



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**

A xestión de competicións deportivas continua a atoparse tremendamente 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.

Concretamente, neste proxecto preténdese crear 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 darlle aos xestores de federacións unha ferramenta na que realizar o seu traballo diario de forma electrónica e VACmatch Mobile integrarase nesta ferramenta.

Decidíuse utilizar tecnoloxías web para realizar este 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 . . . . .	1
1.3. VACmatch . . . . .	3
1.4. Resumo do proxecto . . . . .	3
1.4.1. VACmatch Mobile . . . . .	3
1.4.2. VACmatch . . . . .	4
1.4.3. Estrutura da memoria . . . . .	4
<b>2. Estado da arte</b>	<b>5</b>
2.1. Competidores no mercado . . . . .	6
2.1.1. Follas de cálculo . . . . .	6
2.1.2. Novanet . . . . .	7
2.1.3. Federatio . . . . .	8
2.1.4. miLeyenda . . . . .	9
2.1.5. Esportics . . . . .	10
2.1.6. Sportngin . . . . .	11
2.1.7. Outras plataformas . . . . .	12
2.2. Aplicacións libres no mercado . . . . .	12
2.3. Solución aberta e adaptable . . . . .	12
<b>3. Metodoloxía</b>	<b>13</b>
3.1. Lean Startup . . . . .	13
3.2. eXtreme Programming . . . . .	14
3.3. Scrum . . . . .	14
3.4. Adaptación da metodoloxía . . . . .	14
3.4.1. Desenvolvemento orientado ao cliente . . . . .	14
3.4.2. Sprints con backlog adaptable . . . . .	15
3.4.3. Reunións semanáis . . . . .	15
3.4.4. Reunións diarias . . . . .	15
3.4.5. Releases . . . . .	15
3.4.6. Simplicidade . . . . .	15
3.4.7. Tests . . . . .	16
3.4.8. Fluxo de traballo . . . . .	16

<b>4. Análise de requisitos globais</b>	<b>17</b>
4.1. Consultas a xestores de federacións . . . . .	17
4.1.1. Asociacións consultadas . . . . .	17
4.2. Peticións obtidas . . . . .	18
4.3. Requisitos finais . . . . .	19
4.3.1. Usuarios . . . . .	19
4.3.2. Listar actas . . . . .	19
4.3.3. Visualizar actas . . . . .	19
4.3.4. Xeración de actas offline . . . . .	19
4.3.5. Modificación de actas . . . . .	19
<b>5. Planificación e seguimento</b>	<b>21</b>
5.1. Validación de negocio (Xullo 2015 – Novembro 2015) . . . . .	22
5.1.1. I Torneo VACmatch . . . . .	22
5.1.2. Prototipo visual . . . . .	22
5.1.2.1. Planificación temporal . . . . .	22
5.1.2.2. Definición da iteración . . . . .	23
5.1.2.3. Revisión e feedback . . . . .	23
5.1.2.4. Tarefas e seguimento . . . . .	23
5.1.3. MVP funcional . . . . .	24
5.1.3.1. Planificación temporal . . . . .	24
5.1.3.2. Definición da iteración . . . . .	24
5.1.3.3. Revisión e feedback . . . . .	24
5.1.3.4. Tarefas e seguimento . . . . .	25
5.2. Desenvolvemento de produto (Novembro 2015 – Xaneiro 2016) . . . . .	25
5.2.1. Participación na I Lonxa de Financiamento Responsable . . . . .	25
5.2.2. 1ª iteración. Creación do proxecto . . . . .	26
5.2.2.1. Planificación temporal . . . . .	26
5.2.2.2. Definición da iteración . . . . .	26
5.2.2.3. Revisión e feedback . . . . .	26
5.2.2.4. Tarefas e seguimento . . . . .	27
5.2.3. 2ª iteración. Xestión de actas . . . . .	27
5.2.3.1. Planificación temporal . . . . .	27
5.2.3.2. Definición da iteración . . . . .	27
5.2.3.3. Revisión e feedback . . . . .	27
5.2.3.4. Tarefas e seguimento . . . . .	27
5.2.4. 3ª iteración. Eventos . . . . .	27
5.2.4.1. Planificación temporal . . . . .	27
5.2.4.2. Definición da iteración . . . . .	28
5.2.4.3. Revisión e feedback . . . . .	28
5.2.4.4. Tarefas e seguimento . . . . .	28
5.2.5. 4ª iteración. Xestión de usuarios e creación offline de actas . . . . .	28
5.2.5.1. Planificación temporal . . . . .	29
5.2.5.2. Definición da iteración . . . . .	29
5.2.5.3. Revisión e feedback . . . . .	29
5.2.5.4. Tarefas e seguimento . . . . .	29
5.2.6. 5ª iteración. Sinaturas . . . . .	29

---

5.2.6.1. Planificación temporal . . . . .	29
5.2.6.2. Definición da iteración . . . . .	30
5.2.6.3. Revisión e feedback . . . . .	30
5.2.6.4. Tareas e seguimiento . . . . .	30
5.3. De empresa a comunidade (Xaneiro 2016 – Maio 2016) . . . . .	30
5.3.1. 6 <sup>a</sup> e 7 <sup>a</sup> iteración. Optimización e melloras . . . . .	30
5.3.1.1. Planificación temporal . . . . .	30
5.3.1.2. Definición da iteración . . . . .	31
5.3.1.3. Revisión e feedback . . . . .	31
5.3.1.4. Tareas e seguimiento . . . . .	31
5.3.2. 8 <sup>a</sup> iteración. Testing e integración continua . . . . .	31
5.3.2.1. Planificación temporal . . . . .	31
5.3.2.2. Definición da iteración . . . . .	31
5.3.2.3. Revisión e feedback . . . . .	31
5.3.2.4. Tareas e seguimiento . . . . .	31
5.3.3. 9 <sup>a</sup> e 10 <sup>a</sup> iteración. Inxección de dependencias . . . . .	31
5.3.3.1. Planificación temporal . . . . .	32
5.3.3.2. Definición da iteración . . . . .	32
5.3.3.3. Revisión e feedback . . . . .	32
5.3.3.4. Tareas e seguimiento . . . . .	32
5.3.4. Release 0.2.0: Usabilidade en menús . . . . .	32
5.3.4.1. Planificación temporal . . . . .	32
5.3.4.2. Definición da iteración . . . . .	33
5.3.4.3. Concurso Universitario de Software Libre . . . . .	33
5.3.4.4. Revisión e feedback . . . . .	33
5.3.4.5. Tareas e seguimiento . . . . .	33
5.3.5. Release 0.2.1: I18n e app híbrida . . . . .	33
5.3.5.1. Planificación temporal . . . . .	33
5.3.5.2. Definición da iteración . . . . .	34
5.3.5.3. Revisión e feedback . . . . .	34
5.3.5.4. Tareas e seguimiento . . . . .	34
5.3.6. Release 0.2.2: Imáxe corporativa e revisión de errores . . . . .	34
5.3.6.1. Planificación temporal . . . . .	34
5.3.6.2. Definición da iteración . . . . .	34
5.3.6.3. Revisión e feedback . . . . .	34
5.3.6.4. Tareas e seguimiento . . . . .	34
5.3.7. Release 0.3.0: Usabilidade móvil e entrega continua . . . . .	34
5.3.7.1. Planificación temporal . . . . .	34
5.3.7.2. Definición da iteración . . . . .	34
5.3.7.3. Revisión e feedback . . . . .	34
5.3.7.4. Tareas e seguimiento . . . . .	34
<b>6. Fundamentos tecnolóxicos</b>	<b>35</b>
6.1. Linguaxes e frameworks empregados . . . . .	35
6.2. Bases de datos . . . . .	36
6.3. Estándares de comunicación . . . . .	36
6.4. Repositorios de código . . . . .	36

---

6.5. Ferramentas de xestión . . . . .	36
6.6. Ferramentas documentáis . . . . .	37
<b>7. Deseño e implementación</b>	<b>38</b>
7.1. ReactJS e Flux . . . . .	39
7.1.1. Introdución e elección da tecnoloxía . . . . .	39
7.1.2. Elementos básicos . . . . .	39
7.1.2.1. Compoñentes de React . . . . .	39
7.1.2.2. Arquitectura Flux . . . . .	40
7.1.2.3. Implementación de Flux. Reflux. . . . .	41
7.1.3. Estructura da aplicación . . . . .	42
7.2. Bases de datos e funcionamento offline . . . . .	43
7.2.1. PouchDB . . . . .	43
7.2.2. CouchDB . . . . .	43
7.2.3. Sincronización e xestión de conflictos . . . . .	43
7.3. App híbrida con Apache Cordova . . . . .	44
7.4. Interface gráfica e usabilidade . . . . .	45
7.4.1. Elementos comúns . . . . .	45
7.4.1.1. Menú lateral esquierdo . . . . .	45
7.4.1.2. Enlaces do menu superior dereito . . . . .	46
7.4.1.3. Información e axustes . . . . .	46
7.4.1.4. Barra de notificacións . . . . .	46
7.4.1.5. Autenticación . . . . .	47
7.4.1.6. Lista de pestanas . . . . .	48
7.4.2. Iniciar sesión . . . . .	48
7.4.3. Listado de actas . . . . .	49
7.4.4. Acta . . . . .	49
7.4.4.1. Convocar xogadores . . . . .	50
7.4.4.2. Inicio e fin do partido . . . . .	51
7.4.5. Finalización do encontro . . . . .	51
7.4.6. Modificación de estilos . . . . .	51
7.5. Multideporte . . . . .	52
7.5.1. Deporte . . . . .	52
7.5.2. Roles de usuarios . . . . .	53
7.5.3. Eventos . . . . .	53
7.6. Deseño da DB . . . . .	54
7.7. I18n . . . . .	54
7.8. Inxección de dependencias . . . . .	55
7.9. Testing e Integración continua . . . . .	56
7.9.1. TDD e BDD . . . . .	56
7.9.2. Jest . . . . .	57
7.9.3. Travis CI . . . . .	59
<b>8. Conclusóns e traballo futuro</b>	<b>60</b>
8.1. Recoñecementos . . . . .	60
8.1.1. Finalista no Certamen de Proyectos Libres da UGR . . . . .	60
8.1.2. Premio Universitario de Software Libre . . . . .	61

---

8.2. Conclusións . . . . .	61
8.3. Traballo futuro . . . . .	61
8.3.1. Melloras de desenvolvemento . . . . .	61
8.3.2. Creación de comunidade . . . . .	62
.1. Configurar a aplicación. . . . .	64
.2. Execución da aplicación. . . . .	64
.2.1. Base de datos remota. . . . .	64
.2.2. Compilación e execución web. . . . .	64
.2.3. Compilación para móvil. . . . .	64
<b>A. Glosario de acrónimos</b>	<b>65</b>
<b>B. Glosario de términos</b>	<b>66</b>
<b>Bibliografía</b>	<b>68</b>

# Índice de figuras

Figura	Página
2.1. Folla de cálculo . . . . .	6
2.2. Aplicación web de Novanet . . . . .	7
2.3. Web da FGVB co sistema Federatio . . . . .	8
2.4. APP móvil de MiLeyenda . . . . .	9
2.5. Torneo de tenis na web de Esportics . . . . .	10
2.6. APP móvil de Sportngin . . . . .	11
5.1. Web do I Torneo VACmatch . . . . .	22
5.2. I Lonxa de Financiamento Responsable . . . . .	26
5.3. Finalistas CUSL . . . . .	33
7.1. Esquema da arquitectura Flux . . . . .	40
7.2. Arquitectura de unha aplicación con Apache Cordova . . . . .	44
7.3. Estructura da aplicación gráficamente. . . . .	45
7.4. Menú lateral esquierdo. . . . .	45
7.5. Menú desplegable derecho. . . . .	46
7.6. Ventanas de información e axustes. . . . .	46
7.7. Barra de notificacións. . . . .	47
7.8. Vista de inicio de sesión. . . . .	49
7.9. Listado de actas. . . . .	49
7.10. Vista principal de un acta. . . . .	50
7.11. Seleccionar xogadores presentes no encontro. . . . .	50
7.12. Fin de encontro. . . . .	51
7.13. Sinatura de un acta. . . . .	52
7.14. Esquema base de datos de Person. . . . .	54

# Índice de cuadros

Tabla	Página
-------	--------



# Capítulo 1

## Introducción

### Índice general

---

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

---

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

### 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 cas persoas que compiten nas mesmas.

### 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.

**Inscripciones** 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.

**Aplazamientos de partidos** Moitas esíxenllas 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.

**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 despóis 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 desenvolvo 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 comprender o traballo que lles supón aos xestores de federacións é interesante ver cómo se realiza actualmente o proceso de xestión das actas dun encontro:

1. A federación crea un calendario de encontros que publica na súa web.
  2. Un árbitro recolle un **acta** no local da federación e leva dita acta ao campo ou a pista na que se disputa o encontro que debe arbitrar.
  3. Cada xogador leva a súa **ficha identificativa** ao encontro.
  4. O árbitro cubre o **acta** cos datos de tódalas **fichas** dos xogadores.
  5. Durante o encontro, o árbitro de mesa ou o 4º árbitro enche o **acta** manualmente, cubrindo as estadísticas do encontro.
  6. O **acta** asínase polo árbitro e un representante de cada equipo.
  7. Cada clube guarda unha copia do **acta** e o árbitro transalada a súa ata a federación.
  8. A federación revisa o **acta**, comprobando que os xogadores que xogaron non estaban sancionados, se pertence ao equipo e copiando tódolos datos recollidos, na aplicación de xestión ou a folla de cálculo da que dispoñan.
-

É por isto polo que se decidiu crear unha aplicación móvil que permita que os árbitros xestionen as súas actas de forma electrónica, directamente dende o seu teléfono móvil, 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 nun sistema de xestión de competicións 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* e *Telefónica Galicia OpenFuture*...

Actualmente a iniciativa atópase afincada no *Viveiro de empresas da Universidade da Coruña*

### 1.4. Resumo do proxecto

O proxecto desenvolvo componse de dúas partes diferenciadas que permite 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 ou IOS 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 aos árbitros dos encontros 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 reasignan 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 comezado 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 un xogador de cada equipo poderán asinar a acta con un código PIN.

---

**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.

#### 1.4.2. VACmatch

A aplicación móvil explicada antes tamén se atopa integrada nunha aplicación web de xestión de competicións a través da cal, as federacións poden xestionar as súas competicións, modificar o calendario, engadir novos xogadores ou equipos, xestionar arbitraxes, etc.

A parte que corresponde a xestión de actas de encontros tamén pertence a este proxecto e permite que a federación cree as actas, os árbitros a sincronicen nos seus teléfonos e unha vez cubertas, todos os datos sexan publicados automáticamente nesta aplicación de xestión.

Esto permite manter os resultados e as clasificacións actualizadas en todo momento nas federacións se apenas intervención humana, aforrando un enorme traballo na revisión das actas e no transporte das mesmas ata a sede da federación.

#### 1.4.3. Estrutura da memoria



Engadir estrutura

## Capítulo 2

# Estado da arte

### Índice general

---

<b>2.1. Competidores no mercado</b>	<b>6</b>
2.1.1. Fóllas de cálculo	6
2.1.2. Novanet	7
2.1.3. Federatio	8
2.1.4. miLeyenda	9
2.1.5. Esportics	10
2.1.6. Sportngin	11
2.1.7. Outras plataformas	12
<b>2.2. Aplicacións libres no mercado</b>	<b>12</b>
<b>2.3. Solución aberta e adaptable</b>	<b>12</b>

---

NESTE capítulo mostraranse as diversas alternativas no mercado da xestión de competicións así como se fará unha análise do software libre neste campo e do proceso de estandarización que se busca con este proxecto.

## 2.1. Competidores no mercado

### 2.1.1. Follas de cálculo

As follas 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 algunas tarefas como por exemplo o cálculo da clasificación dos equipos en función dos seus resultados ao longo da competición.

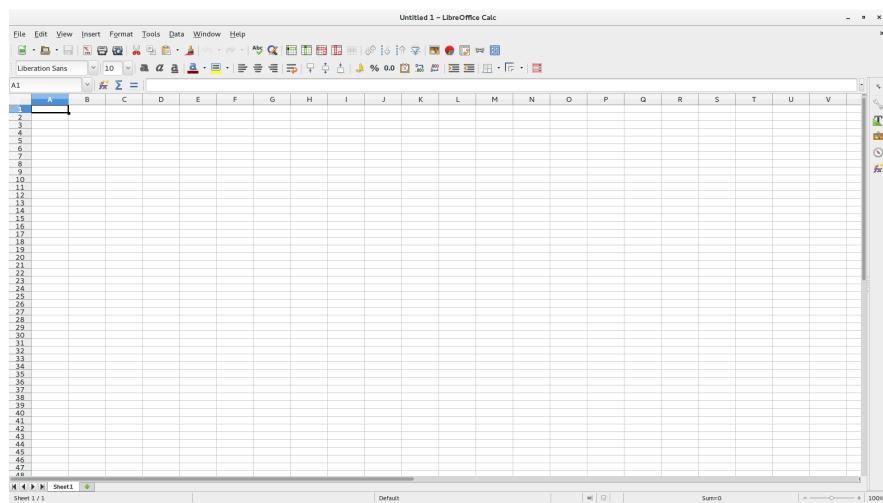


Figura 2.1: Folla de cálculo

O sistema é moi versatil para usuarios experimentados 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 erros na interpretación dos datos son habituais.

### 2.1.2. Novanet

Novanet é unha empresa especializada na xestión de competicións de fútbol e o seu producto compónse únicamente dunha aplicación web desde 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.

Ademáis, 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 logo o árbitro sube os datos a 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.

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



Figura 2.2: Aplicación web de Novanet

### 2.1.3. 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 a 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 a móbiles, o que dificulta enormemente a labor dos árbitros.

**XORNADA NÚMERO 5**  
12/12/2015

**Resultados desta xornada**

LOCAL	VISITANTE	SETS
IES Politécnico	3 - 2 ** XAV Cambados B	25-22/23-25/16-25/26-24/16-14
CVP Gómez Lor Multípticas	3 - 0 ** Voleibol Manitas	25-8/28-26/25-18

\*\* Résultado pendente de verificación

**Clasificación trala Xornada numero 5**

EQUIPO	P.J.	PG	PP	N.P.	J.F.	X	T.F.	TC	PUNTOS
1. CVP Gómez Lor Multípticas	4	4	0	0	12	0	303	180	8
2. ES Politécnico	5	2	3	0	10	12	443	480	7
3. XAV Cambados B	4	2	2	0	8	9	353	372	6
4. Voleibol Manitas	5	1	4	0	5	14	367	434	6

**Seguinte Xornada**

LOCAL	VISITANTE	DATA	HORA	LUGAR
CVP Gómez Lor Multípticas	IES Politécnico	19/12/2015	10:00	PM A Xunqueira 2
Voleibol Manitas	XAV Cambados B	19/12/2015	16:00	CEIP Espíñeira

**OPCIÓN**

- [Volver á competición](#)
- [Enviarlle esta páxina a un amigo](#)
- [Todos os partidos e resultados](#)
- [Calendario completo](#)
- [Versión impresa](#)

**Tweets**

- La Voz de Barbana 13 Dec El Náutico superó con facilidad al farolito rojo de la categoría [bit.ly/1DmHqg](#)
- Retrieved by FGVB 13 Dec Show Summary
- FGVB 15 Dec @VoleibCalego Parabéns! [twitter.com/joseluisunargo...](#)
- FGVB 15 Dec Tenéis ata o venres 18 para apuntarvos ao Curso de árbitros Territorial B. Non deixades pasar a oportunidade! #FGVB [bit.ly/1nLcJap](#)
- Expand

Figura 2.3: Web da FGVB co sistema Federatio

### 2.1.4. miLeyenda

miLeyenda é unha plataforma de xestión de competicións na nube que permite aos administradores de federacións dispor tamén de unha aplicación móvil nativa para IOS e outra para Android.

Dende esta aplicación poden xestionar gran parte dos parámetros das súas competicións entre os que se atopan 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 das aplicacións elévase enormemente ao ter que soportar ata 4 apps móbil 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.

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



Figura 2.4: APP móvil de MiLeyenda

### 2.1.5. Esportics

Esportics é unha startup española centrada na xestión de competicións deportivas e enfocada tremadamente cara o tenis e o padel así como os deportes electrónicos.

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

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

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

The screenshot shows the Esportics website interface. At the top, there's a dark header with the logo 'eSportics' and links for 'Buscar competición', 'Buscar jugador', 'Nuevo equipo', and user information ('castrinho10@gmail.com ( eSPin: 3921 )'). There's also a 'Salir' button and a 'Seleccionar idioma' dropdown.

The main content area displays a tournament registration form for 'LIGA ADT OTOÑO 2015 (CAN PRAT)'. It includes fields for 'Provincia: Barcelona', 'Deporte: Pádel', 'Fecha torneo: 28/09/2015 - 20/12/2015', and 'Sede: Can Prat Pádel Club Mollet'. Below this is a 'Descripción:' text area.

Below the registration form is a navigation bar with tabs: 'Categorías', 'Orden de juego' (selected), 'Participantes', and 'Clasificación'.

The 'Orden de Juego' section contains a table of match results:

Jornada/Ronda	Horario	Pista	Jugadores	Categoría	Resultado
Jornada 1	28/09/15 20:30	-	Lluís Martínez Manlls - 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 Vallecillos 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 Inserte Poyatos	Masculino Open	6-3 1-6 6-4

On the left side of the main content area, there's a sidebar with social sharing buttons for Facebook, Google+, and Twitter, and a 'Contacto Organización' button.

Figura 2.5: Torneo de tenis na web de Esportics

### 2.1.6. Sportngin

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

De feito, Sportngin permite a personalización da aplicación móvil a cada federación, cos seus logotipos, colores corporativos e incluso certos menús personalizables e incluso permite traballar de forma offline.

A parte das funcions 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.

citar desde  
o texto as  
figuras



Figura 2.6: APP móvil de Sportngin

### 2.1.7. Outras plataformas

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

Tamén existen pequenas extensións coa idea de extender outras plataformas xenéricas para adaptalas a xestión de competicións como o *Joomla! CMS sport extension* pero a función final é tremadamente limitada.

Tamén temos outras propostas como *Siguetuliga* que permite as persoas que se atopan vendo o partido, subir os resultados pero simplemente é un complemento, non facilita nin elimina o traballo dos xestores de competicións.

## 2.2. Aplicacións libres no mercado

O software libre é un sector en alza na actualidade, os proxectos colaborativos que forman o mundo *Open Source* estanse a impor en múltiples mercados fronte as correspondentes alternativas privativas que adoitan a ser tremadamente costosas.

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

Concretamente no mundo do deporte é tremadamente complicado atopar algún exemplo de aplicación baseada en software libre e as poucas existentes como *zuluru* ou *phpmysport* atópanse tremadamente atrasadas tanto en funcionalidades como no aspecto visual e por suposto sen ningún tipo de aplicación móvil para facilitar a xestión das actas polo que é importante propor unha alternativa ás aplicacións privativas habituais como é VACmatch Mobile.

## 2.3. 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 móviles 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 móviles para outros sistemas de xestión.

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 comunicacións aberta para a xestión de actas electrónicas e que permita a súa integración noutros sistemas de xestión entre os cales se atopa o sistema de VACmatch, unha implementación libre para a xestión de competicións.

---

## Capítulo 3

# Metodoloxía

### Índice general

---

<a href="#">3.1. Lean Startup</a>	13
<a href="#">3.2. eXtreme Programming</a>	14
<a href="#">3.3. Scrum</a>	14
<a href="#">3.4. Adaptación da metodoloxía</a>	14
<a href="#">3.4.1. Desenvolvemento orientado ao cliente</a>	14
<a href="#">3.4.2. Sprints con backlog adaptable</a>	15
<a href="#">3.4.3. Reunións semanáis</a>	15
<a href="#">3.4.4. Reunións diarias</a>	15
<a href="#">3.4.5. Releases</a>	15
<a href="#">3.4.6. Simplicidade</a>	15
<a href="#">3.4.7. Tests</a>	16
<a href="#">3.4.8. Fluxo de traballo</a>	16

---

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

### 3.1. Lean Startup

Lean Startup é unha metodoloxía para abordar o lanzamento de negocios e produtos a través da validación, a experimentación e a iteración no lanzamento dos mesmos para 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 MPV (Mínimo Producto Viable), 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 de dito produto.

### 3.2. eXtreme Programming

eXtreme Programming é unha metodoloxía de desenvolvemento áxil e incremental baseada na integración do cliente no desenvolvemento así como na simplicidade do código, apostando por facer cousas sinxelas e ter que facer un pequeno traballo por modificalo se é preciso, fronte a facer un gran traballo para quizás nunca utilizalo.

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

Por último aboga por introducir o TDD (Test Driven Development), implementando primeiro os tests, verificar que fallan e a continuación implementar o código que fai que pasen os tests. 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 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 de entre unha e catro semanas e que é definida polo equipo e cada unha das cales proporciona un incremento de software entregable.

A totalidade das tarefas do proxecto atópanse definidas e priorizadas no *Product Backlog* e para cada sprint selecciónanse aquellas que determinarán o *Sprint Backlog*, que serán implementadas durante dito sprint e que non poden variar ata rematar.

Durante todo o ciclo de traballo realizáñense reunións diarias para comprobar o estado do proxecto así como outras ao fin e ao comezo dos sprints co fin de analizar o anterior e planificar o seguinte.

### 3.4. Adaptación da metodoloxía

#### 3.4.1. Desenvolvemento orientado ao cliente

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

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 reais 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 semanas de duración e que cada unha proporciona unha serie de novas funcións totalmente .

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 segúin o aprendido nos anteriores.

falar do backlog que se adapta durante o sprint, non como en scrum que é fixo

### 3.4.3. Reunións semanáis

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 correxilas.

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 de uns 10 minutos de duración para revisar o realizado no día anterior e planificar de forma más 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 postos en produción.

Durante o desenvolvemento producíronse 3 entregas (*releases*) totalmente funcionáis, 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 fundamentase 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 necesarias finalmente.

### 3.4.7. Tests

A importancia de creación de tests automatizados está totalmente demostrada, atopándose en auxe metodoloxías como *TDD* (*Test Driven Development*) ou *BDD* (*Behaviour Driven Development*) que tratan de dirixir o desenvolvemento a través dos tests, que son realizados antes do mesmo.

Durante a primeira parte do desenvolvemento non se aplicóu 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.

### 3.4.8. Fluxo de traballo

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

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

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

Posteriormente realizase unha *Pull Request* a través do mecanismo que proporciona o repositorio de código Github, esperando a unha revisión de código para ser integrado na rama principal do proxecto.

Cada certo tempo revisáse as *Pull Requests* abertas, analízase o código e se todo é correcto, aceptase para integrar na rama principal.

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

---

## Capítulo 4

# Análise de requisitos globais

### Índice general

---

<a href="#">4.1. Consultas a xestores de federacións</a>	17
<a href="#">4.1.1. Asociacións consultadas</a>	17
<a href="#">4.2. Peticións obtidas</a>	18
<a href="#">4.3. Requisitos finais</a>	19
<a href="#">4.3.1. Usuarios</a>	19
<a href="#">4.3.2. Listar actas</a>	19
<a href="#">4.3.3. Visualizar actas</a>	19
<a href="#">4.3.4. Xeración de actas offline</a>	19
<a href="#">4.3.5. Modificación de actas</a>	19

---

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

### 4.1. Consultas a xestores de federacións

Para a realización deste apartado decidíuse 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 obtendo incluso a priorización segundo as necesidades más urxentes.

#### 4.1.1. Asociacións consultadas

**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 y incluso novos colaboradores para poder desenvolver un producto de calidad.

Realizáronse ata 5 visitas a federación co fin de mostrarllles a evolución do proxecto e 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 prototipo do proxecto con usuarios reais, tanto árbitros como xogadores e no que se comprobou as necesidades dos usuarios e as súas 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

**4.2.0.0.1. 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.

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

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

**4.2.0.0.4. Persoas 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.

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

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

**4.2.0.0.7. 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.

**4.2.0.0.8. 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 traballo dos árbitros.
- **Visualizar actas cubertas dun árbitro** Debe mostrar as actas cubertas anteriormente e todos os seus datos pero non modificalas.
- **Actualización automática de actas descargadas ante modificacíons** 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 xeráis dun acta** Débese mostrar do xeito máis simplificado posible os datos xeráis 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** Tanto a aplicación de xestión como a aplicación móvil deben 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 que 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 ánda 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 hay 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>	<b>22</b>
5.1.1. I Torneo VACmatch	22
5.1.2. Prototipo visual	22
5.1.3. MVP funcional	24
<b>5.2. Desenvolvemento de produto (Novembro 2015 – Xaneiro 2016)</b>	<b>25</b>
5.2.1. Participación na I Lonxa de Financiamento Responsable	25
5.2.2. 1 <sup>a</sup> iteración. Creación do proxecto	26
5.2.3. 2 <sup>a</sup> iteración. Xestión de actas	27
5.2.4. 3 <sup>a</sup> iteración. Eventos	27
5.2.5. 4 <sup>a</sup> iteración. Xestión de usuarios e creación offline de actas	28
5.2.6. 5 <sup>a</sup> iteración. Sinaturas	29
<b>5.3. De empresa a comunidade (Xaneiro 2016 – Maio 2016)</b>	<b>30</b>
5.3.1. 6 <sup>a</sup> e 7 <sup>a</sup> iteración. Optimización e melloras	30
5.3.2. 8 <sup>a</sup> iteración. Testing e integración continua	31
5.3.3. 9 <sup>a</sup> e 10 <sup>a</sup> iteración. Inxección de dependencias	31
5.3.4. Release 0.2.0: Usabilidade en menús	32
5.3.5. Release 0.2.1: I18n e app híbrida	33
5.3.6. Release 0.2.2: Imáxe corporativa e revisión de erros	34
5.3.7. Release 0.3.0: Usabilidade móvil e entrega continua	34

---

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

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

### 5.1.1. 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 un entorno real e controlado, os primeiros prototipos desenvoltos.

Finalmente participaron 6 equipos e mais de 50 persoas durante dous días.

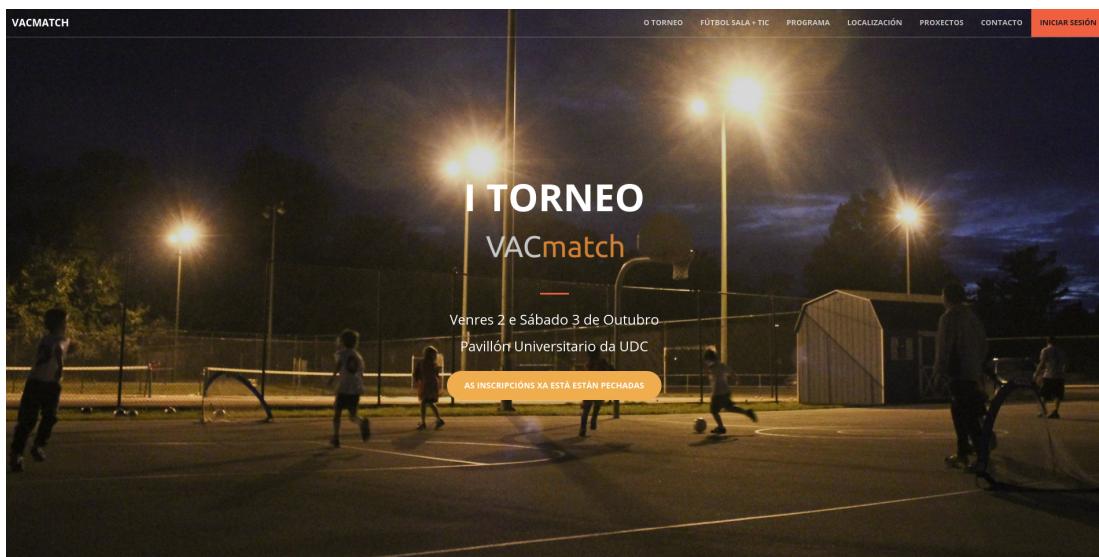


Figura 5.1: Web do I Torneo VACmatch

### 5.1.2. Prototipo visual

#### 5.1.2.1. Planificación temporal

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.

### 5.1.2.2. Definición da iteración

Durante este periodo de un mes de duración planificouse o desenvolvemento de unha aplicación moi sinxela e sen funcionalidade, que únicamente permitise 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.

### 5.1.2.3. Revisión e feedback

Durante as visitas as federacións obtivéronse diversas prospostas 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 hay cobertura.

### 5.1.2.4. Tarefas e seguimento

As tarefas de esta 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.

Planificaronse X horas, fixéronse X por isto

---

### 5.1.3. MVP funcional

#### 5.1.3.1. Planificación temporal

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

#### 5.1.3.2. Definición da iteración

Unha vez finalizadas as probas visuais e de usabilidade procedese 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 compoñentes 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.

#### 5.1.3.3. Revisión e feedback

Durante as visitas as federacións obtivérónse 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 a acta en lugar de escribir as incidencias.

#### 5.1.3.4. Tarefas e seguimento

- Como árbitro quero poder cargar a lista de actas pendentes.
- Como árbitro quero poder cargar a lista de actas rematadas.
- Como árbitro quero poder ver os datos dun partido.
- Como árbitro quero poder ver a lista de xogadores dun equipo.
- Como árbitro quero poder seleccionar os xogadores presentes no partido.
- Como árbitro quero poder engadir un gol a un xogador.
- Como árbitro quero poder engadir unha falta a un xogador.
- Como árbitro quero poder engadir unha tarxeta (amarela ou vermella) a un xogador.
- Como árbitro quero poder ver a lista de eventos de un partido.
- Como árbitro quero poder borrar eventos de un partido.
- Como árbitro quero poder engadir incidencias a un partido.

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

Tras analizar os problemas e os erros cometidos durante a realización do I Torneo VACmatch, e concretamente no funcionamento da aplicación móvil durante o mesmo, decíduse comezar novamente dende o principio un proxecto novo en lugar de facer unha refactorización do prototipo.

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 de VACmatch, no que contar os avances que van sucedendo durante o desenvolvemento do proxecto.

### 5.2.1. 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.

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.



Figura 5.2: I Lonxa de Financiamento Responsable

### 5.2.2. 1<sup>a</sup> iteración. Creación do proxecto

#### 5.2.2.1. Planificación temporal

Esta iteración transcorre entre o día 1 e o 15 do mes de Novembro de 2015

#### 5.2.2.2. Definición da iteración

A presente iteración centrarse 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.

Crease tamén o proxecto base engadindo a licencia e definese 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.

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 engadilas e eliminarlas de xeito sinxelo para facer as primeiras probas.

#### 5.2.2.3. Revisión e feedback

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

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.

#### 5.2.2.4. Tarefas e seguimento

- Definir modelo de datos.
- Deseñar mockups finais.
- Crear proxecto base.
- Estudio da tecnoloxía. React, Reflux, PouchDB, Redmine
- Crear modelos en PouchDB.
- Como árbitro quero poder obter a lista de actas a cubrir.
- Permitir crear e borrar actas para tarefas de test.
- Engadir licencia e Readme

#### 5.2.3. 2<sup>a</sup> iteración. Xestión de actas

##### 5.2.3.1. Planificación temporal

A iteración comeza o día 16 de Novembro e remata o día 30 do mesmo mes.

##### 5.2.3.2. Definición da iteración

Nesta 2º iteración planifícase o desenvolvemento das páxinas principais e básicas para a xestión da acta dun encontro.

As funcionalidades que se abordarán neste sprint centranse 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.

##### 5.2.3.3. Revisión e feedback

Ao non realizar ningunha visita a federacións, non se obtivo feedback pero si é importante mencionar que se produciron retrasos e polo tanto varias tarefas foron transladadas ao seguinte sprint de desenvolvemento.

##### 5.2.3.4. Tarefas e seguimento

- Como árbitro quero ver o resumo da acta.
- Como árbitro quero controlar o tempo do partido.
- Como árbitro quero manter o tempo do partido aínda que cambie de páxina.

#### 5.2.4. 3<sup>a</sup> iteración. Eventos

##### 5.2.4.1. Planificación temporal

Dende o 30 de Novembro ata o 13 de Decembro.

Mencionar tarefas retrasadas a que pasan ao seguinte sprint

#### 5.2.4.2. Definición da iteración

Esta iteración céntrase principalmente na xestión de eventos dos encontros, tarefas que requiren 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 se plantexa crear tipos de eventos que modifiquen tamén a propia acta, actualizando na mesma tanto o resultado como as faltas cometidas.

Por último incorporase tamén na iteración convocar e editar persoas dun equipo así como algún pequeno erro detectado na última iteración.

#### 5.2.4.3. Revisión e feedback

#### 5.2.4.4. Tarefas e seguimento

- Como árbitro quero poder engadir un evento.
- Como árbitro quero ver a lista de xogadores para asignar un evento.
- Como árbitro quero poder confirmar engadir un evento.
- Como árbitro quero engadir unha causa a un evento.
- Cómo árbitro quero poder ver a lista de eventos de un partido.
- Como árbitro quero que se xeneren eventos de comezo e fin de partido.
- Como árbitro quero que se xeneren eventos ao cambiar de parte.
- Como árbitro quero que se actualice o resultado ao engadir un gol.
- Como árbitro quero que se actualicen as faltas automáticamente.
- Como árbitro quero poder convocar e desconvocar xogadores.
- Como árbitro quero poder cambiar o dorsal de un xogador nun partido determinado.
- Como árbitro quero poder borrar un evento.
- Como árbitro quero ver a lista de eventos de un partido ordenada por tempo e parte.
- Mostrar únicamente xogadores convocados ao engadir un evento.
- Como árbitro quero que ao borrar un evento se actualicen os resultados e as faltas na Acta.
- Erro: Cando non hay ningún evento de cambio de parte hay un error.
- Erro: Corexir erro cando se engade un evento con causa.

#### 5.2.5. 4ª iteración. Xestión de usuarios e creación offline de actas

login, logout, metese staff, crear actas offline Memoria, 3 primeros apartados

### 5.2.5.1. Planificación temporal

Esta iteración ten lugar entre o 14 e o 30 de Decembro de 2015.

### 5.2.5.2. Definición da iteración

Esta iteración centrase 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.

### 5.2.5.3. Revisión e feedback

Post no blog -¿fundamentos tecnolóxicos e estado do proxecto.

### 5.2.5.4. Tarefas e seguimento

- Como usuario quero poder facer log in.
- Engadir diferenciación entre Persoal e Xogadores.
- Como usuario quero poder facer log out.
- Como árbitro quero poder crear un partido manualmente.

Memoria Definir introdución.

Memoria Estado da arte.

Memoria Fundamentos tecnolóxicos.

Memoria Engadir modelo.

- Como árbitro quero poder editar un acta creada dende o móvil.
- Erro: Correxir erro coa variable dialogIsOpen ao editar o dorsal de un xogador.

## 5.2.6. 5<sup>a</sup> iteración. Sinaturas

Sinaturas con PIN, engadir incidencias do sprint anterior: crear usuario, crear árbitro ao crear user

### 5.2.6.1. Planificación temporal

Esta iteración comeza o día 1 de Xaneiro de 2016 e remata o día 8.

---

### 5.2.6.2. Definición da iteración

A presente iteración centrase na última vista da aplicación, a que permite xestionar a finalización do encontro dando a posibilidade de asinar a acta e engadir comentarios por parte do árbitro.

Na última iteración detectaronse varias melloras a incluir como facer que por defecto as contas creadas dende a aplicación móvil sexan todas de tipo árbitro e a todas as actas que cree esa conta, se engada a dito usuario como árbitro do encontro.

### 5.2.6.3. Revisión e feedback

#### 5.2.6.4. Tarefas e seguimento

- Como árbitro quero poder asinar un acta.
- Como árbitro quero que unha ou varias persoas convocadas de cada equipo poidan asinar un acta.
- Como árbitro quero poder engadir comentarios a un acta.
- 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.

## 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.

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 disponible para continuar co proxecto.

### 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 su realizóu ningunha tarefa polo que se decidíu unir ambas iteracións.

#### 5.3.1.1. Planificación temporal

Estas iteracións comezan o 14 de Febreiro de 2016 e rematan o 29 de Febreiro.

### 5.3.1.2. Definición da iteración

Durante esta iteración 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.

### 5.3.1.3. Revisión e feedback

### 5.3.1.4. Tarefas e seguimento

- Refactorizar servicios.
- Crear clases para cada entidad.
- Revisar cómo se crean os identificadores dos obxectos en base de datos.

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

### 5.3.2.1. Planificación temporal

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

### 5.3.2.2. Definición da iteración

Decidíuse engadir tests unitarios para previr futuros errores 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 provas, en este caso Travis CI.

### 5.3.2.3. Revisión e feedback

### 5.3.2.4. Tarefas e seguimento

- Engadir tests aos servizos de Eventos, Persoas, Equipos e Actas.
- Engadir tests aos servizos de Auth, Árbitros e Sinaturas.
- Engadir integración continua con Travis CI.
- Engadir campos para confirmar contrasinal e código PIN ao crear un usuario.

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

Motivado de novo pola falta de tempo para traballar no proxecto, decidíuse integrar de novo estas dúas iteracións.

---

### 5.3.3.1. Planificación temporal

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

### 5.3.3.2. Definición da iteración

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

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

Pero a tarefa máis importante xurdíu ao aparecer un problema de dependencias circulares que obrigou 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 errores detectados na iteración anterior.

### 5.3.3.3. Revisión e feedback

### 5.3.3.4. Tarefas e seguimento

- Engadir textos de error na aplicación.
- Crear barra de notificacións para utilizar en calquera compoñente.
- Engadir DAOs.
- Engadir Inxección de Dependencias.
- Engadir estados na Acta.
- Erro: Como árbitro non debería poder convocar/borrar a un xogador que ten eventos asignados.
- Erro: Mostrar que o partido remató na acta.
- Erro: Correxir problema de dependencias circulares.
- Erro: Correxir errores varios derivados de engadir inxección de dependencias.
- Erro: Un evento de comezo de partido non se pode borrar si existe algún outro evento creado.
- Erro na xestión do estado da Acta.
- Erro: Travis CI non funciona correctamente

### 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.

---

#### 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.



Figura 5.3: Finalistas CUSL

#### 5.3.4.4. Revisión e feedback

#### 5.3.4.5. Tarefas e seguimento

- Revisar os links nos menús e engadir información.

### 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 aplicacíons 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 erros

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

## 5.3.7. Release 0.3.0: Usabilidade móvil e entrega continua

Por cordova: dialogos -> ventás, transición carga, migrar taiga.io, Travis + Docker

### 5.3.7.1. Planificación temporal

### 5.3.7.2. Definición da iteración

### 5.3.7.3. Revisión e feedback

### 5.3.7.4. Tarefas e seguimento

# Capítulo 6

## Fundamentos tecnolóxicos

### Índice general

---

6.1. Linguaxes e frameworks empregados . . . . .	35
6.2. Bases de datos . . . . .	36
6.3. Estándares de comunicación . . . . .	36
6.4. Repositorios de código . . . . .	36
6.5. Ferramentas de xestión . . . . .	36
6.6. Ferramentas documentáis . . . . .	37

NESTE capítulo mostraranse as diversas tecnoloxías que foron empregadas durante o desenvolvemento do proxecto así como ferramentas de xestión e documentáis, todas, tecnoloxías Software Libre.

### 6.1. Linguaxes e frameworks empregados

**ReactJS** React é unha librería de Javascript para a creación de Single Page Applications (SPAs), permitindo crear aplicacóns no frontend de forma sinxela a través de diversos compoñentes que se agrupan e que permiten crear aplicacións multiplataforma.

**Reflux** É unha implementación da arquitectura Flux impulsada por Facebook e que permite un fío de datos unidireccional, en lugar do bidireccional que é habitual nas aplicacións web. Permite tamén a creación de Stores nas que se mantén o estado das aplicacións e que permite compartir dito estado entre os diversos compoñentes da aplicación.

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

Añadir  
página  
web de cada  
tecnología: EJ:  
ReactJS  
([1]) y el  
link en  
la bibliogra  
fía

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

**Material UI** 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

**PouchDB** 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.

**CouchDB** 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.

Permite gran cantidade de funcionalidades como servir aplicacións directamente desde CouchDB así como un sistema de replicación incremental e de detección de conflictos.

## 6.3. Estándares de comunicación

**JSON** 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 tremendamente sinxelo traballar con él dende esta linguaxe.

## 6.4. Repositorios de código

**Github** 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. Cómpre 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** É un sistema de control de versións software libre de gran potencia e utilizado en millóns de proxectos que aporta unha versatilidade enorme ao ser distribuída, permitindo traballar incluso de forma offline.

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

**Redmine** 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** É unha ferramenta de integración continua que permite automatizar a execución de tests ou o despregamento automático de código. Ademáis dispón dunha integración con Github polo que resulta moi sinxelo automatizar estas tarefas.

**Docker** 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** Un editor de texto software libre deseñado inicialmente por Github de gran potencia e extensibilidade gracias a un sinxelo sistema de plugins. Ademáis é un editor impulsado por Facebook (creadores de ReactJS) para facilitar o traballo con esta tecnoloxía.

## 6.6. Ferramentas documentáis

**LaTeX** 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 gran calidade.

**Dia** 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

# Deseño e implementación

### Índice general

---

<b>7.1. ReactJS e Flux . . . . .</b>	<b>39</b>
7.1.1. Introducción e elección da tecnoloxía . . . . .	39
7.1.2. Elementos básicos . . . . .	39
7.1.3. Estructura da aplicación . . . . .	42
<b>7.2. Bases de datos e funcionamento offline . . . . .</b>	<b>43</b>
7.2.1. PouchDB . . . . .	43
7.2.2. CouchDB . . . . .	43
7.2.3. Sincronización e xestión de conflictos . . . . .	43
<b>7.3. App híbrida con Apache Cordova . . . . .</b>	<b>44</b>
<b>7.4. Interface gráfica e usabilidade . . . . .</b>	<b>45</b>
7.4.1. Elementos comúns . . . . .	45
7.4.2. Iniciar sesión . . . . .	48
7.4.3. Listado de actas . . . . .	49
7.4.4. Acta . . . . .	49
7.4.5. Finalización do encontro . . . . .	51
7.4.6. Modificación de estilos . . . . .	51
<b>7.5. Multideporte . . . . .</b>	<b>52</b>
7.5.1. Deporte . . . . .	52
7.5.2. Roles de usuarios . . . . .	53
7.5.3. Eventos . . . . .	53
<b>7.6. Deseño da DB . . . . .</b>	<b>54</b>
<b>7.7. I18n . . . . .</b>	<b>54</b>
<b>7.8. Inxección de dependencias . . . . .</b>	<b>55</b>
<b>7.9. Testing e Integración continua . . . . .</b>	<b>56</b>
7.9.1. TDD e BDD . . . . .	56
7.9.2. Jest . . . . .	57
7.9.3. Travis CI . . . . .	59

---

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

## 7.1. ReactJS e Flux

### 7.1.1. Introdución e elección da tecnoloxía

Actualmente o mundo das aplicacións 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.

A sociedade está a cambiar un soporte tradicional como é o papel en todas os 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 aplicacións móbiles con tecnoloxías web e con unha experiencia de usuario exactamente igual a unha aplicación nativa tradicional.

Unha vez selecciodo React, optamos por seguir a arquitectura máis habitual dentro de este tipo de aplicacións, 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 tremendamente eficiente.

### 7.1.2.2. Arquitectura Flux

Flux é unha arquitectura que básicamente propón o seguinte esquema:

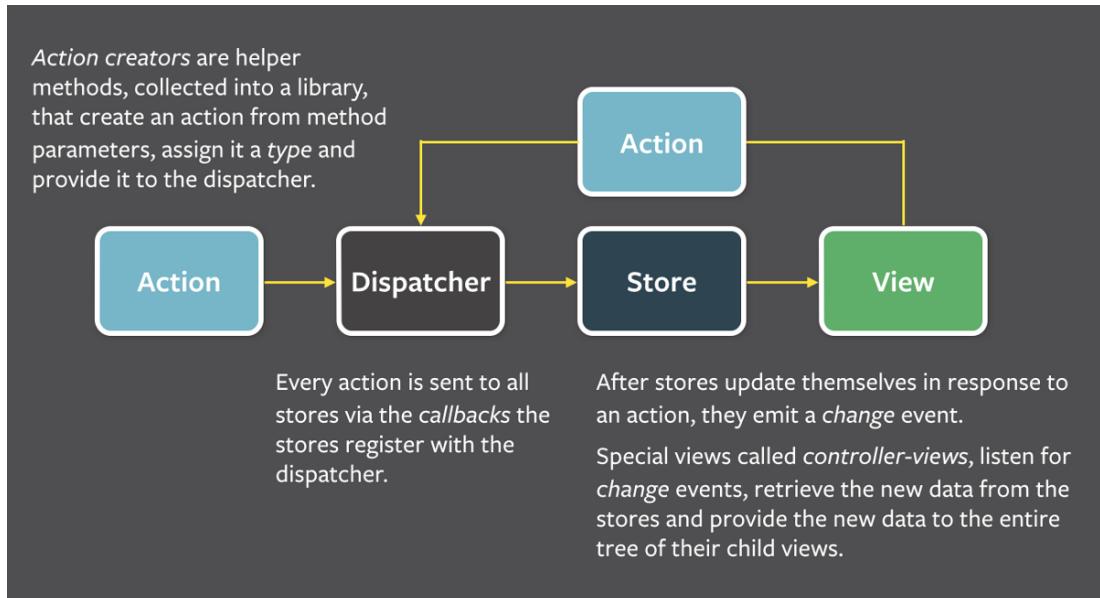


Figura 7.1: Esquema da arquitectura Flux

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

**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 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 o feito de que non existe un único Dispatcher central que enruta as Actions se non que todas as Stores están escoitando e reaccionan cando teñen un método para responder a Action.

Vexamos un exemplo práctico:

```
1 let TextActions = Reflux.createActions([
2   "updateText"
3 ])
```

Definiríamos unha Action que queremos lanzar para tratar de actualizar o texto e crearíamos unha Store que escoite as TextActions e que implemente unha función onUpdateText que será a que reciba o novo valor e actualizará o estado.

```
1 let TextStore = Reflux.createStore({
2   listenables: TextActions,
3   init: function () {
4     this.state = ''
5   },
6   getInitialState: function () {
7     return this.state
8   },
9   onUpdateText: function (newText) {
10    this.state = newText
11    this.trigger(this.state)
12  }
13})
```

Por último definir o componente. Este deberá incluir unha lista de mixins onde indicarle cales son as Stores que reaccionan ante eventos lanzados dende este componente, neste caso, a TextStore.

O componente tamén terá un método render no que definir o que poderíamos chamar informalmente o “modelo” 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.

```
1 let EndReport = React.createClass({
2   mixins: [
3     Reflux.connect(TextStore, 'text')
4   ],
5   handleChangeTextClick: function () {
6     TextActions.updateText('Nuevo texto a renderizar')
7   }
8   render: function () {
9     return (
10       <p>{this.state}</p>
11       <a onClick={this.handleChangeTextClick}>Change text</a>
```

```
12      )
13    }
14 })
```

### 7.1.3. Estructura da aplicación

A continuación mostráños os diversos elementos xenerais 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 traductor a súa labor.

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

**src/app** Estructura principal da aplicación

**Actions** Lista de ficheiros de 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 compoñentes 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 compoñentes 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 compoñentes 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, 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 LocalStorage do navegador, pensada dende o primeiro momento para crear aplicacións web que funcionen de forma desconectada.

### 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, multinavegador 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 na aplicación.

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, aínda 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 revisións 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 utilízalo 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 desenvolvemento 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 escribilas.

A continuación podemos ver un esquema da arquitectura dunha aplicación sobre Apache Córdova:

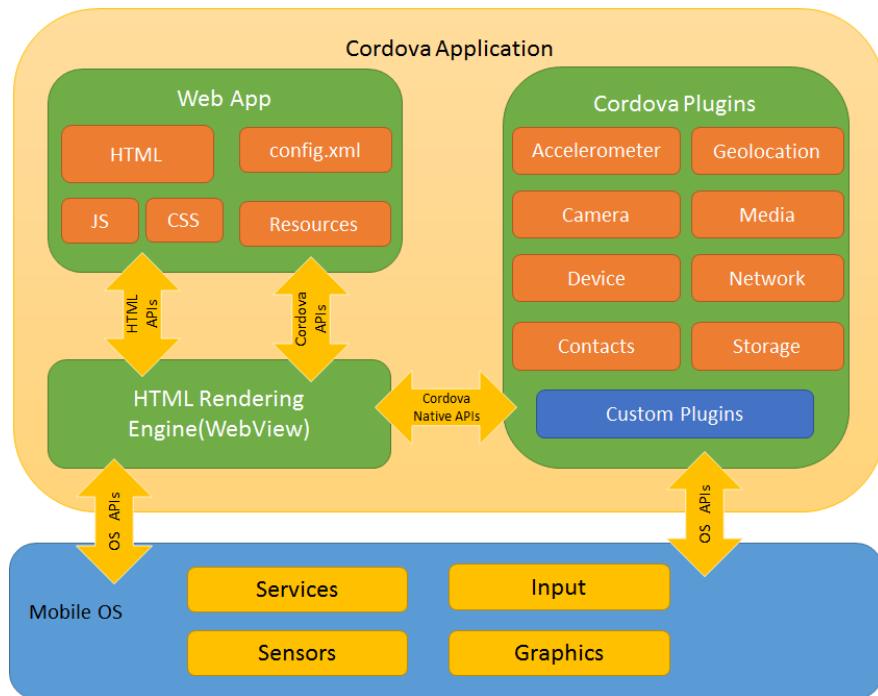


Figura 7.2: Arquitectura de unha aplicación con Apache Cordova

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 deseñe 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:

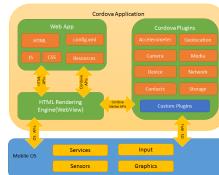


Figura 7.3: Estructura da aplicación gráficamente.

### 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.

Este menú desplegable mostra actualmente o logotipo de VACmatch e unha serie de enlaces entre os que se atopan as páxinas de configuración e 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 "LeftMenuItems". 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.

Utilizase 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.

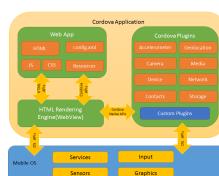


Figura 7.4: Menú lateral esquierdo.

#### 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 elementos a mostrar no menú desplegable.

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

**text** Texto a mostrar polo item.

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

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

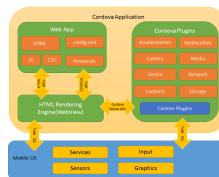


Figura 7.5: Menú desplegable dereito.

#### 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.

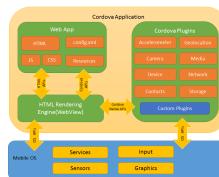


Figura 7.6: Ventanas de información e axustes.

#### 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”, 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.

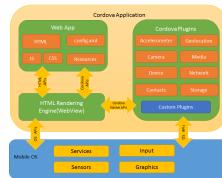


Figura 7.7: Barra de notificacións.

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 código seguinte.

```

1 let SnackBarItem = React.createClass({
2   mixins: [
3     Reflux.connect(SnackBarStore, 'snackBar')
4   ],
5
6   onStatusChange: function (status) {
7     this.refs.snack.show()
8   },
9
10  componentDidMount: function () {
11    this.listenTo(SnackBarStore, this.onStatusChange)
12  },
13
14  render: function () {
15    return <Snackbar key={'generic-snackbar'}>
16      ref='snack'
17      message={this.state.snackBar.message}
18      autoHideDuration={4000} />
19    }
20 })

```

Listing 7.1: Barra de notificacións.

#### 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.

Unha clase sinxela que deben extender aqueles compoñentes que so poidan ser accedidos se o usuario está autenticado, o “AuthenticatedComponent” redirixe a aplicación cara páxina de iniciar sesión, no caso de non poder atopar un usuario activo na sesión.

```

1 export default (ComposedComponent) => {
2   let AuthenticatedComponent = React.createClass({
3     mixins: [
4       Reflux.connect(AuthStore, 'auth'),
5       History
6     ],
7
8     componentWillMount: function (transition) {
9       if (config._env !== 'development') {
10         // This method is called before transitioning to this component.
11         // If the user is not logged in, we'll send him or her to the Login page.
12         if (!AuthStore.isLoggedIn()) {
13           this.history.pushState(null, urls.login.show)
14         }
15       }
16     },
17
18     render: function () {
19       return (
20         <ComposedComponent
21           {...this.props} />
22       )
23     }
24   })
25   return AuthenticatedComponent
26 }

```

Listing 7.2: Compoñente de autenticación.

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 a 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 podendo introducir en ela calquera calquera tipo de elemento de tipo Item.

#### 7.4.2. Iniciar sesión

Aquí 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, aparte 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.

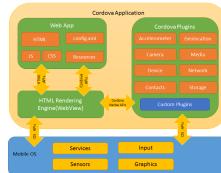


Figura 7.8: Vista de inicio de sesión.

#### 7.4.3. Listado de actas

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

Cada un dos elementos que se mostra 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 xeitor eliminar a acta e todos os elementos que a componen.

Ademáis, 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 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.

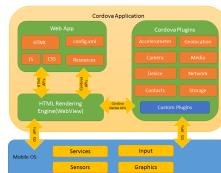


Figura 7.9: Listado de actas.

#### 7.4.4. Acta

Esta é a parte central da aplicación, con total seguridade será o lugar onde os usuarios pasarán a gran maioría 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 cantidade 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.

Finalmente móstranse os nomes dos equipos que están a competir con un enlace para ver os seus xogadores, dous campos de resultados, un cronómetro, un botón de edición, un botón para mostrar os eventos que ocorreron e os diversos eventos que poden ser utilizados para este encontro.

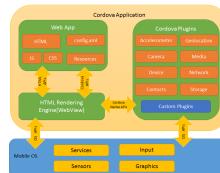


Figura 7.10: Vista principal de un acta.

#### 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 todos os xogadores que se atopan inscritos no equipo correspondente coa sua foto, 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.

Tamén ten a posibilidade de editar o dorsal do xogador para este encontro en concreto xa que en algunas 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

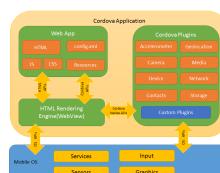


Figura 7.11: Seleccionar xogadores presentes no encontro.

#### 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 áinda o mesmo 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 todos os eventos introducidos, levan 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 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 algúns tipo de agresión ou o motivo polo que un partido tivo que ser finalizado antes de tempo.

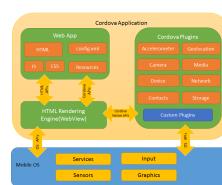


Figura 7.12: Fin de encontro.

Dende o menú superior dereito pódese acceder directamente aos eventos que ocorreron durante o partido para mostrarlle as persoas que asinarán de cada equipo, únicamente co fin de que poidan confirmar que realmente os eventos gardados na acta, 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.

Para realizar a sinatura, a persoa debe dispoñer de conta creada na base de datos da federación na que debiu 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.

#### 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.

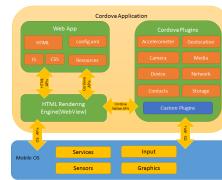


Figura 7.13: Sinatura de un acta.

Certa parte da comunidade de React defende que o estilo das aplicacións web, que tradicionalmente se xestioná 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, coa idea de acabar con diversos erros de deseño en CSS.

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 propósito xa que, agora os estilos serán variables locales que os “minifiers” 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 puidera 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 Store 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.

Por outro lado, cada deporte debe extender a unha clase xenérica Sport que define unha serie de métodos que todos os deportes deben implementar e que indican cómo se realizan certas accións sobre os datos en cada deporte.

**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 ícono que identifica un Evento de un tipo para renderizalo en algúnhia 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 tremenda mente sinxelo engadir un novo deporte, básicamente deberíase 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 definíu 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 producíu, 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.

Existe un compoñente de React para incorporar os eventos de control dentro da vista que lista os eventos dun partido e que calquera evento de este tipo, definido unha clase como a establecida para os eventos de control cos detalles do seu funcionamento, pode utilizar.

Por outra banda temos tamén un compoñente de React para introducir os eventos deportivos na lista de eventos dun partido, e cada evento novo, debe crear un novo compoñente que defina a vista a mostrar cando se desexe engadir un evento de este tipo no encontro.

engadir  
imaxe de  
exemplo

En definitiva, para crear un novo evento débese definir unha clase Javascript coas propiedades do mesmo e que extenda ao SportEvent.

Tamén é preciso engadir unha clase de React que defina a vista que se mostrará ao intentar engadir o evento durante o encontro (habitualmente unha ventá modal) e por último débese engadir o evento creado anteriormente, dentro da lista de eventos que ten ese deporte, e que se atopan na clase que define as propiedades de dito deporte.

## 7.6. Deseño da DB

VACmatch Mobile pensóuse inicialmente para integrarse 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 estaba pensado 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 comprimironse dúas entidades en unha soa como é, o de Call e Member que foron unidas dentro de Person como se indica no gráfico seguinte.

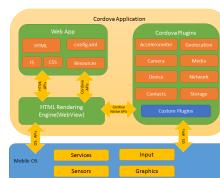


Figura 7.14: Esquema base de datos de Person.

## 7.7. I18n

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

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

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

```

1 <RaisedButton label={
2     <FormattedMessage
3         id='report.show.events'
4         description='Events button'
5         defaultMessage='Events' />
6     } secondary={true} style={style.button}/>

```

---

Listing 7.3: Exemplo de internacionalización na label de un botón.

Do mesmo xeito, a librería permite facer internacionalización de xeito programático sen utilizar componentes de react, a través de unha serie de funcións de javascript.

Ademáis é preciso realizar as traduccíons aos diferentes idiomas, cada unha en un ficheiro de javascript diferente e con un formato de pares cadea/valor.

## 7.8. Inxección de dependencias

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

A existencia destas 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.

A solución atópase inspirada na implementación que realiza o framework Spring, e na que dispoñemos dunha factoría 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.

A continuación pódese ver o código que implementa parte da factoría que se encarga de crear os servizos:

```

1 let ServiceFactory = {
2   isInitialized: false,
3   _servicesList: new Map(),
4
5   constructor () {
6     this._personService = new PersonService()
7     this._reportService = new ReportService()
8     this._refereeService = new RefereeService()
9
10    // Without dependencies
11    this._teamService = new TeamService()
12
13    // Inject dependencies
14    this._eventService.ReportService = this._reportService
15    this._eventService.PersonService = this._personService
16    this._eventService.TeamService = this._teamService
17    ...
18
19    // Add services to the exposed list
20    this._servicesList.set('ReportService', this._reportService)
21    this._servicesList.set('PersonService', this._personService)
22    this._servicesList.set('TeamService', this._teamService)
23    this.isInitialized = true
24  },
25
26  getService (type) {
27    if (!this.isInitialized) {
28      this.constructor()
29    }
30
31    let service = this._servicesList.get(type)
32
33    return service
34  }
35}

```

---

```

29     if (!service) {
30         console.log('Error getting ', type, ' service')
31     }
32     return service
33 }
34
35 }
```

Listing 7.4: Fragmento da ServiceFactory.

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) e Behaviour Driven Development (BDD) 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 de unha 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 céntanse en comprobar o funcionamento 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 o cal permiten simular o comportamento dos obxectos dependentes.

Esto 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 aplicácións ReactJS que facilita...

A continuación imos mostrar un test de exemplo do módulo de eventos no que se pode observar os diversos componentes que o forman. Pódese observar o código de exemplo na figura

Inicialmente defíñese 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“.

```
1 describe('create Sport Event', function () {
2   ...
3 })
```

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 das variables que utilizan.

Dentro do bloque ”describe“ defínense os diversos tests de esta historia de usuario, concretamente dentro dos bloques ”it“.

```
1 it('Create a new Sport Event with valid parameters', function () {
2   ...
3 })
```

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 seguinte sentencia:

```
1 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 seguinte trozo de código 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.

```
1 spyOn(reportService, 'findById').andCallFake(function (anyReportId, callback) {
2   callback(defaultReport, null)
3 })
```

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.

engadir figura

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.

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

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

```
1   describe('create Sport Event', function () {
2
3     beforeEach(function () {
4       defaultPerson = new Person(null, '', '', '', '', false, false, '',
5       '', '')
6       defaultTeam = new Team(null, 'Team name')
7       defaultReport = new Report(null, '', '', ReportStatus.READY,
8       defaultTeam, defaultTeam, [])
9       defaultEvent = new EventElements.Event('event', '1', defaultPerson,
10      defaultTeam, 'goal', 1, 'cause', 1)
11      reportService = new ReportService(jasmine.createSpy('PersonService'),
12      jasmine.createSpy('TeamService'), jasmine.createSpy('EventService'),
13      jasmine.createSpy('SignService'))
14      personService = new PersonService(jasmine.createSpy('ReportService'),
15      jasmine.createSpy('TeamService'), jasmine.createSpy('AuthService'))
16      teamService = new TeamService()
17      eventService = new EventService(reportService, personService,
18      teamService)
19    })
20
21    it('Create a new Sport Event with valid parameters', function () {
22
23      spyOn(reportService, 'findById').andCallFake(function (anyReportId,
24      callback) {
25        callback(defaultReport, null)
26      })
27
28      spyOn(personService,
29      'findByPersonIdReportIdAndTeamId').andCallFake(function (anyPersonId,
30      anyReportId, anyTeamId, callback) {
31        callback(defaultPerson, null)
32      })
33
34      spyOn(teamService, 'findById').andCallFake(function (anyTeamId,
35      callback) {
36        callback(defaultTeam, null)
37      })
38
39      spyOn(EventDao, 'create').andCallFake(function (reportId, person,
40      team, eventType, matchTime, cause, timestamp, callback) {
41        callback(defaultEvent, null)
42      })
43
44      eventService.create(defaultEvent.reportId, defaultEvent.person,
45      defaultEvent.team,
46      defaultEvent.type, defaultEvent.matchTime, defaultEvent.text,
```

```

47     defaultEvent.timestamp,
48     (event, err) => {
49       expect(event).toEqual(defaultEvent)
50       expect(event).not.toBe(null)
51       expect(err).toBe(null)
52     })
53
54   expect(reportService.findById).toHaveBeenCalled()
55
56 expect(personService.findByPersonIdReportIdAndTeamId).toHaveBeenCalled()
57   expect(teamService.findById).toHaveBeenCalled()
58   expect(EventDao.create).toHaveBeenCalled()
59 }

```

Listing 7.5: Exemplo de test no módulo e eventos utilizando Jest.

### 7.9.3. Travis CI

A medida que a aplicación foi medrando, xurdíu a necesidade de simplificar traballos como a búsquedas 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 decidíu 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 todos os tests.

Para isto utilizouse 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 dependencia 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 construción 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>	60
8.1.1. Finalista no Certamen de Proyectos Libres da UGR	60
8.1.2. Premio Universitario de Software Libre	61
<b>8.2. Conclusíons</b>	61
<b>8.3. Traballo futuro</b>	61
8.3.1. Melloras de desenvolvemento	61
8.3.2. Creación de comunidade	62
.1. Configurar a aplicación.	64
.2. Execución da aplicación.	64
.2.1. Base de datos remota.	64
.2.2. Compilación e execución web.	64
.2.3. Compilación para móvil.	64

---

**N**este capítulo contaremos as conclusíons obtidas da realización do proxecto, os recoñe-  
**E**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 poder 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. Conclusiones

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 esto, creando a primeira aplicación libre para xestionar actas de encontros deportivos.

A aplicación foi testeada por usuarios reais polo que está validada por profesionais para ser utilizada en un 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 e, sobre todo, deportes.

## 8.3. 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 xurdiron novas propostas, das cales algunas non foron ainda implementadas e forman parte de liñas de traballo futuro.

### 8.3.1. Melloras de desenvolvemento

**8.3.1.0.1. 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.

**8.3.1.0.2. 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.

**8.3.1.0.3. 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 áinda en maior medida o traballo do árbitro e do xestor da federación deportiva.

**8.3.1.0.4. 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 da base de datos propia.

**8.3.1.0.5. 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.

### 8.3.2. 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.

**8.3.2.0.1. 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.

**8.3.2.0.2. 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.

**8.3.2.0.3. 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.

# Apéndices

- .1. Configurar a aplicación.
- .2. Execución da aplicación.
  - .2.1. Base de datos remota.
  - .2.2. Compilación e execución web.
  - .2.3. Compilación para móvil.

## Apéndice A

### Glosario de acrónimos

**API** *Application Programming Interface.*

**Rest** *Representational State Transfer.*

**TDD** *Test Driven Development*

**BDD** *Behaviour Driven Development*

## Apéndice B

# 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 incluida 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 Rest**

**Lean Startup**

**eXtreme Programming**

**Scrum**

**Sprint**

**Release**

**Commit**

**Pull Request**

**Startup**

# Bibliografía