorren no mesmo como infraccións, goles ou tarxetas de forma que é moi sinxelo engadir novos deportes e eventos.

**Sinatura de actas** Para rematar o encontro, o árbitro poderá engadir comentarios a mesma acta e tanto él como os xogador ou persoal dos equipos, poderán asinar a acta con un código PIN do que dispón cada un.

**Actas offline** O árbitro poderá crear actas incluso aínda que non tivese sincronizados todos os datos do partido, permitindo cubrir as actas incluso no peor escenario posible.

---

<sup>1</sup>Programa de formación empresarial do Centro Internacional Santander Emprendimiento

<sup>2</sup>Programa de mentorización e formación de negocio de Telefónica con un premio de 2.000 €

### 1.4.2. VACmatch Web

A aplicación móvil explicada anteriormente atópase nas primeiras fases de un proceso de integración con unha aplicación web, a través da cal as federacións poden xestionar completamente as súas competicións, modificar o calendario, engadir novos xogadores ou equipos, xestionar arbitraxes, etc.

Actualmente permítese que a federación cree as actas dos encontros, os árbitros as sincronicen nos seus teléfonos e unha vez cubertas, todos os datos sexan publicados automáticamente na web da federación a través dun pequeno plugin.

Esto facilita que a federación poida dar o primeiro paso de sustituir as actas e fichas físicas por versións dixitais pero deben continuar facendo a integración dos datos das actas de xeito manual, xa que ambos sistemas funcionan por separado, e polo tanto é preciso comprobar as actas e mover os seus datos ao sistema de xestión de VACmatch Web onde se gardan os datos finais e verificados pola federación.

Nun futuro cercano permitirase manter as clasificacións actualizadas en todo momento sen apenas intervención humana, e aforrando un enorme traballo na revisión das actas.

### 1.4.3. Estrutura da memoria

Engadir  
estrutura

## Capítulo 2

# Estado da arte

### Índice general

---

<b>2.1. A xestión de competicións e actas deportivas</b>	5
<b>2.2. Competidores no mercado</b>	7
2.2.1. Follas de cálculo	7
2.2.2. Ferramentas tradicionais	8
2.2.3. Ferramentas na nube	10
2.2.4. Outras plataformas	12
<b>2.3. Aplicacións libres no mercado</b>	13
<b>2.4. Solución aberta e adaptable</b>	13

---

NESTE capítulo mostrárase cómo as federacións deportivas están xestionando actualmente as súas competicións, cales son as diversas alternativas no mercado para isto e tamén se fará unha análise do software libre neste campo.

Búscase mostrar a forte necesidade existente de impulsar unha alternativa libre e con boa usabilidade que actualmente está a reclamar o mercado para a xestión de actas electrónicas.

### 2.1. A xestión de competicións e actas deportivas

A xestión de competicións e eventos deportivos é unha tarefa que dende fai moito tempo se ven realizando de xeito manual e nos últimos anos comezan a aparecer no mercado as primeiras aplicacións para facilitar este traballo.

O proceso comeza cando os xogadores son inscritos a través do seu equipo, nunha competición que xestiona unha federación ou asociación deportiva. Cando os clubes realizan o pago da inscrición, reciben unha ficha identificativa<sup>1</sup> para cada xogador e persoal do mesmo e que deberán levar aos encontros para poder participar.

---

<sup>1</sup>É un documento con fotografía e similar ao DNI que permite identificar ao integrante do equipo

Unha vez tódolos equipos están inscritos, a federación procede a crear un calendario que contén unha listaxe de enfrentamentos a realizar entre os equipos.

Semanalmente os árbitros da federación son asignados aos partidos que durante esa semana se disputa e deben pasar polo local da mesma para recoller un modelo en papel da acta dos encontros similar ao que se mostra na Figura 2.1 e para coñecer os partidos que terán ao seu cargo, a data e a localización.

O árbitro débese desplazar ao campo ou pista onde se disputa o encontro, recolle as fichas dos xogadores de ambos equipos e copia dentro da acta aqueles que están presentes, descartando os que non asistiron.

Figura 2.1: Exemplo de acta deportiva.

Durante o partido o árbitro, ou o segundo árbitro segundo corresponda, encárgase de ir cubrindo tódolos eventos que se producen durante o xogo, por exemplo goles, faltas, etc.

Unha vez rematado, os capitáns ou adestradores de ambos equipos deben revisar a acta e asinala se están de acordo co que alí se indica, para que posteriormente o árbitro traslade unha copia da mesma ao local da federación.

Alí os empregados da federación rematan o proceso revisando as actas e introducindo os datos nunha ferramenta de xestión como as que se comentan no seguinte apartado, co fin de actualizar os resultados e a clasificación e para finalmente publicalos na páxina web da federación.

## 2.2. Competidores no mercado

Como se comentaba anteriormente, existen diversas ferramentas para xestionar competicións, unhas más avanzadas ca outras pero igualmente funcionais e que se van a introducir a continuación, resaltando certas vantaxes e carencias das mesmas.

### 2.2.1. Follas de cálculo

A folla de cálculo é o sistema utilizado por excelencia para xestionar competicións.

Un sistema rudimentario pero funcional, algunas federacións combinan en certa medida follas de cálculo e bases de datos sinxelas para crear as táboas das clasificacións e dos resultados, utilizando funcións que permiten automatizar algunhas tarefas como por exemplo o cálculo da clasificación dos equipos en función dos seus resultados ao longo da competición.

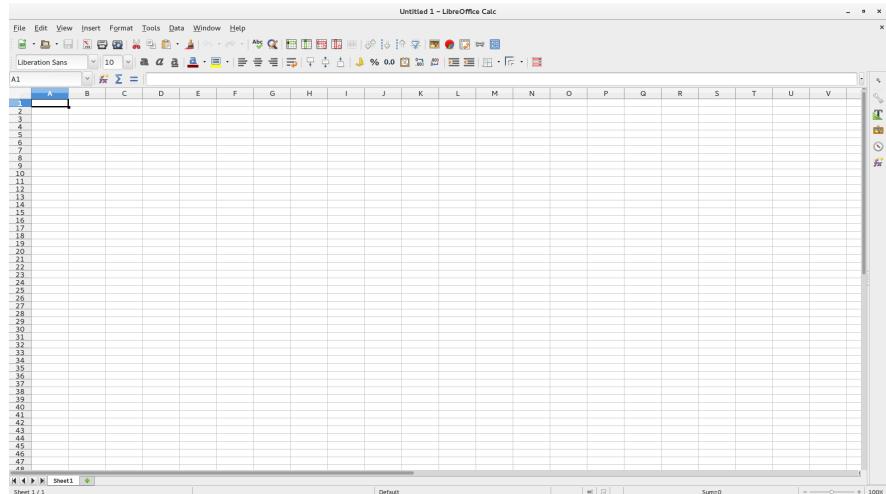


Figura 2.2: Folla de cálculo

O sistema é moi versatil para usuarios experimentados xa que permite adaptar as possibles necesidades variables da federación, na forma de calcular datos estadísticos, na forma en que se organizan as diferentes competicións, etc pero implica un gran traballo manual e pode chegar a ser un suplicio para usuarios con poucos coñecementos de ofimática.

As actas seguen a chegar en papel a federación e os datos deben ser introducidos nas diversas follas de cálculo (que en competicións de gran tamaño, vólvense inmanexables), revisando as sancións de xeito manual, polo que os errores na interpretación dos datos son habituais e por suposto os resultados non son mostrados en tempo real.

### 2.2.2. Ferramentas tradicionais

Tradicionalmente existiron varias ferramentas que comparten características de aplicacóns web clásicas xa que son soluciones especializadas que deben ser instaladas individualmente para cada federación e personalizadas para cada unha a través dunha labor de consultoría.

A continuación imos presentar as dúas alternativas máis extendidas no panorama nacional.

#### 2.2.2.1. Novanet

Novanet é unha empresa especializada na xestión de competicións de fútbol e o seu producto compónse únicamente dunha aplicación web dende a que crear as clasificacións pero tamén modificar as actas dos partidos ou ver os resultados.

No caso que nos incumbe, non dispón dunha aplicación que poida ser instalable nun teléfono móvil e os árbitros vense na obriga de acceder directamente a páxina web da federación para modificar as actas, pero que cando menos adáptase correctamente a terminais de menor tamaño como se pode observar na Figura 2.3.

Ademais, non permite o funcionamento do sistema de forma offline xa que non foi pensado inicialmente para o caso, o que obriga a levar a acta en papel que se cubre igualmente a pesar de que o árbitro sube posteriormente os datos a través da web cando chega a súa casa.

A interfaz é bastante complexa, o cal é un problema xa que a meirande parte dos árbitros son de avanzada idade e polo tanto resúltalles difícil adaptarse aínda que está a mellorar pouco a pouco.

Por último mencionar que únicamente está pensada para ser utilizada en fútbol e fútbol sala.

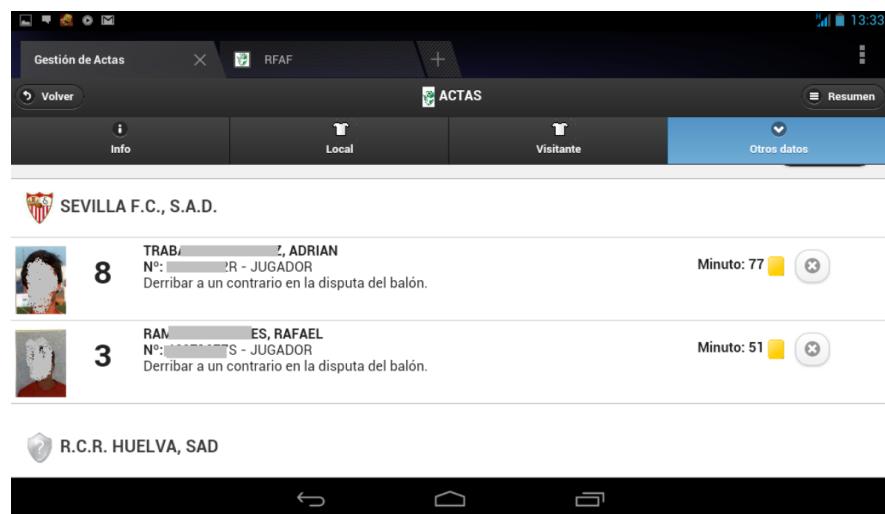


Figura 2.3: Aplicación web de Novanet

### 2.2.2.2. Federatio

O caso de Federatio é moi similar ao anterior, xa que tampouco dispón dunha aplicación específica para a xestión de actas electrónicas de forma sinxela e polo tanto, os árbitros deben acceder a través da páxina web ao chegar a casa para pasar os datos da acta física á versión electrónica.

A interfaz dista de ser atractiva xa que apenas se renovou dende que comezou a funcionar, entorno ao ano 2005, e non se atopa adaptada para terminais móbiles, o que dificulta enormemente a labor dos árbitros.

A pesar de isto, é unha solución amplamente utilizada en deportes como o voleibol e o balonmano como se pode observar na Figura 2.4 na que podemos observar a páxina web da Federación Galega de Voleibol que dispón do sistema Federatio integrado.

The screenshot shows the official website of the Federación Galega de Voleibol (FGVB). At the top, there's a navigation bar with links like 'Competicións', 'Volei-prá', 'Descarga', 'Novas', 'Dir. Técnica', 'Clubes', 'Adestradores', 'Árbitros', 'Comité de Competición', 'Prensa', 'Galería', and 'Mapa do Web'. Below the header, there's a banner for the 'FEDERACIÓN GALEGA DE VOLEIBOL' featuring a volleyball player. The main content area displays information about the '2ª División Autonómica Infantil' and 'Segunda División Infantil Feminina', specifically the '1ª Fase' and 'Grupo: 1ª Fase Grupo C (Zonal) LIGA 4 Vueltas'. It shows the results for 'XORNADA NÚMERO 5' on 12/12/2015, with four teams listed: IES Politécnico, CVP Gómez Lor Multípticas, XAV Cambados B, and Voleibol Manatos. A note indicates that the result for CVP Gómez Lor Multípticas is pending verification. To the right, there's a sidebar titled 'Acceso Restringido' with login fields for 'Usuario' and 'Contraseña', and a link to 'Login'. Another sidebar titled 'Horarios e Resultados' shows a small icon of a calendar. On the far right, there's a 'Tweets' section with several tweets from the FGVB's Twitter account (@VoleiCalego), including one from La Voz de Barbana (@vocabarba) and another from FGVB (@VoleiCalego) about a competition. At the bottom, there are links for 'OPCIÓNIS', 'Volver á competición', 'Enviarlle esta páxina a un amigo', 'Calendario completo', 'Version impresa', and 'Todos os partidos e resultados'.

Figura 2.4: Web da FGVB co sistema Federatio

### 2.2.3. Ferramentas na nube

Nos últimos anos xurdiron unha serie de tecnoloxías que democratizaron o acceso a nube e por iso xurdiron varias produtos específicos e centrados en xestionar dende unha única plataforma, múltiples federacións e asociacións deportivas.

#### 2.2.3.1. miLeyenda

miLeyenda é unha de estas plataformas para a xestión de competicións na nube que permite aos administradores de federacións dispor tamén de unha aplicación móvil nativa para IOS e outra para Android que se pode ver na Figura 2.5.

Dende esta aplicación poden xestionar gran parte dos parámetros das súas competicións entre os que se atopan as clasificacións, as altas de xogadores e por suposto, as actas dos encontros.

Así mesmo tamén dispoñen dunha aplicación para que os xogadores e clubes poidan ver os resultados e as clasificacións polo que o custe de mantemento ditas aplicacións elévase enormemente ao ter que soportar ata 4 apps móbiles diferentes.

Esta é unha das grandes vantaxes de utilizar as tecnoloxías web que emprega VACmatch Mobile, permitindo utilizar unha soa aplicación para calquera sistema operativo e que incluso pode funcionar en un simple navegador web.

A usabilidade das aplicacións é tamén salientable e permite a xestión de diversos deportes pero a cambio non permite de momento cubrir as actas de forma offline, un problema habitual ante a falta de cobertura nos diversos pavillóns e campos deportivos.



Figura 2.5: APP móvil de MiLeyenda

### 2.2.3.2. Esportics

Esportics é outra solución tamén española e que se centra na xestión de competicións deportivas de tenis, padel e deportes electrónicos e que lles permite aos xestores crear o seu propio espazo dentro do portal como o exemplo que temos na Figura 2.6.

A pesar de que a aplicación funciona para múltiples deportes, a súa adaptación é bastante forzada en certos menús e á hora de estruturar as competicións.

Únicamente dispón dunha páxina web adaptable a móbiles, aínda que a adaptación é tamén mellorable, e non dispón dunha aplicación específica para que os árbitros poidan cubrir as actas dende o seu teléfono.

A usabilidade é aceptable, engadindo unha complexidade que non é precisa en certos menús e que provoca que os árbitros de avanzada idade, lles resulte tamén pouco intuitivo.

Jornada/Ronda	Horario	Pista	Jugadores	Categoría	Resultado
Jornada 1	28/09/15 20:30	-	Lluís Martínez Manils - David Pages VS Robert Delgado - RAUL BERENGUEL MARTINEZ	Masculino Segunda	4-6 6-7
Jornada 1	28/09/15 21:00	-	Alex Butjosa Novales - David Sánchez Vallejos VS Artur Costa - Pedro Lazaro Miguel	Masculino Segunda	6-4 4-6 5-7
Jornada 1	29/09/15 09:30	-	Oscar Bermudez - Christian Palomares VS Adrià Marin Chaparro - Arnau Inserre Poyatos	Masculino Open	6-3 1-6 6-4

Figura 2.6: Torneo de tenis na web de Esportics

### 2.2.3.3. Sportngin

Sportngin é unha solución integral para a xestión deportiva, probablemente un dos proxectos de referencia xa que dispón de aplicacións web e móvil para a xestión e a visualización de competicións.

De feito, Sportngin permite a personalización da aplicación móvil para cada federación, cos seus logotipos, colores corporativos e incluso certos menús personalizables, dando a posibilidade de cubrir as actas de forma offline, todo con unha interfaz moi amigable como a que podemos ver na Figura 2.7.

A parte das funcións habituais para xestionar as competicións e a creación de actas, con unha usabilidade moi coidada, aporta como valor engadido a visualización e xestión de noticias, fotos ou estadísticas da competición.

Por último tamén permite aos entrenadores planificar adestramentos e incluso comunicarse cos seus xogadores a través da mensaxería interna.

É sen dúbida o proxecto máis completo e a referencia a seguir a pesar de non ser software libre.



Figura 2.7: APP móvil de Sportngin

#### 2.2.4. Outras plataformas

Engadir  
taboa  
compara-  
tiva

Existen moitas ferramentas para a xestión de competicións e resultados pero apenas ningunha facilita aos árbitros unha plataforma sinxela e con unha usabilidade coidada.

Tamén existen pequenos plugins coa idea de extender outras plataformas xenéricas para adaptalas a xestión de competicións como o *Joomla! CMS sport extension*<sup>2</sup> pero a función final é moi limitada.

Por último temos outras propostas como *Sigueliga*<sup>3</sup> que permite que as persoas que se atopan vendo o partido, poidan subir os resultados pero simplemente é un complemento, non facilita nin elimina o traballo dos xestores de competicións.

<sup>2</sup>Plugin para o sistema de xestión de contidos Joomla.

<sup>3</sup>Rede social para o deporte aficionado.

### 2.3. Aplicacións libres no mercado

O software libre é un sector en crecemento na actualidade, os proxectos colaborativos que forman o mundo *Open Source* estanse a impor en múltiples mercados fronte ás correspondentes alternativas privativas que adoitan a ser tremadamente custosas e que atan ao usuario a tecnoloxías pechadas, impedíndolle ser o dono real do seu software, sen poder adaptalo ou engadirlle novas funcionalidades, nin tan sequera ver cómo se atopa feito e poder certificar así a seguridade da súa información.

Mesmo as grandes compañías TIC están a apostar por liberar parcial ou totalmente as súas tecnoloxías e produtos, favorecendo un desenvolvemento colaborativo fronte a idea atrasada do individualismo do modelo productivo tradicional.

Concretamente no mundo do deporte é tremadamente complicado atopar algúun exemplo de aplicación baseada en software libre e as poucas existentes como *zuluru* ou *phpmysport* non destan pola súa usabilidade nin polas súas funcionalidades, moi por detrás de outras solucións comerciais como as comentadas anteriormente e por suposto sen ningún tipo de aplicación móvil para facilitar a xestión das actas polo que é importante propor unha alternativa como VACmatch Mobile ás aplicacións privativas.

### 2.4. Solución aberta e adaptable

Os xestores de competicións habitualmente realizan unha considerable inversión económica para que unha empresa de consultoría lles cree unha aplicación web ou de escritorio a medida para a súa xestión. Algunhas mesmo dispoñen de aplicacións móbiles para os árbitros pero que son específicas para dito sistema de xestión polo que a reutilización de aplicacións non é posible.

Ademáis, estos sistemas son propietarios e o código non se atopa accesible polo que é imposible tratar de adaptar ditas aplicacións para outros sistemas de xestión ou de engadirlle funcionalidades sen depender da empresa que comercializa o software.

É por isto polo que se chegou a conclusión de que é preciso crear unha plataforma aberta como é VACmatch Mobile que porporciona unha aplicación software libre adaptable a diversos deportes e integra unha API de comunicacóns aberta para a xestión de actas electrónicas e que permite a súa integración noutros sistemas de xestión entre os cales se atopa o sistema de VACmatch Web, unha implementación libre para a xestión de competicións.

---

## Capítulo 3

# Metodoloxía

### Índice general

---

<a href="#">3.1. Lean Startup</a>	15
<a href="#">3.2. eXtreme Programming</a>	15
<a href="#">3.3. Scrum</a>	15
<a href="#">3.4. Adaptación da metodoloxía</a>	16
<a href="#">3.4.1. Desenvolvemento orientado ao cliente</a>	16
<a href="#">3.4.2. Sprints con backlog adaptable</a>	16
<a href="#">3.4.3. Reunións semanais</a>	16
<a href="#">3.4.4. Reunións diarias</a>	17
<a href="#">3.4.5. Releases</a>	17
<a href="#">3.4.6. Simplicidade</a>	17
<a href="#">3.4.7. Tests</a>	17
<a href="#">3.4.8. Fluxo de contribución ao proxecto</a>	18

---

NESTE capítulo imos analizar as diversas metodoloxías utilizadas para a xestión do proxecto e explicar a adaptación das mesmas que finalmente se utilizó.

Comezaremos falando da metodoloxía orientada ao modelo de negocio xa que, como se indica na introdución, este proxecto xurdíu dentro de unha iniciativa empresarial e polo tanto dende o primeiro momento se traballou orientado cara o cliente, convertíndoo nun dos pilares do desenvolvemento.

Seguidamente comentaremos as metodoloxías áxiles que serviron de base para definir a metodoloxía utilizada finalmente, unha adaptación das mencionadas e que tamén se indica para rematar o capítulo.

### 3.1. Lean Startup

Lean Startup [Rie11] é unha metodoloxía para abordar o lanzamento de negocios e produtos a través da validación, a experimentación e a iteración no lanzamento dos mesmos co fin de acortar o ciclo de desenvolvemento.

É unha metodoloxía de traballo moi habitual nas startups que se centra na idea de *Crear - Medir - Aprender*, desenvolvendo pequenos produtos e realizando tests de mercado reais con verdadeiros clientes co fin de medir o seu grao de satisfacción e aprender para mellorar o producto en seguintes iteracións.

Habitualmente céntrase na idea de crear un MVP (Minimum Viable Product), unha versión do producto que permite os desenvolvedores recoller co mínimo esforzo a máxima cantidade de coñecemento validado por parte dos clientes, evaluando as hipóteses de se os clientes realmente estarían dispostos a pagar polo producto e implicando a dito cliente no desenvolvemento do producto.

### 3.2. eXtreme Programming

eXtreme Programming [SG15] é unha metodoloxía de desenvolvemento áxil e incremental baseada na integración do cliente no desenvolvemento así como na simplicidade do código.

A metodoloxía apostá por facer as cousas sinxelas, sen preocuparse por ter que facer un pequeno traballo por adaptalas se é preciso, fronte a idea tradicional de facer un gran traballo para quizás nunca chegar a utilizar parte do mesmo.

As entregas funcionais son frecuentes e outras características como a importancia de introducir a programación en parellas para reducir o número de errores que se producen ao programar.

Por último aboga por introducir o TDD (Test Driven Development) [Mar08], implementando primeiro os tests, verificando que fallan, para a continuación implementar o código que fai que pasen correctamente os mesmos. A idea é que os requisitos sexan convertidos a probas e de este modo cando os tests se pasen, poderemos garantizar que o código cumple os requisitos.

### 3.3. Scrum

Scrum [DGG12] tamén é unha metodoloxía incremental de desenvolvemento cunha serie de roles definidos para o proceso, cada un coas súas responsabilidades e que divide o proxecto en varios *Sprints* que son ciclos de desenvolvemento.

Cada un de eles ten unha duración definida polo equipo de, habitualmente, entre unha e catro semanas, proporcionando un incremento de software entregable ao final de cada *Sprint*.

A totalidade das tarefas do proxecto atópanse definidas e priorizadas en unha lista chamada *Product Backlog*. Para cada sprint, selecciónanse aquellas tarefas que determinarán a lista a

---

implementar durante a presente iteración, o *Sprint Backlog*, e que non pode variar ata rematar o sprint.

Durante todo o ciclo de traballo realizánse reunións diarias para comprobar o estado do proxecto así como outras ao finalizar e ao comezar os sprints, co fin de analizar a iteración anterior e planificar a seguinte, facendo un seguimento continuo do proxecto e facilitando a adaptación do mesmo a posibles novos requisitos.

### 3.4. Adaptación da metodoloxía

#### 3.4.1. Desenvolvemento orientado ao cliente

O proxecto ten lugar dentro dunha iniciativa empresarial polo que se decidíu utilizar un modelo de desenvolvemento orientado ao cliente en todo momento, baseandose no pilar central da metodoloxia *Lean Startup*.

Para isto realizáronse diversas visitas as federacións para comprobar as súas necesidades a través dunha serie de entrevistas estructuradas para coñecer os problemas e a súa prioridade a hora de resvelos.

Do mesmo modo realizáronse dous prototipos, un primeiro únicamente con plantillas HTML para testear a organización da interfaz de usuario e un segundo xa funcional para comprobar a resposta dos usuarios finais ante o seu funcionamento.

#### 3.4.2. Sprints con backlog adaptable

A organización do desenvolvemento organizouse de xeito moi similar a idea proposta en *Scrum*, dividindo o proceso en sprints, pequenas iteracións de díás ou tres semanas de duración e que cada unha proporciona unha serie de novas funcións.

Cada sprint comeza con unha reunión de aproximadamente 30/45 minutos de duración na que realizar a planificación do mesmo en función do traballo realizado no sprint anterior, o que permite realizar melloras nas previsións segundo o aprendido dos anteriores.

A diferencia do proposto por Scrum, decidíuse optar por sprints de duración variable e con un backlog adaptable según as necesidades xa que proporciona unha maior flexibilidade e liberdade.

#### 3.4.3. Reunións semanais

Todas as semanas faise unha reunión de 30/45 minutos de duración na que analizar o realizado na semana anterior e comprobar o seguimento da iteración co fin de atopar desviacións e corrixilas.

Cando unha reunión semanal coincide co fin de un sprint, dita reunión sirve para realizar a planificación do seguinte sprint de xeito moi similar as reunións de sprint que se realizan en

---

*Scrum.*

#### 3.4.4. Reunións diarias

Ao comezar o día realiza unha análise duns 10 minutos de duración para revisar o realizado no día anterior e planificar de forma máis concreta o que se vai facer ese mesmo día.

#### 3.4.5. Releases

Durante o desenvolvemento do proxecto trátase de aplicar a idea de realizar unha serie de pequenos entregables en cada iteración.

Todas as entregas ao finalizar unha iteración son totalmente funcionais pero non todas son versións entregables reais para ser postas en produción.

Durante o desenvolvemento producíronse 3 entregas (*releases*) totalmente funcionais, a primeira foi un prototipo, a segunda foi a versión real do proxecto e a terceira incorporou tests e diversas características para asegurar unha primeira versión estable.

#### 3.4.6. Simplicidade

Utilizouse o principio de simplicidade que promove *eXtreme Programming* durante todo o desenvolvemento baixo a máxima de implementar únicamente o imprescindible en cada momento, sempre pensando en programar para hoxe e non para mañá.

A idea fundaméntase en realizar refactorizacións de código para engadir novas funcionalidades a medida que son necesarias en lugar de invertir demasiado tempo na planificación e implementación de funcións que se supoñen necesarias e, algunas das cales, é probable que non sexan utilizadas finalmente.

#### 3.4.7. Tests

A importancia de creación de tests automatizados está totalmente demostrada, atopándose en auxe metodoloxías como TDD (Test Driven Development)<sup>1</sup> ou BDD (Behaviour Driven Development)<sup>2</sup> que tratan de dirixir o desenvolvemento a través dos tests e que son realizados antes da implementación da funcionalidade.

Durante a primeira parte do desenvolvemento non se aplicou ningunha de estas metodoloxías pero a partir da primeira *release* e da integración dos primeiros tests, decidíuse optar por aplicar TDD no desenvolvemento, realizando probas unitarias nos servicios utilizados.

---

<sup>1</sup>Desenvolvemento dirixido polos tests.

<sup>2</sup>Desenvolvemento dirixido polo comportamento

---

### 3.4.8. Fluxo de contribución ao proxecto

O fluxo de traballo utilizado dende o primeiro día trata de simular o traballo diario de equipo e permite controlar a evolución do código de xeito máis ordenado.

Dispónse de unha rama *master* na que se atopa a versión estable de desenvolvemento así como de unha rama *development* que é máis inestable e que ao final de cada sprint, é integrada dentro de *master*.

Unha nova funcionalidade ou erro é resolto nunha nova rama independente, creada a partir de *development* e tratando que todas estas novas funcionalidades sexan independentes entre si.

Así mesmo tratase de que todos os *commits* sexan funcionais e o máis independentes posibles, evitando ter algúns que non compile ou que non pase os tests.

Posteriormente realizase unha *pull request*<sup>3</sup> a través do mecanismo que proporciona o repositorio de código de GitHub, esperando que alguén revise o código para ser integrado na rama de desenvolvemento.

Cada certo tempo revisánse as *pull requests* abertas, analízase o código e se todo é correcto, acéptase.

Ao final de cada sprint realizaase una nova *pull request* para integrar a funcionalidade creada no sprint actual e que se atopa na rama de desenvolvemento, dentro da rama estable.

Dende a introducción de tests no proxecto, todo código subido ao repositorio é analizado a través dun sistema de integración continua —que se comenta con máis detalle na Sección ??,— e que comproba se os cambios engadidos pasan os tests ou non, e avisan por correo electrónico do resultado.

---

<sup>3</sup>Unha *pull request* é unha petición para integrar unha rama de Git en outra a través de GitHub, un mecanismo habitual para engadir novas funcionalidades ou corregir errores.

## Capítulo 4

# Análise de requisitos globais

### Índice general

---

<a href="#">4.1. Consultas a xestores de federacións</a>	19
<a href="#">4.2. Peticións obtidas</a>	20
<a href="#">4.3. Requisitos finais</a>	21
<a href="#">4.3.1. Usuarios</a>	21
<a href="#">4.3.2. Listar actas</a>	21
<a href="#">4.3.3. Visualizar actas</a>	21
<a href="#">4.3.4. Xeración de actas offline</a>	22
<a href="#">4.3.5. Modificación de actas</a>	22

---

NESTE capítulo exporemos o proceso de análise de requisitos para o desenvolvemento do proxecto, explicando as diversas visitas que se realizaron a múltiples federacións.

Durante meses traballouse da man de varias de estas federacións e asociacións deportivas tratando de comprender, non só as necesidades reais dos clientes se non tamén traballando na usabilidade da aplicación da súa man.

Comezaremos vendo as suxerencias recibidas das diversas federacións e finalmente expoñeremos os requisitos que finalmente se decidiu engadir ao proxecto.

### 4.1. Consultas a xestores de federacións

Para a realización deste apartado decidiuse consultar con diversas asociacións deportivas e federacións das que obter suxerencias e peticións acerca das necesidades que actualmente están a demandar, co fin de obter certas funcionalidades a implementar e incluso a priorización segundo as súas necesidades máis urxentes.

Na realización deste apartado contouse coa colaboración das asociacións e federacións deportivas que se detallan a continuación.

Tras varias reunións con eles, obtívose unha lista de requerimentos e suxerencias respecto das súas necesidades, que están detalladas na Sección 4.2, e dos que finalmente foron destilados os requisitos finais, que son presentados como Sección 4.3 de este capítulo.

**Asociación de Peñas de Fútbol de A Coruña.** É a asociación máis interesada polo proxecto e coa que se leva colaborando dende o primeiro momento, aportando suxerencias e incluso novos colaboradores para poder desenvolver un producto de calidade.

Realizáronse ata 5 visitas á federación co fin de mostrarllles a evolución do proxecto, comprobar a usabilidade da aplicación e o estado das funcionalidades.

**UPOFU.** A Asociación de peñas ten boa relación coa UPOFU polo que nos facilitou o seu contacto e ofrecéronse da mesma maneira a colaborar co proxecto, interesados tamén en incorporalo na súa xestión.

**Torneo VACmatch.** Organizouse un torneo de fútbol sala co fin de testear o primeiro prototípo do proxecto con usuarios reais, tanto árbitros como xogadores e no que se comprobou as dificultades dos usuarios e se verificou a súa necesidade de dispor deste tipo de ferramentas.

**Outras.** Tamén se realizaron visitas a outras federacións incluso de outros deportes como o voleibol para comprobar os seus problemas na xestión e verificar a importancia de que o proxecto sexa facilmente adaptable a outros deportes.

## 4.2. Peticións obtidas

**Cubrir acta en tempo real.** A aplicación móvil debe permitir cubrir as actas e actualizar os resultados en tempo real co fin de manter a web da federación actualizada en todo momento.

**Permisos.** Débese dispor dun sistema de permisos para diferenciar a árbitros e outros xestores da competición.

**Sincronización.** A aplicación móvil debe sincronizar os datos coa plataforma central onde se atopa o sistema de xestión da federación e a súa web.

**Pessoas convocadas.** É preciso poder dispor de todas as persoas inscritas nun equipo e poder indicar de xeito sinxelo si esas persoas están ou non no encontro.

**Eventos.** A aplicación debe poder crear novos eventos, borrarlos e mostralos de xeito sinxelo e ao mesmo tempo de xeito xenérico que permita integrar calquera deporte.

**Motivación dun evento.** Debe poderse incluir en certos eventos un motivo polo que se creou ese evento, dispoñendo dunha lista de motivos por defecto e incluso permitindo ao xestor da federación, engadir novos motivos personalizados para a súa federación.

**Editar dorsal dun xogador.** Xa que en moitas competicións un xogador pode xogar cada partido con un dorsal diferente, debe poder cambiarse o dorsal por defecto dende a aplicación móvil.

**Persoa con varios roles.** Debe terse en conta a posibilidade de que unha persoa poida ter varios roles, tanto de xogador como de entrenador dentro de un equipo.

### 4.3. Requisitos finais

#### 4.3.1. Usuarios

- **Facer login e logout.** A aplicación móvil debe permitir iniciar e pechar sesión para os árbitros.
- **Permisos para edición de actas.** A aplicación de xestión disporá de permisos diferenciados para editar as actas xa que os árbitros únicamente poden editar as actas que teñen asignadas.

#### 4.3.2. Listar actas

- **Visualizar próximas actas a cubrir dun árbitro.** Mostrar a lista de próximas actas que ten para cubrir un árbitro, mostrando o lugar e a data do mesmo para facilitar o seu traballo.
- **Visualizar actas cubertas dun árbitro.** Debe mostrar as actas cubertas anteriormente e todos os seus datos.
- **Actualización automática de actas descargadas ante modificacións.** As actas deben actualizarse de forma automática na aplicación do árbitro unha vez o xestor da federación realiza a asignación dun partido a un colexiado.

#### 4.3.3. Visualizar actas

- **Listar o personal e xogadores dun equipo.** Móstrase o personal e os xogadores do equipo na aplicación móvil.
- **Visualizar datos xerais dun acta.** Débese mostrar do xeito máis simplificado posible os datos xerais da acta nunha pantalla inicial para facilitar que sexa cuberta interactivamente durante o desenvolvemento do encontro.
- **Visualizar eventos dun acta.** Permitirse visualizar os eventos ordenados cronolóxicamente para facilitar a súa consulta.

#### 4.3.4. Xeración de actas offline

A aplicación debe permitir a creación de actas de forma offline xa que pódese dar o caso de que a aplicación móvil non actualice as novas actas e o árbitro se vexa na obriga de crear unha acta de forma manual.

#### 4.3.5. Modificación de actas

- **Modificación de propiedades da acta.** Débese permitir modificar propiedades da acta tales como a localización do encontro ou a data do mesmo.
- **Convocar un xogador ou entrenador.** A aplicación móvil permitirá indicar qué persoal dos equipos están presentes no encontro así como editar certos datos dos mesmos como o seu dorsal.
- **Engadir un xogador que non está no equipo.** Pode darse o caso de que a un xogador débeselle permitir xogar un encontro aínda que non fose dado de alta na federación correspondente polo que é preciso poder engadir novos xogadores.
- **Editar datos de persoal creado.** Débese permitir editar certos datos dun xogador que foi creado dende a aplicación móvil como o nome, o dorsal ou o equipo o que pertencen.
- **Poder engadir motivos dun evento xerado.** A federación debe poder engadir novos motivos personalizados para poder engadir a un evento dende a aplicación de xestión.
- **Cambiar de parte.** A aplicación móvil debe permitir cambiar de parte no encontro.
- **Modificar o tempo.** A aplicación móvil disporá dun cronómetro que permita seguir o tempo do encontro así como permitirá modificalo manualmente por si hai algúin desaxuste durante o encontro.
- **Engadir observacións na acta.** O árbitro debe poder engadir observacións as actas dos encontros.
- **Asinar a acta.** Tanto persoal do equipo como árbitros deben poder asinar as actas con un código PIN do que disporá cada un.
- **Engadir eventos deportivos.** A aplicación debe facilitar a adaptación de novos deportes e a posibilidade de engadir de xeito sinxelo novos eventos.

## Capítulo 5

# Planificación e seguimento

### Índice general

---

<b>5.1. Validación de negocio (Xullo 2015 – Novembro 2015)</b>	24
5.1.1. Prototipo visual	24
5.1.2. MVP funcional	25
<b>5.2. Desenvolvemento de produto (Novembro 2015 – Xaneiro 2016)</b>	27
5.2.1. 1 <sup>a</sup> e 2 <sup>o</sup> iteración. Creación do proxecto e xestión de actas	28
5.2.2. 3 <sup>a</sup> iteración. Eventos	30
5.2.3. 4 <sup>a</sup> iteración. Xestión de usuarios e creación offline de actas	32
5.2.4. 5 <sup>a</sup> iteración. Sinaturas	34
<b>5.3. De empresa a comunidade (Xaneiro 2016 – Maio 2016)</b>	34
5.3.1. 6 <sup>a</sup> e 7 <sup>a</sup> iteración. Optimización e melloras	35
5.3.2. 8 <sup>a</sup> iteración. Testing e integración continua	36
5.3.3. 9 <sup>a</sup> e 10 <sup>a</sup> iteración. Inxección de dependencias	37
5.3.4. Release 0.2.0: Usabilidade en menús	38
5.3.5. Release 0.2.1: I18n e app híbrida	40
5.3.6. Release 0.2.2: Imáxe corporativa e revisión de erros	40

---

NESTE capítulo detallaremos a planificación e o seguimento do proxecto, un proxecto que por diversas circunstancias se dividú principalmente en tres grandes etapas, as dúas primeiras mentres VACmatch era unha iniciativa emprendedora baseada nun proxecto software libre e a última durante a cal se comezou unha conversión da iniciativa cara un proxecto comunitario.

Durante o desenvolvemento xuridiron tamén diversos acontecementos relacionados co proxecto e que tamén é importante resaltar debido a súa influencia no desenvolvemento.

**Agosto 2015 - Outubro 2015** VACmatch. Validación de negocio.

**Outubro 2015 - Xaneiro 2016** VACmatch. Desenvolvemento de producto.

**Xaneiro 2016 - Xuño 2016** De empresa a comunidade.

## 5.1. Validación de negocio (Xullo 2015 – Novembro 2015)

A duración de esta etapa é de aproximadamente 4 meses e ven determinada polos primeiros pasos de VACmatch como iniciativa empresarial e que levan a orientar o desenvolvemento do produto cara o cliente, comezando con unha serie de prototipos para coñecer as suas necesidades e validar a idea de negocio.

Durante o primeiro mes planíficase a realización dun prototipo visual co fin de comprobar a usabilidade e consolidar os requisitos dos clientes. De seguido, plantéxase crear un pequeno prototipo funcional, un MVP (Mínimo Producto Viable) na metodoloxía Lean Startup, co obxectivo de testear as necesidades dos clientes e definir o producto final a desenvolver.

### 5.1.1. Prototipo visual

#### 5.1.1.1. Planificación e definición da iteración

Esta iteración dura un total de 4 semanas de desenvolvemento entre o 15 de Xullo e o 16 de Agosto e realizase unha visita semanal ao cliente para obter feedback e mostrarlle a evolución do prototipo.

Durante este periodo de un mes de duración planificouse o desenvolvemento de unha aplicación moi sinxela e sen funcionalidade, que únicamente permite analizar a usabilidade do sistema e comprobar si é factible adaptar o proceso de creación de un acta deportiva, a unha aplicación móvil.

Así mesmo, ao longo do período realizaranse ata tres visitas a federación coa que se traballou dende o primeiro momento para comprobar a experiencia de un futuro usuario real da aplicación e obter feedback para futuras melloras.

#### 5.1.1.2. Revisión e feedback

Durante as visitas as federacións obtivéreronse diversas propostas que levaron a adaptar o prototipo:

- Facer interactiva a aplicación e non mostrar grandes táboas con datos.
- Todas as accións deben xirar ao redor da acta.
- Crear partidos cando non hai cobertura.

#### 5.1.1.3. Tarefas e seguimento

A descomposición das tarefas desta iteración son as seguintes:

- Crear esqueleto da aplicación.

- Como árbitro quero poder consultar as próximas actas a cubrir.
- Como árbitro quero poder consultar as actas xa cubertas.
- Como árbitro quero poder editar un acta.
- Como árbitro quero poder ver un acta.
- Como árbitro quero poder ver os xogadores de ambos equipos.
- Como árbitro quero poder engadir un evento.
- Como árbitro quero poder rematar ou suspender un partido.
- Como árbitro quero poder logearme.
- Como árbitro quero poder seleccionar cales xogadores de cada equipo se atopan no encontro.
- Como árbitro quero poder borrar un evento.
- Estudio sobre React e Flux.

### 5.1.2. MVP funcional

#### 5.1.2.1. Planificación e definición da iteración

Esta iteración dura un total de 3 meses e desenvólvese entre o 16 de Agosto e o 15 de Novembro, realizando múltiples visitas a federacións e asociacións deportivas.

Unha vez finalizadas as probas visuais e de usabilidade procédese a planificar o desenvolvemento para adaptar o prototipo e engadirlle funcionalidade sinxela, sen validacións e sen funcionalidade offline, co obxectivo de obter un prototipo funcional que poida ser utilizado por usuarios reais nun entorno controlado.

Engadirase funcionalidade para as vistas creadas na iteración anterior, comezando polo listado de actas pendentes e rematadas e diversos componentes xenéricos como os que se utilizan para listar xogadores e outros elementos como poden ser as actas.

Únicamente se engadirá a funcionalidade básica imprescindible para xestionar un encontro, excluindo requisitos como a sinatura de actas ou a creación offline das mesmas co fin de axilizar as primeiras probas.

Durante a iteración tamén se farán visitas a federación para mostrar o estado do desenvolvemento e para buscar que tamén árbitros reais vexan os progresos e proporcionen feedback.

Unha vez rematada a iteración realizarase o torneo onde probar o prototipo desenvolto nun caso real; pódese ver o desenvolvemento de dita competición na Sección [5.1.2.4](#).

### 5.1.2.2. Revisión e feedback

Durante as visitas as federacións obtivéronse múltiples propostas e melloras, moitas das cales será incorporadas ao backlog do proxecto mentres que outras serán rexeitadas polo momento ao non considerarse prioritarias ou por ser casos de usos moi concretos para esa federación e difícilmente extrapolables a outras.

#### Listaxe de tarefas engadidas ao backlog.

- Meter xogadores manualmente xa que poden non terse creado no sistema de xestión.
- Poder ver os eventos de forma sinxela dende a vista de fin de partido para que os equipos vexan o que están firmando.
- Editar dorsal dos xogadores xa que poden cambiar.
- Ter opción de non poñer motivo para as tarxetas.
- Mostrar foto de xogador ao engadir un evento.

#### Listaxe de tarefas rexeitadas polo momento.

- Ao marcar doble amarela, avisar da expulsión. Moi concreta para un deporte, non se implementa de momento.
- Mostrar confirmación de que se engadíu un evento.
- Avisar aos delegados/personal do clube cando se sube un acta.
- Posibilidade de que o árbitro engada un anexo na casa á acta en lugar de escribir as incidencias.

### 5.1.2.3. Tarefas e seguimento

As tarefas que se realizarán durante esta iteración son as seguintes:

**MVP1** Como árbitro quero poder cargar a lista de actas pendentes.

**MVP2** Como árbitro quero poder cargar a lista de actas rematadas.

**MVP3** Como árbitro quero poder ver os datos dun partido.

**MVP4** Como árbitro quero poder ver a lista de xogadores dun equipo.

**MVP5** Como árbitro quero poder seleccionar os xogadores presentes no partido.

**MVP6** Como árbitro quero poder engadir un gol a un xogador.

**MVP7** Como árbitro quero poder engadir unha falta a un xogador.

**MVP8** Como árbitro quero poder engadir unha tarxeta (amarela ou vermella) a un xogador.

**MVP9** Como árbitro quero poder ver a lista de eventos de un partido.

**MVP10** Como árbitro quero poder borrar eventos de un partido.

**MVP11** Como árbitro quero poder engadir incidencias a un partido.

Planificouse un total de 65 horas para desenvolver esta iteración que non resultaron suficientes, obligando a facer un total de 10,5 horas extra motivado polo descoñecemento da tecnoloxía, formación na que foi preciso invertir un maior número de horas das esperadas.

#### 5.1.2.4. I Torneo VACmatch

Durante este periodo tamén se planificou a organización dun torneo de fútbol sala a finais do mes de Outubro coa idea de probar en nun entorno real e controlado, os primeiros prototipos desenvoltos.

Finalmente participaron 6 equipos e mais de 50 persoas durante os dous días que durou o evento, obtendo todo tipo de suxerencias e detectando múltiples errores que serán analizados e comentados na seguinte iteración.

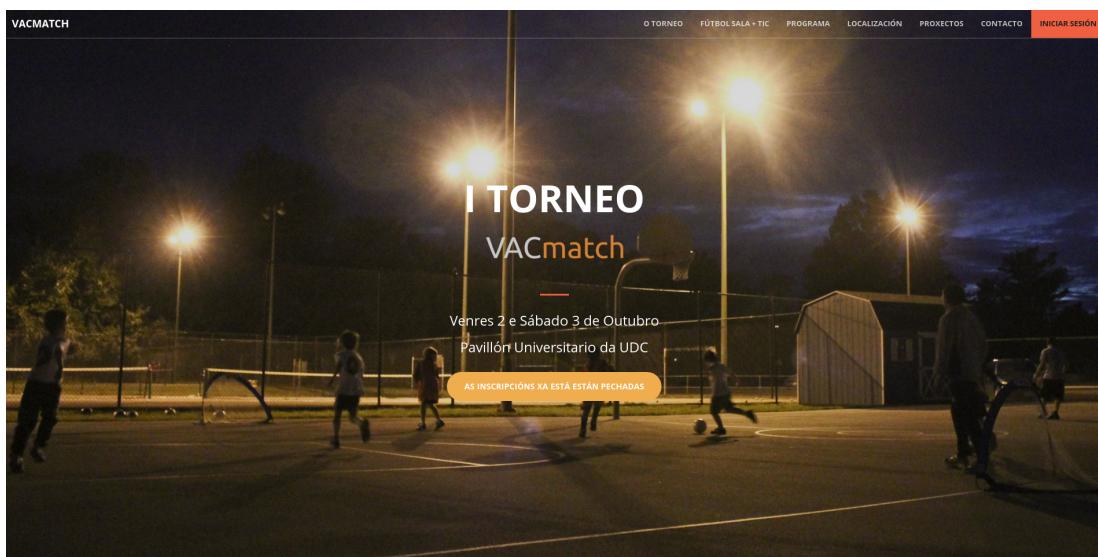


Figura 5.1: Web do I Torneo VACmatch

## 5.2. Desenvolvemento de produto (Novembro 2015 – Xaneiro 2016)

Tras analizar os problemas e os errores cometidos durante a realización do I Torneo VACmatch, e concretamente no funcionamento da aplicación móvil durante o mesmo, decidíuse comezar

novamente dende o principio un proxecto novo en lugar de facer unha refactorización do prototípo.

Durante este tempo decidimos inscribir o proxecto no “Concurso Universitario de Software Libre” o que incentivou a comezar a escribir un blog técnico, a través da conta de Medium<sup>1</sup> de VACmatch, no que contar os avances que suceden durante o desenvolvemento do proxecto.

En este periodo de 3 meses de duración planifícase dedicar unha media de 4 horas diárias a VACmatch Mobile debido a que en paralelo se está a traballar na versión de VACmatch Web.

Así mesmo descontando os múltiples días festivos calculase un total de 208 horas de traballo divididas en 5 iteracións.

### 5.2.1. 1<sup>a</sup> e 2<sup>o</sup> iteración. Creación do proxecto e xestión de actas

#### 5.2.1.1. Planificación e definición da iteración

Estas iteracións transcorren ao longo do mes de Novembro, adicando 15 días a cada unha.

A primeira céntrase en comezar o desenvolvemento da aplicación pensando dende o primeiro momento no funcionamento tanto online como offline e controlando os posibles conflictos que poidan suceder entre os datos.

Comézase co estudo da tecnoloxía en profundidade xa que os prototipos non utilizaban nin Reflux nin PouchDB e simplemente enviaban os datos a unha API remota.

Créase tamén o proxecto base engadindo a licencia e defínese o modelo de datos, que vai cambiar bastante do modelo inicial, pensando para unha base de datos relacional que é a que podíamos atopar na API remota de VACmatch Web.

É por iso que os datos serán desnortinalizados e pasarán a almacenarse en documentos en lugar de en táboas.

Por último comezase a implementación do listaxe de actas a cubrir e de un botón para engadirlos e eliminarlos de xeito sinxelo para facer as primeiras probas.

Durante a 2<sup>o</sup> iteración planifícase o desenvolvemento das páxinas principais e básicas para a xestión da acta dun encontro.

As funcionalidades que se abordarán neste sprint céntranse principalmente na vista na que se mostra o resumo actual da acta así como o control do tempo, co fin de permitir xestionar o encontro en tempo real de forma interactiva.

#### 5.2.1.2. Revisión e feedback

Durante esta iteración analizáronse os problemas detectados durante o torneo, optando por comenzar a implementación do proxecto dende cero como se comentou anteriormente.

---

<sup>1</sup>Medium é unha rede social que permite crear e seguir blogs de múltiples temáticas

Tamén se realizaron diversas publicacións do blog para comentar a realización do torneo e sobre todo analizar aspectos como a elección tecnolóxica, a metodoloxía de desenvolvemento e as próximas funcionalidades a abordar.

Ao non realizar ningunha visita a federacións, non se obtivo o seu feedback.

### 5.2.1.3. Tarefas e seguimento

Durante estas iteracións realizáronse as seguintes tarefas:

**S1.1** Definir modelo de datos.

**S1.2** Definir arquitectura y tecnología.

**S1.3** Deseñar mockups.

**S1.4** Crear proxecto base.

**S1.5** Crear modelos en PouchDB.

**S1.6** Estudio da tecnoloxía. React, Reflux, PouchDB, Redmine

**S1.7** Como árbitro quero poder obter a lista de actas a cubrir.

**S1.8** Permitir crear e borrar actas para tarefas de test.

**S1.9** Engadir licencia e Readme

**S2.1** Como árbitro quero ver o resumo da acta.

**S2.2** Como árbitro quero controlar o tempo do partido.

**S2.3** Como árbitro quero manter o tempo do partido aínda que cambie de páxina.

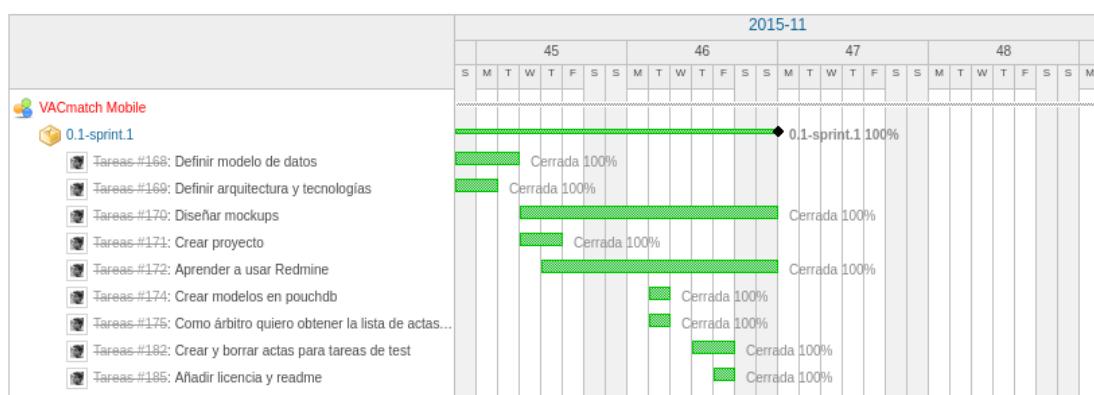


Figura 5.2: Diagrama de Gant do sprint 1

A planificación inicial de 71 horas de desenvolvemento finalmente cumpliuse correctamente, incluso reducindo o tempo empregado ata as 68.

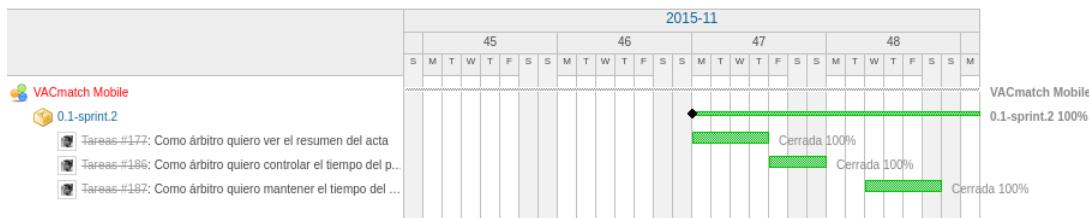


Figura 5.3: Diagrama de Gant do sprint 2

#### 5.2.1.4. Participación na I Lonxa de Financiamento Responsable

Durante este tempo tamén cómpre destacar a participación de VACmatch na I Lonxa de Financiamento Responsable en Galicia, permitíndonos presentar o noso proxecto ante diversos inversores preocupados pola responsabilidade social das empresas.



Figura 5.4: I Lonxa de Financiamento Responsable

Foi unha experiencia única que nos permitiu introducirnos por primeira vez no mundo da inversión en startups e coñecer aos diversos proxectos que se presentaron, obtendo tamén feedback dos asistentes e participantes.

#### 5.2.2. 3<sup>a</sup> iteración. Eventos

##### 5.2.2.1. Planificación e definición da iteración

Esta iteración comeza o 30 de Novembro e remata o 13 de Decembro, unha iteración de dúas semanas de duración e céntrase principalmente na xestión de eventos dos encontros, tarefas que requieren unha forte planificación e análise xa que se busca que ditos eventos sexan o máis xenéricos posibles e facilmente adaptables aos diversos deportes.

Da mesma forma plantéxase crear tipos de eventos que modifiquen tamén a propia acta, actualizando na mesma tanto o resultado como as faltas cometidas.

Por último incorpórase tamén na iteración a posibilidade de convocar e editar persoas dun equipo así como algúns pequenos errores detectados na última iteración.

### **5.2.2.2. Revisión e feedback**

Esta iteración centróuse en avanzar a maior velocidade no desenvolvemento, pódese observar cómo o número de horas asignadas supera a media teórica planificada para o período completo.

Non se realizou ningunha visita a federacións e centrouse prácticamente todo o esforzo, fora do desenvolvemento, en preparar a presentación para a I Lonxa de Financiamento Responsable en Galicia da que se falou anteriormente na Sección [5.2.1.4](#) e na que se obtiveron diversos comentarios para enfocar o modelo de negocio e o desenvolvemento da aplicación cara o cliente.

### **5.2.2.3. Tarefas e seguimento**

A continuación móstranse as diversas tarefas realizadas na iteración:

**S3.1** Como árbitro quero poder engadir un evento.

**S3.1.1** Como árbitro quero ver a lista de xogadores para asignar un evento.

**S3.1.2** Como árbitro quero poder confirmar engadir un evento.

**S3.1.3** Como árbitro quero engadir unha causa a un evento.

**S3.2** Cómo árbitro quero poder ver a lista de eventos de un partido.

**S3.3** Como árbitro quero que se xeneren eventos de comezo e fin de partido.

**S3.4** Como árbitro quero que se xeneren eventos ao cambiar de parte.

**S3.5** Como árbitro quero que se actualice o resultado ao engadir un gol.

**S3.6** Como árbitro quero que se actualicen as faltas automáticamente.

**S3.7** Como árbitro quero poder convocar e desconvocar xogadores.

**S3.8** Como árbitro quero poder cambiar o dorsal de un xogador nun partido determinado.

**S3.9** Como árbitro quero poder borrar un evento.

**S3.10** Como árbitro quero ver a lista de eventos de un partido ordenada por tempo e parte.

**S3.11** Mostrar únicamente xogadores convocados ao engadir un evento.

**S3.12** Como árbitro quero que ao borrar un evento se actualicen os resultados e as faltas na Acta.

---

**S3.12** Erro: Cando non hai ningún evento de cambio de parte hai un error.

**S3.13** Erro: Corexir erro cando se engade un evento con causa.

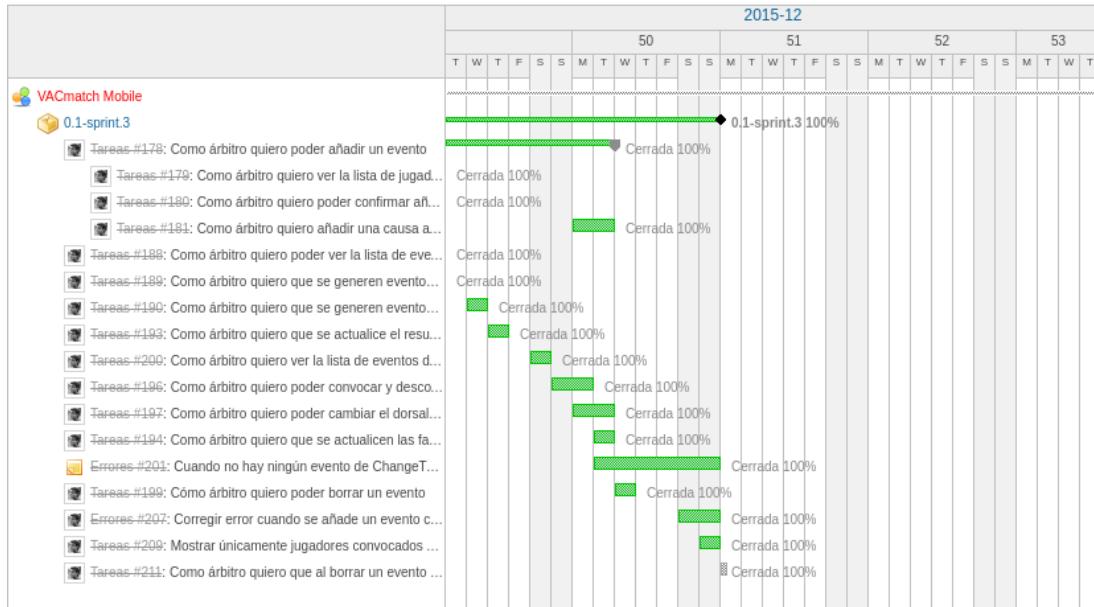


Figura 5.5: Diagrama de Gant do sprint 3

Finalmente a planificación inicial de 51 horas cumplíuse correctamente quedando un extra de 6 horas que foron reasignadas ao outro proxecto.

### 5.2.3. 4<sup>a</sup> iteración. Xestión de usuarios e creación offline de actas

#### 5.2.3.1. Planificación e definición da iteración

Esta iteración ten lugar entre o 14 e o 30 de Decembro de 2015 e céntrase na autenticación da aplicación que debe integrar unha base de datos remota para que os árbitros poidan conectarse co seu usuario e tamén inclúe a creación e edición manual de actas que en anteriores iteracións foi engadida únicamente para probas internas pero sen realizar as comprobacións necesarias.

Tamén se planifican tarefas para comezar as primeiras partes da memoria do proxecto que incluen a introdución, o estado da arte os fundamentos tecnolóxicos.

#### 5.2.3.2. Revisión e feedback

Durante este periodo escribíuse unha nova entrada no blog do proxecto na que se fixo unha introdución aos fundamentos tecnolóxicos do mesmo, explicando a motivación da elección das tecnoloxías e mostrando un exemplo moi sinxelo.

Por último mostrouse tamén a estrutura da aplicación e as partes más importantes do proxecto, pensando en facilitar a introdución no desenvolvemento aos posibles interesados en colaborar no mesmo.

### 5.2.3.3. Tarefas e seguimento

O listado de tarefas abordadas durante esta iteración é o seguinte:

**S4.1** Como usuario quero poder facer log in.

**S4.2** Engadir diferenciación entre Persoal e Xogadores.

**S4.3** Como usuario quero poder facer log out.

**S4.4** Como árbitro quero poder crear un partido manualmente.

**S4.5** [Memoria] Definir introdución.

**S4.6** [Memoria] Estado da arte.

**S4.7** [Memoria] Fundamentos tecnolóxicos.

**S4.8** [Memoria] Engadir modelo.

**S4.9** Como árbitro quero poder editar un acta creada dende o móvil.

**S4.10** Erro: Correxir erro coa variable dialogIsOpen ao editar o dorsal de un xogador.

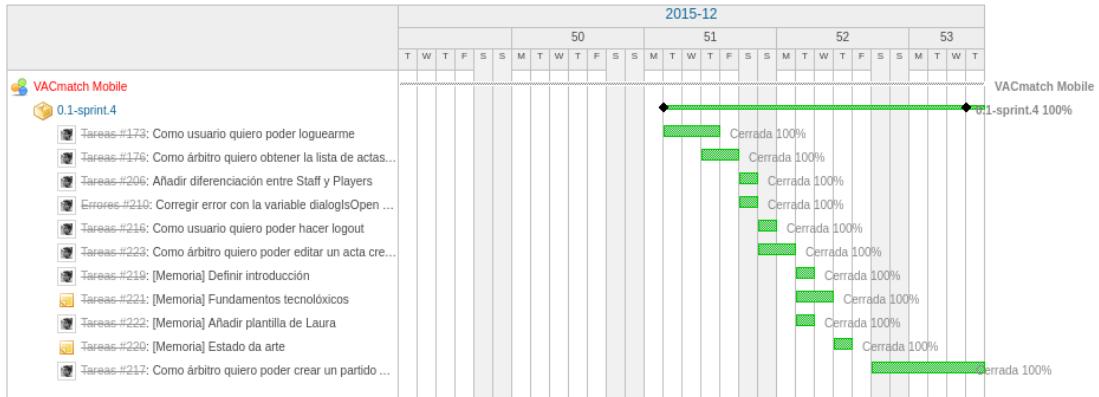


Figura 5.6: Diagrama de Gant do sprint 4

Esta vez a planificación non foi moi acertada debido a diversos imprevistos externos ao proxecto, o que levou a necesidade de mover diversas tarefas á seguinte iteración para axustar a planificación incial de 53.5 horas as 44 que se realizaron finalmente.

### 5.2.4. 5<sup>a</sup> iteración. Sinaturas

#### 5.2.4.1. Planificación e definición da iteración

Esta iteración comeza o día 1 de Xaneiro de 2016 e remata o día 8 e centrarse na última vista da aplicación, aquela que permite xestionar a finalización do encontro dando a posibilidade de asinar a acta e engadir comentarios por parte do árbitro.

Na iteración anterior detectáronse varias melloras a incluir como facer que por defecto as contas creadas dende a aplicación móvil sexan todas de tipo árbitro. Así mesmo facer que se engada a dito usuario como árbitro do encontro en tódalas actas que cree esa conta.

#### 5.2.4.2. Revisión e feedback

Ao finalizar esta iteración publicóuse unha nova entrada no blog do proxecto expoñendo a metodoloxía ágil utilizada para o desenvolvemento así como aquelas nas que se basea (Scrum e eXtreme Programming) e a metodoloxía de negocio orientada cara o cliente (Lean Startup).

Por último tamén se publicou o fluxo de traballo que se realiza para contribuir ao proxecto co sistema de control de versións Git, co fin de facilitar o traballo a futuros contribuidores.

#### 5.2.4.3. Tarefas e seguimento

Durante esta iteración realizáronse as seguintes tarefas e das cales se pode observar o seu diagrama de Gant na Figura 5.7.

**S5.1** Como árbitro quero poder asinar un acta.

**S5.2** Como árbitro quero que unha ou varias persoas convocadas de cada equipo poidan asinar un acta.

**S5.3** Como árbitro quero poder engadir comentarios a un acta.

**S5.4** Como árbitro quero poder borrar un xogador da lista de convocados.

Crear un árbitro ao crear un novo usuario na aplicación móvil.

Engadir o árbitro que ten o usuario asignado nas actas que crea.

Nesta iteración a planificación incial de 32 horas foi sobreestimada polo sobraron 5 horas de desenvolvemento que foron reasignadas a outro proxecto.

## 5.3. De empresa a comunidade (Xaneiro 2016 – Maio 2016)

Durante o mes de Xaneiro de 2016 decidíuse abandonar o proxecto de VACmatch como iniciativa empresarial por diversos motivos e continuar con él únicamente como proxecto comunitario de software libre.

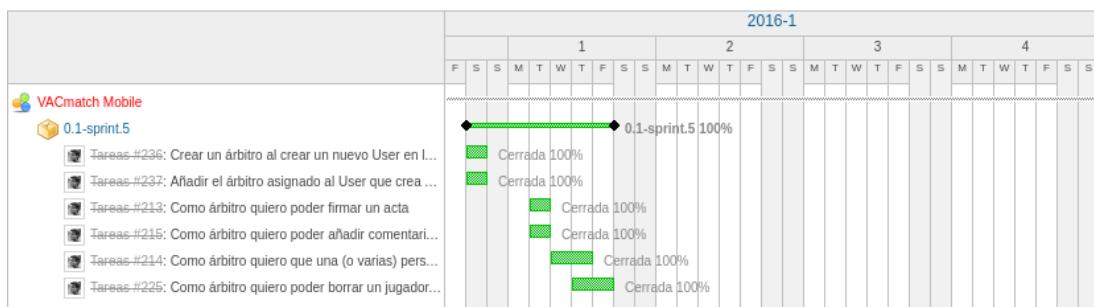


Figura 5.7: Diagrama de Gant do sprint 5

Motivado por isto, producíuse un parón de aproximadamente un mes de duración e ao mesmo tempo comezouse a traballar nunha empresa externa polo que durante este período diminuí considerablemente o tempo dispoñible para continuar co proxecto.

O número de horas total estimado para este período de 6 iteracións é de 189, todo nun total de 5 meses, o que supón unha media aproximada de 2 horas diarias de desenvolvemento.

### 5.3.1. 6<sup>a</sup> e 7<sup>a</sup> iteración. Optimización e melloras

Motivado pola inestabilidade xerada polos cambios mencionados anteriormente, non se realizou unha correcta planificación da 6<sup>a</sup> iteración e finalmente non foi posible realizar ningunha tarefa polo que se decidíu unir ambas iteracións.

#### 5.3.1.1. Planificación e definición da iteración

Estas iteracións comezan o 14 de Xaneiro de 2016 e rematan o 29 de Febreiro, a pesar de que non se realizou traballo efectivo ata as últimas dúas semanas de Febreiro.

Durante este periodo planificouse unha importante refactorización de código co fin de simplificar certas partes do código e facilitar o mantemento da aplicación así como revisar a forma na que se crean os identificadores dos obxectos en base de datos.

#### 5.3.1.2. Revisión e feedback

Como se comentou anteriormente, este foi un sprint marcado por un longo parón de aproximadamente un mes e medio durante o que o proxecto non avanzó nin se obtivo ningún tipo de feedback.

En cambio durante este tempo si se publicaron varias entradas no blog do proxecto co fin de poñer ao día de xeito público os últimos avances do proxecto e contar diversas decisións técnicas que se adoptaron como a selección da licencia para o proxecto, a implementación do sistema de eventos ou a sinatura das actas.

### 5.3.1.3. Tarefas e seguimento

Durante esta iteración realizáronse poucas tarefas, todas imputadas ao sprint número 7, pero de unha duración considerable como se pode observar no diagrama de Gant da Figura 5.8.

**S7.1** Refactorizar servicios.

**S7.2** Crear clases para cada entidade.

**S7.3** Revisar cómo se crean os identificadores dos obxectos en base de datos.

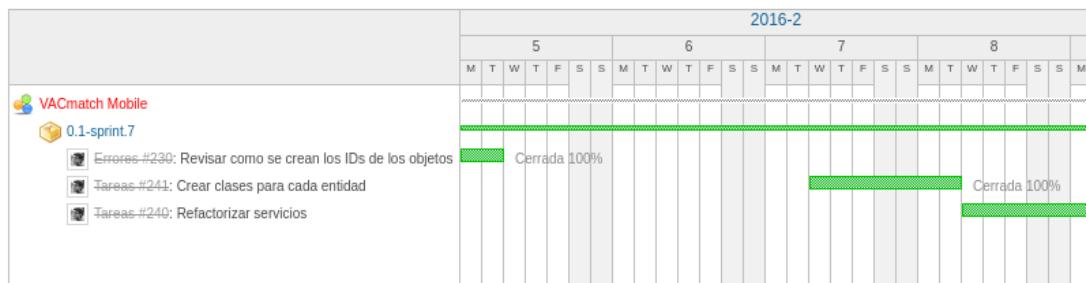


Figura 5.8: Diagrama de Gant do sprint 7

Para a realización de esta iteración planificouse un total de 22 horas e non se precisou ningunha hora extra para completar o traballo.

### 5.3.2. 8<sup>a</sup> iteración. Testing e integración continua

#### 5.3.2.1. Planificación e definición da iteración

A iteración transcorre entre os días 1 e 14 de Marzo.

Decidíuse engadir tests unitarios para previr futuros erros e facilitar o mantemento da aplicación xa que en todo proxecto de certa embergadura, e máis en proxectos libres nos que calquera pode colaborar, é importante asegurar que os novos cambios que se engadan, non xeneren problemas no funcionamento da aplicación.

Relacionado con este tema tamén se planificou a integración do repositorio de código con unha ferramenta de integración continua que facilite a execución de este tipo de probas, en este caso Travis CI.

#### 5.3.2.2. Revisión e feedback

Durante esta iteración foi preciso publicar unha folla de ruta do proxecto no blog do mesmo, requisito fundamental solicitado pola organización do Concurso Universitario de Software Libre no que VACmatch se atopa inscrito dende o mes de Outubro aproximadamente.

É por iso que se fixo unha entrada resumo para mostrar as tarefas existentes no Redmine do proxecto donde se realiza toda a xestión de incidencias así como se resaltou os seguintes pasos a seguir no mesmo como por exemplo engadir a internacionalización ou permitir poñer en funcionamento a aplicación nun dispositivo móbil.

### 5.3.2.3. Tarefas e seguimento

As seguintes tarefas son as realizadas en esta iteración:

**S8.1** Engadir tests aos servizos de Eventos, Persoas, Equipos e Actas.

**S8.2** Engadir tests aos servizos de Auth, Árbitros e Sinaturas.

**S8.3** Engadir integración continua con Travis CI.

**S8.4** Engadir campos para confirmar contrasinal e código PIN ao crear un usuario.

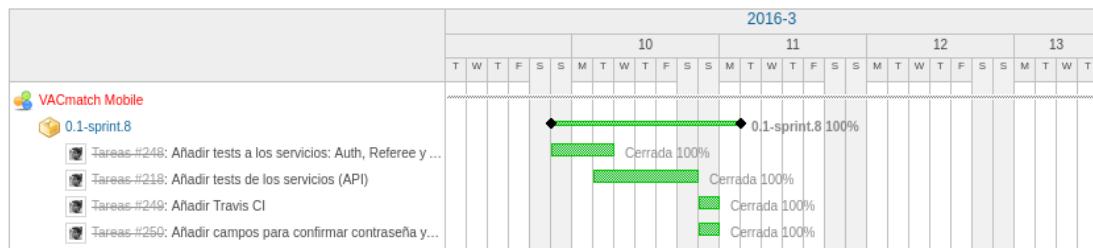


Figura 5.9: Diagrama de Gant do sprint 8

No diagrama da Figura 5.9 pódese observar a evolución das tarefas ao longo do sprint que inicialmente tivo unha planificación de 37 horas pero finalmente alongouse 16 horas extra, obligando a ampliar a xornada de traballo en 2 fines de semana a media xornada co obxectivo de non retrasar a execución das tarefas.

### 5.3.3. 9<sup>a</sup> e 10<sup>a</sup> iteración. Inxección de dependencias

Motivado de novo pola inestabilidade no que se dispón para realizar o proxecto, finalmente decidíuse integrar de novo estas dúas iteracións en unha.

#### 5.3.3.1. Planificación e definición da iteración

Estas iteracións comezan o día 15 de Marzo e rematan o 25 de Abril.

Detectáronse problemas en Travis xa que non detectaba errores nos tests polo que é o primeiro que había que corrixir.

Tamén se decidíu crear unha barra de notificacións compartida para todos os componentes da aplicación e engadíronse estados diferentes para as actas co fin de mostrar cando un encontro non comezou, cando se está a xogar e cando rematou o mesmo.

Pero a tarefa máis importante xurdíu ao aparecer un problema de dependencias circulares que obligou a engadir unha factoría para realizar unha inxección de dependencias entre os servizos da aplicación xa que varios, dependían uns de outros.

Finalmente planificouse tamén a corrección de diversos erros detectados na iteración anterior.

### 5.3.3.2. Revisión e feedback

Durante esta iteración non se publicou ningunha entrada no blog e tampouco se recibiron suxerencias.

### 5.3.3.3. Tarefas e seguimento

As tarefas definidas para esta iteración son as seguintes:

**S10.1** Engadir textos de error na aplicación.

**S10.2** Crear barra de notificacións para utilizar en calquera compoñente.

**S10.3** Erro: Correxir problema de dependencias circulares.

**S10.3.1** Engadir DAOs.

**S10.3.2** Engadir Inxección de Dependencias.

**S10.4** Erro: Mostrar que o partido remató na acta.

**S10.4.1** Engadir estados na Acta.

**S10.5** Erro: Como árbitro non debería poder convocar/borrar a un xogador que ten eventos asignados.

**S10.6** Erro: Correxir errores varios derivados de engadir inxección de dependencias.

**S10.7** Erro: Un evento de comezo de partido non se pode borrar si existe algún outro evento creado.

**S10.8** Erro na xestión do estado da Acta.

**S10.9** Erro: Travis CI non funciona correctamente

A estimación inicial de horas para esta iteración foi de un total de 78, resultando unha vez más optimista e permitindo que a diferencia de horas coas reais, 66, poidesen ser reasignadas ao proxecto de VACmatch Web.

### 5.3.4. Release 0.2.0: Usabilidade en menús

#### 5.3.4.1. Planificación temporal

Esta iteración transcorre entre o día 25 de Abril e o 9 de Maio.

---

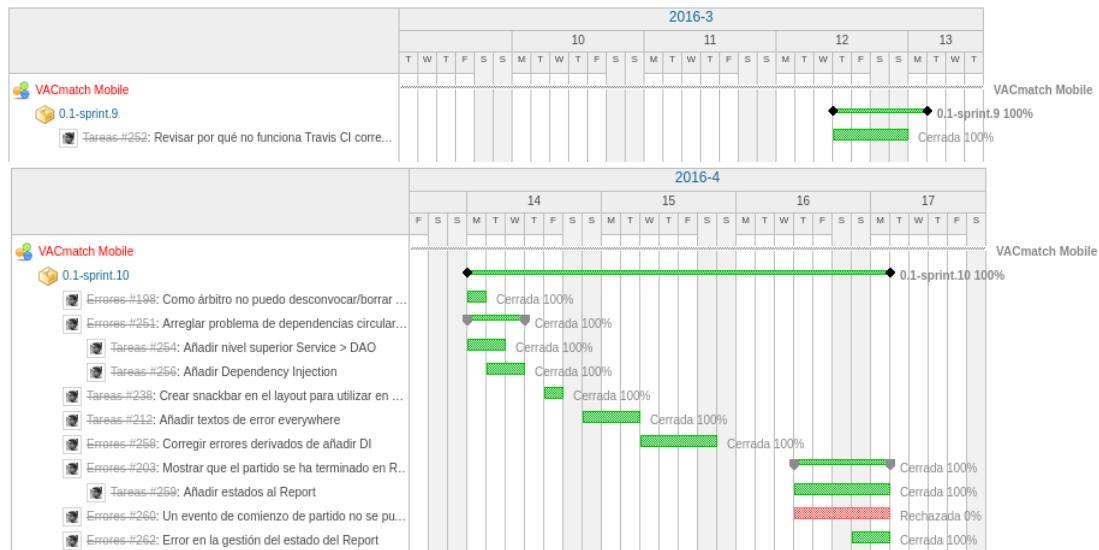


Figura 5.10: Diagramas de Gant dos sprints 9 e 10

#### 5.3.4.2. Definición da iteración

Unha iteración con unha sola tarefa pero de un tamaño suficiente para cubrir a totalidade do sprint, centrada en engadir os enlaces que faltaban no menú lateral esquerdo e no superior derecho, certos botóns de retroceso e corexidos diversos erros menores.

#### 5.3.4.3. Concurso Universitario de Software Libre

Así mesmo, durante este iteración recibíuse o “Premio al mejor proyecto de tecnologías móviles” do Concurso Universitario de Software Libre (CUSL), un concurso onde participaron máis de 75 estudiantes de toda España e donde se expuxeron 47 proxectos de código libre.

Fomos invitados a participar na fase final que tivo lugar os días 5 e 6 de Maio na Universidade de Sevilla na que presentar o proxecto ante representantes de diversas empresas de software libre españolas que tamén participaron con diversas charlas sobre os seus modelos de negocio e as vantaxes do software libre a nivel empresarial.

Foi unha gran experiencia a compartida con todos os finalistas e asistentes e, por suposto, os custes da viaxe foron subvencionados pola organización e o premio finalmente foi de 500 € en metálico.

#### 5.3.4.4. Revisión e feedback

#### 5.3.4.5. Tarefas e seguimento

- Revisar os links nos menús e engadir información.



Figura 5.11: Finalistas CUSL

### 5.3.5. Release 0.2.1: I18n e app híbrida

#### 5.3.5.1. Planificación temporal

Esta iteración comeza o día 9 e remata o 24 Maio.

#### 5.3.5.2. Definición da iteración

A tarefa de maior tamaño que se planificou en esta iteración foi a internacionalización coa librería React Intl xa que supón modificar todas as vistas da aplicación.

Posteriormente engadíuse Apache Cordova para permitir crear aplicacions híbridas que funcionen en diversos sistemas operativos móveis e por suposto, tamén se incluíu no repositorio de código, a documentación sobre cómo arrancar unha base de datos CouchDB para utilizar como backend e sobre cómo realizar a compilación da aplicación para executar nun sistema operativo móvil.

#### 5.3.5.3. Revisión e feedback

#### 5.3.5.4. Tarefas e seguimento

- Engadir internacionalización.
- Crear app híbrida con Apache Córdova.

### 5.3.6. Release 0.2.2: Imáxe corporativa e revisión de errores

Imaxe corporativa VACmatch, bugfix Memoria a saco

**5.3.6.1. Planificación temporal**

**5.3.6.2. Definición da iteración**

**5.3.6.3. Revisión e feedback**

**5.3.6.4. Tarefas e seguimento**

## Capítulo 6

# Fundamentos tecnolóxicos

### Índice general

---

6.1. Linguaxes e frameworks empregados . . . . .	42
6.2. Bases de datos . . . . .	43
6.3. Estándares de comunicación . . . . .	43
6.4. Repositorios de código . . . . .	44
6.5. Ferramentas de xestión . . . . .	44
6.6. Ferramentas documentais . . . . .	45

---

NESTE capítulo móstraranse as diversas tecnoloxías que foron empregadas durante o desenvolvemento do proxecto así como ferramentas de xestión, bases de datos, repositorios de código e ferramentas documentais, todas, tecnoloxías Software Libre.

### 6.1. Linguaxes e frameworks empregados

#### ReactJS [facebook.github.io/react](https://facebook.github.io/react)

React é unha librería de Javascript para a creación de Single Page Applications (SPAs), permitindo crear aplicacións completas que se executen no navegador de forma sinxela a través de diversos componentes que se agrupan e que permiten crear aplicacións multiplataforma.

#### Reflux [github.com/reflux/refluxjs](https://github.com/reflux/refluxjs)

É unha implementación da arquitectura Flux e que permite un fluxo de datos unidireccional, en lugar do bidireccional que é habitual nas aplicacións web. Permite tamén a creación de Stores nas que se mantién o estado das aplicacións e que permite compartir dito estado entre os diversos componentes da aplicación.

#### Jest [facebook.github.io/jest](https://facebook.github.io/jest)

É unha ferramente sinxela creada sobre o framework de testing para Jasvascript, Jasmine, que facilita a utilización de mocks e a creación de tests unitarios.

**React Intl (i18n)** [formatjs.io](https://formatjs.io)

React Intl é unha ferramenta para facilitar a internacionalización de aplicacíons Javascript e concretamente as aplicacíons baseadas en React.

**React Router** [github.com/reactjs/react-router](https://github.com/reactjs/react-router)

Unha librería para o enrutado de aplicacíons baseadas en ReactJS proveendo unha API sinxela con funcionalidades de gran potencia como a carga preguiceira de código ou o enrutado dinámico.

**Material UI** [material-ui.com](https://material-ui.com)

Un conxunto de compoñentes para React que implementan o Material Design impulsado por Google, unha nova linguaxe visual baseada na representación en 3D dos obxectos que non deben intersecarse se non que a través de sombras para simular diferentes profundidades, os obxectos debe superpoñerse uns sobre os outros.

## 6.2. Bases de datos

**CouchDB** [couchdb.apache.org](https://couchdb.apache.org) É unha base de datos pensada para web que permite almacenar os datos en formato JSON e acceder aos mesmos a través dun navegador via HTTP, funcionando como unha API REST (Representational state transfer).

Permite múltiples funcionalidades pouco habituais entre os sistemas de xestión de bases de datos como servir aplicacións directamente dende CouchDB así como un sistema de replicación incremental e de detección de conflictos.

**PouchDB** [pouchdb.com](https://pouchdb.com)

Unha base de datos NoSQL baseada en Javascript e inspirada en CouchDB, pensada para facilitar o funcionamento de aplicacións web de forma offline.

PouchDB permite almacenar os datos localmente no navegador web cando non hay conexión a internet e sincronizar de forma sinxela ditos datos en remoto con CouchDB e outros servidores compatibles.

## 6.3. Estándares de comunicación

**JSON** [couchdb.apache.org](https://couchdb.apache.org)

É un formato estándar para o intercambio de datos e que pola súa simplicidade estase a imponer como formato habitual por exemplo, para a comunicación con APIs Rest e debido a súa similitude coa definición de obxectos en Javascript, permite que sexa tremadamente sinxelo traballar con él dende esta linguaxe.

---

## 6.4. Repositorios de código

### Gerrit [gerritcodereview.com](http://gerritcodereview.com)

Gerrit é un repositorio de código baseado no sistema de control de versións Git e centrado en proveer un xeito sinxelo de realizar revisións de código dende unha plataforma web. Foi utilizado durante o desenvolvemento do prototipo da aplicación pero finalmente decidíuse trasladar o código a GitHub para facilitar a colaboración de outros posibles desenvolvedores.

### GitHub [github.com](http://github.com)

GitHub é un repositorio de código que se está a convertir no lugar máis importante de publicación de aplicacións Software Libre e que permite aloxar proxectos como o presente, de forma totalmente gratuita. Cómprase destacar que esta é a única ferramenta utilizada para o desenvolvemento do proxecto que non é software libre pero si proporciona unha visibilidade de cara a comunidade de gran importancia neste tipo de proxectos.

## 6.5. Ferramentas de xestión

### Git [git-scm.com](http://git-scm.com)

É un sistema de control de versións software libre con grandes funcionalidades e que é utilizado en millóns de proxectos. Aporta unha versatilidade enorme ao ser distribuído, permitindo traballar incluso de forma offline.

### Gulp [gulpjs.com](http://gulpjs.com)

Un sistema que permite a automatización de tarefas durante o desenvolvemento de aplicacións como por exemplo compilar automáticamente o código Javascript escrito na súa última versión á versión máis antiga para que poida ser executada por calquera navegador web.

### Babel [babeljs.io](http://babeljs.io)

É un compilador de Javascript que permite a traducir código fonte escrito no estándar ECMAScript 6 a ECMAScript 2015, soportado pola gran maioría de navegadores.

### Browserify [browserify.org](http://browserify.org)

É unha ferramenta que permite escribir os módulos da aplicación como se fosen módulos para unha aplicación escrita en Node.js e que os compila para poder ser utilizados no navegador web.

### Redmine [redmine.org](http://redmine.org)

É unha ferramenta de xestión de proxectos flexible, multiplataforma e software libre con diversos plugins para facilitar a planificación de iteracións e traballar con metodoloxías áxiles de desenvolvemento.

### Travis CI [travis-ci.org](http://travis-ci.org)

É unha ferramenta de integración continua que permite automatizar a execución de tests

---

ou o despregamento automático de código. Ademais dispón dunha integración con Github polo que resulta moi sinxelo automatizar estas tarefas.

**Docker** [docker.com](https://www.docker.com)

É un sistema que permite empaquetar e despregar de xeito sinxelo aplicacións coas súas dependencias en unidades estándar chamadas contenedores, abstraendo e automatizando a virtualización da plataforma na que correrá a aplicación.

**Atom** [atom.io](https://atom.io)

Un editor de texto software libre deseñado inicialmente por GitHub e centrado na súa extensibilidade gracias a un sinxelo sistema de plugins. Ademais, no ecosistema de ReactJS, existen compoñentes para Atom co obxectivo de facilitar a edición de aplicacións React, elaboradas polos mesmos impulsores da propia librería.

**Apache Cordova** [cordova.apache.org](https://cordova.apache.org)

Unha ferramenta de desenvolvemento que permite usar tecnoloxías web estándar (HTML, CSS3 e Javascript) para crear aplicacións móveis multiplataforma.

## 6.6. Ferramentas documentais

**LaTeX** [latex-project.org](https://www.latex-project.org)

Un sistema para a composición de documentos que inclúe todo tipo de funcionalidades para a edición de textos científicos ou técnicos, moi adecuado para este proxecto e que xenera documentos de xeito sinxelo e automáticamente estructurados.

**Dia** [dia-installer.de](https://dia-installer.de)

É unha aplicación para a creación de diagramas entre os que se atopan os diagramas UML e que permite a exportación dos mesmos a imáxenes vectoriais.

---

## Capítulo 7

# Diseño e implementación

## Índice general

---

<b>7.1. ReactJS e Flux . . . . .</b>	<b>47</b>
7.1.1. Introducción e elección da tecnoloxía . . . . .	47
7.1.2. Elementos básicos . . . . .	48
7.1.3. Estructura da aplicación . . . . .	50
<b>7.2. Bases de datos e funcionamiento offline . . . . .</b>	<b>51</b>
7.2.1. PouchDB . . . . .	52
7.2.2. CouchDB . . . . .	52
7.2.3. Sincronización e xestión de conflictos . . . . .	52
<b>7.3. App híbrida con Apache Cordova . . . . .</b>	<b>52</b>
<b>7.4. Interface gráfica e usabilidade . . . . .</b>	<b>54</b>
7.4.1. Elementos comúns . . . . .	54
7.4.2. Iniciar sesión . . . . .	59
7.4.3. Listado de actas . . . . .	59
7.4.4. Acta . . . . .	60
7.4.5. Finalización do encontro . . . . .	63
7.4.6. Modificación de estilos . . . . .	64
<b>7.5. Multideporte . . . . .</b>	<b>65</b>
7.5.1. Deporte . . . . .	65
7.5.2. Roles de usuarios . . . . .	66
7.5.3. Eventos . . . . .	66
<b>7.6. Deseño da DB . . . . .</b>	<b>68</b>
<b>7.7. I18n . . . . .</b>	<b>69</b>
<b>7.8. Inxección de dependencias . . . . .</b>	<b>69</b>
<b>7.9. Testing e Integración continua . . . . .</b>	<b>70</b>
7.9.1. TDD e BDD . . . . .	71
7.9.2. Jest . . . . .	71
7.9.3. Travis CI . . . . .	74

---

NESTE capítulo veremos os detalles de deseño e implementación de diversas partes do proxecto e falaremos das decisións tomadas con respecto a utilización das múltiples tecnoloxías utilizadas.

Comezaremos facendo unha pequena introducción á terminoloxía e ás tecnoloxías utilizadas no proxecto na Sección 7.1, xa que resulta fundamental para comprender as diversas explicacións técnicas sobre o deseño e a implementación.

Falaremos tamén das bases de datos non relacionais na Sección 7.2, comentaremos a necesidade da creación da aplicación como App híbrida e todo isto seguido dunha extensa análise sobre a usabilidade e as múltiples decisións que se tomaron ao respecto, debido a gran importancia que ten este campo dentro do proxecto.

Por último tamén comentaremos algúns conflitos que xurdíu durante o desenvolvemento a nivel de dependencias de paquetes ou testing e as decisións que se tomaron para solventalos.

## 7.1. ReactJS e Flux

### 7.1.1. Introdución e elección da tecnoloxía

Actualmente o mundo das aplicacíons móbiles está en pleno crecemento e cada vez é máis sinxelo ver cómo pequenos comercios ou incluso eventos de ocio como festivais de música ou conferencias, teñen a súa propia aplicación móvil<sup>1</sup>.

A sociedade está a eliminar un soporte tradicional como é o papel, en tódolos aspectos da vida, dende a publicidade ata a propia xestión do traballo, e todo estase a dixitalizar, facilitando o traballo humano e reducindo os custes a medio prazo.

Neste contexto decidíuse optar por realizar unha aplicación móvil híbrida, con tecnoloxías web, xa que se pode observar un movemento nos últimos anos cara este tipo de tecnoloxías que permiten desenvolver únicamente unha aplicación e executala nos diversos sistemas operativos móbiles existentes, sen ter que desenvolver unha aplicación concreta para cada un de eles.

Tras unha análise das diversas linguaxes e frameworks que podían ser utilizados para este fin, finalmente optouse por utilizar ReactJS pola sua flexibilidade fronte a outras alternativas como AngularJS ou EmberJS.

Estas alternativas céntranse en ofrecer un framework moi completo buscando cubrir todos os aspectos de unha aplicación como o enrutado de urls, a internacionalización ou os servizos, fronte a React que únicamente trata de cubrir a xestión das vistas e do seu estado, dando total liberdade para escoller a tecnoloxía que se precise para o resto de compoñentes da aplicación.

Así mesmo tamén cómpre destacar a existencia de React Native, unha tecnoloxía baseada en React que permite crear aplicacíons móbiles con tecnoloxías web e con unha experiencia de

---

<sup>1</sup>Eventos como o FOSDEM, a LibreCon ou OpenExpo xa dispoñen de aplicacíons móbiles para facilitar os asistentes organícen as actividades ás que asistirán.

usuario exactamente igual a unha aplicación nativa tradicional.

Unha vez seleccioado React, optamos por seguir a arquitectura máis habitual dentro de este tipo de aplicacóns, Flux, a través da librería Reflux.

### 7.1.2. Elementos básicos

#### 7.1.2.1. Compoñentes de React

React é unha librería de Javascript que nos permite xestionar de xeito sinxelo as vistas da nosa aplicación a través de diversos elementos denominados compoñentes.

React permítenos escribir os nosos compoñentes con unha sintaxis moi similar a HTML pero posteriormente traduce esta sintaxis a código javascript habitual polo que podemos decir que estamos escribindo páxinas web únicamente con funcións de javascript.

A idea principal é reutilizar e agrupar compoñentes para formar o que tradicionalmente chamamos vistas e que a súa vez poden ter un estado. Este estado pode variar modificando as vistas, como por exemplo cando se engade un novo gol na lista de eventos e React encárgase de volver a renderizar únicamente a parte da vista que cambiou, polo que é tremendaente eficiente.

#### 7.1.2.2. Arquitectura Flux

Flux [Bod16] é unha arquitectura que básicamente propón o esquema mostrado na Figura 7.1.

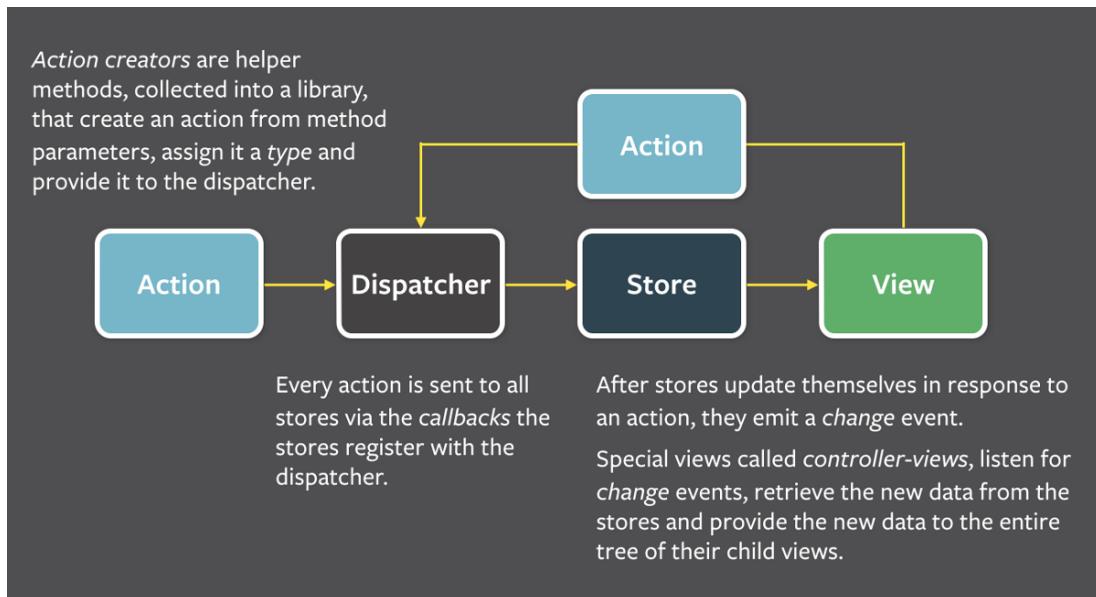


Figura 7.1: Esquema da arquitectura Flux

Unha aplicación que opte por seguir esta arquitectura debe conter os seguintes compoñentes:

**7.1.2.2.1. Actions** A vista atópase formada por unha serie de componentes de React que son capaces de disparar as Actions, por exemplo ao pulsar un botón, e que funcionan de xeito similar a eventos, notificando ao Dispatcher cando se produce a súa execución.

**7.1.2.2.2. Dispatcher** O Dispatcher é o encargado de recibir e enrutar as Actions disparadas cara as Stores.

**7.1.2.2.3. Stores** Estas Stores son as encargadas de xestionar o estado dos componentes e toda a lóxica necesaria para actualizar o estado co novo cambio.

Se por exemplo ao pulsar un botón debería cambiar un texto que está no componente, poderíamos ter este texto almacenado no estado dentro dunha Store e inicializado a un valor baleiro. Unha vez se fai click no botón, unha Action chamará a unha Store que é a que contén a información sobre cómo cambiar o texto do estado e React encargarase de volver a pintar o novo valor do estado, dentro do seu componente.

### 7.1.2.3. Implementación de Flux. Reflux.

Flux é unha arquitectura, unha especie de patrón de desenvolvemento pero existen diversas implementacións da mesma e neste proxecto decidímonos por utilizar Reflux [?].

O funcionamento é moi similar pero con algúns matices; por exemplo, non existe un único Dispatcher central que enruta as Actions se non que todas as Stores están escutando e reaccionan cando teñen un método para responder a Action.

Vexamos un exemplo práctico:

Listing 7.1: Exemplo de Action.

```
let TextActions = Reflux.createActions([
    "updateText"
])
```

Definiríamos unha Action (Fragmento de código 7.1). que queremos lanzar para tratar de actualizar o texto e crearíamos unha Store (Fragmento de código 7.2) que escute as TextActions e que implemente unha función onUpdateText que será a que reciba o novo valor e actualizará o estado.

Listing 7.2: Exemplo de Store.

```
let TextStore = Reflux.createStore({
    listenables: TextActions,
    init: function () {
        this.state = ''
    },
    getInitialState: function () {
        return this.state
    },
})
```

---

```
onUpdateText: function (newText) {
  this.state = newText
  this.trigger(this.state)
}
})
```

---

Por último definir o compoñente (Fragmento de código 7.3). Este deberá incluir unha lista de mixins onde indicarlle cales son as Stores que reaccionan ante eventos lanzados dende este compoñente, neste caso, a TextStore.

O compoñente tamén terá un método render no que definir o que poderíamos chamar informalmente a “vista” do que queremos renderizar, e que inclúe un enlace para cambiar o texto mostrado.

Dito enlace reacciona cando se fai click e chama ao método handleChangeTextClick que lanza a Action para que a Store a intercepte, cambie o estado e React renderice de novo, únicamente o que cambiou.

Listing 7.3: Exemplo de compoñente de React.

```
let EndReport = React.createClass({
  mixins: [
    Reflux.connect(TextStore, 'text')
  ],
  handleChangeTextClick: function () {
    TextActions.updateText('Nuevo texto a renderizar')
  },
  render: function () {
    return (
      <p>{this.state}</p>
      <a onClick={this.handleChangeTextClick}>Change text</a>
    )
  }
})
```

---

### 7.1.3. Estructura da aplicación

A continuación móstranse os diversos elementos xerais que contén a aplicación, as carpetas principais e certos ficheiros fundamentais na execución da aplicación.

**tests** Carpeta onde se poden atopar os tests da aplicación. De momento só se dispón de tests da capa de servizos.

**i18n** Nesta localización pódese atopar a internacionalización, coas cadeas dentro da subcarpeta de “messages“ e coa descripción das mesmas para cada compoñente de React dentro de “descriptors“, coa idea de darlle contexto e facilitarlle ao tradutor a súa labor.

**cordova app** Aquí podemos atopar a configuración para compilar a aplicación aos diversos dispositivos móbiles.

**src/app** Estrutura principal da aplicación

**Actions** Lista de ficheiros con accións a disparar, agrupados cada un pola store que o implementa.

**API** Contén a configuración, as urls da aplicación e a factoría utilizada para a inxección de dependencias dos servizos que explicaremos máis adiante.

**Components** Lugar onde se poden atopar todos os componentes que forman as vistas da aplicación.

**Daos** Entidades que abstraen dos servicios a definición do acceso a base de datos, facilitando que sexa sinxelo cambiar dita base de datos se é preciso.

**Models** Contén a definición dos modelos da aplicación, tanto as entidades que se almacenan en base de datos como de certas clases que son utilizadas pola aplicación, como por exemplo as que definen os deportes ou a implementación dos tipos de eventos.

**Services** Estas clases conteñen a lóxica de negocio da aplicación.

**Stores** Lista de ficheiros que conteñen e xestionan o estado da aplicación. Cómpre diferenciar a lóxica de negocio, que se almacena nos servizos, da xestión do estado dos componentes de React que podemos atopar nas Stores.

**app.js** Ficheiro de inicialización da aplicación que se encarga de arrancar o enrutador e executa a aplicación ben en modo web ou ben en modo aplicación móvil en función da configuración definida.

**router.jsx** É o encargado de poñer en relación as urls da aplicación cos componentes que corresponden a cada unha e onde se introduce tamén, a información sobre a internacionalización.

## 7.2. Bases de datos e funcionamento offline

O mundo das bases de datos está a cambiar enormemente nos últimos anos coa aparición das bases de datos non relacionais [Tiw11], neste caso concreto, era preciso dispoñer de unha base de datos no cliente que permitise almacenar os datos das actas dos encontros de forma offline xa que o árbitro do encontro, pode non ter cobertura e debería poder cubrir a súa acta da mesma forma.

Ante este requisito cabe pensar en bases de datos lixeiras como SQLite que se utilizan habitualmente nas aplicacións móbiles pero ditas bases de datos non poden ser executadas directamente nun navegador web, polo que finalmente se optou por utilizar PouchDB, unha base de datos orientada a documentos e creada sobre o GlobalStorage<sup>2</sup> do navegador, pensada dende o primeiro momento para crear aplicacións web que funcionen de forma desconectada.

<sup>2</sup>Un obxecto que mantén múltiples áreas de almacenamento privado para almacenar datos durante un largo período de tempo

### 7.2.1. PouchDB

As actas electrónicas son documentos que unha vez sexan creadas, apenas serán modificadas e neste tipo de información, son as bases de datos orientadas a documentos as que presentan un mellor rendemento.

A elección de PouchDB [?] foi sobre todo motivada por ser tremadamente lixeira, multianvador e sobre todo, facilitar a sincronización contra unha base de datos remota, neste caso, CouchDB.

### 7.2.2. CouchDB

CouchDB [?] é a base de datos non relacional que se utiliza como base de datos central en remoto, e coa cal sincronizará PouchDB que se atopa no navegador web do cliente.

Esta base de datos é tremadamente versatil e sinxela de utilizar xa que implementa unha API REST e, a forma de interactuar con ela basease simplemente en enviar documentos JSON a través de sinxelas peticións HTTP cara a súa API.

Destaca pola súa extensa comunidade e por ser altamente dispoñible e tolerante a erros pero eventualmente inconsistente, áinda que foi crítico na súa elección a xestión que CouchDB fai dos conflictos de datos e que comentamos a continuación

### 7.2.3. Sincronización e xestión de conflictos

Probablemente o maior reto á hora de enfrentarse a unha aplicación con funcionamento offline é a xestión de conflictos entre os datos.

En este caso en concreto, existe a problemática de que por múltiples motivos, un acta pode ser modificada en dous lugares ao mesmo tempo, tanto polo árbitro no seu teléfono como pola persoa encargada da xestión da federación polo que resulta imprescindible non perder información e ser capaces de mostrarlle a totalidade da información ao xestor da federación para que este poida seleccionar cales datos son os reais.

Neste punto é donde CouchDB resulta moi útil xa que a propia base de datos se encarga de almacenar unha árbore co histórico de revisíons que se producen sobre un documento e así se poderían mostrar ao usuario para que seleccione a correcta en caso de conflicto.

## 7.3. App híbrida con Apache Cordova

O obxectivo do proxecto é poder utilizalo en múltiples dispositivos móbiles co fin de facilitar que calquera árbitro poida utilizala independentemente do sistema operativo que teña o seu teléfono ou tableta.

É por isto que é fundamental a utilización dun sistema que permita este desenvolvemen-

to multiplataforma como é o caso de Apache Cordova [?] que, ademáis, facilita o acceso aos elementos do dispositivo como sensores, datos, estado de rede... a través dunha serie de APIs estandar.

Actualmente non se está a utilizar ningunha de estas funcionalidades pero si é posible que nun futuro cercano se decidan implementar novas características que si precisen o acceso a este tipo de elementos como pode ser por exemplo, o micrófono, para que o árbitro poida gardar as incidencias dun partido como notas de voz en lugar de escribelas.

Na Figura 7.2 podemos ver un esquema da arquitectura dunha aplicación sobre Apache Córdova.

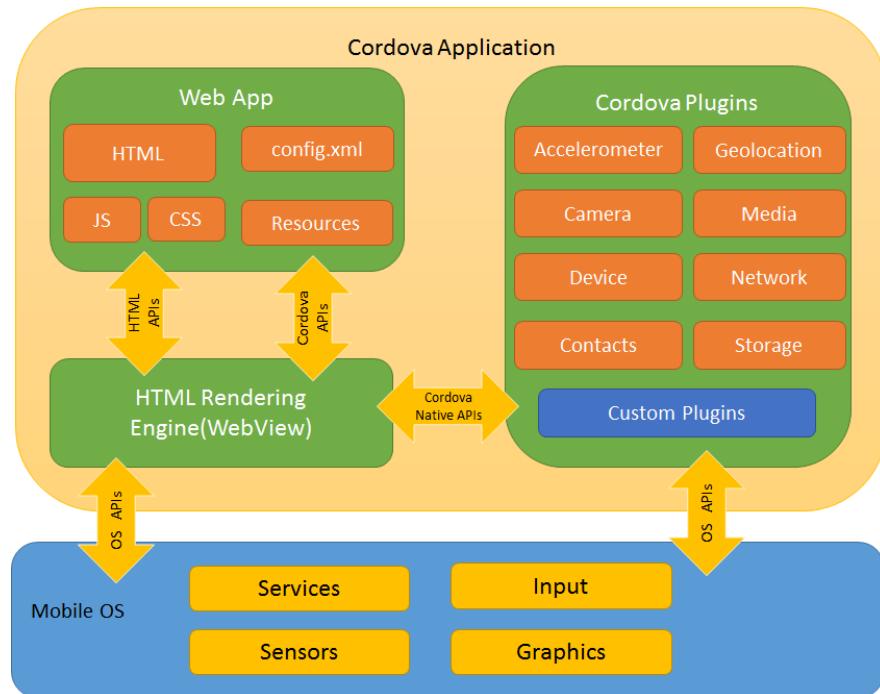


Figura 7.2: Arquitectura de unha aplicación con Apache Cordova

Básicamente a aplicación executase como unha aplicación web normal sobre un WebView, un motor de renderizado de HTML que pode interactuar coas APIs nativas do dispositivo a través dunha serie de plugins que Cordova provee.

Para executar a aplicación únicamente é preciso crear un proxecto de Cordova, engadir os ficheiros da aplicación, as plataformas para as que se desexe xerar o proxecto e construílo.

## 7.4. Interface gráfica e usabilidade

A importancia da interface gráfica en este proxecto é crucial, debido a que a aplicación será utilizada por persoas de un amplísimo rango de idades, a usabilidade é o valor máis importante.

A nivel de deseño a aplicación separase en catro grandes bloques:

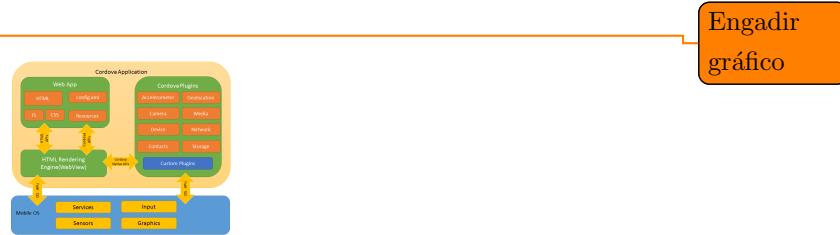


Figura 7.3: Estructura da aplicación gráficamente.

### 7.4.1. Elementos comúns

Para facilitar o desenvolvemento da aplicación creáronse varios compoñentes de React xenéricos que permiten realizar tarefas comúns a múltiples partes da aplicación, e polo tanto, que son utilizados dende outros compoñentes de React.

#### 7.4.1.1. Menú lateral esquierdo

É o menú principal da aplicación e que se mostra ao pulsar no botón superior esquierdo dende a maior parte de vistas.

Como se pode ver na Figura 7.4, este menú desplegable mostra actualmente o logotipo de VACmatch e unha serie de enlaces entre os que se atopan as páxinas de configuración e a páxina "acerca de".

Para xestionar os enlaces que se mostran en cada páxina de forma internacionalizada, e para permitir dispoñer dos elementos nun sitio centralizado, creouse o compoñente `LeftMenuItem`. Este compoñente contén varias listas con items para introducir dentro do menú lateral esquierdo e que se poden editar ou engadir outras se é preciso.

Utilízase tamén a `MenuStore` para xestionar os elementos que se atopan actualmente no menú e pódese utilizar a Action `setLeftMenu` para modificar os elementos que o compoñen en calquera momento, por exemplo, ao entrar en unha nova vista.

#### 7.4.1.2. Enlaces do menu superior dereito

O funcionamento de este menú é moi similar ao anterior só que este non dispón de compoñente propio e únicamente é preciso chamar a Action `setRightMenu` para colocar a listaxe cos novos

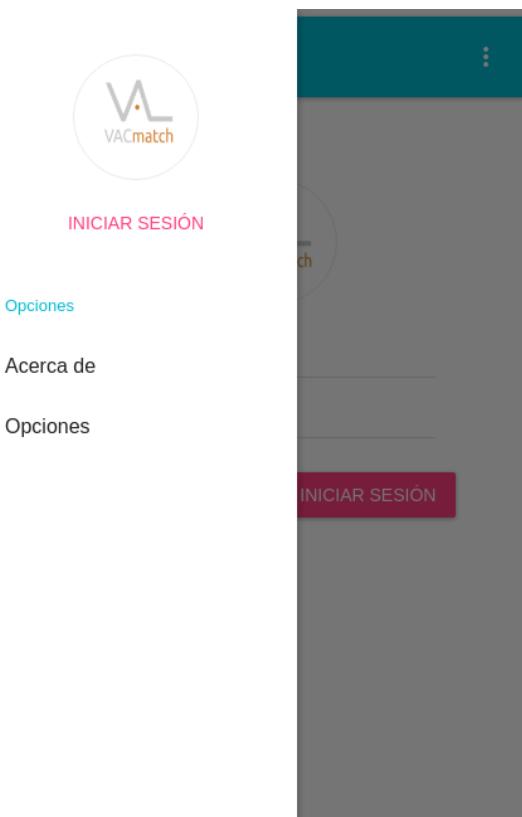


Figura 7.4: Menú lateral esquierdo.

elementos a mostrar no menú desplegable como se ve na Figura 7.5

A función debe recibir unha lista de obxectos que conterán os seguintes atributos:

**text** Texto a mostrar polo item.

**url** Url a onde ir ao facer click no elemento.

**callback** Función de callback que se executará unha vez se redirixa a aplicación á url indicada (Opcional).

#### 7.4.1.3. Información e axustes

Toda aplicación debe conter un apartado de información sobre a mesma, e en este caso móstrase ademáis as diversas redes sociais do proxecto e o repositorio de código en GitHub para que calquera poida contribuir ao mesmo.

Por outro lado tamén se ten en conta que toda aplicación debe ser personalizable en certa medida, de momento non se dispón de opcións máis ca cambiar de idioma pero cando exista a necesidade de incorporar novos elementos de configuración, este será o apartado onde os usuarios poderán modificar ditas opcións.

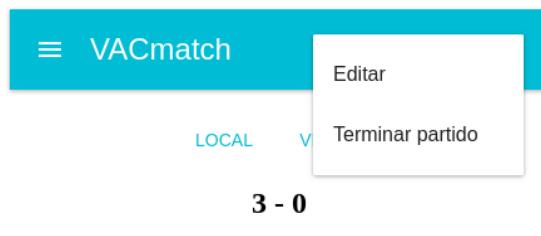


Figura 7.5: Menú desplegable dereito.

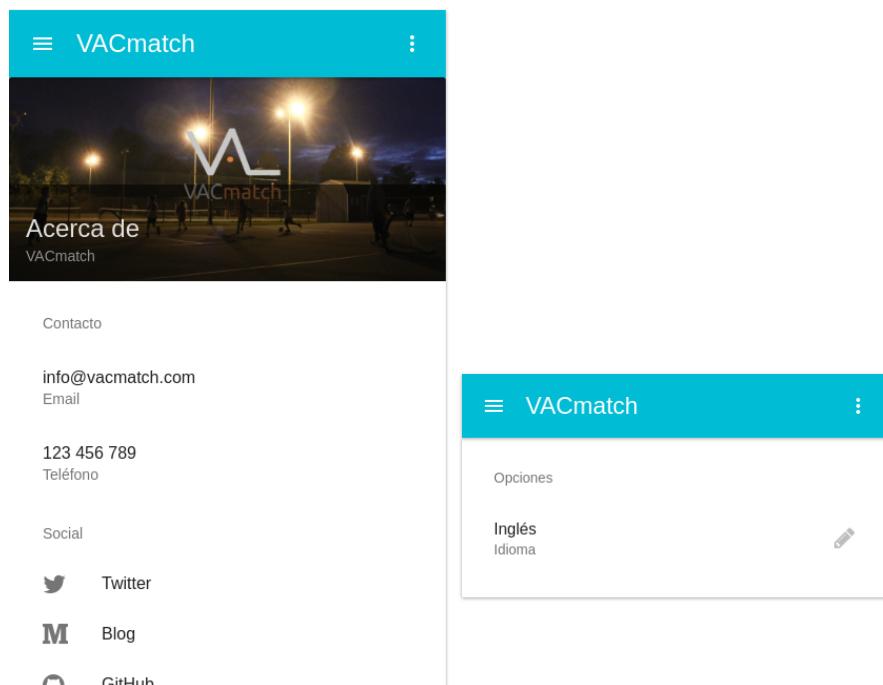


Figura 7.6: Ventás de información e axustes.

#### 7.4.1.4. Barra de notificacións

Inicialmente xurdíu a necesidade de comunicarlle ao usuario os errores que se producen na aplicación, casos concretos como introducir un usuario incorrecto ou tratar de eliminar un evento de comezo de partido antes de eliminar o de fin do mesmo, deben mostrar un aviso ao usuario.

É por iso que se decidíu crear unha “Barra de notificacións” (Figura 7.7), unha pequena ventá que xurde a modo de aviso na parte inferior da pantalla durante uns segundos para mostrar información.

Inicialmente pensouse para mostrar os errores pero tamén resulta de utilidade a hora de mostrarlle outra información ao usuario como cando un evento é engadido correctamente.

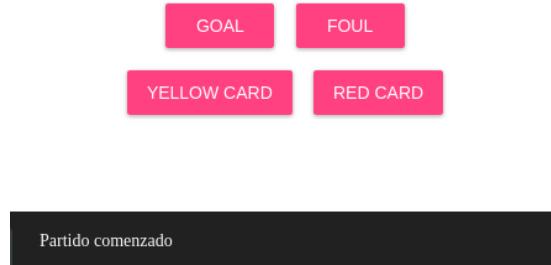


Figura 7.7: Barra de notificacións.

As novas mensaxes son almacenadas na `SnackBarStore` creada para o caso, que permite diferenciar as mensaxes de erro e as de información, facendo que a implementación da barra sexa moi sinxela como se mostra no Fragmento de código 7.4.

Listing 7.4: Barra de notificacións.

```
let SnackBarItem = React.createClass({
  mixins: [
    Reflux.connect(SnackBarStore, 'snackBar')
  ],
  onStatusChange: function (status) {
    this.refs.snack.show()
  },
  componentDidMount: function () {
    this.listenTo(SnackBarStore, this.onStatusChange)
  },
  render: function () {
    return <Snackbar key={'generic-snackbar'}
      ref='snack'
      message={this.state.snackBar.message}
      autoHideDuration={4000} />
  }
})
```

#### 7.4.1.5. Autenticación

A autenticación é fundamental en esta aplicación xa que únicamente os usuarios con acceso poden crear ou modificar actas, neste contexto foi preciso crear un compoñente de autenticación que se encargue de realizar a comprobación de se o usuario iniciou sesión na aplicación sempre que se carguen os compoñentes.

Creouse unha clase sinxela que deben extender aqueles compoñentes que só poidan ser accedidos se o usuario está autenticado, o “AuthenticatedComponent” (Fragmento de código 7.5) e que redirixe a aplicación cara páxina de iniciar sesión, no caso de non poder atopar un usuario

activo na sesión.

Listing 7.5: Compoñente de autenticación.

```
export default (ComposedComponent) => {
  let AuthenticatedComponent = React.createClass({
    mixins: [
      Reflux.connect(AuthStore, 'auth'),
      History
    ],
    componentWillMount: function (transition) {
      if (config._env !== 'development') {
        // This method is called before transitioning to this component.
        // If the user is not logged in, we'll send him or her to the Login page.
        if (!AuthStore.isLoggedIn()) {
          this.history.pushState(null, urls.login.show)
        }
      }
    },
    render: function () {
      return (
        <ComposedComponent
          {...this.props} />
      )
    }
  })
  return AuthenticatedComponent
}
```

Ao mesmo tempo, este compoñente permítenos aportar unha funcionalidade diferente en caso de atoparnos en modo “desenvolvemento”, saltando a comprobación de se o usuario ten sesión iniciada e permitíndonos acceder a calquera páxina sen ter que iniciar sesión cada vez que se recarga a páxina.

#### 7.4.1.6. Lista de pestanas

Durante o desenvolvemento percatámonos de que é habitual a necesidade de utilizar unha lista de elementos divididos en varias pestanas, tanto á hora de mostrar a lista de xogadores dos equipos como á hora de asinar un acta ou engadir novos eventos.

É por isto que se creou un compoñente xenérico que permite incorporar unha lista de pestanas e unha lista de elementos para cada unha de elas, xenerando de xeito sinxelo e xenérico estas vistas. Pódense introducir en elas calquera calquera tipo de elemento sempre que se encontre envolto dentro de un tipo xenérico definido como Item.

#### 7.4.2. Iniciar sesión

Na Figura 7.8 podemos atopar a páxina inicial da aplicación que permite a un usuario iniciar sesión cos seus datos de acceso así como, pulsando o botón de rexistro, crear un novo usuario dentro da aplicación que lle permita xestionar actas de xeito manual.

Entre os múltiples parámetros que o usuario pode cubrir, atópase un código PIN que lle permitirá asinar as actas dos encontros que arbitre.

Cómpre mencionar que, á parte das páxinas de configuración da aplicación e de “acerca de”, esta é a única vista accesible para usuarios sen autenticar e calquera intento de acceso a algunha do resto de páxinas, redirecionará ao usuario cara esta páxina de inicio de sesión.

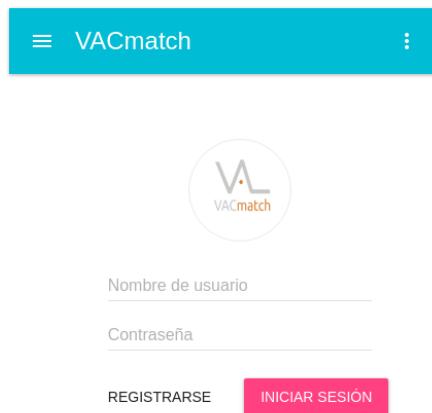


Figura 7.8: Vista de inicio de sesión.

#### 7.4.3. Listado de actas

Nesta sección podemos ver a lista de actas que un árbitro ten asignadas na Figura 7.9, divididas en dúas pestanas, por un lado as de próximos partidos a arbitrar e por outro aquellas nas que o encontro xa rematou e que non é habitual que sexan accedidas.

Cada un dos elementos permítenos modificar a información básica da acta, tanto o nome dos equipos como o lugar de celebración ou a data por se é preciso edita ditos datos, permitindo do mesmo xeito eliminar a acta e tódolos elementos que a componen.

Ademais, esta páxina danos a opción de engadir unha nova acta se é preciso, unha funcionalidade imprescindible xa que por múltiples razóns poderíase dar o caso de que a aplicación de

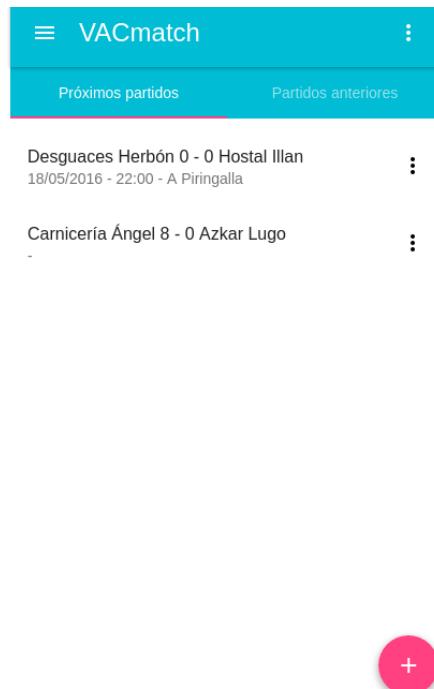


Figura 7.9: Listado de actas.

un árbitro non sincronice as súas actas coas existentes na base de datos da federación, polo que o árbitro debe poder crear un acta baleira de xeito manual incluso sen ter conexión a internet.

#### 7.4.4. Acta

Esta é a parte central da aplicación, con total seguridade será o lugar onde os usuarios pasarán a gran mayoría do seu tempo de uso da aplicación xa que é o punto central do desenvolvemento dun partido.

A acta dun encontro deportivo é un elemento con gran cantidad de información e que pode ser visualizada sen apenas esforzo xa que se atopa toda representada únicamente en unha folla de papel.

É por iso que dende o primeiro momento se tiña claro que había que tratar de manter esa sinxeleza de uso pero, evidentemente, era imposible acceder a toda esa información en un só golpe de vista dentro de un soporte de un tamaño tan pequeno como é un teléfono móvil.

Entón o enfoque foi un pouco diferente e tratouse de facer que o proceso de cubrir un acta resultase o máis interactivo posible, mostrando na pantalla central únicamente a información indispensable para o árbitro que está a cubrila.

O deseño final pódese ver na Figura 7.10 na que se mostran os seguintes campos:

- Os nomes dos equipos que están a competir con un enlace para ver os seus xogadores.

- Dous campos de resultados.
- Un cronómetro.
- Un botón para comezar o encontro e xestionar o cronómetro unha vez comezou.
- Un botón de edición
- Un botón para mostrar os eventos que ocorreron.
- Botóns para engadir un dos diversos eventos que poden ser utilizados para este encontro



Figura 7.10: Vista principal de un acta.

#### 7.4.4.1. Convocar xogadores

Nas accións habituais que debe realizar un árbitro, o primeiro paso é cubrir no papel os nomes dos xogadores que se atopan presentes no encontro coas fichas identificativas que cada un de eles aporta.

Como o obxectivo é eliminar o papel e simplificar o traballo do árbitro e da federación, a acta do encontro contén tódolos xogadores que se atopan inscritos no equipo correspondente coa sua foto como se pode ver na Figura 7.11, polo que xa non son precisas as fichas en papel de cada un de eles e o árbitro non ten que escribir os nomes dos integrantes dos equipos, se non que únicamente debe seleccionar cales de eles se atopan no campo.

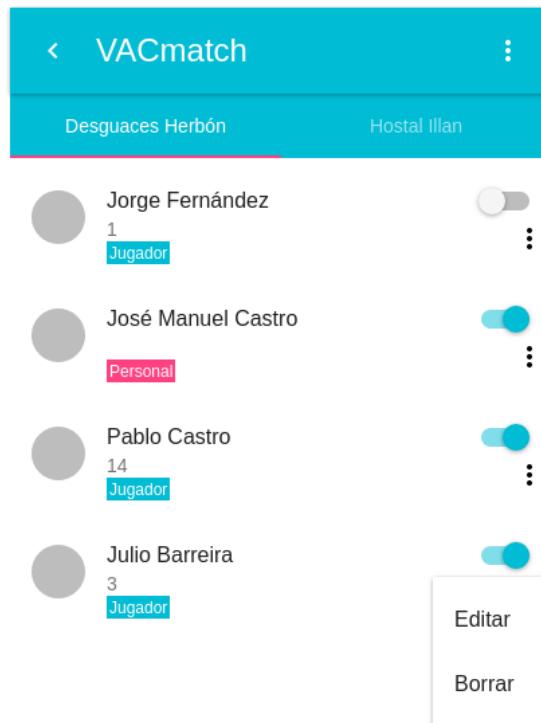


Figura 7.11: Seleccionar xogadores presentes no encontro.

Tamén ten a posibilidade de editar o dorsal do xogador para este encontro en concreto xa que en algunhas competicións é habitual que os xogadores cambien de camiseta segundo o partido.

Por último existe a opción de engadir un novo xogador de xeito manual, útil para o caso no que o árbitro teña que crear un acta manualmente ou para cando un novo xogador é engadido a un clube pero non é dado de alta na aplicación a tempo, e se o árbitro accede, este pode ser engadido a acta de xeito manual e competir sen problemas.

#### 7.4.4.2. Inicio e fin do partido

O seguinte paso á hora de xestionar a acta dun encontro é indicar cando comeza o mesmo, para isto o usuario dispón de un botón que só se mostra cando non comezou aínda dito partido e que se encarga de introducir un evento de comezo.

No momento en que se comece o partido, atoparase dispoñible o botón que permite arrancar e deter o **cronómetro**, co fin de poder xestionar o encontro de xeito interactivo e que tódolos eventos introducidos leven incorporado o momento no que se produciron.

Non é obligatorio utilizar o cronómetro polo que os eventos poden ser engadidos en calquera momento e non se almacenará o minuto no que se produciron.

Do mesmo xeito, pódese engadir un evento de fin de encontro nunha das opcións desplegables do menú superior dereito e que redirixirá ao árbitro a páxina de asinado de acta.

#### 7.4.5. Finalización do encontro

Cando un árbitro decide rematar un encontro, accederá a esta última vista da aplicación que podemos ver na Figura 7.12 onde poderá engadir un texto coa información de posibles **incidencias** que ocorreran durante o encontro e que deben ser postas en coñecemento da federación, como algúin tipo de agresión ou o motivo polo que un partido tivo que ser finalizado antes de tempo.



Figura 7.12: Fin de encontro.

Dende o menú superior dereito pódese acceder directamente aos eventos que ocorreron durante o partido. Isto resulta útil para mostrarlle as persoas que asinan a acta toda a información da acta para que poidan confirmar que realmente os eventos gardados na mesma, son certos.

A vista ten varias pestanas, unha para o árbitro e outra para cada un dos equipos, onde poderán asinar a acta tantas persoas como sexa preciso.

Unha vez se faga click no botón de asinar de algunha das pestanas, mostrárase unha listaxe coas persoas que poden asinar de cada equipo ou do equipo arbitral como podemos ver na Figura 7.13.

Para realizar a sinatura, a persoa debe dispoñer de conta creada na base de datos da federación na que debeu de indicar previamente código PIN que lle permitirá asinar a acta.

No caso de xogadores creados manualmente dende a propia aplicación móvil, poderase asinar a acta sen dispoñer de código PIN, deixando o oco baleiro.

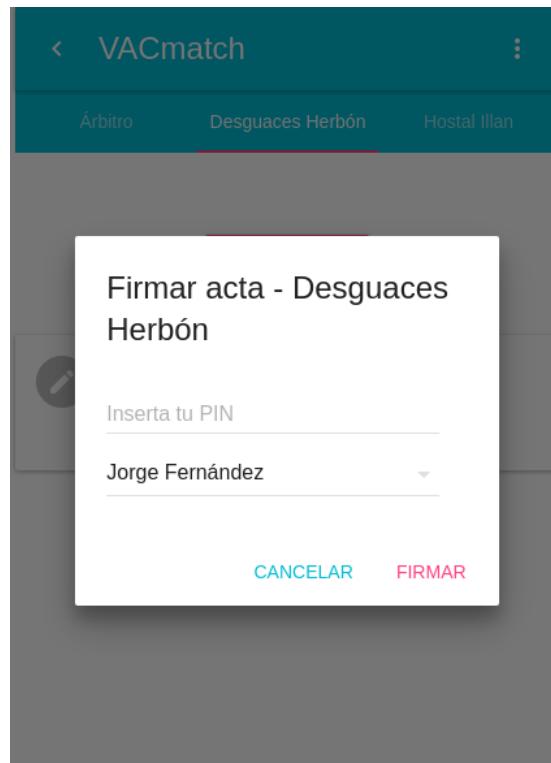


Figura 7.13: Sinatura de un acta.

#### 7.4.6. Modificación de estilos

A xestión de estilos na aplicación varía de xeito considerable fronte ás aplicacións web habituais polo que é importante facer unha pequena reseña.

Certa parte da comunidade de React defende que o estilo das aplicacións web, que tradicionalmente se xestionava a través de follas CSS, debe mudar e pasar a realizarse directamente en Javascript.

Os estilos son inxectados directamente no compoñente de xeito “inline” a través do atributo “style” de HTML<sup>3</sup>, coa idea de acabar con diversos errores de deseño en CSS, algúns dos cales se comentan a continuación.

O feito de que os estilos sexan código global accesible dende calquera parte das aplicacións tradicionais, fai que esta sexa pouco consistente ante os cambios, unha pequena modificación pode cambiar gran parte da aplicación e pode ser difícil de atopar dito cambio cando os proxectos adquiren un certo tamaño polo que a xestión de xeito máis localizado a través de variables Javascript, facilitaría este mantemento.

A eliminación de código morto tamén é outra das vantaxes de utilizar Javascript para este

<sup>3</sup>A diferencia da maneira tradicional na que os estilos se componen en clases CSS a través do atributo “class” e pódense anidar unhas con outras de forma declarativa

propósito xa que, agora os estilos serán variables locales que os “minifiers”<sup>4</sup> de Javascript eliminarán automáticamente.

Por último tamén aporta unha total flexibilidade, dando incluso a posibilidade de tratar os estilos como parte do estado da aplicación e convertilos de este xeito, en totalmente dinámicos.

## 7.5. Multideporte

Un dos obxectivos iniciais era conseguir que a aplicación poidera adaptarse de xeito moi sinxelo a novos deportes, inicialmente comezaríase co fútbol pero todo o deseño debía estar orientado cara esta finalidade.

### 7.5.1. Deporte

Para iso creouse unha `sportStore` que contén a información do deporte que a aplicación está xestionando actualmente, e que pode ser actualizado e cambiado por outro, de xeito moi sinxelo.

Dende múltiples compoñentes como por exemplo o `PersonList`, pódese obter o deporte actual que está almacenado na Store e as múltiples propiedades que o definen, como se pode ver na Figura 7.14.

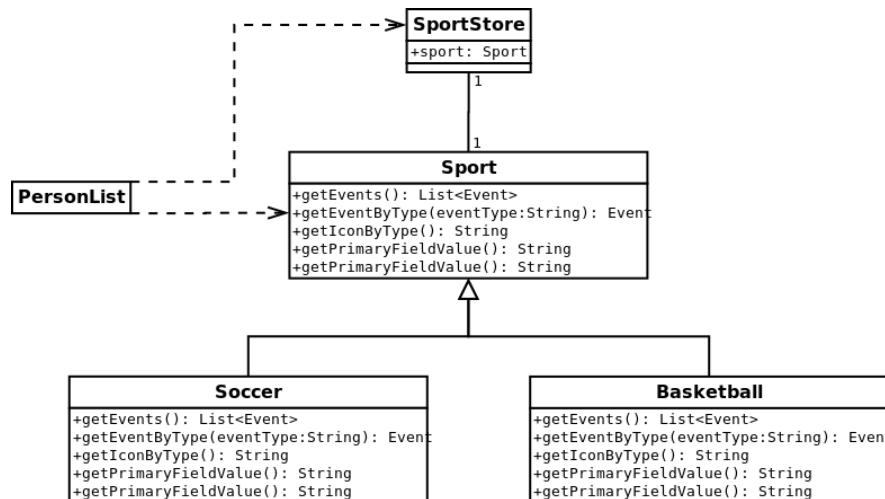


Figura 7.14: Patrón estratexia para o deporte.

Cada deporte debe extender a unha clase xenérica `Sport` nun patrón Strategy [GHJV94] que se pode observar na Figura 7.14 e que define unha serie de métodos que tódolos deportes deben implementar, métodos que indican cómo se realizan certas accións sobre os datos en cada deporte.

A continuación móstranse os parámetros que ten un `Sport` actualmente:

<sup>4</sup>Programas que eliminan código e caracteres inecesarios como os espacios ou tabulacións

**Eventos** A lista de eventos que se poden realizar en ese deporte.

**Evento por tipo** Permite recuperar un obxecto Evento a partir da clave do seu tipo.

**Icono por tipo** Devolve o icono que identifica un Evento de un tipo para renderizalo en algúnha parte da aplicación como a lista de eventos.

**Campo primario** É unha función que calcula o resultado para almacenar no campo primario da acta en función da lista de eventos para ese encontro e de cada deporte. En fútbol devolve únicamente a suma de eventos de tipo gol.

**Campo secundario** Similar ao campo anterior pero para introducir información no campo secundario da acta, no caso do fútbol, o campo secundario almacena as faltas do encontro e esta función devolve a suma de faltas que hay no partido.

Esta implementación fai que sexa tremadamente sinxelo engadir un novo deporte, básicamente debería crear unha nova clase que extendese os métodos correspondentes coa funcionalidade concreta de ese deporte.

### 7.5.2. Roles de usuarios

Actualmente a aplicación únicamente pode ser accedida por árbitros, usuarios con un rol concreto, pero resulta trivial engadir novos roles, como por exemplo o de administrador de unha federación, para que este poida crear tamén encontros dende a aplicación móvil.

### 7.5.3. Eventos

Do mesmo xeito, cada deporte ten os seus propios eventos polo que se definiu unha forma de engadir eventos novos de xeito sinxelo, co fin de que a incorporación de un novo deporte resultase o máis trivial posible.

Actualmente podemos dividir os eventos en dous tipos:

**De control** Son eventos xenéricos de xestión como cambiar de parte, ou comenzar un encontro, que non é habitual que vaian a cambiar xa que son compartidos pola inmensa maioría de deportes.

**De deporte** Son eventos concretos de cada deporte como engadir un gol ou unha tarxeta, a súa vez divídense en:

**Evento** É un tipo de evento de deporte que únicamente permite ser engadido ou eliminado, non dispón de un comportamento especial.

**Evento con causa** Este tipo de evento, engade tamén unha causa pola que se produciu, útil para indicar por exemplo a motivación que levou a un árbitro a expulsar a un xogador con unha tarxeta vermella.

---

**Evento de puntuación** Tipo de evento que permite actualizar os campos de resultados, primario e secundario de forma automática ao ser engadido.

Na Figura 7.15 podemos ver a relación existente entre a `EventStore` que contén o estado dos eventos e os componentes da vista que os renderizan.

Temos dous componentes de React para renderizar os eventos dentro da vista que lista tódolos eventos que suceden durante o partido, os componentes `SportEvent` e `Event`.

Por outra banda temos tamén dous componentes de React que definen a vista que se ten que mostrar cando se deseñe engadir un evento de este tipo dende a aplicación, son `Event` e `EventWithCause` que ademais mostrará unha lista de causas que poden ser engadidas ao evento.

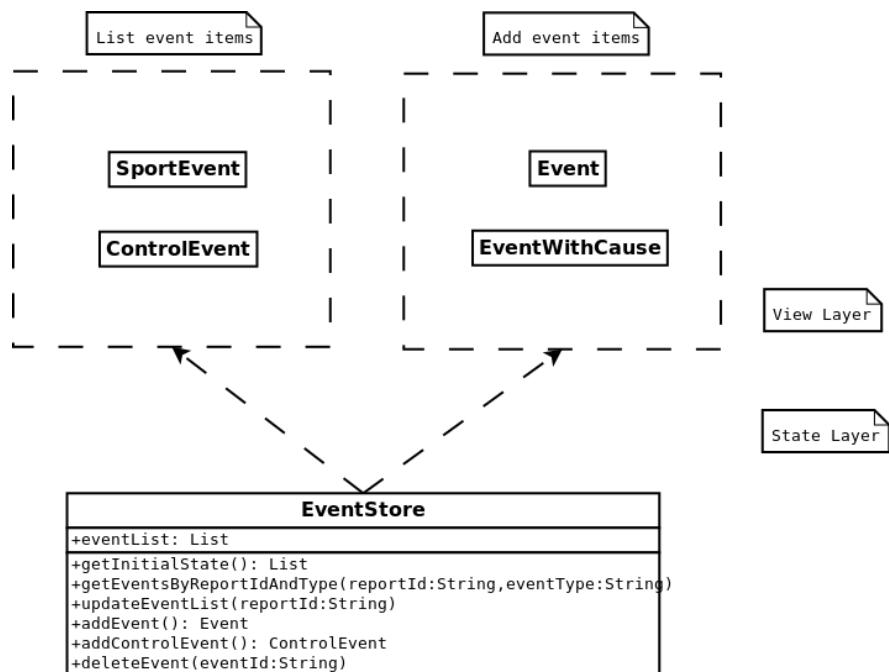


Figura 7.15: Diagrama parcial da vista dos eventos.

En definitiva, para crear un novo evento débese definir unha clase Javascript coas propiedades do mesmo e que implemente ao `SportEvent` como se pode ver na Figura 7.16 a través dos exemplos de `GoalEvent` ou `FoulEvent`.

Tamén é preciso engadir unha clase de React que defina as vistas que se mostrarán ao intentar engadir e listar un evento durante o encontro ou utilizar as xa definidas previamente que se poden ver na Figura 7.15.

Por último, como podemos observar de novo na Figura 7.16, débese engadir o evento creado anteriormente, dentro da lista de eventos que ten ese deporte, e que se atopan na clase que implementa dito deporte concreto.

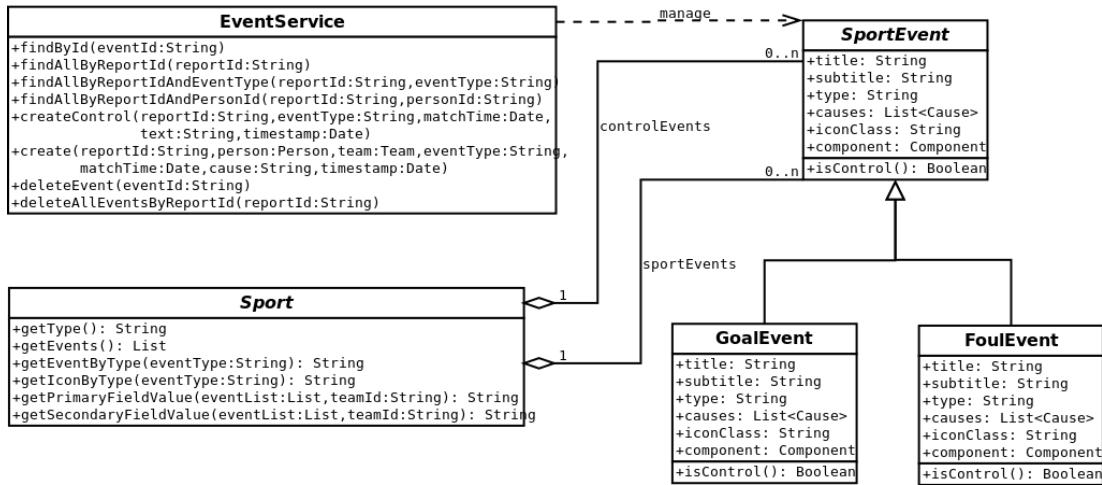


Figura 7.16: Diagrama de clases para a xestión de eventos.

## 7.6. Deseño da DB

VACmatch Mobile pensóuse inicialmente para ser integrado co sistema de xestión de competicións que estamos a desenvolver con VACmatch Web e que dispón dunha API de comunicacións que se atopa montada sobre unha base de datos relacional.

É por iso que o modelo de datos inicial estaba plantexado para unha base de datos relacional e tivo que ser adaptado para utilizar unha non relacional orientada a documentos como é PouchDB.

Prácticamente todas as entidades foron desnormalizadas para incluir información necesaria e en algúin caso comprimíronse dúas entidades en unha soa como por exemplo en **Call**<sup>5</sup> e **Member**<sup>6</sup> que foron unidas dentro de **Person** como se pode ver na Figura 7.17

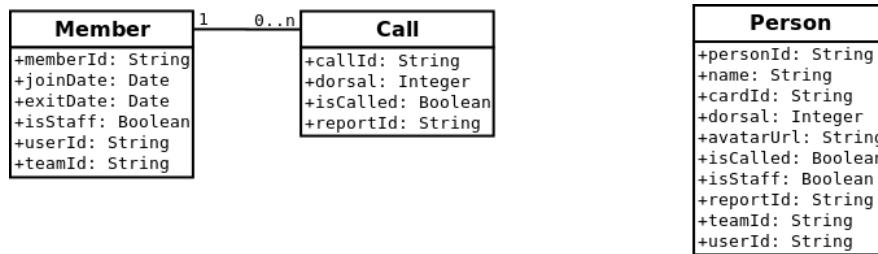


Figura 7.17: Esquema base de datos de Person.

<sup>5</sup>Convocatoria de un xogador para un partido en concreto

<sup>6</sup>Membro de un equipo

## 7.7. I18n

Para a xestión da internacionalización da aplicación utilizouse a librería React Intl que provee dunha serie de compoñentes de React e unha API sinxela para formatear datas, números e strings, incluindo pluralización e xestión de traducións.

Esta librería dá a posibilidade de engadir un contexto aos textos que se internacionalizan, o cal facilita a labor do tradutor que se encargue de engadir ou modificar un idioma, proporcionándolle unha información extra sobre o texto que debe traducir.

Únicamente é preciso definir os compoñentes cos seus correspondentes atributos como se mostra no Fragmento de código 7.6 e a propia ferramenta xenera ficheiros *.json* seguindo o esquema de carpetas e ficheiros da aplicación con toda a información recopilada.

Listing 7.6: Exemplo de internacionalización na label de un botón.

```
<RaisedButton label={  
    <FormattedMessage  
        id='report.show.events'  
        description='Events button'  
        defaultMessage='Events' />  
} secondary={true} style={style.button}/>
```

Do mesmo xeito, a librería permite facer internacionalización de xeito programático sen utilizar compoñentes de React, a través de unha serie de funcións de Javascript.

Por último, é preciso realizar as traduccions aos diferentes idiomas, cada unha en un ficheiro de Javascript diferente, seguindo un formateado de pares cadea/valor.

## 7.8. Inxección de dependencias

Durante o realización do proxecto xurdíu problema entre os servizos xa que moitos precisan dos outros para realizar comprobacóns de forma circular.

A existencia de estas dependencias circulares leva a que sexa imposible de determinar a relación entre eles polo que é preciso deseñar un pequeno sistema de inxección de dependencias que permite solventar esta problemática.

A solución escollida finalmente consiste en crear unha factoría seguindo o patrón Singleton [GHJV94] que se encarga de crear un obxecto para cada servizo e a continuación inxéctanse uns dentro dos outros.

Cada vez que se chama a factoría para recuperar un servizo, esta devolve o servizo coas súas dependencias inxectadas no seu interior polo que resolvemos o problema de xeito sinxelo.

Na Figura 7.7 pódese ver parcialmente o código que implementa a factoría que se encarga de crear os servizos e devovelos cando é preciso.

Listing 7.7: Fragmento da ServiceFactory.

```
let ServiceFactory = {
    isInitialized: false,
    _servicesList: new Map(),

    constructor () {
        this._personService = new PersonService()
        this._reportService = new ReportService()
        this._refereeService = new RefereeService()
        // Without dependencies
        this._teamService = new TeamService()
        // Inject dependencies
        this._eventService.ReportService = this._reportService
        this._eventService.PersonService = this._personService
        this._eventService.TeamService = this._teamService
        ...
        // Add services to the exposed list
        this._servicesList.set('ReportService', this._reportService)
        this._servicesList.set('PersonService', this._personService)
        this._servicesList.set('TeamService', this._teamService)
        this.isInitialized = true
    },

    getService (type) {
        if (!this.isInitialized) {
            this.constructor()
        }
        let service = this._servicesList.get(type)
        if (!service) {
            console.log('Error getting ', type, ' service')
        }
        return service
    }
}
```

Cómpre tamén comentar as dificultades que implica a inxección de dependencias ao realizar testing unitario xa que non se poden aproveitar as funcionalidades que aporta Jest, o framework de testing, que habitualmente permite automatizar a creación de mocks e, en este caso, obríganos a crear manualmente os mocks dos servizos e inxectalos dentro da clase a testear.

## 7.9. Testing e Integración continua

O testing é unha parte fundamental nun proxecto e concretamente nun proxecto de software libre no que a comunidade pode colaborar e onde poden participar persoas sen un coñecemento moi profundo da aplicación, polo que se fai totalmente imprescindible garantizar que calquera cambio non rompa a integridade da mesma.

### 7.9.1. TDD e BDD

Neste proxecto seguironse diversas prácticas de Test Driven Development (TDD) [Mar08] e Behaviour Driven Development (BDD) [WH12] comezando por unha definición das diversas tarefas do proxecto como tests de aceptación das funcionalidades e rematando pola realización de tests unitarios.

**7.9.1.0.1. Tests de aceptación** Estos tests son destinados a determinar se foron cumplidos os requisitos dunha certa funcionalidade, sempre centrándose na parte funcional e alonxándose dos detalles de implementación.

Para isto utilízanse unha linguaxe simple de dominio co fin de definir os requisitos funcionais, en este proxecto baseámonos no seguinte:

**Como ....** afectado ou realizador da funcionalidade.

**Quero ...** acción a realizar.

**Cando ...** momento ou caso no que debe realizarse.

Un exemplo de test de aceptación:

*Como árbitro quero que o resultado se actualice automáticamente ao engadir un novo gol.*

**7.9.1.0.2. Tests unitarios** Estos tests céntranse en comprobar o funcionamiento de xeito illado dos diversos sistemas e facendo fincapé en que cada proba sexa un caso totalmente independente do resto.

En este proxecto crearonse tests unitarios centrados nos servizos da aplicación, que son os elementos más importantes da mesma xa que son os que definen a lóxica de negocio.

Para a realización de este tipo de tests é fundamental a utilización de mocks, obxectos que imitan o comportamento de obxectos reais de xeito controlado e que permiten simular o comportamento dos obxectos dependentes.

Isto é fundamental para asegurar o illamento da funcionalidade e eliminar a dependencia de outros elementos a hora de realizar as probas unitarias.

### 7.9.2. Jest

É unha librería para realización de testing automático en aplicacóns ReactJS que facilita automatizar a creación de mocks ou executar os tests con unha implementación falsa do DOM<sup>7</sup>.

No Fragmento de código 7.13 pódese observar un test de exemplo do módulo de eventos e os diversos compoñentes que o forman.

---

<sup>7</sup>Modelo en Obxectos para a Representación de Documentos

Inicialmente defínese dentro da sección `describe` unha "historia de usuario" que contén a información xeral sobre o caso de uso, dentro da cal pódense realizar diversos tests, no exemplo que vemos, o caso de uso é "Crear un evento deportivo".

Listing 7.8: Definición de unha historia de usuario en Jest.

```
describe('create Sport Event', function () {  
  ...  
})
```

Dentro do caso de uso vemos un bloque `beforeEach` que permite inicializar variables para cada novo test que se realice, eliminando de este xeito a dependencia entre os tests que as utilizan.

Dentro do bloque `describe` defínense os diversos tests desta historia de usuario, concretamente dentro dos bloques `it`.

Listing 7.9: Definición de un test en Jest.

```
it('Create a new Sport Event with valid parameters', function () {  
  ...  
})
```

Como comentábamos anteriormente, ao realizar tests unitarios é habitual a utilización de obxectos mock que imitan a funcionalidade de outros. En Jest todos os obxectos son mocks e non é preciso definilos todos así. Cambiando un pouco a forma de traballar habitualmente, aquí débese definir ao comezo do ficheiro aqueles que non van a ser mockeados a través da sentencia definida no Fragmento de Código 7.10.

Listing 7.10: Sentencia para indicar que non se creen un mock.

```
jest.dontMock('../src/app/models/report/status/StartedStatus')
```

Tamén é habitual querer darlle un comportamento por defecto aos mocks que se utilizan, por exemplo no Fragmento de Código 7.11 estamos facendo que cando se chame a función `findById` do obxecto `reportService` co un só parámetro, o obxecto devolverá un callback cos parámetros indicados, neste caso unha Acta por defecto (`defaultReport`) e un valor nulo como segundo parámetro.

Listing 7.11: Sentencia para asignar comportamento a un mock.

```
spyOn(reportService, 'findById').andCallFake(function (anyReportId, callback) {  
  callback(defaultReport, null)  
})
```

Unha vez definido o comportamento que queremos que teñan os mocks, debemos chamar ao método real do obxecto que estamos testeando e así teremos totalmente illado o comportamento de dito método.

Finalmente utilizamos a cláusula `expect` para indicar resultados esperados, no caso de exemplo que mostramos a continuación, estamos a indicar que a chamada ao método `findById` de

`reportService` foi realizada. Tamén é posible indicar valores que debe devolver, negacións ou outras validacións incluso más complexas.

Listing 7.12: Sentencia para comprobar a execución de un test.

```
expect(reportService.findById).toHaveBeenCalled()
```

A continuación mostramos o caso de test anterior completo a modo de exemplo.

Listing 7.13: Exemplo de test no módulo de eventos utilizando Jest.

```
describe('create Sport Event', function () {
  beforeEach(function () {
    defaultPerson = new Person(null, '', '', '', '', false, false, '', '', '')
    defaultTeam = new Team(null, 'Team name')
    defaultReport = new Report(null, '', '', ReportStatus.READY,
      defaultTeam, defaultTeam, [])
    defaultEvent = new EventElements.Event('event', '1', defaultPerson,
      defaultTeam, 'goal', 1, 'cause', 1)
    reportService = new ReportService(jasmine.createSpy('PersonService'),
      jasmine.createSpy('TeamService'), jasmine.createSpy('EventService'),
      jasmine.createSpy('SignService'))
    personService = new PersonService(jasmine.createSpy('ReportService'),
      jasmine.createSpy('TeamService'), jasmine.createSpy('AuthService'))
    teamService = new TeamService()
    eventService = new EventService(reportService, personService,
      teamService)
  })

  it('Create a new Sport Event with valid parameters', function () {
    spyOn(reportService, 'findById').andCallFake(function(anyReportId,callback) {
      callback(defaultReport, null)
    })

    spyOn(personService, 'findByPersonIdReportIdAndTeamId').andCallFake(function
      (anyPersonId, anyReportId, anyTeamId, callback) {
      callback(defaultPerson, null)
    })

    spyOn(teamService, 'findById').andCallFake(function (anyTeamId, callback) {
      callback(defaultTeam, null)
    })

    spyOn(EventDao, 'create').andCallFake(function (reportId, person, team,
      eventType, matchTime, cause, timestamp, callback) {
      callback(defaultEvent, null)
    })

    eventService.create(defaultEvent.reportId, defaultEvent.person,
      defaultEvent.team, defaultEvent.type, defaultEvent.matchTime,
      defaultEvent.text, defaultEvent.timestamp,
      (event, err) => {
        expect(event).toEqual(defaultEvent)
        expect(event).not.toBe(null)
      })
  })
})
```

```
    expect(err).toBe(null)
  })

expect(reportService.findById).toHaveBeenCalled()
expect(personService.findByIdReportIdAndTeamId).toHaveBeenCalled()
expect(teamService.findById).toHaveBeenCalled()
expect(EventDao.create).toHaveBeenCalled()
}))})
```

---

### 7.9.3. Travis CI

A medida que a aplicación foi medrando, xurdíu a necesidade de simplificar traballos como a búsqueda de errores ou o mantemento da aplicación, buscando garantir que se mantén a integridade da mesma en todo momento e non se introducen errores polo que se decidíu engadir tests unitarios na aplicación.

Da mesma maneira lanzóuse unha primeira versión estable do proxecto polo que tamén se decidiu engadir un sistema de integración continua ao mesmo co fin de garantizar que a aplicación xerada na rama de producción é funcional, compila en todo momento e pasa tódolos tests.

Para isto utilizóuse Travis CI como sistema integración para o que se engadíu un ficheiro *.travis.yml* no que se lle indican diversos parámetros de configuración, como as versións de Javascript coas que debe poñer a proba o funcionamento do programa, os scripts a executar para instalar dependencias e para executar o programa, as ramas de desenvolvemento sobre as que debe actuar, no noso caso `master` e `development` ou as notificacións que debe enviar ao finalizar un traballo, por exemplo a través de correo electrónico ou utilizando un servizo de mensaxería instantánea como Slack.

Nesta integración xurdíu un problema derivado do sistema de construcción utilizado no proxecto, Gulp, que nos axuda a automatizar tarefas como a compilación, a execución dos tests ou o deploy da aplicación, e derivado tamén do framework de tests, Jest.

Cando este último se executaba a través de Gulp, non devolvía os errores ao resolver os tests como resultado dunha función, se non que o facía a través de un callback. É por iso polo que foi preciso engadir na tarefa "test" definida en Gulp, o control de este caso para evitar falsos positivos na execución dos tests unitarios dende Travis.

# Capítulo 8

## Conclusíons e traballo futuro

### Índice general

---

<b>8.1. Recoñecementos</b> . . . . .	<b>75</b>
8.1.1. Finalista no Certamen de Proyectos Libres da UGR . . . . .	75
8.1.2. Premio Universitario de Software Libre . . . . .	76
<b>8.2. Traballo futuro</b> . . . . .	<b>76</b>
8.2.1. Melloras de desenvolvemento . . . . .	76
8.2.2. Creación de comunidade . . . . .	77
<b>8.3. Conclusíons</b> . . . . .	<b>77</b>

---

NESTE capítulo contaremos as conclusíons obtidas da realización do proxecto, os recoñe-  
mentos obtidos polo mesmo e as liñas de traballo futuro que temos plantexadas.

### 8.1. Recoñecementos

Durante a realización do proxecto tamén se participóu en varios certames e concursos, non só coa idea de obter recoñecementos se non tamén co obxectivo de difundir o proxecto e buscar colaboradores para a comunidade de software libre que estamos a crear.

#### 8.1.1. Finalista no Certamen de Proyectos Libres da UGR

O día 6 de Xuño celebróuse na Facultade de Informática da Universidade de Granada (UGR) a final do “Certamen de Proyectos Libres” que organiza a Oficina de Software Libre de dita universidade dende xa fai varios anos.

VACmatch Mobile convertíuse en un dos 7 finalistas e foi invitado a participar presentando o proxecto ante un xurado composto por diversos profesionais do sector, presentación que se realizou en video ao non poder asistir ao evento personalmente.

### 8.1.2. Premio Universitario de Software Libre

Participouse no Concurso Universitario de Software Libre, un certame no que competiron ata 75 persoas con máis de 40 proxectos de software libre de todo España e no que VACmatch Mobile foi premiado.

Durante o mes de Maio celebróuse na Facultade de Informática da Universidade de Sevilla a fase final nacional na que fomos invitados xunto cos outros tres finalistas, a participar e expoñer o noso proxecto ante diversos profesionais e empresas do sector.

Finalmente recibimos o premio ao Mellor Proxecto para Dispositivos Móbiles con unha remuneración de 500 € en metálico e por suposto todos os gastos do desplazamento e estancia durante a fase final foron cubertos pola organización do concurso.

## 8.2. Traballo futuro

Como calquera aplicación actual, este proxecto non é un proxecto totalmente concluído xa que certos requisitos iniciais non foron aínda implementados e múltiples deportes poden ser engadidos.

Tamén gracias a utilización de metodoloxías áxiles de desenvolvemento, ao longo do proxecto surdiron novas propostas, das cales algunas non foron aínda implementadas e forman parte de liñas de traballo futuro.

### 8.2.1. Melloras de desenvolvemento

**8.2.1.0.1. Creación de unha versión de demostración** Sería interesante engadir unha funcionalidade para realizar entregas continuas que permita automatizar a posta en produción dunha versión do programa como demostración para posibles federacións interesadas en coñecer o funcionamento da aplicación.

**8.2.1.0.2. Resolución de conflictos** Actualmente cando un elemento se modifica ao mesmo tempo en dous lugares a vez (un acta na federación e no campo, por exemplo), os conflictos non son resoltos se non que se almacenan ambas versións para escoller cal sería a correcta, polo que sería interesante proveer aos usuarios de unha interfaz para resolver ditos conflictos de xeito sinxelo.

**8.2.1.0.3. Integración con VACmatch Web** O punto forte da aplicación sería poder integrar o sistema de xestión de competicións de VACmatch coa aplicación móvil co fin de simplificar aínda en maior medida o traballo do árbitro e do xestor da federación deportiva.

**8.2.1.0.4. Módulo para a xestión multifederación** Outra funcionalidade interesante sería a de engadir a posibilidade de realizar a autenticación dos árbitros contra diversos hosts

de federacións, pensando na opción de que cada federación poida ter unha instancia propia da base de datos.

**8.2.1.0.5. Módulo de notificacións** Sería interesante que o árbitro recibise notificacións sobre cando a aplicación sincroniza novas actas de encontros, co fin de facilitarlle coñecer o lugar, data e hora a onde se debe desplazar para árbitrar e incluso engadir un sistema de recordatorios dos encontros que ten asignados.

## 8.2.2. Creación de comunidade

Nun proxecto de software libre é fundamental dispor dunha comunidade de desenvolvedores que axuden a mantelo vivo polo que tamén queremos mencionar diversas liñas de traballo a seguir en este punto.

**8.2.2.0.1. Hackathons de desenvolvemento.** Organizaranse diversos hackathons de desenvolvemento de 1 ou 2 días de duración nos que presentar o proxecto e tratar de buscar desenvolvedores para impulsar unha pequena funcionalidade ou un pequeno prototipo ao redor de VACmatch co fin de introducilos no proxecto.

**8.2.2.0.2. Proxectos de Fin de Grao ou Master.** Traballarase con profesores e asociacións para promover proxectos de fin de grao e master baseados en VACmatch, en lugar de crear pequenas aplicacións a medida para unha asociación ou federación, traballar sobre un proxecto grande e múltideporte.

**8.2.2.0.3. GPUL Summer of Code.** A asociación GPUL está desenvolvendo un programa de apoio a proxectos de software libre para que estudiantes universitarios colaboren en ditos proxectos durante un verán polo que trataremos de propor VACmatch como un dos proxectos no que estos estudiantes poidan colaborar.

## 8.3. Conclusións

O software libre é fundamental na sociedade actual, a inmensa maioría dos avances tecnolóxicos baseanse en solucións libres e cada vez son más os que as desenvolven.

No campo da tecnoloxía aplicada ao deporte, pouco a pouco comézanse a ver as primeiras solucións para informatizar a xestión pero, concretamente o software libre, atópase aínda con un longo camiño por recorrer e con este proxecto conseguimos sentar unhas bases para isto, creando a primeira aplicación libre para xestionar actas electrónicas de encontros deportivos.

VACmatch Mobile permite que os árbitros das competicións deportivas poidan xestionar as actas directamente no seu teléfono móvil e estas sincronizar en tempo real coa páxina web da

federación, mantendo os resultados actualizados en todo momento e permitindo incluso cubrir as actas cando non dispoñen de conexión a internet.

A aplicación é multiplataforma e foi testeada por usuarios reais polo que está validada por profesionais para ser utilizada nun contexto real, así mesmo atópase escrita pensando na extensibilidade, facilitando que os desenvolvedores interesados polo proxecto, poidan ampliala de xeito sinxelo e engadir novas funcionalidades, sobre todo, deportes.

# Apéndices

## **Apéndice A**

### **Configurar a aplicación.**

**A.1. Base de datos remota.**

**A.2. Aplicación.**

## **Apéndice B**

### **Executar a aplicación.**

**B.1. Compilación e execución web.**

**B.2. Compilación para móvil.**

## Apéndice C

# Glosario de acrónimos

**API** *Application Programming Interface.*

**REST** *Representational State Transfer.*

**TDD** *Test Driven Development.*

**BDD** *Behaviour Driven Development.*

**DOM** *Document Object Model.*

**HTML** *HyperText Markup Language.*

**CSS3** *Cascading Style Sheets*

## Apéndice D

# Glosario de términos

**VACmatch** VACmatch é unha plataforma de xestión de competicións deportivas que permite realizar todo tipo de trámites coas federacións deportivas de forma electrónica e reduce enormemente o traballo que estas deben realizar no seu día a día.

**VACmatch Mobile** é unha aplicación que permite que os árbitros deportivos poidan xestionar as actas dos seus encontros de forma electrónica.

**Actas** É o lugar onde se almacena a información sobre un encontro deportivo, inclue os equipos, os lugares onde se xogou e o resto de estadísticas de cada xogador durante o partido.

**Fichas** Unha ficha é un documento con fotografía incluída que identifica a un xogador que compite nunha competición e que debe levar a tódolos encontros para poder disputar os partidos.

**Xestor da competición** Persoa encargada da xestión do calendario, da recepción das actas dos encontros, da súa revisión, da súa publicación e, en xeral, da xestión dunha competición.

**Árbitro** Persoa que se encarga de velar polo cumplimento do regulamento dun deporte durante un encontro e así mesmo debe tomar nota na acta, das estadísticas e dos diversos eventos que ocurren nun encontro.

**API** Interfaz de programación de aplicacións que abstrae unha serie de funcións para a súa utilización dende outro software.

**Rest** É un tipo de arquitectura de desenvolvemento web que se basea no protocolo HTTP coa idea de utilizar os diversos verbos que define o protocolo para interactuar cos recursos de unha API.

**Lean Startup** Metodoloxía de desenvolvemento de negocio centrada no cliente e na aprendizaxe validada.

**eXtreme Programming** Metodoloxía de desenvolvemento de software que se centra na necesidade de adaptarse aos cambios no avance do proxecto co fin de dar un punto de vista máis realista que intentar definir todos os requisitos ao comezo do proxecto.

**Scrum** Metodoloxía de desenvolvemento de software ágil centrada en mellorar a xestión dun proxecto a través dunha serie de prácticas de revisión e xestión por iteracións de desenvolvemento chamadas sprints.

**Sprint** Iteración de desenvolvemento dentro das metodoloxías áxiles que habitualmente ten unha duración de entre 1 e 3 semanas.

**Release** Versión entregable do produto.

**Pull Request** Petición de integración de unha rama de Git dentro de outra. É a forma habitual de colaborar en proxectos que se atopan en GitHub.

**Startup** É un tipo de empresa en construcción, habitualmente con produto propio, moi ágil e que busca operar con uns costos mínimos e obter unhas ganancias de medren exponencialmente.

**Javascript** Linguaxe de programación interpretada, moi habitual para desenvolver aplicacóns web no lado do cliente, aínda que últimamente está a ganar protagonismo no servidor.

**SQLite** Sistema de xestión de bases de datos relacional, moi lixeiro e utilizado habitualmente en aplicacións móbilas nativas.

---

# Bibliografía

- [Bod16] Adam Boduch. *Flux Architecture*. Packt Publishing, May 2016.
- [DGG12] Rafael de las Heras del Dedo, Carmen Lasá Gómez, and Alonso [et al ] Álvarez García. *Métodos Ágiles y Scrum*. Anaya Multimedia-Anaya Interactiva, Madrid, January 2012.
- [GHJV94] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Addison Wesley, Reading, Mass, 01 edition edition, October 1994.
- [Mar08] Robert C. Martin. *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall, Upper Saddle River, NJ, 1 edition edition, August 2008.
- [Pan16] Mahesh Panhale. *Beginning Hybrid Mobile Application Development*. Apress, 1st ed. 2016 edition edition, May 2016.
- [Rie11] Eric Ries. *The Lean Startup: How Constant Innovation Creates Radically Successful Businesses*. Portfolio Penguin, London; New York, October 2011.
- [SG15] Andrew Stellman and Jennifer Greene. *Learning Agile: [understanding Scrum, XP, Lean, and Kanban]*. O'Reilly, Beijing, 1st ed edition, 2015. OCLC: 897773274.
- [Tiw11] Shashank Tiwari. *Professional NoSQL*. Wrox, Indianapolis, IN, 1 edition edition, September 2011.
- [WH12] Matt Wynne and Aslak Hellesoy. *The Cucumber Book: Behaviour-Driven Development for Testers and Developers*. Pragmatic Bookshelf, Dallas, Tex, 1 edition edition, February 2012.