ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADO EN INGENIERÍA DEL SOFTWARE

APLICACIÓN DE GESTIÓN DE ACTIVIDADES DEPORTIVAS PARA DISPOSITIVOS MÓVILES

APPLICATION OF MANAGEMENT OF SPORT ACTIVITIES FOR MOBILE DEVICES

Realizado por
David Doña Corrales

Tutorizado por
Javier Cubo Villalba
Ernesto Pimentel Sánchez

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, DICIEMBRE 2015

Fecha defensa:
El Secretario del Tribunal

Agradecimientos y dedicatoria

A mis hermanos y especialmente a mi madre, por estar ahí para lo bueno y para lo malo, por su paciencia conmigo, por su cariño, y por todas las cualidades que reflejan en ella a una madre ejemplar.

A Ángela, por ser tan buena conmigo, por escucharme siempre que lo necesito y por compartir su vida conmigo.

A mi amigo Juan, ejemplo de esfuerzo y dedicación, cuyo apoyo y compañía ha sido muy importante para andar este camino.

A mis amigos y compañeros de facultad Gustavo, Santiago, Zhendong Ma, Benjamín, José Antonio, Enrique y Joaquín que me han acompañado en esta aventura y que gracias a ellos la vida en la universidad me ha resultado más amena.

Gracias a Javier Cubo por su confianza en mí y su ayuda en este proyecto.

A mi padre, cuyo apoyo y ejemplo de fuerza y superación a lo largo de mi vida ha dejado en mí un legado imborrable. Tú has sido el verdadero motor que me ha impulsado a llegar aquí. Allá donde estés papá, escribo estas palabras con esperanza de que las puedas leer: ¡Lo conseguí papa!

Resumen:

Este trabajo fin de grado tiene como objetivo satisfacer la necesidad por parte del Club Deportivo de la Universidad de Málaga de disponer de una herramienta software que permita almacenar y gestionar información sobre las competiciones que se realizan en el club de una forma cómoda y rápida para así mejorar los procesos actuales de recogida y manipulación de los datos.

Para lograr ese objetivo se ha desarrollado una aplicación para dispositivos móviles compatible con Windows Phone y Android para la gestión de eventos de atletismo de carácter general y con una licencia de software libre para que cualquier usuario tenga la libertad de usar y modificar la aplicación sin restricciones.

Para el desarrollo de la aplicación se ha hecho uso de los conocimientos y técnicas aprendidas a lo largo de la carrera. En esta memoria se describe todo lo relacionado con el trabajo llevado a cabo para alcanzar los objetivos del proyecto; desde la fase inicial de requisitos hasta la fase de pruebas y validación del producto obtenido, incluyendo artefactos generados en el proceso de diseño y modelado.

Palabras clave:

Aplicación móvil, competición, atletismo, deporte, gestión deportiva, software, Phonegap, Apache Cordova

Abstract:

This final degree project has as goal to satisfy the need of the Club Deportivo at Universidad de Málaga which is to have a software tool that allows storing and managing information about the competitions they perform in the club, in a quick and easy way, by improving the current processes of collection and manipulation of data.

In order to achieve such objective, it has been developed a mobile application in Windows Phone and Android for the management of athletic events as a generic solution with open source license, so anyone could use and modify the application without any constraints.

As regards the development of the software application, it has been used the knowledge and techniques learnt along the degree. This document describes the work done to get the objectives of the project; from the requirements to the testing and validation phase of the product built, including artifacts generated in the process of design and modeling.

Keywords:

Mobile application, competition, athletics, sport, sport management, software, Phonegap, Apache Cordova

ÍNDICE

1. INTRODUCCIÓN

1.1 Problemática y motivación

Los técnicos del Club Deportivo de la Universidad de Málaga tienen entre sus actividades la recogida y elaboración de datos referentes a las distintas actividades físicas y deportivas que se realizan en el complejo y en otros eventos. Este tipo de actividades se realizan a mano apoyándose de programas básicos de ofimática como Word o Excel.
La tarea de elaborar y controlar manualmente todos estos ficheros resulta tediosa además de añadir otros tipos de problemas:

- Riesgo sobre la integridad de los datos.
- Gasto de tiempo considerable.
- Los datos se suelen tomar en papel y luego se pasan a ordenador, pudiendo extraviarse los papeles en el proceso intermedio.
- Riesgo sobre la redundancia de información.
- Aislamiento de los datos.
- Riesgo de errores en procesos de inferencia de los datos, como por ejemplo al obtener la mejor puntuación de una prueba.
- Búsqueda de la información lenta.
- Dificultad de mantenimiento. Un cambio; por ejemplo, en el grupo de un participante, puede llevar al cambio manual en varios ficheros.
- Imposibilidad de acceso simultáneo por parte de varios técnicos a un mismo fichero.
- Dificultad de compartir los datos de los ficheros con otros tipos de programas.

Debido a esto surge la necesidad de controlar de manera automática y centralizada las competiciones y actividades físicas y deportivas que se llevan a cabo en el club a través de una aplicación. Esto permitirá agilizar los procesos mencionados y solventar los problemas descritos.

En concreto, se pretende ***diseñar e implementar una aplicación para dispositivos móviles para la gestión de diferentes tipos de actividades físicas y deportivas.*** Dicha gestión incluirá el control de competiciones, grupos, participantes, equipos, pruebas y registros; además de otros tipos de características que se detallan más adelante. Además, todas estas funcionalidades serán diseñadas para un uso genérico; es decir, se podrán definir todo tipo de pruebas y competiciones relacionadas con el ámbito deportivo.

Como solución para llevar a cabo este desarrollo de software, se ha decidido utilizar la herramienta ***Cordova/Phonegap*** [18] cuya funcionalidad principal es la de poder desarrollar bajo un mismo código versiones de la aplicación para distintos

dispositivos móviles. Esta decisión se ha tomado principalmente porque reduce significativamente el coste de tiempo y esfuerzo; permitiendo desarrollar de una sola vez el código necesario para la aplicación y evitando así tener que realizar el mismo programa pero en distintos lenguajes.

Para el desarrollo de las vistas se ha decidido utilizar como núcleo **Lungo**, que se trata de un framework que permite un diseño adaptativo, creado especialmente para dispositivos móviles, liberado como software libre.

La aplicación será diseñada de manera que sea fácilmente extensible, modificable y adaptable.

La aplicación ha sido implementada con frameworks y librerías open-source, y también del mismo modo será distribuida, ofreciendo las siguientes ventajas:

- Ningún coste del producto.
- Libertad de uso, modificación y distribución.
- Los usuarios que necesiten una gestión similar podrán descargarse el código y, con pequeñas modificaciones sobre el mismo, extender fácilmente la aplicación en base a sus nuevas necesidades.

1.2 Objetivos

El objetivo principal de este trabajo es obtener un software que permita, tanto al Club Deportivo de la Universidad de Málaga como a cualquier otro usuario con la misma necesidad, gestionar competiciones desde sus dispositivos móviles permitiendo además que su uso sea de carácter general y adaptable a distintas disciplinas deportivas.

1.3 Métodos y fases de trabajo

Para la realización del proyecto se definió inicialmente 6 fases en el anteproyecto. Debido a la decisión posterior de desarrollar la aplicación bajo la herramienta Cordova/Phonegap, las fases 4 y 5 se solapan ya que el desarrollo en distintas plataformas móviles se hace bajo un único proceso. Se enumeran las fases a continuación junto a algunas anotaciones:

Fase 1. Captura y análisis de requisitos. Al formar parte este proyecto del mismo ámbito que otro TFG [32], los requisitos son comunes a ambos proyectos, por lo que mi dedicación en esta fase fue analizar los requisitos ya recogidos por parte de mi compañero, adaptar algunos requisitos al ámbito de este proyecto, validarlos y elaborar mi propio documento de requisitos. La estimación inicial para esta fase fue de 5 horas.

Fase 2. Estudio del estado del arte. Durante esta fase se llevó a cabo un análisis de posibles aplicaciones móviles que fueran similares a la que se pretendía desarrollar. Se encontraron algunas como “Mis Torneos” [33], Gestor Deportivo [34], Youth Sports Team Management [35], y Korrio Mobile [36], de las que cabe destacar las dos primeras, las cuales tienen sólo algunas de las funcionalidades pero ninguna reúne todas las características de este software. La estimación inicial para esta fase fue de 10 horas.

Fase 3. Estudio de lenguajes y herramientas para el desarrollo de la aplicación en dispositivos móviles. Inicialmente esta fase estaba destinada a aprender sobre los lenguajes específicos para Windows Phone y Android. Debido al cambio de técnica para el desarrollo (Phonegap) esta fase ha sido dedicada a aprender sobre el entorno Phonegap y herramientas necesarias para la construcción del software. La estimación de esta fase era de 25 horas.

Fase 4. Diseño, implementación y pruebas iniciales de la aplicación para Android y Windows Phone. El resultado de esta fase ha sido una primera versión completa del software junto a un conjunto de pruebas de diferente tipo para cada módulo del programa. Se estimó una dedicación de 180 horas.

Fase 5. Instalación final de las dos versiones, puesta en marcha en sistema de producción, y pruebas. Una vez concluidos los desarrollos y las pruebas iniciales, se instalará para su mantenimiento la aplicación en sus diferentes versiones, Android y Windows Phone, pudiendo ser empleada por parte de los técnicos deportivos del Club Deportivo de la UMA, dando un período de mantenimiento. Esta fase se estima en una dedicación 15 horas.

1.4 Estructura de la memoria

A lo largo de la memoria se explica todo lo relacionado con el proceso de desarrollo de esta aplicación, desde las tecnologías utilizadas hasta las últimas fases de pruebas y validación. A continuación se explica de forma breve cada apartado que existe en la memoria como objeto de resumen.

- **Capítulo 1. Introducción**

En este capítulo se describe el problema y la motivación que ha llevado a la realización de este proyecto, además de describir los objetivos del mismo y las fases en las que se ha dividido.

- **Capítulo 2. Tecnologías empleadas**

A lo largo de este capítulo se detalla las principales características de cada una de las herramientas, lenguajes y frameworks utilizados.

- **Capítulo 3. Especificación de requisitos**

Este apartado recoge todo lo relacionado con la fase de requisitos, describiendo cada uno de ellos y se expone algunos de los casos de uso asociados a estos requisitos. También se añade una tabla sobre definiciones y abreviaturas propias del ámbito de la aplicación.

- **Capítulo 4. Diseño de la aplicación**

En este capítulo se muestran distintos artefactos producidos por el modelado de la aplicación, tales como el diagrama relacional, diagrama de clases y diagrama de navegación. También se muestran algunas maquetas de interfaz.

- **Capítulo 5. Implementación de la aplicación**

Abarca información sobre todo el proceso de implementación. Desde el tipo de arquitectura que se ha usado hasta patrones generales y propios del lenguaje JavaScript. También se detalla otros elementos destacables de la implementación, como por ejemplo los objetos Promise.

- **Capítulo 6. Pruebas y validación**

A través de este capítulo se explican las pruebas que se han realizado sobre los componentes de la implementación y su posterior validación. También se hace referencia a un tipo especial de pruebas que se han diseñado para hacer testing al sistema completo a través de la interfaz de usuario.

- **Capítulo 7. Conclusiones y líneas futuras**

En este capítulo se exponen las conclusiones finales del TFG, señalando los aspectos a destacar durante el desarrollo y planteando ideas sobre posibles futuras ampliaciones que se pudieran hacer sobre el mismo.

- **Anexos técnicos**

Este capítulo comprende todos los casos de uso, diagramas de secuencia y mockups creados en el desarrollo del proyecto.

2. TECNOLOGÍAS EMPLEADAS

Para el desarrollo de este software se ha decidido optar por la herramienta Phonegap; que permite a través de HTML, Javascript y CSS obtener desde la misma implementación versiones para Windows Phone, Android, y; aunque no estaba contemplado abarcarlo inicialmente, también iOS.

Como lenguaje de programación se ha utilizado Javascript, que es el lenguaje necesario para desarrollar bajo la herramienta Phonegap. Junto a este lenguaje se han utilizado librerías tales como Zepto (una librería basada en JQuery) para el manejo de eventos y del DOM, RequireJS para modularizar, QuoJS que viene en conjunto con Lungo y SweetAlert como complemento para los mensajes de alerta.

Para poder llevar a cabo el requisito de imprimir en formato PDF, se ha optado por jsPDF, una librería Javascript con suficiente madurez y que se adecua a lo que se necesita. Junto a esta librería también se ha utilizado AutoTable, un plugin creado por terceros para facilitar la creación de las tablas.

Las pruebas se han desarrollado con QUnit, un framework en JavaScript.

En la persistencia de datos se ha utilizado Web SQL Database, que se trata de una API que incorpora HTML5 para crear bases de datos locales y que, aunque actualmente la W3C ha dejado de dar soporte para esta especificación, sigue teniendo un uso importante y está soportado por la mayoría de los navegadores de escritorio y de móvil. Además, se trata de una base de datos relacional bastante potente y forma parte de la especificación de Cordova/Phonegap la cual da soporte en todos los dispositivos móviles. Para el acceso y uso de Web SQL se ha hecho uso del lenguaje SQL.

La parte de las vistas está construida en HTML5 y CSS3, teniendo como núcleo el framework Lungo, que está especialmente diseñado para dispositivos móviles y ofrece un diseño y características adecuadas para este proyecto.

En cuanto a las herramientas utilizadas para el desarrollo del software y sus procesos se han utilizado REM, Trello, MagicDraw, NinjaMock, Phonegap Build, MySQL Workbench, Sublime Text 2, GapDebug, Chrome y XAMPP.

En los siguientes apartados se detalla información sobre cada tecnología y herramienta utilizada.

2.1 Lenguajes

En esta sección se explica con detalle cada lenguaje utilizado y su uso específico en la implementación de la aplicación.

2.1.1 JavaScript

Nacido en 1995, JavaScript [1] (oficialmente denominado ECMAScript, aunque no se le suele referir por este nombre) es un lenguaje de programación orientado a objetos, débilmente tipado, usado principalmente en páginas webs y que se ejecuta directamente desde el navegador. Se trata de un lenguaje interpretado por lo que no se requiere compilarlo para poder ejecutarlo. Su sintaxis es parecida a C, aunque posee las características del paradigma orientado a objetos.

En sus comienzos, nació como un lenguaje encapsulado en el lado del cliente para poder realizar pequeñas funciones, tales como validar formularios o generar efectos visuales. Esto permitía extender las funciones que otorgaba el HTML. Su uso inicial fue bastante limitado e infravalorado; incluso había programadores que opinaban que no era un lenguaje de programación en sí y que estaba condenado a desaparecer con el tiempo. Pero a lo largo de los años ha ido ganando terreno gracias al auge de las aplicaciones web modernas llegando a niveles de complejidad y prestaciones tan grandes como cualquier otro lenguaje de primer nivel.

Actualmente su uso no se ha limitado sólo en el client-side, sino que también se ha implicado en el server-side, formando parte como lenguaje de servidores tales como Node.js, Jaxer y RingoJS. Aplicaciones web tan importantes como Hotmail, Facebook o Google utilizan como parte importante JavaScript.

Todos los navegadores modernos ofrecen soporte para la ejecución de este lenguaje, lo que le otorga una gran versatilidad pudiendo ser ejecutado en cualquier sistema operativo con tal solo utilizar un navegador. Otra de sus ventajas es que actualmente posee un gran apoyo por parte de la comunidad de programadores, proliferando una gran cantidad de librerías y frameworks útiles y de gran calidad.

También se utiliza para desarrollar aplicaciones en Phonegap, siendo por lo tanto el lenguaje usado para llevar a cabo la implementación de esta aplicación.

Al comienzo del proyecto, no se vislumbró todas las ventajas de usar este lenguaje; y es que después de haber estudiado su uso en Phonegap llegué a la conclusión de que el software a desarrollar bajo JavaScript poseía muy bajo acoplamiento con el uso del framework lo que le otorga la característica de poderse ejecutar de manera independiente (standalone). Esto significa que como resultado de este proyecto se tiene, además de una aplicación para móviles, también un programa de escritorio.

2.1.2 HTML

HTML [2] (HyperText Markup Language) es un lenguaje de etiquetado para elaborar páginas webs. Se trata de un estándar a cargo de la W3C que nace en 1991 con el objetivo de definir una estructura básica para añadir contenido en

páginas webs.
Está compuesto por una serie de etiquetas que los navegadores interpretan y a través de los cuales se traduce de forma visual. Estas etiquetas son escritas de la forma <*nombre_etiquera*>*algún texto*</*nombre_etiqueta*> donde el nombre de la etiqueta determina el contexto o transformación del texto encerrado. Por ejemplo una etiqueta con nombre da énfasis en el texto que encierra mientra que una etiqueta <button> crea visualmente un botón con el nombre del texto encerrado.
Dentro de cada etiqueta se pueden añadir atributos que configuran o añaden propiedades a las mismas. Por ejemplo, una etiqueta <table> que se le añade el atributo "name" quedaría <table name="*un_nombre*"> el cual define un nombre para la etiqueta.

HTML ha suprimido y añadido características a lo largo de los años adaptándose a los cambios y necesidades del momento. Actualmente HTML va por su versión 5 la cual añade mejoras frente a sus predecesores como por ejemplo las etiquetas <section> y <article> que ayudan a mejorar la semántica del documento. También elimina otras etiquetas que estaban en desuso como <frame> o .

La estructura básica de un documento HTML5 es:

```
<!doctype html>
 <html>

  <head>
    <meta charset="utf-8"/>
    <title>Título de la página</title>
  </head>

  <body>
    Contenido de la página
  </body>

</html>
```

Figura 1. Código básico de una página HTML5

Donde el elemento <body> encierra el contenido de la página. Dentro del cuerpo, una estructura básica que se suele utilizar para organizar las partes de la página es la siguiente:

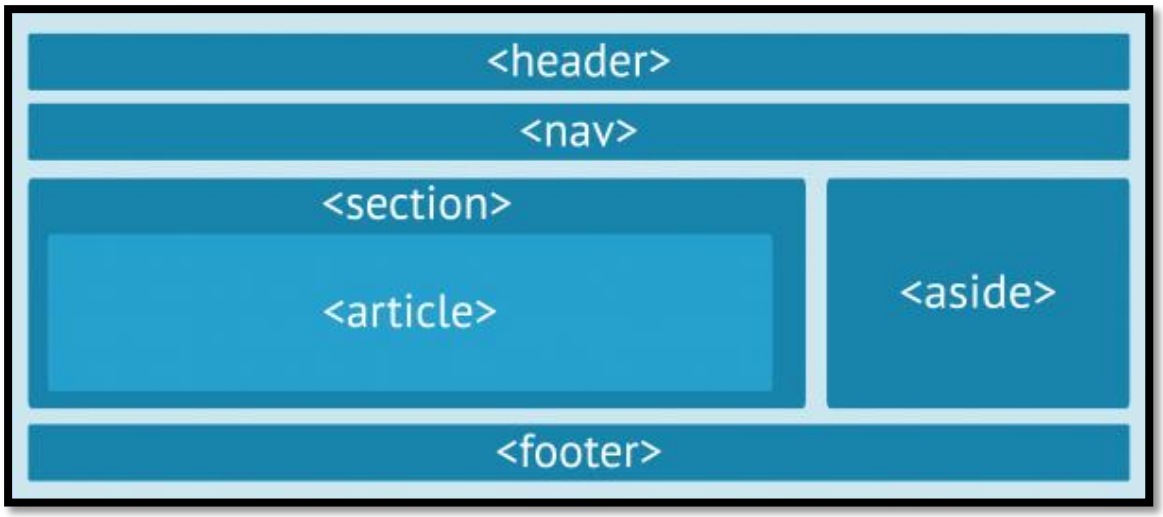


Figura 2. Estructura común de un HTML5

HTML5 ofrece una mejora semántica sobre las etiquetas; así, etiquetas como <header> guarda todo lo relacionado con los encabezados de la página y etiquetas como <nav> guardan elementos de navegación. Esto, además de mejorar la comprensión a la hora de observar un código HTML, supone una ventaja para las técnicas de SEO mejorando la indexación a los buscadores.

Las vistas que se han creado en el desarrollo de de esta aplicación están totalmente construidas con HTML5 junto al framework Lungo el cual extiende las propiedades que se pueden añadir a las etiquetas HTML permitiendo hacer estructuras complejas con poco código.

2.1.3 CSS

CSS [3] (por sus siglas en inglés Cascading Style Sheets) es un lenguaje creado para definir estilos visuales asociados a documentos HTML y XML. Estos estilos describen cómo se verá por pantalla un elemento asociado a este estilo, como por ejemplo el color, su borde, tamaño, alineación, opacidad, etc. Se trata de un estándar publicado en 1996 al cargo de la W3C. La versión actual es la 3 e incorpora importantes mejoras y funciones avanzadas que permiten dotar a las páginas web de mayor dinamismo y efectos como el atributo “overflow” o “box-shadow” entre muchos.

El principal objetivo de CSS es separar el código estructural del documento de su presentación, manteniendo una definición de estilos y asociando los elementos de la página con estas definiciones.

Un CSS se compone de reglas. Cada una de estas reglas está compuesta de dos partes: el selector y la declaración. La función del selector es indicar a qué elementos del documento se aplicará la regla y la declaración contiene el conjunto

de propiedades que se aplicarán al elemento. Cada una de estas propiedades serán las que les den un determinado estilo al objeto asociado.

Las reglas se aplican al documento automáticamente o no, dependiendo de cómo se especifique el selector. Si el selector implica nombres de etiquetas el navegador asignará automáticamente el estilo a todas esas etiquetas del documento. Si el selector hace referencia a una determinada clase, sólo se aplicará a aquellos tags que explícitamente pertenezcan a esa clase mediante el atributo “class”.

Los estilos CSS se pueden definir en un documento aparte, en la misma página de contenidos o aplicarlas directamente sobre las etiquetas bajo el atributo “style”, siendo esta última práctica la menos recomendada debido a su alto acoplamiento y poca reutilización.

Lungo, el framework usado como soporte para las vistas de esta aplicación, hace un uso importante de CSS3, incorporando en su librería su propia hoja de estilos diseñados para dar un aspecto de interfaz apropiada para móviles y tablets. Esto ha facilitado que el desarrollo de las vistas se lleve a cabo con un menor esfuerzo al no tener que recurrir al diseño visual de cada elemento, dando como resultado una interfaz coherente y agradable. Aunque eso no quita que haya tenido que añadir una hoja de estilos propia para reescribir o incorporar ciertos aspectos visuales que se adapten más a las necesidades de la aplicación.

2.1.4 SQL

SQL [4] (Structured Query Language) es un lenguaje de tipo declarativo de alto nivel de abstracción, que permite el acceso y manejo de bases de datos relacionales. SQL nace en 1986 (año de estandarización) siendo una evolución del lenguaje SEQUEL (Structured English Query Language) el cual fue originado por IBM. Su utilización se basa en sentencias que crean o modifican la estructura e información contenida en una base de datos.
SQL es un lenguaje muy potente y maduro. Cuenta con una gran popularidad y su uso es muy extendido formando parte de SGBD como Oracle o MySQL.

Este lenguaje ofrece una gran variedad de operaciones para tratar las bases de datos, dividiéndose estas en dos grupos principales:

- DDL (Data Definition Language). Este conjunto de sentencias de SQL se encarga de crear y modificar la estructura de la base de datos junto a sus relaciones. Unas de las operaciones básicas a destacar de este grupo son CREATE para crear tablas, ALTER para modificar una tabla existente, DROP para eliminar tablas y TRUNCATE para vaciar todo el contenido de registros de una tabla.

- DML (Data Manipulation Language). Estas sentencias permiten el acceso y la manipulación de los registros contenidos en las tablas. Varias operaciones básicas de este grupo son SELECT para recuperar información almacenada de las tablas, UPDATE para actualizar registros, DELETE para borrar registros e INSERT para insertar un nuevo registro.

- DCL (Data Control Language). El principal propósito de este sub-lenguaje es conceder o revocar permisos de acceso y control de los datos y tablas contenidos en la base de datos. El uso en este proyecto de este tipo de instrucciones ha sido nulo, debido a que la implementación de Web SQL Database está hecha con SQLite el cual carece de soporte para acceso y control de usuarios [5].

Debido al uso que se ha hecho de Web SQL en este proyecto, este lenguaje de consultas ha formado una parte importante del mismo. La implementación que se ha llevado a cabo para la persistencia de datos recae en el uso de SQL como lenguaje de acceso a los datos.

2.1.5 UML

UML [6,7] (Unified Modeling Language) es un lenguaje visual diseñado para el modelado de sistemas software. Se aprueba como estándar en 1997 por el grupo OMG (Object Management Group) en su versión 1.1. Actualmente está en su versión 2.5 de forma oficial, recientemente lanzada en Junio de 2015.
Este lenguaje tiene como objetivo ayudar en el desarrollo del software ofreciendo una serie de herramientas gráficas basadas en esquemas y objetos que permiten diseñar conceptualmente distintos aspectos del proceso de modelado.
Además de las herramientas gráficas, UML incorpora OCL (Object Constraint Language), un lenguaje formal que permite añadir restricciones sobre modelos UML y que ayuda a especificar aspectos relevantes del sistema que de manera gráfica no es posible alcanzar.

Algunas de las ventajas que se obtienen del uso de UML son:

- Lenguaje formal estandarizado y altamente aceptado por la industria.
- Amplia gama de herramientas para abarcar todas las características principales del modelado.
- Gran flexibilidad.
- Independiente de la tecnología.
- Capacidad para modelar en distintos niveles de abstracción.
- Validación por parte del usuario. UML posee un nivel de abstracción lo suficientemente alto como para ser entendible por stakeholders que no posean conocimientos técnicos.

- Está basado en el concepto de orientación a objetos, por lo que cuenta con los beneficios que ofrece este tipo de paradigma.
- Reduce los tiempos de desarrollo de software.
- Sirve como documentación para el proyecto.
- Independiente de la metodología de desarrollo.

UML contiene 14 tipos de diagramas que se dividen en 3 categorías:

- **Diagramas estructurales**, que se encargan de modelar la estructura desde distintas perspectivas, como por ejemplo los diagramas de clases o los diagramas de componentes.
- **Diagramas de comportamiento**, enfocados en describir el comportamiento dinámico de partes del sistema mediante esquemas como los diagramas de máquinas de estado o diagramas de flujo.
- **Diagramas de interacción.** Se tratan más bien de un subconjunto de los diagramas de comportamiento, pero estos hacen una especial enfatización en la interacción entre objetos y sus mensajes e intercambio de información.

2.2 Herramientas

En los siguientes apartados se enumeran y detallan las herramientas usadas como apoyo en los procesos y en la implementación de la aplicación.

2.2.1 Sublime Text

Sublime Text [8] es un editor de texto especialmente diseñado para multitud de lenguajes. Fue lanzado en 2008 y actualmente está en su versión estable 2.0.2 (la versión usada en este proyecto) junto a la versión 3 beta. Se trata de un software propietario que posee una versión gratuita. Está disponible para plataformas Windows, Linux y Mac OS.
Debido a su elegante estilo visual y al gran conjunto de herramientas que incorpora, se ha convertido en un editor muy popular.

Entre sus características cabe destacar:

- Coloreado de código para
- Resaltado de los cierres de llaves y paréntesis al señalarlos permitiendo una visión cómoda y rápida de la búsqueda de cierres.
- Autocompletar. Al no tratarse de un IDE carece del potencial del autocompletado de IDE’s como por ejemplo Eclipse. Pero igualmente tiene un buen uso y ayuda a la hora de escribir el código.
- Selector múltiple. Permite seleccionar varias líneas y poder editarlas a la misma vez.

- Liviano tanto en tamaño (de 5 a 7 MegaBytes) como en tiempos de carga.
- Ampliable mediante plugins.

A la hora de elegir el editor de HTML5, CSS3 y JavaScript para el proyecto se barajó optar por otros editores como Notepad++ o Brackets. Este último es creación de Adobe con licencia de software libre y tiene una apariencia y funcionalidades parecidas a Sublime Text, salvo por ciertas funcionalidades que incorpora Brackets, entre las que se puede destacar "Live Preview" que permite ver los cambios directamente en el navegador sin recargar la página.
Pero por la sencillez de uso, su rapidez y por todas las herramientas tan útiles mencionadas de Sublime Text se decidió usar en este proyecto como herramienta de escritura de código.

2.2.2 REM

REM [9] (REquirements Management) es un software de gestión de requisitos basado en XML y XSLT que genera documentación HTML. Creado en la tesis doctoral "Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información" de Amador Durán Toro para la Universidad de Sevilla.
Es una herramienta de uso académico gratuito; para adquirirla se debe de registrar en la página de la universidad de Sevilla y adquirir un código de descarga.

REM es una herramienta ligera y flexible, pudiéndose adaptar a diseños específicos mediante plantillas. Incorpora una interfaz intuitiva diferenciándose tres partes principales, el menú, el navegador de documentos a la izquierda y como marco principal el visor del documento.

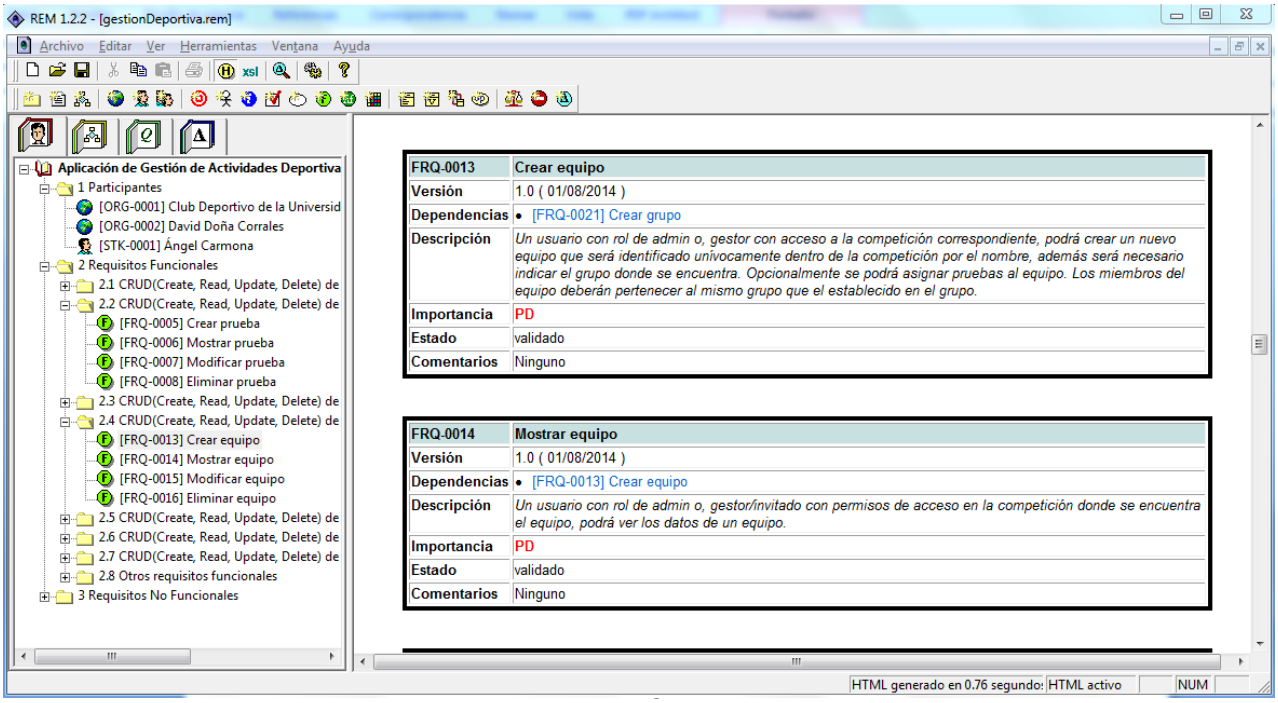


Figura 3. Ventana principal de REM

Posee 4 tipos de documentos para gestionar los requisitos:

- ***Documento de requisitos del sistema***. Permite crear un documento en el cuál se podrá desarrollar el conjunto de los requisitos.
- ***Documento de análisis del sistema.*** Este tipo de documento permite agrupar información de análisis sobre el sistema la cual es especialmente útil para tener un desarrollo inicial y un concepto general del la solución a desarrollar; ofreciendo para ello casos de uso, objetos, actores, clases de negocio y otros artefactos más.
- ***Registro de conflictos y defectos***. Este documento tiene como objetivo gestionar la parte de conflictos y defectos que pueden aparecer en los requisitos (proceso ligado a la calidad del software) ayudando a documentar, por ejemplo, problemas sobre contradicciones entre requisitos o ambigüedad.
- ***Registro de peticiones de cambio***. Permite gestionar las posibles peticiones de cambio de requisitos que puedan surgir en el proyecto.

En un primer momento a la hora de pensar qué herramienta se podría utilizar para gestionar los requisitos, se pensó en Cameo Requirements+ junto con su licencia académica. Pero su uso en la carrera más bien me causó una experiencia negativa por el hecho de que tuvo un funcionamiento (lo suficientemente) inestable y era un programa con bastante carga computacional, por lo que me decidí a buscar otras alternativas que además fueran gratuitas.
Tras mucho investigar, y debido a la escasez de software libre o gratuito ofertado, me decanté por la herramienta REM que si bien no tiene una gran cantidad de herramientas como otros productos software propietarios, ofrece en proyecto lo suficiente para gestionar los requisitos.
Cabe decir que para su uso he adaptado la plantilla original a las necesidades de este proyecto, eliminando algunas partes de la tabla de descripción de requisitos, como por ejemplo la columna “fuentes” o “autor”.

2.2.3 TRELLO

Trello [10] es una aplicación web para la organización de proyectos de diferente índole inspirada en el sistema Kanban, creado en 2011 por la compañía Fog Creek Software. Ofrece soporte para acceder desde Windows Phone, Android y iOS tanto como versión de aplicación como mediante el navegador de móvil. Incorpora además un entorno multiusuario con el que se puede asignar tareas a diferentes usuarios, entre otras acciones. En ingeniería del software es especialmente útil para metodologías ágiles, como por ejemplo Scrum.

Algunas de sus características destacables son:

- Facilidad de uso gracias a una interfaz visual muy intuitiva.
- Posee una versión gratuita.
- Dispone de una API pública para poder integrarse con otros elementos.
- Capacidad de integrarse con cualquier tipo de proyecto.
- Orientado a un entorno colaborativo.

Trello se organiza en **tableros** (boards) que representan proyectos.
Dentro de cada tablero se distribuyen las **listas de tareas** (task lists) donde cada tarea es representada mediante una **tarjeta** (card). Cada tarjeta puede contener diversa información extra, como fecha de vencimiento, documentos adjuntos, comentarios,…
La idea del uso de varias listas es reflejar los diferentes estados por los que pasan las tareas, permitiéndose en la aplicación pasar las tarjetas de una lista a otra para tal fin.

En este proyecto se ha hecho uso de esta aplicación como herramienta de apoyo para la planificación.

2.2.4 NinjaMock

NinjaMock [11] es una herramienta web especialmente preparada para diseño de mockups de interfaces móviles, aunque también ofrece soporte para diseño web. Tiene una licencia gratuita y otras de pago que añaden ciertas características, como por ejemplo entorno para multiusuario.

Permite diseños basados en estilos de Android, iPhone, Windows Phone junto a otros más, ofreciendo un entorno basado en proyectos lo que permite guardar los diseños y gestionarlos desde la misma aplicación.
Entre sus funcionalidades se puede destacar también la posibilidad de exportar los diseños tanto en imágenes como en pdf.

Se trata de una herramienta bastante completa y suficiente en su licencia gratuita que ha agilizado mucho el proceso de diseño de las vistas de esta aplicación.

2.2.5 MagicDraw

MagicDraw [12,13] es un software CASE (Computer Aided Software Engineering) perteneciente a la empresa No Magic que congrega una serie de herramientas para modelar distintos aspectos de un sistema basándose en UML. Ofrece la creación de una amplia gama de diagramas, entre los que se puede destacar:

- Diagramas de clases.
- Diagramas de secuencia.
- Diagramas de actividad.
- Diagramas de colaboración.
- Diagramas de casos de uso.
- Diagramas de componentes.
- Diagramas de máquinas de estado.

MagicDraw da soporte a otro tipo de funcionalidades, tales como:

- **Generación de código en MDD** (Model-Driven Development). Esto permite, por ejemplo, generar a partir de un diagrama de clases artefactos específicos para distintos lenguajes.
- **Trabajo colaborativo**. Mediante la herramienta MagicDraw's Teamwork Server distintas personas pueden trabajar sobre un mismo proyecto simultáneamente.
- **Extensión mediante plugins**.
- **Capacidad de integración con otras herramientas**, como es el caso de Cameo Requirements+ (perteneciente a la misma compañía), o herramientas externas como por ejemplo Eclipse.

Para su uso en este proyecto se ha utilizado la licencia académica (Personal Edition) que ofrece la UMA, en la versión 17.0.2. Los diagramas de secuencia, de clases y de casos de uso que se presentan en este documento han sido diseñados con esta herramienta. Junto a MagicDraw se ha utilizado un plugin llamado magicUWE. Este plugin está basado en la especificación UWE [14] (UML-based Web Engineering), que extiende el lenguaje UML con nuevos elementos diseñados para aplicaciones web. Mediante este plugin se ha diseñado también un diagrama de navegación.

La elección de esta herramienta para el modelado ha sido porque, frente a otras alternativas gratuitas como es el caso de Umbrello, se trata de una herramienta muy potente y completa.

2.2.6 Phonegap Build

Phonegap Build [15] es un servicio basado en la nube proporcionado por Adobe para la compilación de proyectos Cordova/Phonegap.
Su principal característica es la de ofrecer un entorno simple donde se pueda compilar en la nube un proyecto con esta tecnología sin necesidad de hacer de forma manual el proceso. Esto evita tener que gestionar dependencias de plugins y mantener SDK's para cada tipo de S.O. móvil en el PC.

Este servicio ofrece distintos tipos de cuentas. La cuenta gratuita permite tener un número ilimitado de proyectos en la nube aunque sólo uno se permite tener como proyecto privado (además de otro tipo de restricciones). Los demás serán automáticamente licenciados como open-source.

El proceso de compilación en la nube se basa en subir todo el proyecto comprimido en un zip a la web junto con un XML dentro que especifica la configuración y plugins que se han utilizado en el proyecto. PhoneGap Build a continuación se encarga de añadir las dependencias y compila de forma automática para los sistemas iOS, Windows Phone y Android.

Este servicio ha sido seleccionado para el proyecto por todas las ventajas que se han mencionado anteriormente ayudando a agilizar el desarrollo gracias a la obtención de versiones de la aplicación de forma inmediata.

2.2.7 MySQL Workbench

MySQL Workbench [16] es una herramienta perteneciente a Oracle que integra diseño y gestión de base de datos de MySQL. Este software es un sucesor de DBDesigner4 con versiones open-source y comercial.
Permite elaborar diagramas de bases de datos de forma visual, hacerles ingeniería inversa y además sincronizarlos con una base de datos real.

MySQL Workbench ofrece una gran variedad de funciones, pero su uso en este proyecto se ha limitado al diseño visual de la base de datos y su posterior ingeniería inversa para obtener un archivo SQL que contenga las sentencias de creación de dicha base de datos. La versión usada de este software ha sido la 6.0 CE.

2.2.8 GapDebug

GapDebug [37] es una herramienta que permite depurar las aplicaciones basadas en Phonegap que están instaladas en el teléfono móvil. Se apoya en Google Chrome para la depuración (de hecho requiere tener instalado dicho navegador) por lo que su función básica es comunicar la aplicación que se está ejecutando en el móvil con Google Chrome.

Al ser una aplicación web lo que se está ejecutando dentro de un webview y estar asociado a ese webview Google Chrome, esto permite depurar la aplicación instalada mediante Chrome como si de una aplicación web normal se tratase, con todas las ventajas que esto añade y que se detallan en la descripción de la herramienta Chrome. Otras funcionalidades que permite GapDebug son: instalación “drag and drop” de las aplicaciones en el móvil y screenshots.

Una desventaja que tiene es que no está preparado para depurar las aplicaciones en un Windows Phone.

Mediante GapDebug se ha podido realizar en este proyecto una parte muy importante en él, que es la depuración del programa en el dispositivo real.

2.2.9 Chrome

Google Chrome [38], además de ser una aplicación para la navegación web, dispone de un conjunto de herramientas muy potentes para la inspección y depuración del código de aplicaciones web. Para ser accedida esta herramienta a través del navegador se pulsa botón derecho sobre cualquier elemento web y a continuación se pulsa en “Inspeccionar elemento”, o también se puede acceder pulsando la tecla F12.

Algunas de las funcionalidades principales que Chrome incorpora para los desarrolladores web son:

- **Inspector de elementos**. Permite señalar cualquier elemento de la página con el ratón y automáticamente visualizarse todo el código relacionado con él.
- **Modo dispositivo**. Este modo permite mostrar la aplicación en un marco simulando distintos tamaños de dispositivos móviles, además de personalizar otras características avanzadas como por ejemplo la simulación de tipos de red (gprs, 3G…).
- **Elements**. Esta opción muestra el código HTML que se está visualizando.
- **Network**. Permite ver el tráfico de datos de la web.
- **Source**. Este apartado permite visualizar y depurar de forma muy completa toda la lógica de la aplicación web.
- **Resources**. Muestra todo lo referente al almacenamiento de la web (Web SQL, Session Storage,…) que se está visualizando.
- **Console.** Muestra información de diferente tipo, entre lo que cabe destacar la consola de JavaScript.
- **Edición dinámica de código.** La herramienta de desarrollador permite editar prácticamente todo el código que existe en la página web mostrada (HTML, CSS, JavaScript,...) y ver los cambios de forma instantánea.

Junto a Chrome también se ha hecho uso del plugin Ripple, que proporciona un entorno de simulación de móviles con cualidades parecidas a las del propio navegador, pero que está especialmente diseñado para el framework Phonegap.

El uso de Google Chrome como herramienta de depuración en este proyecto ha sido esencial y de gran ayuda. Durante la fase de implementación de la aplicación las pruebas, visualización y depuración del software se han llevado a cabo primero

desde el navegador, esto es porque la aplicación está compuesta de HTML, CSS y JavaScript; permitiendo con esta acción un rápido visionado y desarrollo en comparación con tener que compilar e instalar en el móvil a cada momento.

Existe otra alternativa igual de potente, que es el Firebug de Mozilla, pero debido a ciertas peculiaridades incompatibles de este proyecto (uso del framework Lungo y de Web SQL), no ha sido posible usarlo.

2.2.10 XAMPP

XAMPP [17] (X Apache MySQL PHP Perl) es un servidor e intérprete de Apache, MySQL, PHP y Perl construido bajo licencia de software libre. Fue desarrollado por Apache Friends y publicado en el 2002, siendo su versión actual la 5.6.14. Es un sistema multiplataforma (la 'x' en el nombre es por esta propiedad) cuya principal cualidad es su fácil y rápida instalación.

Su uso en este proyecto se ha limitado a la funcionalidad de servidor web para las pruebas de la aplicación en PC. La decisión de su uso ha sido porque Chrome deshabilita el acceso a archivos cuando una página web está siendo visionada en un modo "offline" y, debido al importante uso de la API Web SQL y Lungo (los cuales requieren acceso a archivos), se ha optado por esta opción. Se barajó inicialmente usar Firefox como navegador y depurador, pero este no cuenta con la API Web SQL Database.

2.3 Frameworks y API's

El uso de Frameworks y API's en el proyecto ha jugado un papel muy importante en el desarrollo. Se describen en los siguientes capítulos cada uno de ellos.

2.3.1 Cordova/Phonegap

Phonegap [18] es un framework para el desarrollo de aplicaciones móviles multiplataforma hechas en HTML, CSS y JavaScript. Fue creado en 2008 por la compañía Nitobi y liberado como software libre bajo licencia Apache en 2011. Poco después, Adobe compró la compañía Nitobi y con ella la marca Phonegap.

El hecho de que al mismo producto se le llame Cordova o Phonegap indistintamente es porque realmente se trata del mismo software, salvo que Adobe mantiene una marca propia para poder explotar y añadir herramientas propias al

framework. Apache, para diferenciarlo por los motivos anteriormente mencionados, decide nombrar la distribución como Cordova.
Ambos productos son lo mismo; son open-source y contienen el mismo código, sólo se diferencian en el nombre. Pero debido al uso que se ha hecho de la herramienta específica de compilación en la nube perteneciente a Adobe, se le referirá en el resto de la memoria sólo como Phonegap.

Este framework, como ya se menciona en la introducción, resulta de gran ayuda a la hora de implementar una aplicación para distintos S.O. móviles. Su principal bondad es que permite desarrollar para las principales plataformas a través de un mismo código, reduciendo así los costes de tiempo y esfuerzo notablemente.

Las aplicaciones que da como resultado la compilación bajo Phonegap son aplicaciones híbridas. Esto significa que, ni son aplicaciones que están hechas con el código nativo del dispositivo, pero tampoco se tratan de aplicaciones web exactamente, pues tienen acceso a recursos internos del dispositivo.

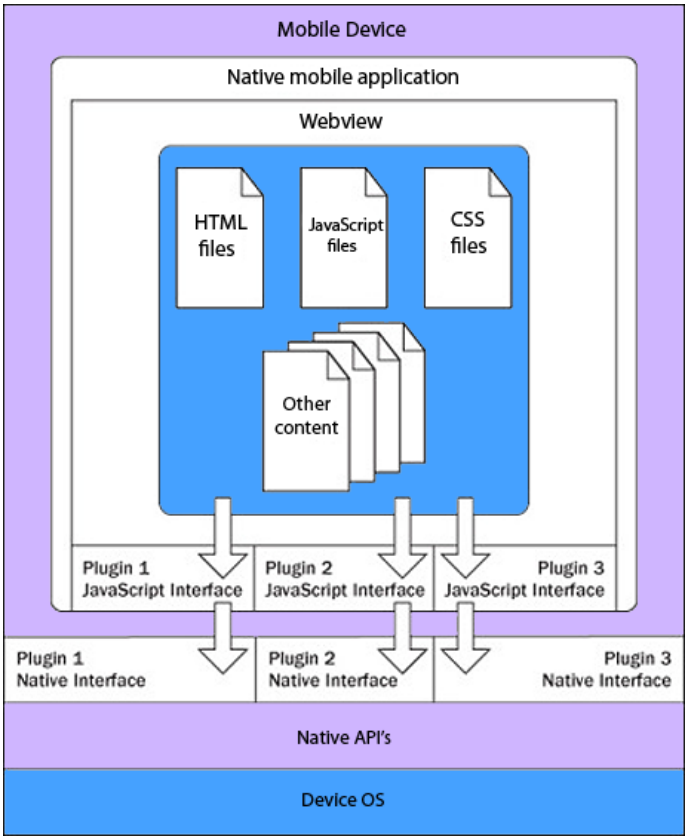


Figura 4. Arquitectura Phonegap

Phonegap actúa como intermediario entre la aplicación basada en web y el dispositivo (figura 4). Cuando se requiere acceder a determinadas funciones (cámara, contactos, almacenamiento…), la aplicación web (que básicamente está iniciada en un webview) hace uso de una librería JavaScript que aporta el framework

la cual sirve como API entre la propia aplicación web y el Framework. Luego, Phonegap se encarga de obtener los recursos pedidos comunicándose con el dispositivo en el lenguaje propio de cada sistema.

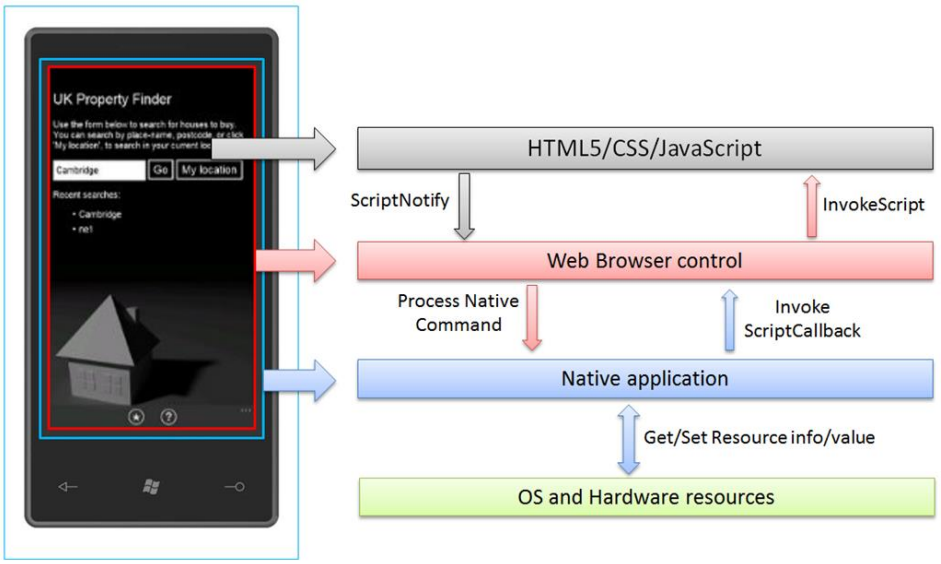


Figura 5. Ejemplo de funcionamiento interno de Phonegap

2.3.2 Lungo

Lungo [19] es un framework escrito en JavaScript para desarrollar páginas webs basadas en HTML5, CSS3 y JavaScript. Este software fue creado por TapQuo, una pequeña empresa española y es un proyecto de código abierto bajo licencia GPLv3. Se originó en 2010 y actualmente está en la versión 2.2. Está orientado principalmente para móviles, aunque posee un robusto diseño adaptativo el cual permite que se pueda ver correctamente en cualquier tamaño de pantallas. Requiere de la librería QuoJS, creada también por la misma empresa.

Lungo establece una estructura de diseño y maquetación basado en HTML5 y atributos especiales propios del framework. Esto permite crear de una manera más sencilla estructuras visuales complejas, tales como menús, listas, formularios, “pull and refresh”, etc. También incorpora una API para manejar el DOM junto a otras funcionalidades que dotan a este framework de unas capacidades suficientes para crear aplicaciones web potentes.
Una característica interesante de Lungo es que permite cargar elementos sobre la misma página de forma asíncrona mediante el atributo “data-async”, esto evita tener que navegar a otras páginas y por ende mejora el tiempo de respuesta.

Como se ha mencionado antes, lungo está orientado principalmente para entornos móviles, por lo que todos los elementos vienen preparados para responder ante eventos de tipo “tap”, “swipe”, “hold”, etc.

El uso de Lungo en este proyecto ha sido totalmente enfocado en el desarrollo del front-end, evitando el uso interno de sus librerías con el motivo de que cualquier desarrollador en el futuro pueda cambiar el diseño de vistas por otro alterando lo mínimo la parte del controlador de la aplicación.

Aunque resulta ser una herramienta muy completa, aún tiene algunas particularidades de funcionamiento que son mejorables, especialmente el sistema de eventos, el cual me ha hecho más de una vez investigar durante horas el código interno para sacar información sobre algunos de sus comportamientos.

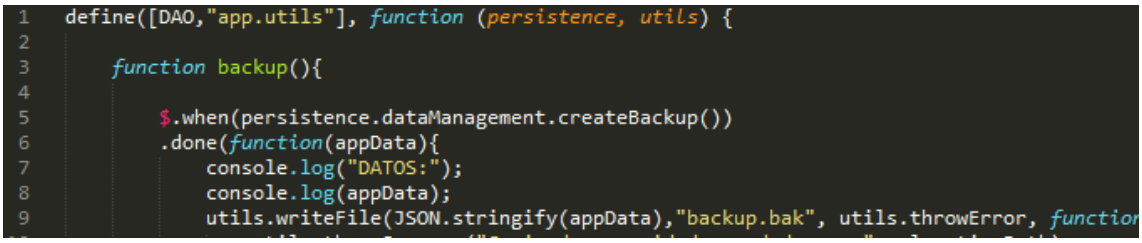
2.3.3 RequireJS

El uso de JavaScript requerido por este proyecto, junto a otros elementos, ha resultado (casi) nuevo para el proyectando. Ha sido un lenguaje apenas usado durante la carrera por lo que he tenido que estudiar su uso durante el proyecto (cosa que ha sido muy enriquecedora). El aprendizaje continuo durante el desarrollo me ha llevado a replantear en algunas ocasiones la forma de implementar ciertas partes del código. Uno de estos casos es el planteamiento del sistema de modulación.

JavaScript resulta ser; para los que estamos acostumbrados a lenguajes de programación fuertemente tipados y con una estructura bien definida (como por ejemplo Java), un lenguaje cuyo código es fácil de desestructurarse y emborronarse. Es por eso que primeramente me planteé dividir en archivos cada componente lógico de la aplicación y así evitar en lo posible archivos grandes y con poca cohesión. Pero esto me llevó a otros dos problemas; uno es que al tener que añadir todos los scripts mediante links terminaban “ensuciando” el código HTML y el otro problema es el orden en que debían de ser añadidos ya que el código añadido de JavaScript se procesa secuencialmente como HTML (no hay un pre-análisis de dependencias), y se corría el riesgo de posibles dependencias circulares.
La solución la encontré en RequireJS.

RequireJS [20] es un framework creado en 2011 basado en la especificación CommonJS que permite implementar programación modular a través de dependencias. Tiene compatibilidad con los principales navegadores y es open source. Su uso se basa en crear módulos nombrados, en los que se especifican sus dependencias con otros módulos.

Un ejemplo de declaración de módulo es el siguiente:



```
define([DAO,"app.utils"], function (persistence, utils) {

    function backup(){

        $.when(persistence.dataManagement.createBackup())
        .done(function(appData){
            console.log("DATOS:");
            console.log(appData);
            utils.writeFile(JSON.stringify(appData),"backup.bak", utils.throwError, functio
```

Figura 6. Ejemplo de definición de módulo

Mediante “define” se declara el módulo. Los argumentos que recibe esta función son los siguientes (por orden):

- **Nombre del módulo** (opcional, si no se especifica toma el nombre del archivo).
- **Array de dependencias** (opcional).
- **Función con el contenido del módulo**. Esta función contendrá dentro todo el código del módulo. Además, se le pasará como parámetros las dependencias indicadas en el array de dependencias para ser accedidos como objetos dentro de dicha función.

Existe otras configuraciones para definir un módulo, pero la anterior descrita es la más usada en el proyecto y la más común.

Además de la forma explicada anteriormente de añadir dependencias en la declaración de un módulo, se puede hacer una llamada desde cualquier parte del código mediante la función “require”.

Cada módulo debe exportar los elementos (funciones, variables) a los que podrán acceder otros módulos. Esto se hace retornando un objeto con las propiedades que se desean exportar.

Como característica importante, RequireJS implementa el patrón AMD [21] (Asynchronous Module Definition) cuya finalidad es poder descargar los módulos de forma asíncrona, permitiendo agilizar la carga de archivos bajo demanda y dar tiempos de respuesta más rápidos (generalmente).

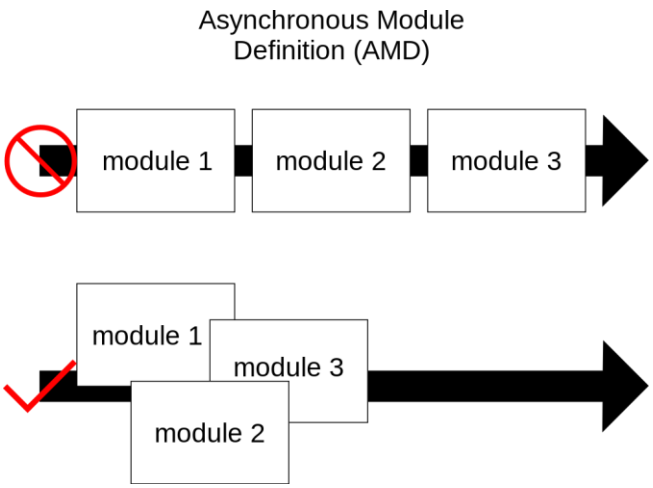


Figura 7. Funcionamiento de AMD

Algunas de las ventajas que se obtienen al usar el patrón modular con RequireJS son:

- Se mantiene una API pública, protegiendo funciones y variables internas (acercándonos así a un concepto parecido a las clases mediante la encapsulación).
- Permite hacer código reusable.
- Se mantiene el namespace de la aplicación más limpio.
- Se carga sólo los scripts que se necesitan en cada momento, no todo el código.
- Separar más el código JavaScript del HTML.

Otras alternativas a RequireJS que se encontraron son Browserify, Shepherd.js y CommonJS.
Shepherd.js está basado en la nueva especificación de ECMAScript 6 en la que se dota a JavaScript de gestión de módulos de forma nativa (una especificación muy reciente y aún no implementada por la mayoría de navegadores).
CommonJS es otro módulo JavaScript también para dar soporte a módulos y en el que se basa la herramienta usada en este proyecto.

El hecho de que eligiera RequireJS en lugar de las demás es por ser una herramienta muy aceptada por los desarrolladores y madura; y a pesar de que RequireJS se basa en la especificación de CommonJS, este último está orientado a su uso en Node.js, al contrario que RequireJS cuya implementación está pensada para aplicaciones webs (y recordemos que esta aplicación está basada en tecnología web).

2.3.4 Web SQL Database

Web SQL Database es una especificación de la W3C para el uso de base de datos relacionales en páginas webs del lado del cliente. Está soportada por la mayoría de los navegadores actuales y, como su nombre indica, está basada en el lenguaje SQL.
A pesar de que la W3C dejó de dar soporte a Web SQL desde 2010, sigue teniendo un uso importante y en Phonegap se utiliza como base de datos relacional, dando soporte completo para esta implementación.

Se trata de una BBDD que se aloja en el lado del navegador del cliente, siendo persistente después del cierre del navegador. La API que implementa es asíncrona, por lo que el acceso a sus resultados se basa en “callbacks”.

Aun siendo una base de datos bastante potente, carece de ciertas funcionalidades como el control de usuarios, simplificándose el control de acceso sólo al origen de creación de la BBDD, tal como lo hace por ejemplo Local Storage.

La mayoría de los navegadores implementan esta especificación con SQLite, por lo que se cuenta con la ligereza y rapidez de este sistema.

Como alternativa de uso para este proyecto a Web SQL Database se encontró a IndexedDB, pero esta última se trata de una base de datos no relacional, por lo que no es compatible con la implementación deseada del proyecto, además de que no está soportada de forma nativa por muchos navegadores (no hay que olvidar que Phonegap utiliza siempre un navegador interno para mostrar la aplicación).
Otra alternativa muy competitiva fue Google Lovefield, pero tratándose de una librería JavaScript relativamente nueva, ofreciendo básicamente lo mismo que Web SQL, me decanté por esta última.

2.3.5 QUnit

QUnit [22] es un framework para hacer pruebas de unidad hecho en JavaScript desarrollado por el equipo de JQuery bajo licencia MIT. Se trata de una herramienta muy flexible con una amplia gama de funcionalidades para el testing, permitiendo ejecutar una gran cantidad de tipos de asertos.
Posee una API muy intuitiva y los resultados de las pruebas se muestran en un HTML con la información de resultados bien distribuida y cómoda de mirar, permitiendo filtrar por módulos y otros tipos de propiedades.

Figura 8. Ejemplo de resultados de pruebas en QUnit

La configuración básica que requiere QUnit para ejecutar las pruebas son:

- Página HTML con la maquetación que especifica el propio QUnit para mostrar los resultados.
- Biblioteca de QUnit enlazada en el HTML.
- Incorporar en el HTML el código JavaScript a ser testeado.
- Incorporar en el HTML las pruebas diseñadas con la librería QUnit.

Las pruebas que se han hecho bajo QUnit se están divididas en módulos utilizando RequireJS, de este modo todas las pruebas quedan mejor organizadas.

Junto a QUnit se ha utilizado un plugin llamado **Blanket** con el cual se ha podido hacer análisis de cobertura de código.

Como alternativa a QUnit para este proyecto se barajó el uso de Jasmine, y, teniendo ambos características similares, elegí QUnit porque su forma de redactar las declaraciones de pruebas me parecieron más intuitivas, además de ser un framework con más años en el mercado.

2.4 Librerías

Por medio de las librerías (o también llamadas bibliotecas) se ha podido implementar ciertas funcionalidades de la aplicación y realizar tareas complejas con menos esfuerzo. Se detallan a continuación cada una de las que se han utilizado.

2.4.1 Zepto

Zepto es una biblioteca JavaScript open source para la manipulación del DOM, manejo de eventos y, en general, para la interacción con los documentos HTML. Su API está totalmente basado en el API de JQuery, siendo el objetivo de uso de ambos el mismo y pudiéndose cambiar de una biblioteca a otra sin necesidad de modificar el código. La principal característica que diferencia a Zepto de la biblioteca en la que se inspira (JQuery), es su minúsculo peso, poco más de 5 Kilobytes (en su versión básica). Otra característica interesante es que posee una arquitectura modular, pudiéndose elegir que elementos incorporar a la biblioteca y así adaptar su volumen a lo que estrictamente se necesita.

La razón principal para el uso de una biblioteca como Zepto es que permite hacer en pocas líneas funciones que de ser implementadas sin la herramienta supondrían muchas líneas y código complejo.

Lungo incorpora una biblioteca (la cual la necesita para poder funcionar) que se llama Quo.js, la cual contiene también herramientas muy parecidas a Zepto, pero se ha evitado su uso en el proyecto con el objetivo de mantener la mayor independencia entre el controlador y las vistas.

2.4.2 jsPDF

jsPDF [23] es una librería con licencia MIT que permite imprimir PDF’s desde el lado de un cliente vía JavaScript. Su API incorpora un conjunto de funciones que permiten añadir al documento imágenes, textos, figuras y otras funciones básicas. Además añade una función bastante interesante que permite renderizar textos expresados en HTML en texto para el PDF.

Para implementar la funcionalidad de imprimir PDF’s se ha utilizado esta herramienta en conjunto con un plugin diseñado para ella llamado **jsPDF-AutoTable** (también con licencia de código abierto), el cual extiende la funcionalidad permitiendo generar tablas fácilmente.

Existe una alternativa a jsPDF igual de viable, que es la biblioteca pdfmake. Ambas tienen características muy parecidas y podría haber optado igualmente por la otra.

2.4.3 SweetAlert

SweetAlert [24] es una pequeña librería con licencia MIT escrita en JavaScript que permite lanzar ventanas modales muy atractivas y altamente configurables. Se pueden crear muchos tipos de ventanas modales, tanto de información como de entrada de datos, y todo bajo un diseño responsivo.

El framework Lungo usado en este proyecto incorpora ya un módulo de ventanas modales y es con ese módulo con el que se empezó el proyecto. Pero debido a unos requisitos especiales de diseño que eran necesarios para las ventanas modales utilizadas en la parte de edición de registros, SweetAlert se adaptaba mejor que el módulo de Lungo.

3. ESPECIFICACIÓN DE REQUISITOS

Cuando se trata de describir qué debe de hacer un producto, el recurso más importante en la ingeniería del software es la obtención de un documento de especificación de requisitos software (ERS), el cual recoge todas las necesidades que el cliente requiere del producto.
El documento ERS debe de contener de manera concisa e inequívoca toda la información relevante para que el diseño y desarrollo se pueda llevar a cabo.
Esta información además sirve como consenso entre las partes cliente y desarrollador, teniendo que ser validado por ambas partes.

Existen diversas propuestas sobre los contenidos que debe de comprender un ERS, siendo de los más importantes el documento 830-1998 propuesto por el instituto IEEE.

Este capítulo reúne toda la información relacionada con los requisitos que debe de cumplir este software, además de los actores y participantes involucrados en él. Si bien no comprende toda la información requerida en el documento ERS propuesto por la IEEE, abarca toda la información que se ha necesitado para llevar a cabo el desarrollo.

3.1 Definiciones y abreviaturas

A lo largo de este documento y dentro de la aplicación se utilizan ciertas palabras y abreviaturas que pertenecen al ámbito de la aplicación y que resulta conveniente definirlas para su buen entendimiento.
A continuación se separan en tablas distintas aclaraciones en orden alfabético.

Definiciones

Asignación	Hace referencia a la forma en la que se ha asignado una prueba. La asignación puede ser de tipo manual o aleatoria.
Competidor	Sinónimo de participante.
Dorsal	Numeración identificativa que se le asigna a un competidor.
Equipo	Conjunto de participantes.
Grupo	Conjunto lógico de participantes y equipos.
Intento	Marca que obtiene un participante o equipo en una prueba.
Marca	Valor que se obtiene como resultado de una prueba.
Participante	Persona individual que participa en una competición.

Abreviaturas

MMP	Mejor Marca por Participante

3.2 Actores y participantes

Se detallan a continuación los participantes (stakeholders) y los actores que interactuarán directamente con el sistema.

Stakeholders

- *Club Deportivo de la UMA*: aunque la aplicación está diseñada para un uso general, el proyecto está motivado por este club.
- *Ángel Carmona*: representante técnico del Club Deportivo de la UMA.
- *Agencia Española de Protección de Datos*: esta institución afecta de manera indirecta el proyecto a través de la LOPD (Ley Orgánica de Protección de Datos).

Actores

- *Invitado*: este tipo de actor interactúa de forma directa con el sistema y sus acciones están limitadas a leer/obtener información de las competiciones a las que tiene acceso.
- *Gestor*: este tipo de actor posee los permisos de un invitado y además es capaz de crear y borrar información de las competiciones a las que tiene acceso.
- *Administrador*: Los actores con este rol cuentan con todos los permisos de lectura y escritura de todas las competiciones, además de permisos extras como la administración de usuarios de la aplicación.

3.3 Requisitos funcionales

En esta sección se describen los requisitos del tipo funcionales. Se han dividido en grupos lógicos para agrupar funcionalidades relacionadas.

RF01 - CRUD (Create, Read, Update, Delete) de competiciones

- **RF01.1 - Crear competición**. Un usuario con rol de admin o gestor podrá crear una competición identificada por un nombre. También se podrá almacenar la

fecha de comienzo y fin, organizador, la ciudad donde se realiza y establecer un logo para la competición. La competición estará compuesta por un conjunto de pruebas, participantes, equipos, grupos y registros que se crearán dentro del contexto de la competición.

- **RF01.2 - Mostrar competición.** Un usuario con rol de admin, o gestor/invitado con permiso en dicha competición podrá ver los datos de la competición así como de todos los componentes que la forman (pruebas, participantes, grupos, equipos y registros).
- **RF01.3 - Modificar competición**. Un usuario con rol de admin, o gestor con permiso en dicha competición, podrá cambiar todos los datos referentes a la competición, así como añadir, modificar o eliminar pruebas, participantes, grupos, equipos o registros.
- **RF01.4 - Eliminar competición.** Un usuario con rol de admin, o gestor con permiso en dicha competición, podrá cambiar todos los datos referentes a la competición, así como añadir, modificar o eliminar pruebas, participantes, grupos, equipos o registros.

RF02 - CRUD (Create, Read, Update, Delete) de pruebas

- **RF02.1 - Crear prueba.** Un usuario con rol de admin o gestor podrá crear una nueva prueba que tendrá un nombre único dentro de la competición. Se indicará si es Individual o en Equipo y un tipo de resultado (Distancia, Tiempo o Numérica). También se guardará en la prueba si fue elegida por sorteo o no y cuál se considera que será la mejor marca (el valor más alto o el valor más bajo). La prueba siempre será creada dentro de una competición.
- **RF02.2 - Mostrar prueba.** Se podrá ver información detallada de una prueba como nombre, si es Individual o en Equipo y tipo de resultado.
- **RF02.3 - Modificar prueba.** Un usuario con rol de admin o gestor podrá cambiar el nombre, tipo y tipo de resultado, además de poder cambiar cuál se considerará el mejor valor.
- **RF02.4 - Eliminar prueba.** Un usuario con rol de admin o gestor podrá eliminar una prueba de una competición. Al eliminar una prueba se eliminarán todos los registros asociados a dicha prueba.

RF03 - CRUD (Create, Read, Update, Delete) de participantes

- **RF03.1 - Crear participante.** Un usuario con rol de admin o gestor podrá crear un participante, identificado por un dorsal único dentro de la competición en la que se inscribe. Además será necesario indicar el nombre, apellidos y el grupo al que pertenece. De forma opcional se podrá especificar el sexo, la edad, si pertenece a algún equipo y se le podrá asignar pruebas. Estas asignaciones se podrán hacer manualmente o por sorteo (elegidas por el software).

- **RF03.2 - Mostrar participante.** Un usuario con rol de admin o, gestor/invitado con permisos de acceso en la competición donde se encuentra el participante, podrá ver toda la información de un participante.
- **RF03.3 - Modificar participante.** Un usuario con rol de admin o, gestor con acceso a la competición donde se encuentre el participante, podrá modificar todos los atributos de una participante: nombre, apellidos, sexo, edad, grupo, equipo, dorsal y añadir/eliminar pruebas asignadas. Las pruebas que contengan registros del participante no se podrán eliminar de las asignaciones.
- **RF03.4 - Eliminar participante.** Un usuario con rol de admin o, gestor con acceso a la competición donde se encuentre el participante, podrá eliminar un participante, eliminando a su vez todos sus registros en las pruebas en las que ha participado.

RF04 - CRUD (Create, Read, Update, Delete) de equipos

- **RF04.1 - Crear equipo.** Un usuario con rol de admin o gestor con acceso a la competición correspondiente podrá crear un nuevo equipo que será identificado unívocamente dentro de la competición por el nombre, además será necesario indicar el grupo donde se encuentra. Opcionalmente se podrá asignar pruebas al equipo. Estas asignaciones se podrán hacer manualmente o por sorteo (elegidas por el software).
  Los miembros del equipo deberán pertenecer al mismo grupo que el establecido en el grupo.
- **RF04.2 - Mostrar equipo**. Un usuario con rol de admin o, gestor/invitado con permisos de acceso en la competición donde se encuentra el equipo, podrá ver los datos de un equipo.
- **RF04.3 - Modificar equipo.** Un usuario con rol de admin o, gestor con permisos de acceso en la competición donde se encuentre el equipo podrá modificar el nombre del equipo, el grupo en el que se encuentra y sus pruebas asignadas. Las pruebas que contengan registros del equipo no se podrán eliminar de las asignaciones.
- **RF04.4 - Eliminar equipo.** Un usuario con rol de admin o, gestor con permisos de acceso en la competición donde se encuentre el equipo podrá eliminar un equipo y a su vez los registros asociados a este.

RF05 - CRUD (Create, Read, Update, Delete) de usuarios

- **RF05.1 - Crear usuario.** Un usuario con rol de admin podrá dar de alta nuevos usuarios indicando su nick, contraseña y el rol que tienen (admin, gestor o invitado). Si el rol seleccionado es gestor o invitado, se podrá indicar las competiciones a las que tiene acceso para gestionarlas o verlas respectivamente.
- **RF05.2 - Mostrar usuario**. Un usuario con rol de admin, podrá ver los nicks de los usuarios y las competiciones a las que tiene acceso cada usuario.

- **RF05.3 - Modificar usuario.** Un usuario con rol de admin podrá cambiar el nick o contraseña de cualquier usuario, así como las competiciones a las que tiene acceso.
- **RF05.4 - Eliminar usuario**. Un usuario con rol de admin podrá borrar a un usuario de la aplicación. Siempre deberá existir al menos un usuario con rol de administrador en la aplicación.

RF06 - CRUD (Create, Read, Update, Delete) de grupos

- **RF06.1 - Crear grupo.** Un usuario con rol de admin o, gestor con acceso a la competición correspondiente, podrá crear un grupo en una competición. Este puede ser parte de otro grupo que haya sido creado anteriormente. El nombre de este debe de ser único dentro de la competición.
- **RF06.2 - Mostrar grupo.** Un usuario con rol de admin o,gestor/invitado con acceso a la competición donde se encuentre el grupo, podrá ver la información del grupo.
- **RF06.3 - Modificar grupo.** Un usuario con rol de admin o, gestor con acceso a la competición donde se encuentre el grupo, podrá modificar el nombre del grupo o añadir/eliminar miembros del grupo.
- **RF06.4 - Eliminar grupo.** Un usuario con rol de admin podrá eliminar un grupo de una competición, esto eliminará a todos los participantes y equipos que se encuentren en este grupo y sus registros. Los subgrupos pertenecientes a este grupo también serán eliminados.

RF07 - CRUD (Create, Read, Update, Delete) de registros.

- **RF07.1 - Crear registro.** Un usuario con rol de admin o gestor con permisos podrá crear registros (intentos) de participantes o equipos sobre pruebas que tengan asignadas en una competición determinada. El valor para este registro dependerá del tipo de resultado de la prueba.
- **RF07.2 - Mostrar registro**. Un usuario con rol de admin o, gestor/invitado con acceso a la competición donde se encuentre el registro, podrá ver los registros de los participantes/equipos.
- **RF07.3 - Modificar registro.** Un usuario con rol de admin o, gestor con acceso a la competición donde se encuentre el registro, podrá modificar el registro de un participante/equipo, es decir, la marca que ha realizado.
- **RF07.4 - Eliminar registro**. Un usuario con rol de admin o, gestor con acceso a la competición donde se encuentre el registro, podrá eliminar un registro o todos los registros de una prueba asociados a un participante/equipo.

RF08 - Mostrar participantes por orden alfabético. En aquellos casos donde se muestre una tabla con participantes, esta podrá ordenarse por apellidos ascendentemente y descendentemente.

RF09 - Mostrar los resultados por orden de clasificación. Cuando se muestre una tabla con resultados de una prueba, se podrá ordenar por la marca obtenida.

RF10 - Obtener el mejor registro de un participante en una prueba. El sistema podrá obtener el mejor registro (MMP) de un participante/equipo en una prueba de una competición.

RF11 - La aplicación permitirá exportar e importar los datos de la base de datos. El sistema podrá exportar la base de datos en un fichero y posteriormente podrá ser importada.

RF12 - Se podrá obtener un fichero pdf con los resultados de una competición. Un usuario con rol de admin, gestor o invitado podrá imprimir en un fichero pdf los resultados de una competición. Podrá seleccionar las pruebas y los grupos que desea imprimir para filtrar los resultados. Además, dicho pdf deberá incluir información sobre MMP, puesto y puntos del participante. También se añadirá en caso de existir información sobre la competición y el logo.

RF13 - Filtrar resultados. La aplicación podrá filtrar y mostrar los resultados para una prueba determinada y/o un grupo determinado.

RF14 - Login de usuarios. La aplicación será accedida con una previa validación del usuario indicando su nombre de usuario y contraseña.

RF15 - Seleccionar una prueba por sorteo. La aplicación permitirá, además de la forma manual, asignar una prueba aleatoria a un participante/equipo la cual seleccionará el propio software de entre las existentes en la competición.

3.4 Requisitos no funcionales

Existen ciertos requisitos para la aplicación que son de carácter no funcional. Cada uno de ellos se enumeran a continuación.

RNF1 - La aplicación estará disponible en Android y Windows Phone.

RNF2 - La aplicación será compatible con versiones de Android igual o superior a Android 2.3 (API 9) y versiones de Windows Phone igual o superior a WP7.

RNF3 - La aplicación deberá cumplir con la LOPD (Ley Orgánica de Protección de Datos). El acceso y control de los datos que contiene la aplicación cumplimentará lo establecido por esta ley.

RNF4 - La interfaz de usuario será accesible e intuitiva.

RNF5 - La aplicación deberá ser software libre.

3.1 Reuniones

Al estar los requisitos definidos al principio del proyecto, se llevó a cabo sólo una reunión con el cliente junto al tutor del proyecto al inicio del mismo, con el objetivo de analizar los requisitos existentes y verificarlos. También se estudió la forma de adaptar ciertos requisitos a las peculiaridades de un entorno móvil, y se añadió algunos requisitos no funcionales, como por ejemplo la versión mínima de los S.O. móviles y el requisito de usabilidad (RNF4).

3.2 Casos de uso

Los casos de uso son un recurso muy importante y valioso a la hora de tener un primer acercamiento sobre qué pasos se debe de seguir para llevar a cabo una tarea, todo ello de forma abstracta y sin especificar detalles de implementación.
Dada la extensión de los casos de uso generados en este proyecto, en este apartado se expone sólo algunos de los casos de uso que se han utilizado en la fase de análisis de los requisitos, añadiendo la totalidad de ellos en el apartado “anexos” de esta memoria.

Nombre	Crear una prueba
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario selecciona "crear prueba". 2- El sistema muestra una vista con los campos a rellenar. 3- El usuario introduce los datos de la prueba. 4- El usuario acepta la creación. 5- El sistema crea la nueva prueba.	

Post-condiciones	POSCOND1 (la prueba existe en la competición)
Escenarios alternativos	
4- El usuario cancela la creación. 5- El sistema no modifica nada y devuelve a usuario a una vista anterior.	
4- El sistema detecta que ya existe una prueba con el mismo nombre y avisa al usuario. 5- El usuario cambia el nombre por otro o cancela la creación.	
5- El sistema detecta que existen datos obligatorios sin rellenar. 6- El sistema informa al usuario de dicho error y cancela la operación.	

Nombre	Eliminar un grupo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe algún grupo en la competición)
Escenario principal	
1- El usuario selecciona el grupo a eliminar de la lista de grupos. 2- El usuario selecciona "eliminar". 3- El sistema pide confirmación sobre la eliminación. 4- El usuario acepta eliminar el grupo. 5- El sistema elimina el grupo junto con todos sus registros, participantes, equipos y subgrupos.	
Post-condiciones	POSCOND1 (el grupo seleccionado junto con sus registros, participantes, equipos y subgrupos no existe en la competición).
Escenarios alternativos	
4- El usuario cancela la petición. 5- El sistema no modifica nada.	

Nombre	Obtener pdf con resultados de una competición
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor/invitado con permisos).
Escenario principal	
1- El usuario selecciona la opción que permite exportar un fichero pdf. 2- El sistema muestra al usuario una vista con las pruebas y grupos a seleccionar que existen en la competición. 3- El usuario selecciona las pruebas y grupos que desea. 4- El usuario acepta generar el pdf. 5- El sistema genera el pdf.	
Post-condiciones	
Escenarios alternativos	
4- El usuario cancela la petición. 5- El sistema no genera nada.	

4. DISEÑO DE LA APLICACIÓN

Para poder llevar a cabo el desarrollo de la aplicación, se han llevado a cabo con anterioridad una serie de diseños sobre diferentes aspectos de la aplicación a desarrollar con el objetivo de modelar la aplicación y disponer de una serie de artefactos que guíen el proceso. Cada uno de ellos se detalla en los siguientes apartados, incluyendo además la metodología de desarrollo que se ha utilizado para llevar a cabo la implementación.

4.1 Metodología de desarrollo

Para el desarrollo de la aplicación se ha elegido una metodología **iterativa e incremental**. Este tipo de metodología es una de las bases de las metodologías ágiles y surgió como respuesta para solventar debilidades que poseía el modelo tradicional de cascada.

Esta metodología, tal como se indica en su nombre, está compuesta de dos partes: iterativo e incremental.
La parte **incremental** se basa en dividir el proyecto en etapas de corta duración (iteraciones). Cada iteración repite un conjunto de tareas, cada una de las cuales forman un modelo en cascada (análisis, diseño, implementación y pruebas).

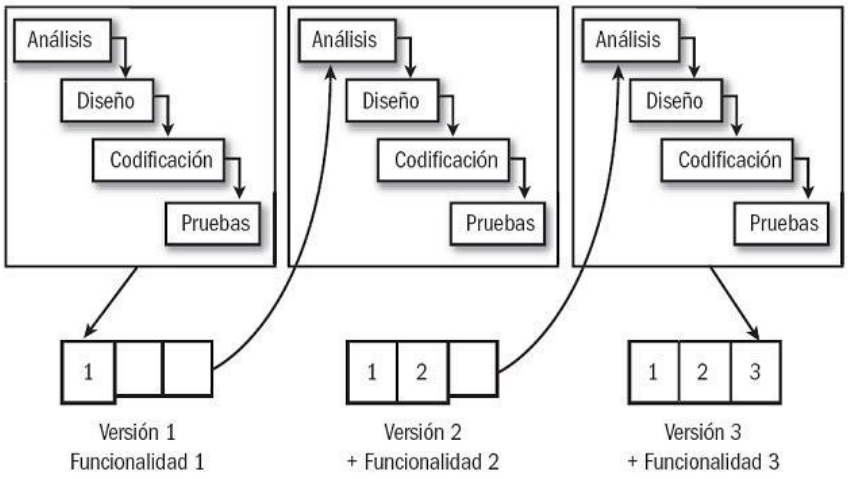


Figura 9. Iteraciones de la metodología iterativa e incremental junto a sus tareas

El objetivo de estas iteraciones es ir obteniendo cada vez una versión más completa del software, obteniendo resultados más inmediatos y minimizando el impacto de algunos de los riesgos que se corren en un proyecto, como por ejemplo cambios en los requisitos.

La parte **iterativa** (no confundir con iteración) trata de revisar y mejorar el producto sin que esto implique añadir funcionalidades al software. Por ejemplo, una práctica que se puede llevar a cabo en esta parte es la refactorización.

Concretamente, el proyecto se ha dividido en las siguientes iteraciones:

- 1ª iteración
  - Módulo de competiciones. Abarca el requisito RF.01.
- 2ª iteración
  - Módulo de pruebas. Abarca el requisito RF.02.
- 3ª iteración
  - Módulo de grupos. Abarca el requisito RF.06.
- 4ª iteración
  - Módulo de equipos. Abarca los requisitos RF.04, RF.15.
- 5ª iteración
  - Módulo de participantes. Abarca los requisitos RF.03, RF.08.
- 6ª iteración
  - Módulo de registros. Abarca los requisitos RF.07, RF.09, RF.10, RF.13.
- 7ª iteración
  - Módulo de usuarios. Abarca los requisitos RF.05, RF.14.
- 8ª iteración
  - Módulo exportar en PDF. Abarca el requisito RF.12.
- 9ª iteración
  - Módulo importar/exportar datos. Abarca el requisito RF.11.

La decisión para el orden de estas iteraciones se hizo en base a un análisis de dependencias entre los requisitos, ordenando de menor a mayor dependencia.

Aunque se ha intentado llevar la metodología mencionada, es cierto que gran parte de la tarea de las pruebas se ha hecho en las fases finales del proyecto.

Como herramienta de planificación para esta metodología, se ha utilizado Trello, el cual se ha definido en el apartado 2.2.3.

4.2 Modelo conceptual

En este apartado se muestra un diagrama de clases que representa a modo conceptual el dominio de la aplicación. Se trata de un modelo abstracto diseñado con el objetivo de ayudar a comprender el contexto del problema y no necesariamente un modelo de implementación. Se explicará con detalle también las partes que componen el diagrama.

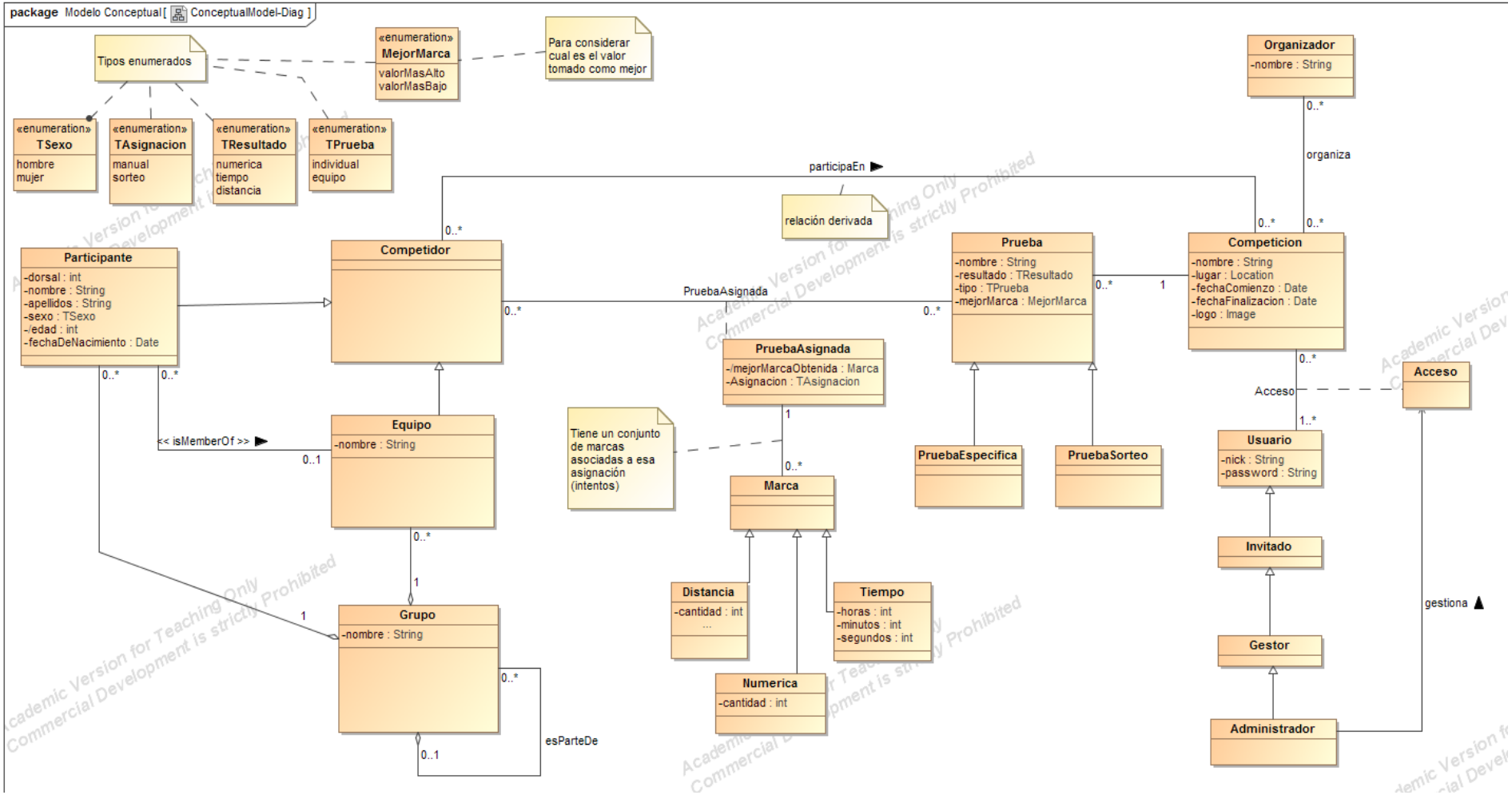


Figura 10. Modelo conceptual

Clase Competición

Esta clase representa a las competiciones que existen dentro de la aplicación. Es la clase principal del sistema, la cual constituye un entorno de grupos, pruebas, equipos, participantes y registros asociados a cada instancia de esta clase. Sus atributos son:

- *nombre*: campo obligatorio de tipo String que representa el nombre de una competición. Este nombre será único en el ámbito de la aplicación.
- *lugar*: campo opcional de tipo String donde se describe la localización de la competición.
- *fechaComienzo*: campo opcional de tipo Date que representa la fecha de comienzo del evento.
- *fechaFinalizacion*: campo opcional de tipo Date que representa la fecha de finalización del evento.
- *logo*: campo opcional de tipo Image la cual contiene una imagen que representa el logo de la competición.

Clase Organizador

Esta clase representa a los organizadores de las competiciones. Su único atributo es *nombre*, el cual es de tipo String y obligatorio.

Clase Prueba

La clase representa las pruebas que se llevan a cabo dentro de cada competición, a partir de las cuales se obtienen las marcas. Los atributos que la componen son:

- *nombre*: campo obligatorio de tipo String que representa el nombre de una competición. Este nombre será único en el ámbito de la competición.
- *resultado*: campo obligatorio del tipo enumerado TResultado, el cual tiene los valores “numérica”, “tiempo” y “distancia”. Representa que clase de valor guardará las marcas de esta prueba.
- *tipo*: campo obligatorio del tipo enumerado TPrueba, el cual puede ser “equipo” o “individual”.
- *mejorMarca*: campo obligatorio del tipo enumerado MejorMarca el cual tiene como objetivo indicar cual marca de un grupo de intentos se considerará la mejor marca, pues no en todos los tipos de pruebas tiene por qué ser el valor más alto o el más bajo el mejor.

Clase Grupo

Esta clase representa agrupaciones lógicas de participantes, equipos. Un grupo además también puede contener otros grupos, cuya función es representada

mediante la relación “esParteDe”. El único atributo que contiene es *nombre*, el cual es de tipo String y obligatorio. El valor de este atributo debe de ser único dentro de una competición.

Clases Usuario, Invitado, Gestor y Administrador

La clase Usuario es una clase abstracta que representa un usuario de la aplicación. Esta clase es implementada por la clase Invitado, la cual se especializa en Gestor y este a su vez se especializa en Administrador. Cada una de las especializaciones representa un nuevo nivel de permisos y control. El invitado sólo tendrá acceso a lectura de los datos pertenecientes a las competiciones que tenga permisos, el Gestor agrega al anterior permisos de creación, modificación y borrado de datos y el administrador tiene acceso total sobre todas las competiciones y además tiene el control de permisos de usuarios.
Estos permisos se representan mediante la clase Acceso, la cual es una clase de la relación Acceso. La clase usuario se compone de:

- *nick*: campo obligatorio de tipo String que representa el nombre de usuario. Este nombre será único en el ámbito de la aplicación.
- *password*: campo obligatorio de tipo String que guarda la contraseña del usuario para el acceso.

Clase Competidor

La clase Competidor es una clase abstracta que representa a un participante o un equipo. Esta clase es implementada por las clases Participante y Equipo. A un Competidor se le asignan pruebas mediante la clase de relación PruebaAsignada.
Contiene un atributo, *nombre*, el cual dependiendo del contexto representa el nombre del equipo o el nombre de un participante.

Clase Participante

Esta clase representa un participante individual en una competición y es una implementación de la clase Competidor. Un participante puede ser opcionalmente miembro de un equipo cuya relación está representada en el diagrama con el estereotipo <<isMemberOf>>.
La clase Participante contiene los siguientes atributos:

- *dorsal*: campo obligatorio de tipo Integer que representa el dorsal del participante. Este valor de este campo será único en el ámbito de una competición.
- *apellidos*: campo obligatorio de tipo String.
- *sexo*: campo opcional de tipo enumerado (TSexo).
- *edad*: atributo opcional de tipo Integer. Es un atributo de tipo derivado.
- *fechaDeNacimiento*: atributo opcional de tipo Date.

Clase Equipo

Esta clase representa un equipo perteneciente a una competición y es una implementación de la clase Competidor. Un equipo está compuesto por participantes que pertenecen al mismo grupo, cuya relación está representada en el diagrama con el estereotipo <<isMemberOf>>.
La clase equipo no incorpora ningún atributo extra respecto a su clase padre.

Clase PruebaAsignada

Esta clase representa las asignaciones de pruebas que se hacen sobre un participante o equipo. Se trata de una inscripción necesaria para posteriormente introducir marcas de esa prueba a ese participante/equipo. Este tipo de asignación se puede hacer de forma manual o automática a través del sistema.
La clase PruebaAsignada contiene los siguientes atributos:

- *mejorMarcaObtenida*: atributo opcional de tipo Integer. Es un atributo de tipo derivado y contiene la mejor marca de un competidor en una prueba.
- *asignacion*: atributo obligatorio de tipo TAsignacion. Representa cómo se ha asignado esa prueba (manual o por sorteo).

Clases Marca, Distancia, Tiempo, Numerica

Marca representa la puntuación obtenida en alguna prueba. Es una clase abstracta la cual está implementada por las clases Distancia, Tiempo y Numerica. Cada una de las implementaciones guarda el tipo especial de dato que está asociado con cada tipo de prueba.
La clase Distancia contiene el atributo obligatorio *cantidad*, el cual representa una distancia mediante un float.
La clase Numerica tiene como atributo obligatorio *cantidad*, que representa una cuantía de algo.
La clase Tiempo tiene como atributos obligatorios *horas*, *minutos* y *segundos*. Cada uno de ellos son de tipo Integer.

4.3 Modelo Relacional

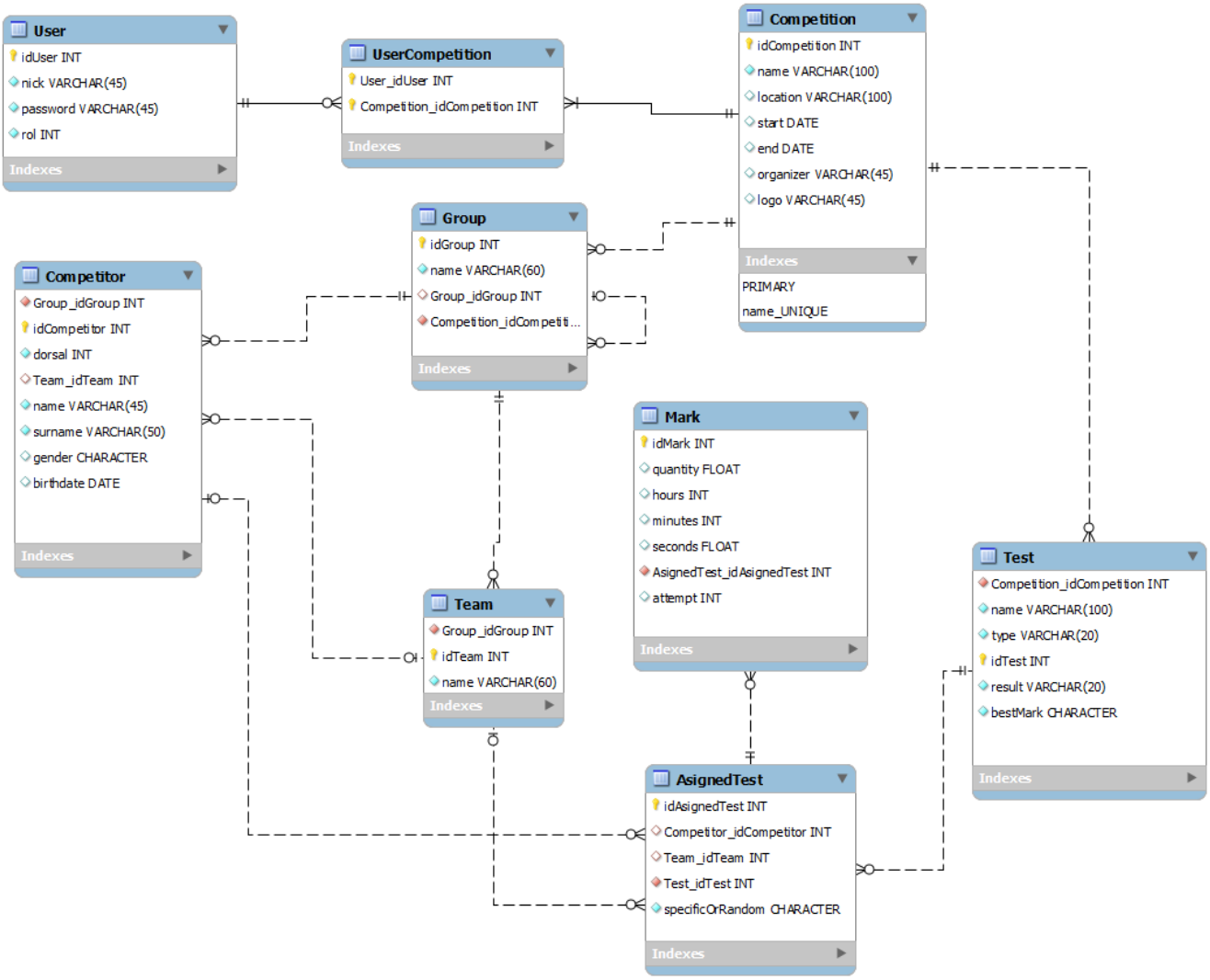


Figura 11. Modelo Relacional

Este modelo es el resultado de analizar el modelo conceptual y adaptarlo para una base de datos. Normalmente, en sistemas más complejos que el que atañe a este proyecto, es recomendable primero diseñar el modelo Entidad-Relación, y a partir de los refinamientos y análisis llegar al modelo Relacional, que es el que se representa en la figura anterior. Pero en el proceso de diseño se ha omitido este paso, pues a partir del diagrama conceptual se tuvo claro cuál sería el modelo final de la base de datos.

Existen algunas restricciones de multiplicidad y de atributos que se han "relajado", es decir, donde debería de ser una multiplicidad de 1-N se ha puesto una de 0-N; y donde un atributo debería de ser obligatorio no lo es. Concretamente son los siguientes elementos:

- campos *cantidad*, *minutos* y *segundos* en la tabla Mark.
- Relación entre Competitor y AsignedTest.
- Relación entre Team y AsignedTest.
- Relación entre AsignedTest y Mark.

Esta decisión se ha tomado por motivos de diseño de la propia base de datos, principalmente para evitar duplicidad de tablas muy parecidas. El control de esas relaciones y atributos “relajados” se ha designado a la capa inmediatamente superior a la base de datos, que es la que se encarga de gestionar la base de datos.

Cada tabla incorpora además un campo “id” auto-incremental (excepto UserCompetition) por motivos de eficiencia para el acceso a los registros; relevando la clave primaria a este atributo. Esto evita que si se cambia por algún motivo la lógica de negocio (por ejemplo, que en la tabla Competitor el campo surname deje de ser obligatorio) no se vea afectada la base de datos.

Por último cabe añadir que, por decisión de nombrar en inglés todas las funciones y variables del código y ya que a partir de este modelo de base de datos desarrollado con MySQL Workbench se obtiene las sentencias SQL, se decidió construir el esquema preparado ya en inglés.

4.4 Diagrama de navegación

UWE [14] es una propuesta de modelado UML que extiende este lenguaje para poder diseñar diagramas especializados en páginas y aplicaciones web. El resultado de este proyecto es una aplicación con versión para varios S.O. móviles hecha mediante Phonegap. Pero no hay que olvidar que el funcionamiento interno y lo que contiene es una aplicación basada en tecnología web. Es por eso que se ha hecho conveniente diseñar un diagrama que muestre el esquema de navegación.

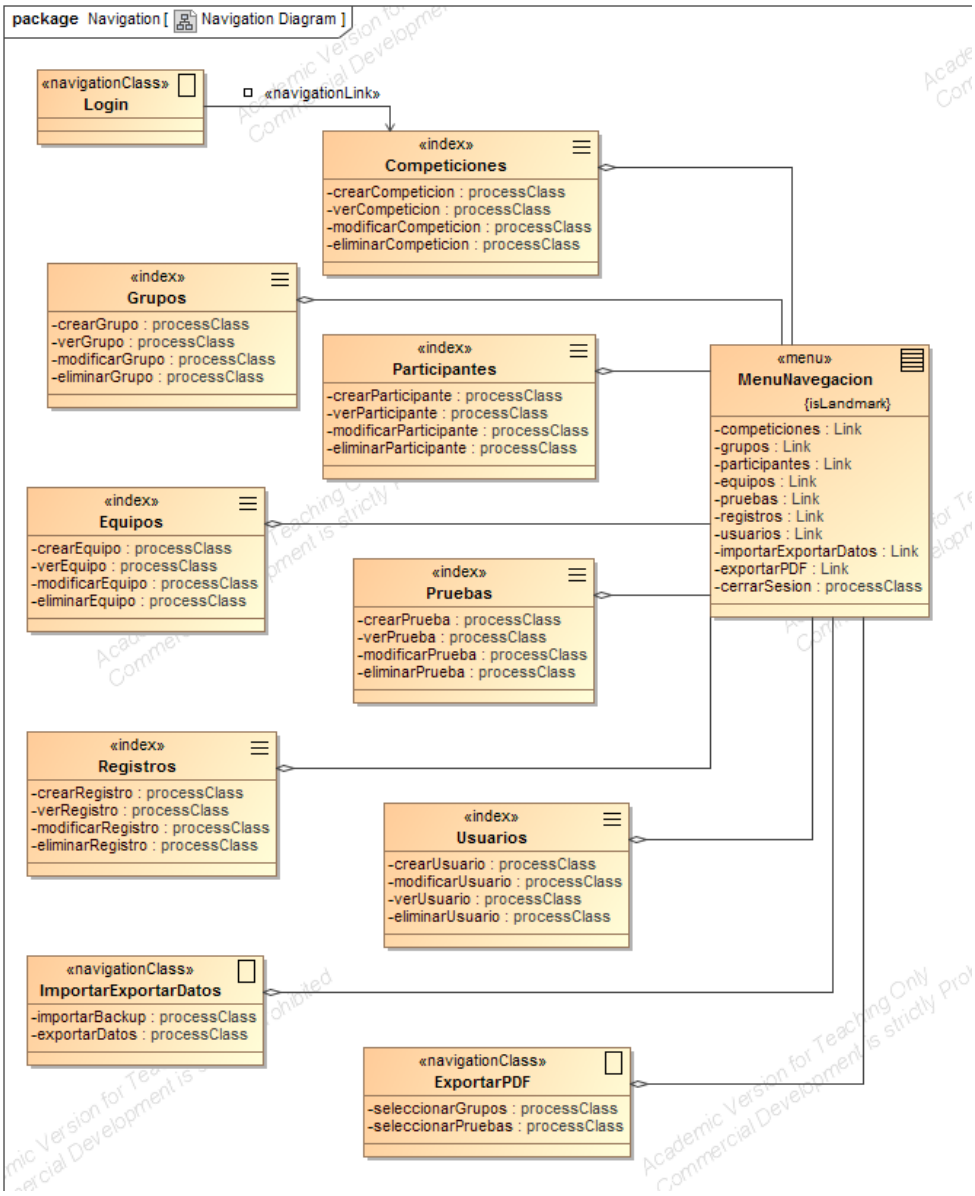


Figura 12. Diagrama de navegación

4.5 Distribución de la aplicación

Para el diseño de la aplicación se ha hecho uso de frameworks, librerías, archivos HTML, CSS, etc. Todos estos artefactos se encuentran distribuidos en directorios los cuales se explican a continuación.

Figura 13. Árbol de directorios

www : de aquí "cuelgan" todos los archivos y directorios de la aplicación.

- *Competitions, competitors, groups, iodata, pdf, records, teams, tests, users*: estos directorios contienen las páginas HTML de la aplicación.
- *css*: contiene los CSS que son propios de la aplicación.
- *img*: algunas imágenes e iconos usados en el programa.
- *js*: Contiene todos los archivos escritos en JavaScript.
- *classes*: clases de la aplicación.
- *controllers*: contiene los controladores de cada módulo.
- *DAO*: Lógica de la persistencia junto a sus dependencias. Se detallará su contenido más adelante.
- *testing*: carpeta que contiene todas las pruebas diseñadas que se han hecho a la aplicación.
- *third*-party-lib: contiene todos los archivos de frameworks y bibliotecas de terceros usadas en la aplicación.
- *app*.utils.js: es un JavaScript con diversas funciones utilizadas en distintos puntos de la aplicación.
- *config*-app.js: configuración de arranque de la aplicación
- *index.html*: Contiene la vista de login.
- *menuleft.html*: es el menú de navegación que se encuentra presente en toda la aplicación (menos en el login).
- *config.xml*: archivo XML con la configuración necesaria para el compilador de Phonegap.

Aunque no aparece en la imagen, uno de los archivos que están dentro del directorio www es *cordova.js*, el cual no aparece porque se añade automáticamente al generar la aplicación con el build de phonegap. Este JavaScript es muy importante

para la aplicación pues será el que sirva de intermediario entre la parte web y la parte nativa de la aplicación.

4.6 Mockups

Un mockup es una maqueta simple que se utiliza en el contexto de la ingeniería del software para realizar un diseño previo a la implementación de las vistas de la aplicación. Para realizar los mockups en este proyecto se ha hecho uso de la ya mencionada herramienta NinjaMock. Todos los mockups diseñados se adjuntan en el apartado “anexos” de la memoria, pero en esta sección se explicará la estructura principal diseñada a partir de los mockups.

Figura 14. Vistas Login y lista de competiciones

La figura 14 muestra a la izquierda la primera ventana que aparece al entrar en la aplicación. En dicha ventana el usuario ingresará su nombre de usuario y su contraseña para poder tener acceso a la aplicación. Una vez se ha conseguido acceder, la siguiente vista que se muestra es la lista de competiciones que es la que se ve en la imagen de la derecha. A través de esta vista se podrá seleccionar, ver, crear, editar y borrar competiciones.

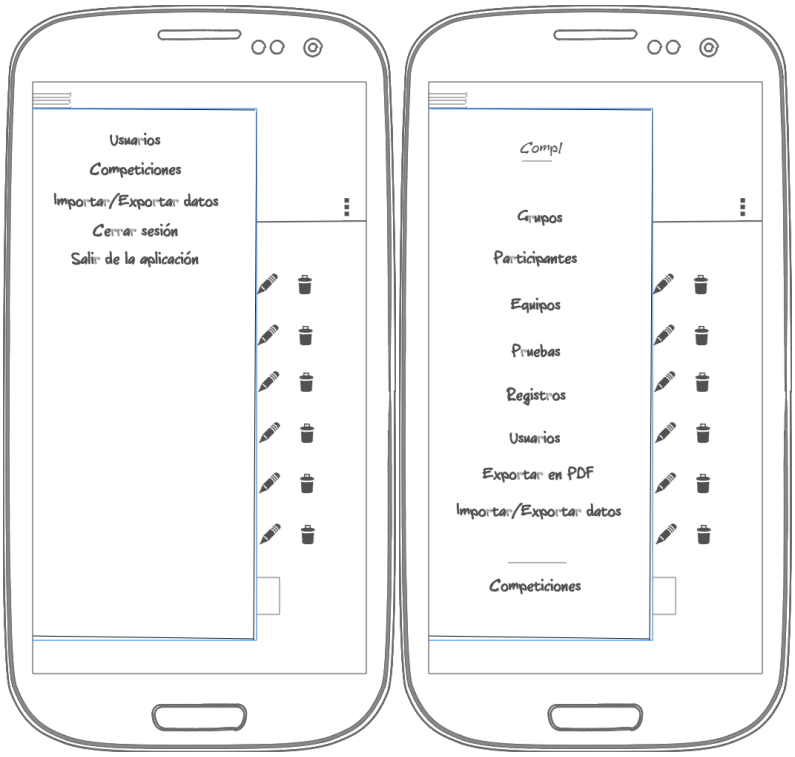


Figura 15. Menú sin competición seleccionada y con competición seleccionada

En la figura 15 se puede ver las opciones que ofrece el menú lateral en sus dos formas: sin haber accedido al ámbito de una competición (izquierda) y habiendo accedido a alguna competición (derecha). Este menú es el principal de la aplicación y permite navegar entre las secciones de la aplicación.

Figura 16. Sección de equipos y vista de datos de un equipo

En la figura anterior se muestra un ejemplo de una vista principal de sección. A través del menú se selecciona la sección “equipos” y la aplicación mostrará una ventana como la imagen de la izquierda, donde aparece la lista de equipos existentes junto a algunas opciones.
En la imagen de la derecha se puede ver el resultado de pulsar sobre el nombre de un elemento de la lista, que muestra un modal con la información del equipo.
Casi todas las secciones tienen un aspecto, botones e interacción parecidos al mostrado en la figura anterior, a excepción de algunas diferencias debido a propiedades específicas de secciones que añaden, cambian o quitan ciertos elementos de la vista.

5. IMPLEMENTACIÓN DE LA APLICACIÓN

Una vez que se han recogido los requisitos, se han analizado, se han obtenido sus casos de uso, y se han modelado distintos aspectos necesarios de la aplicación, se da paso a la fase de implementación. En dicha implementación, se ha dispuesto de una arquitectura y otros aspectos técnicos que se destacan a continuación.

5.1 Arquitectura en 3 capas

La arquitectura bajo la que se ha implementado la aplicación es una arquitectura basada en 3 capas y un nivel. Estas capas son separaciones lógicas cada una de la cual contiene un rol definido. Dichas capas son:

- **Capa de presentación.** Esta capa está compuesta básicamente por archivos HTML los cuales son simples vistas de presentación al usuario que muestran y recogen la información. El objetivo de esta capa es presentar una interfaz al usuario con la que pueda interactuar y comunicar a la capa inferior (capa de lógica del negocio) las acciones que el usuario solicita.
- **Capa de lógica del negocio** o **capa controladora.** Esta capa se encarga de gestionar todos los eventos que la capa de presentación le envía. Contiene la implementación de todas las reglas y pasos a seguir cuando una acción por parte del usuario es requerida. Además se encarga de la parte dinámica de la interfaz, añadiendo y eliminando elementos de las vistas y también controlando la navegación.
- **Capa de datos** o **capa de persistencia**. Esta capa se encarga de realizar la persistencia de los datos y ofrecer un acceso a la información almacenada. Está implementada con el patrón DAO (apartado 5.2.1) y el patrón Factory Method (apartado 5.2.3), con el objetivo de abstraer completamente el funcionamiento e implementación de esta capa de las capas superiores.

A través de esta división en capas conseguimos una separación entre las vistas, la parte lógica de la aplicación y la gestión de los datos; permitiendo así poder hacer cambios en una capa sin que ello afecte a las demás.

Se muestra a continuación un esquema conceptual de esta arquitectura:

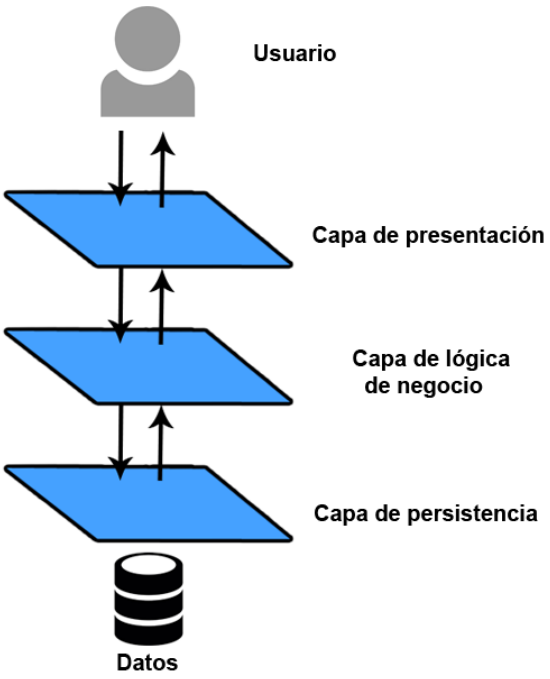


Figura 14. Arquitectura de la aplicación

5.2 Patrones de diseño

Los patrones de diseño son soluciones probadas a problemas comunes en el ámbito del desarrollo software. Un patrón está compuesto solamente por una serie de indicaciones abstractas (evitando nombrar lenguajes o tecnologías específicas todo lo posible) para implementar la solución.
Aunque su uso no es obligatorio ni trata de limitar las decisiones por parte del desarrollador, los patrones siempre son herramientas útiles cuyo uso es muy aconsejable.

A lo largo del desarrollo del software se han detectado situaciones donde se han podido usar algunos de los patrones de diseño, los cuales se detallarán en los siguientes apartados.

5.2.1 Patrón DAO

El patrón DAO [25] (Data Access Object) ofrece un mecanismo para abstraer el acceso y gestión de los datos a través de un objeto de persistencia (DAO) aislando con ello los detalles de implementación y la fuente de los datos usada. Normalmente existe un objeto DAO por cada entidad del dominio el cual suministra métodos para la manipulación de esa entidad (usualmente métodos CRUD). Estos objetos DAO

recibe y devuelve objetos que representan las entidades del dominio y que sirven como transporte y encapsulación de datos.

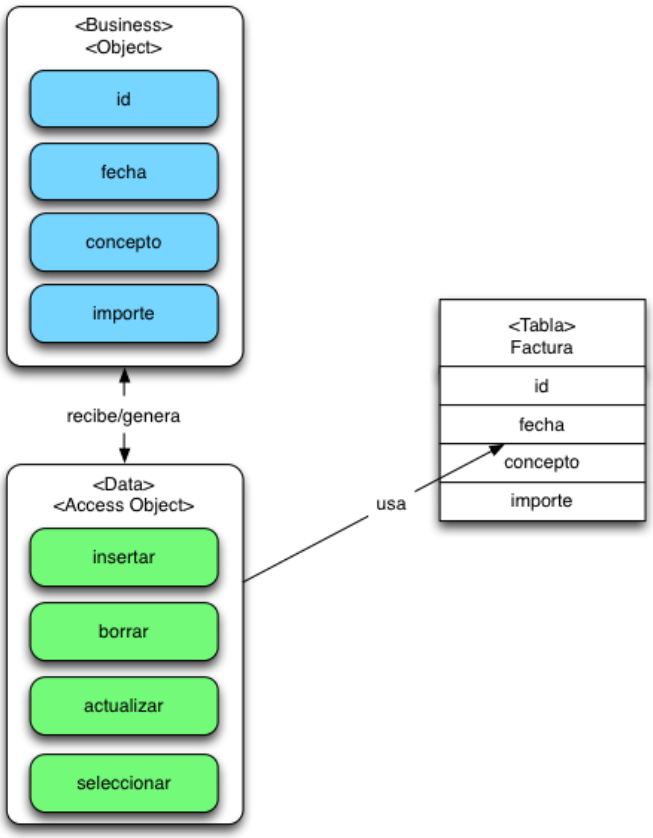


Figura 15. Esquema del patrón DAO

En esta implementación se ha definido una serie de clases con el objetivo de ser usadas como objeto de transporte y comunicación entre la capa de la lógica de negocio y la capa de datos.

A la hora de decidir cómo abstraer la persistencia, se barajó utilizar el patrón **Active Record** [39] el cual es otra posibilidad igual de válida, pero se decidió optar con el patrón DAO porque permite mayor flexibilidad.

La razón del uso de este patrón en la implementación es proporcionar una interfaz sencilla de manipulación de datos y poder aislar completamente la capa de persistencia, proporcionando la capacidad de poder cambiar el origen o implementación de los datos por cualquier otro manteniendo la misma interfaz de acceso.

5.2.2 Patrón Singleton

El patrón Singleton [26] (Singular) es un patrón cuyo objetivo es restringir la creación de objetos a una única instancia y proveer un mecanismo para el acceso a esta. Esto es muy útil cuando se quiere evitar construir nuevos objetos de una misma clase por diversos motivos, uno de ellos puede ser proporcionar un acceso global hacia un único objeto.

En este proyecto se ha utilizado como método para generar una única vez un objeto de conexión con la base de datos y restringiendo así el número de conexiones por cada objeto DAO a una; dando como resultado una mejora en la eficiencia de las conexiones.

5.2.3 Patrón Factory Method

El patrón Factory Method [27] (Método Fábrica) es una simplificación del patrón Abstract Factory. Este patrón consiste en delegar a otra entidad la responsabilidad de la creación concreta de un objeto que implementa una interfaz definida. Esto es muy útil cuando se quiere obtener un objeto pero no se desea saber qué clase de implementación guarda.

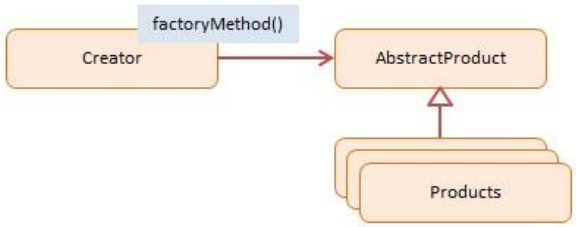


Figura 16. Esquema general de Factory Method

Por un lado está el objeto creador (Creator), que se encarga de elegir qué tipo concreto de producto se creará. Existe una interfaz (AbstractProduct) que define el producto abstracto y de esa interfaz se realiza las implementaciones concretas del producto (Products).

En JavaScript no existe en concepto de interfaz ni clases abstractas; por lo que en la implementación, la interfaz “AbstractProduct” es mantenida bajo el aseguramiento de que cada producto posee unos métodos y atributos públicos iguales entre sí y acordes a una interfaz “virtual” definida.

El uso que se le ha dado a este patrón ha sido entre la capa controladora y la capa de persistencia para ceder la elección de la implementación de la capa DAO en un único lugar. Con esta operación, la aplicación se dota de más flexibilidad para futuras ampliaciones.

5.3 Otros aspectos de la implementación

5.3.1 Implementación modular

Como ya se explicó en el apartado 2.3.3, JavaScript no posee un soporte robusto del uso de módulos, y sobre todo carece de soporte para dependencias de forma nativa.
En un principio se intentó solventar el problema mediante el uso del patrón Módulo, una solución dada por Douglas Crockford (muy conocido por su involucración y aportes al lenguaje JavaScript) la cual permite tener un concepto de encapsulación de funciones y variables privadas, permitiendo exportarlas como API. Pero esta opción seguía sin dar solución al problema de las dependencias. Es por eso que tras un análisis se optó finalmente por el framework RequireJS.

En este proyecto casi todo el JavaScript generado está dividido en módulos con RequireJS. Tanto las clases (prototipos), los controladores, la capa de persistencia y las pruebas en QUnit están implementadas con RequireJS.
Cada módulo define en su cabecera las dependencias (sería un símil a un “include”) que deben de ser cargadas y al final de cada módulo se exportan las funciones y/o variables que se desean tener como API.

5.3.2 Implementación SPA

SPA [28] (Single-Page Application) es un tipo de implementación de aplicaciones webs en la que todo el contenido y navegación se realiza a través de una única página web. Esto aporta a usuario una mejor experiencia al asemejarse más a una aplicación convencional de dispositivos móviles o escritorio.

En las aplicaciones de tipo SPA se cargan de una vez o se van añadiendo de forma dinámica según se vaya requiriendo. La API de Lungo está preparada para este tipo de implementación, permitiendo separar el contenido en distintas páginas webs y cargándolo dinámicamente a lo largo de la navegación en la misma página inicial.

Cada sección lógica de esta aplicación está separada en distintos archivos HTML compuestos por un <section> y varios <article> que se cargan de forma asíncrona en la página de inicio, siendo el controlador el que gestiona dichas cargas.

5.3.3 Objeto Promise

Web SQL Database es una base de datos asíncrona, esto significa que realiza transacciones de forma asíncrona y los resultados deben de ser manejados por “callbacks”, que son funciones que se le pasan como variables a la transacción y reciben el resultado una vez se ha procesado de forma asíncrona.
Por cada transacción se debe de añadir una función callback que puede ser de definida de forma anónima en el momento de la llamada o guardada anteriormente en una variable y pasarla como parámetro. Esto, en operaciones asíncronas con un mínimo nivel de complejidad, da como resultado código poco legible y complica mucho la tarea del tratamiento de errores. En funciones con mayor complejidad, este tipo de operaciones se hacen casi inmanejables, dando como resultado lo que se conoce como callback hell [29] (el infierno de los callbacks).

```
a(function (resultsFromA) {
  b(resultsFromA, function (resultsFromB) {
    c(resultsFromB, function (resultsFromC) {
      d(resultsFromC, function (resultsFromD) {
        e(resultsFromD, function (resultsFromE) {
          f(resultsFromE, function (resultsFromF) {
            console.log(resultsFromF);
          })
        })
      })
    })
  })
});
```

Figura 17. Ejemplo de funciones callbacks anidados

Por este motivo, se decidió buscar en este proyecto una alternativa o forma de hacer operaciones asíncronas que fueran más legibles y fáciles.

Las promises [30] (promesas) son objetos devueltos inmediatamente tras una función asíncrona, que pueden poseer los siguientes estados:

- No cumplida: Cuando aún no se ha llegado a ningún resultado.
- Cumplida: Cuando el código asíncrono ha terminado su función correctamente y ya está disponible el resultado.
- Rechazado: Cuando surge algún tipo de error.

Este objeto, que representa los estados de la transacción, se le pasa una función por cada uno de los resultados posibles (cumplida o rechazada).
A través de esta práctica conseguimos una forma más natural de tratar la transacción, haciendo el código mucho más legible y entendible.

La implementación que se ha usado para llevar a cabo esta práctica (pues JavaScript no la soporta de forma nativa) es la que incorpora la biblioteca Zepto (JQuery) con sus objetos “Deferred”.

5.3.4 Uso de librerías y dependencias

Dado que la aplicación hace un uso importante de módulos y de librerías, se ve conveniente crear un esquema que muestre las dependencias de los distintos componentes del proyecto. A continuación se muestra dicho esquema:

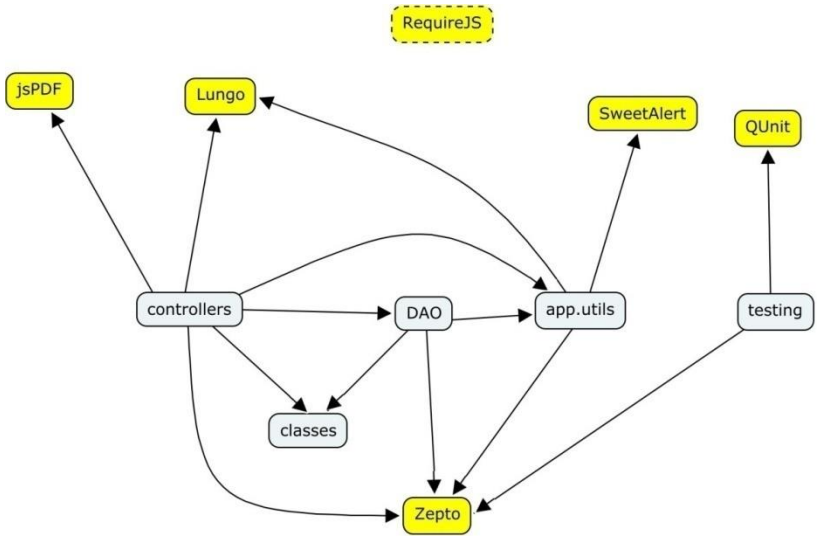


Figura 18. Dependencias de librerías

5.3.5 Convenciones de codificación

En la codificación se ha utilizado la filosofía de self-documenting code, es decir, desarrollar dando a las funciones y variables nombres lo más descriptivos posibles, para así también tener que evitar en la medida de lo posible escribir comentarios sobre qué es o qué hace cada variable o función.
También se ha utilizado JSDoc, que es una herramienta muy similar al Javadoc, para especificar la cabecera y lo que hacen las funciones mediante una estructura que define el propio JSDoc inspirada en Javadoc.

Como guía de codificación general se ha utilizado JSHint, una herramienta que se incorpora como plugin a Sublime Text y que examina el código en busca de codificaciones mal efectuadas o no aconsejables. Esta herramienta está basada en la especificación de escritura de código de Douglas Crockford [31], aunque elimina algunas normas por considerarlas demasiado restrictivas.

6. PRUEBAS Y VALIDACIÓN

En este capítulo se hablará de las pruebas que se han realizado para detectar errores de programación y comprobar el correcto funcionamiento de la aplicación. También se explica con detalle un desarrollo especial que se ha hecho para las pruebas de interfaz de usuario.

6.1 Pruebas realizadas

El proceso de pruebas y validación forma una parte fundamental en los procesos de desarrollo software. El diseño y ejecución de pruebas permite detectar y así corregir errores de programación que pueden llevar al sistema a estados incorrectos.
Se recomienda hacer pruebas sobre un producto software desde sus inicios hasta fase final del mismo; consiguiendo así detectar de manera temprana errores que de ser arrastrados hasta la parte final del producto podrían suponer unos costes de esfuerzo y tiempo importantes.
El objetivo de las pruebas es detectar la presencia de errores, no demostrar la ausencia de los mismos. Un programador que pruebe su propio código es muy desaconsejable pues lo evaluará de acorde a su total conocimiento sobre el mismo, ignorando entradas o situaciones que se podrían dar. Es por eso que lo más recomendable es que las pruebas del software las diseñe y lleve a cabo personas o departamentos ajenos al desarrollo, pues no tienen conocimientos de cómo está implementado. Desafortunadamente este proyecto es de tipo individual y no ha quedado otro remedio que diseñar las pruebas y probarlo por mí mismo, aunque por otra parte se ha tenido en la fase final un poco de ayuda por parte de una persona ajena al ámbito de la aplicación para probar el sistema completo en un entorno de ejecución real, lo cual ha permitido acercarse un poco al contexto explicado anteriormente.

Se describe a continuación las clases de pruebas que se ha llevado a cabo para detectar y mejorar la calidad de la aplicación:

- Pruebas unitarias: son pruebas que abarcan módulos o unidades pequeñas de código. Dentro de la creación de cada módulo del sistema perteneciente a las iteraciones del plan de desarrollo, se ha ido diseñando y desarrollando pruebas unitarias de las distintas clases que representan las entidades de la aplicación y que se usan en la capa de datos y la capa de lógica del negocio.
  Se ha enfocado este tipo de pruebas desde la perspectiva de caja negra (pruebas funcionales) y caja blanca (pruebas estructurales); utilizando para este último enfoque el plugin Blanket.js que permite examinar la cobertura de las pruebas.

- Pruebas de sistema: el objetivo de este tipo de pruebas es probar al sistema como un todo. En este tipo de tests se pone a prueba el sistema completo para comprobar la corrección y completitud del mismo, teniendo como guía los requisitos y los casos de uso especificados. Para la realización de estas pruebas se ha desarrollado un **sistema de automatización de pruebas de interfaz de usuario** usando la biblioteca Zepto y el framework QUnit. En el siguiente apartado se explica con detalle cómo se ha llevado a cabo esta implementación.
  Estas pruebas normalmente son de caja negra, aunque gracias al tipo de tecnología usado se ha podido analizar desde las pruebas automáticas de interfaz aspectos internos de la aplicación.

- Pruebas de regresión: en estas pruebas el objetivo es detectar errores que se hayan podido causar por la modificación de algún componente. Cada vez que se modifica un componente que ha sido validado por las pruebas (ya sea por añadir nuevas funcionalidades o modificar las existentes), debe de hacérsele las pruebas de regresión para asegurarse que su modificación no ha llevado a situaciones de error las funciones existentes.

- Pruebas de validación: el objetivo de estas pruebas es comprobar que el software cumple con las especificaciones, es decir, que el producto desarrollado es lo que el usuario quería. Se puede utilizar este tipo de pruebas al final del proyecto. Pero es recomendable, si se tiene una división funcional del producto, el validar cada una de esas partes a la finalización de cada una. Para llevar a cabo esta validación se ha tenido como ayuda a un usuario ajeno al desarrollo que, conociendo las características de la aplicación, ha realizado pruebas supervisadas en un entorno controlado por la herramienta GapDebug.

6.2 Sistema de automatización de pruebas de UI

Inicialmente, para obtener un sistema que permitiera automatizar pruebas a través de la interfaz de usuario de forma automática, se realizó una investigación de las herramientas actuales disponibles para ello. Además, siendo esta aplicación híbrida, se buscó en los dos posibles entornos: pruebas de interfaz para páginas web y pruebas de interfaz para los entornos móviles.
Herramientas para pruebas de páginas webs gratuitas prácticamente no se encontraron. Selenium e iMacros fueron las herramientas factibles que se encontraron. La primera se descartó por su poca flexibilidad para este tipo de aplicación y la segunda por no funcionar correctamente ante ciertas situaciones.
Por otro lado, las herramientas gratuitas de automatización para aplicaciones

móviles que se encontraron fueron Selendroid y Appium; pero resultaban ser incompletas o directamente inestables.

Tras ese estudio de herramientas disponibles, se decidió implementar un sistema propio para automatizar pruebas de interfaz de usuario, ayudado por QUnit y Zepto. Para ello, se dispuso en una misma ventana de navegador las pruebas en QUnit y un marco (iframe) con la aplicación en ejecución. El funcionamiento básico de esta estructura de pruebas es que a través de QUnit se examina y manda eventos de interacción a la aplicación situada en el marco mediante la librería Zepto.

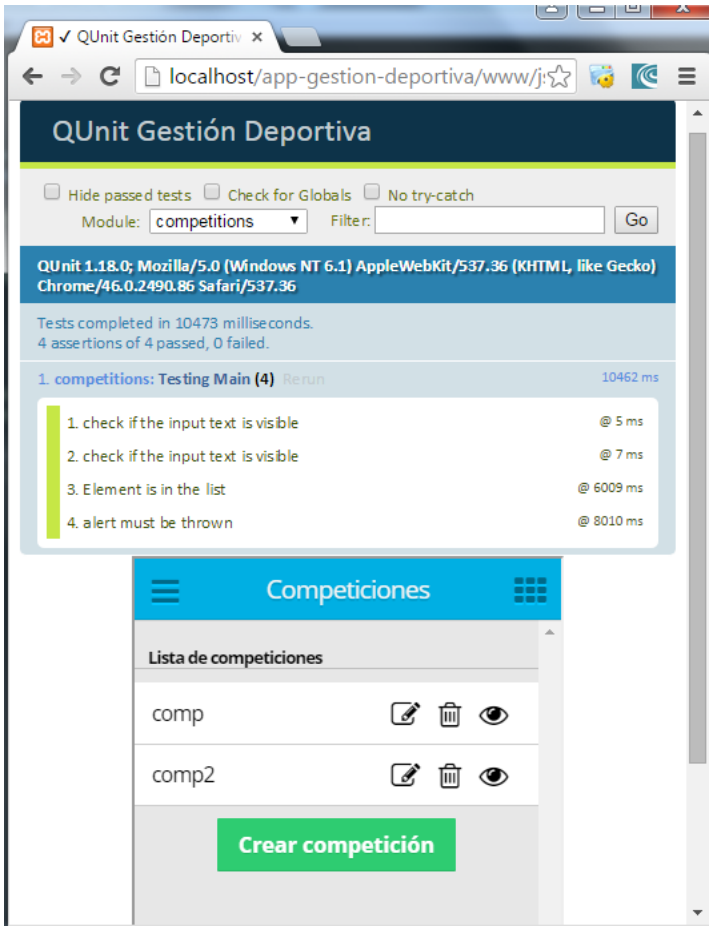


Figura 19. Pruebas de interfaz de usuario

Gracias a la librería Zepto, se tiene un gran control de lo que ocurre en la aplicación, permitiendo no sólo comprobar elementos y estados de la vista, sino además tener acceso a propiedades internas de la aplicación y a la propia base de datos, lo que ha permitido hacer un análisis más profundo de la aplicación.
Hay que señalar que, aunque gracias a esta solución he podido hacer pruebas de casi toda la funcionalidad de la aplicación desde la perspectiva de usuario, no he podido realizar pruebas de las funcionalidades que dependen del dispositivo móvil, las cuales son el acceso a imágenes para insertar como logo de la competición y la funcionalidad de importar/exportar ficheros desde un directorio. Este tipo de pruebas se han tenido que realizar a mano desde un dispositivo real.

7. CONCLUSIONES Y LÍNEAS FUTURAS

Como punto final de esta memoria, se exponen las conclusiones obtenidas del trabajo realizado y se ofrece una serie de líneas futuras de ampliación de la aplicación que podrían ser interesantes como parte de otro proyecto.

7.1 Conclusiones

La realización de este proyecto ha supuesto una experiencia y aprendizaje general de un importante valor para la carrera formativa y la preparación del futuro profesional del autor. El punto más importante y a la vez más difícil que ha supuesto para el desarrollo del proyecto ha sido enfrentarse a una gran cantidad de tecnologías desconocidas (por el estudiante) y novedosas.

Inicialmente el proyecto iba a ser escrito en los diferentes lenguajes nativos para cada sistema operativo móvil (C# y Java) y así fue como se comenzó, pero tras un estudio se dio con la solución que propone Phonegap. Esta decisión eliminó tareas paralelas e iguales que hubieran supuesto mucho más esfuerzo; como por ejemplo tener que hacer el mismo programa en los lenguajes nativos de cada sistema o implementar las mismas pruebas para cada uno.

Pero la elección de la tecnología que ofrece Phonegap no ha supuesto reducir a la mitad el esfuerzo y tiempo, pues esta decisión ha acarreado tratar con más tecnologías y lenguajes desconocidos que de la otra manera; aunque, como ya se ha explicado antes, el haber hecho el mismo proyecto en dos vertientes paralelas de forma nativa (Windows Phone y Android) seguramente hubiera supuesto mayor esfuerzo.

Como bien he podido leer a lo largo del proyecto en foros y sitios de Internet especializados, JavaScript es un lenguaje que “o lo odias, o lo amas”. Esto es así porque permite hacer todo con una flexibilidad enorme en su estructura y elementos; lo cual puede ser algo bueno para unos o algo malo para los que prefieren estructuras con más control y bien definidas.

Desde sus inicios, JavaScript ha sido un lenguaje llamado “caótico” por muchos, debido a que había pocos estudios y estándares para definir una estructura y uso bien definidos; lo que llevaba a muchos programadores a escribir código en este lenguaje sin normas ni estructuras. No ha sido hasta los últimos años que JavaScript ha llamado la atención de la comunidad de programadores y se está trabajando y mejorando muchos de estos aspectos.

El aprendizaje de JavaScript, como de otras tecnologías novedosas para mí (Zepto, Lungo, RequireJS,…), se ha llevado a cabo principalmente durante el desarrollo del proyecto y debido a la falta de conocimiento en los momentos iniciales, se ha tenido que refactorizar mucho código, pues a medida que más se aprendía sobre este lenguaje, más se determinaban buenas o malas decisiones anteriores. Esto junto a todos los demás aspectos del proyecto, ha hecho que haya un consumo considerable de horas respecto a las estimadas.

Se consideran alcanzados los objetivos iniciales del proyecto, pues se ha obtenido una aplicación que satisface los requisitos que se pedían y que está lista para usarse en Windows Phone y Android. Pero además, se han alcanzado otras características del proyecto, pues con la elección de Phonegap se ha obtenido la posibilidad de tener además la aplicación en iOS, BlackBerry OS y fireOS simplemente con realizar una compilación para cada uno de esos sistemas operativos móviles.

En general, ha sido una experiencia muy fructífera en la que el autor ha aprendido sobre muchos tipos de herramientas y tecnologías que desconocía, enfrentándome además a un caso real de desarrollo de software. También, además de contar con el apoyo de los directores, se ha conseguido dar un paso más en el proceso de autoaprendizaje, siendo esto una parte muy importante a considerar en la realización de un proyecto como el presente.

7.2 Líneas futuras

Hay una serie de elementos que se habrían podido añadir o ampliar en la aplicación y que no han tenido cabida en esta versión debido a la envergadura del mismo. Se explican a continuación algunas líneas futuras que se podrían realizar para ampliar la funcionalidad del producto:

- **Centralización de datos**. La aplicación actual almacena y recoge los datos de una base de datos local. Dadas las posibilidades que trae consigo el uso de dispositivo móvil, resulta altamente conveniente, útil y natural que al tratarse de una aplicación con acceso multiusuario los datos sean accedidos desde fuera del dispositivo a través de algún servicio externo. Además, al haberse separado totalmente la capa de persistencia de las demás capas, esta implementación se llevaría a cabo sin tener que alterar ningún otro componente. Se enumeran y explican brevemente a continuación un par de ideas para esta implantación:
    - Utilizar en conjunto JQuery + AJAX + PHP + Servidor. El servidor tendría que tener soporte para el código PHP y daría acceso a la base de datos global.
    - Utilizar en conjunto JQuery + Web Service (SOAP o REST).

- **Aplicación de escritorio**. Al tratarse de una aplicación hecha con tecnología web, posee mucha versatilidad para ser usada en distintos sistemas operativos y navegadores. La aplicación, tal como ya está implementada, funciona casi en su totalidad desde un navegador web, exceptuando las funcionalidades propias de un Smartphone, que son el acceso a imágenes de archivo para el logo de la competición y la exportación e importación de la base de datos. Lo que se podría hacer para obtener una aplicación de escritorio es adaptar esas partes al entorno de escritorio y, utilizando como motor interno de ejecución a Google Chrome, encapsular la aplicación para su instalación y correcto funcionamiento en entornos de escritorio.

- **Adición de nuevas funcionalidades**. Dado el contexto de la aplicación, han surgido una serie de ideas sobre nuevas funcionalidades que pueden resultar útiles de implementar en un futuro. Una es añadir la opción de adjuntar el PDF generado a un correo con la aplicación por defecto de correos que tenga el móvil. Otra funcionalidad que se puede añadir es una sección de incidencias, donde se pueda realizar fotos con el móvil sobre material o recursos de los pabellones que estén rotos o en malas condiciones y ser recogidas por la aplicación para realizar informes.

- **Mayor compatibilidad con otro tipo de disciplinas deportivas**. El diseño de la aplicación ha sido de carácter abierto para distintos tipos de competiciones que se realizan en el ámbito del atletismo. Pero podría añadirse además soporte para deportes de otra naturaleza, tales como tenis, boxeo, baloncesto, fútbol, ciclismo, etc.

REFERENCIAS BIBLIOGRÁFICAS

[1] JavaScript, https://es.wikipedia.org/wiki/JavaScript

[2] HTML, https://es.wikipedia.org/wiki/HTML

[3] CSS, https://en.wikipedia.org/wiki/Cascading_Style_Sheets

[4] SQL, https://es.wikipedia.org/wiki/SQL

[5] Using SQLite, autor Jay A. Kreibich, publicado por O'Reilly Media, Inc. 2010

[6] Definición UML, https://en.wikipedia.org/wiki/Unified_Modeling_Language

[7] Características UML, http://www.omg.org/gettingstarted/what_is_uml.htm

[8] Sublime Text, https://en.wikipedia.org/wiki/Sublime_Text

[9] Introducción a REM, Amador Durán Toro, 2006

[10] Trello, https://en.wikipedia.org/wiki/Trello

[11] Página oficial de NinjaMock, https://ninjamock.com

[12] MagicDraw, https://en.wikipedia.org/wiki/MagicDraw

[13] Características de MagicDraw, http://www.nomagic.com/products/magicdraw.html

[14] UWE, https://es.wikipedia.org/wiki/UWE_UML

[15] Página oficial de Phonegap Build, https://build.phonegap.com

[16] Información sobre MySQL Workbench, https://es.wikipedia.org/wiki/MySQL_Workbench

[17] Página oficial de XAMPP, https://www.apachefriends.org

[18] Phonegap, https://en.wikipedia.org/wiki/PhoneGap

[19] Enlace a repositorio de Lungo, https://github.com/tapquo/Lungo.js/

[20] RequireJS, http://requirejs.org

[21] Patrón AMD, https://en.wikipedia.org/wiki/Asynchronous_module_definition

[22] QUnit, https://en.wikipedia.org/wiki/QUnit

[23] Repositorio Git de jsPDF, https://github.com/MrRio/jsPDF

[24] Enlace a la página oficial en GitHub de SweetAlert, http://t4t5.github.io/sweetalert

[25] Definición de DAO, https://es.wikipedia.org/wiki/Data_Access_Object

[26] Definición de Patrón Singleton, https://es.wikipedia.org/wiki/Singleton

[27] Patrón Factory Method, http://www.dofactory.com/javascript/factory-method-design-pattern

[28] Single-Page Application, https://es.wikipedia.org/wiki/Single-page_application

[29] Información sobre callback hell, http://callbackhell.com/

[30] Objetos promises, https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Promesa

[31] Convención de codificación de Crockford, http://javascript.crockford.com/code.html

[32] TFG Aplicación Para la Gestión de Actividades Deportivas. Juan María Frías Hidalgo. Año 2014

[33] Aplicación Mis torneos, https://play.google.com/store/apps/details?id=com.poquesoft.mistorneos&hl=es

[34] Aplicación Gestor Deportivo, https://play.google.com/store/apps/details?id=com.poquesoft.mistorneos&hl=es

[35] Aplicación Youth Sports Team Management, http://www.teamsnap.com

[36] Aplicación Korrio, http://www.korrio.com

[37] GapDebug, https://www.genuitec.com/products/gapdebug

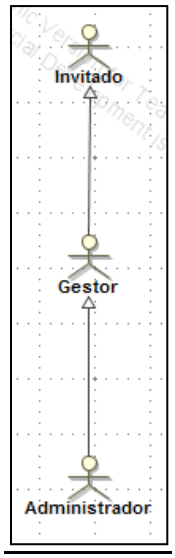
[38] Información sobre características de Chrome, https://es.wikipedia.org/wiki/Google_Chrome

[39] Active Record, https://es.wikipedia.org/wiki/Active_record

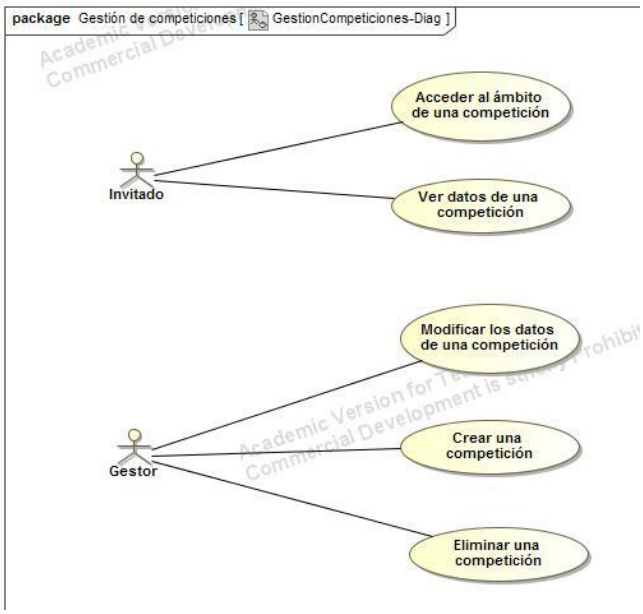
ANEXOS TÉCNICOS

En este capítulo se muestra, divididos en grupos lógicos, todos los casos de uso, diagramas de secuencia y mockups usados para el desarrollo de la aplicación. Por cada caso de uso se muestra una tabla con la especificación del mismo y sus diagramas y mockups relacionados.

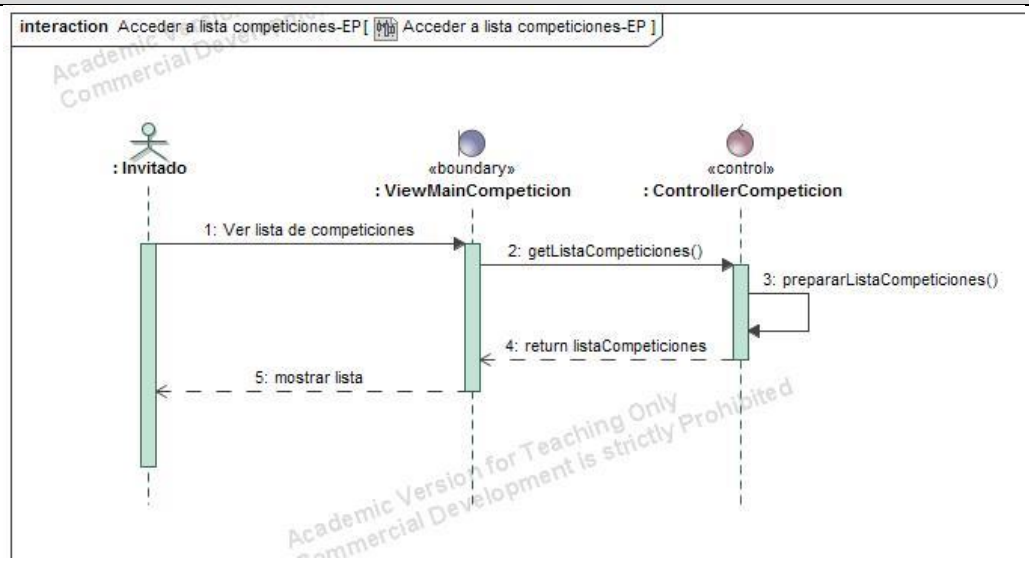
Relación entre los actores de los casos de uso:

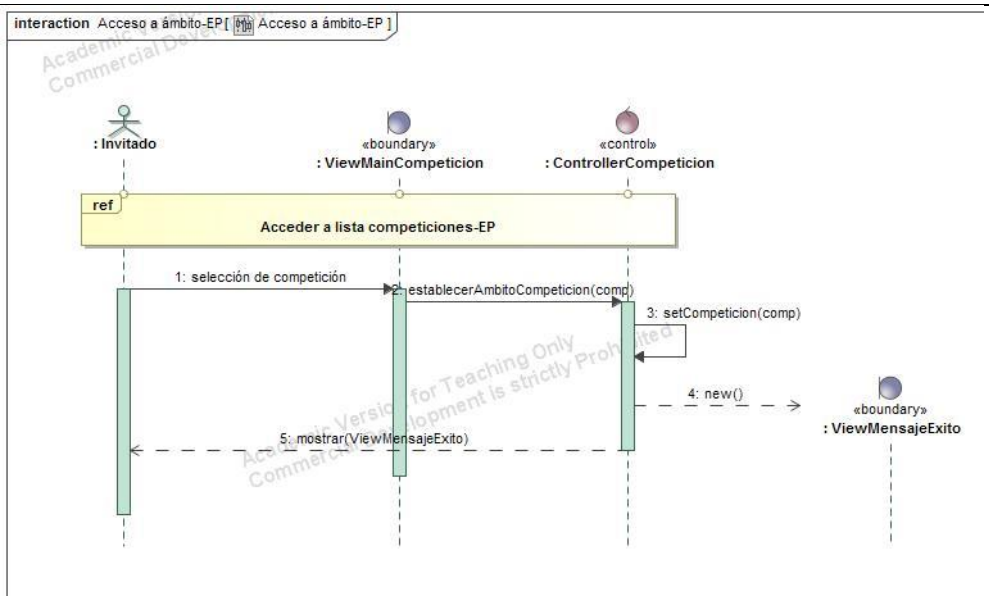


1. Gestión de competiciones

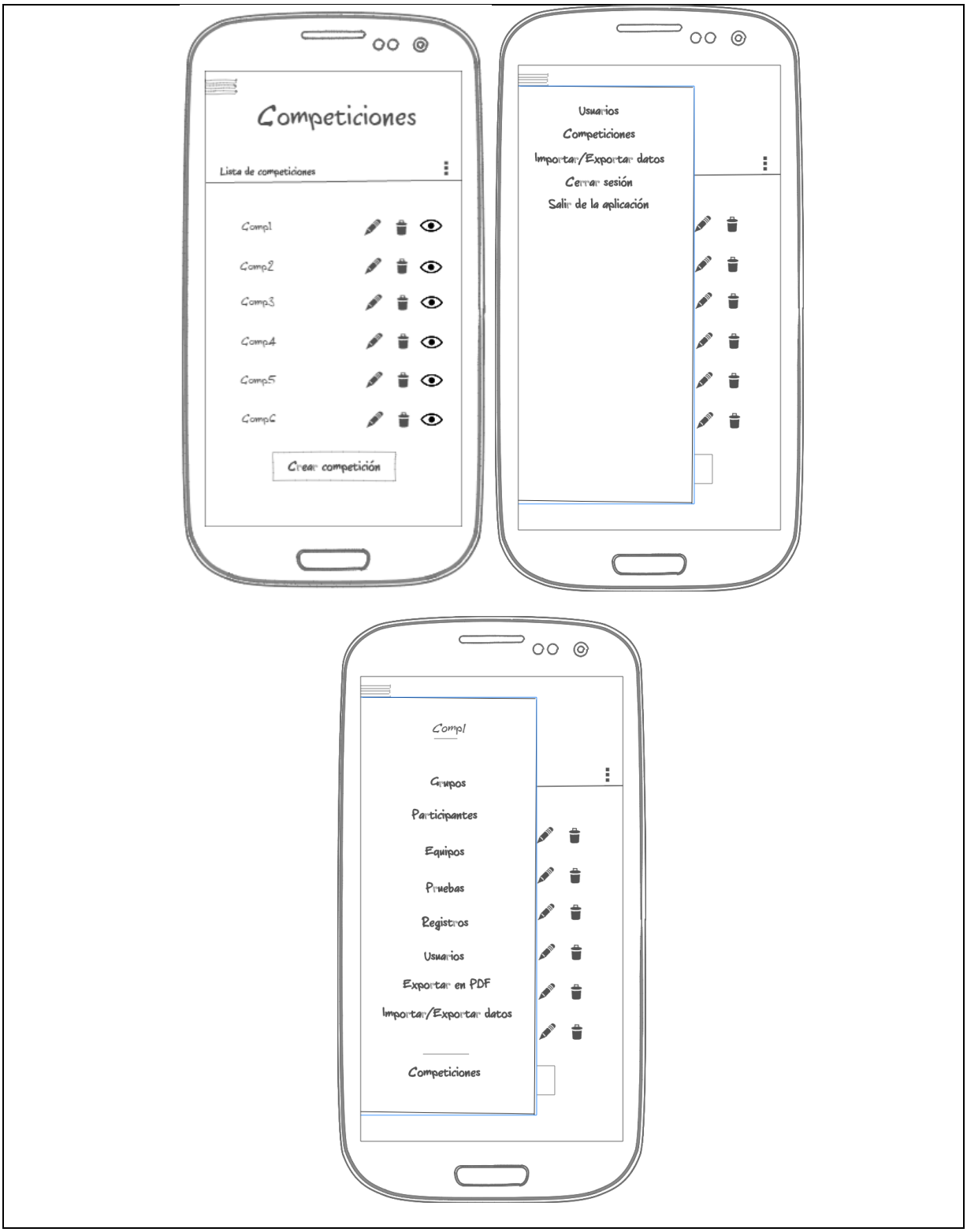


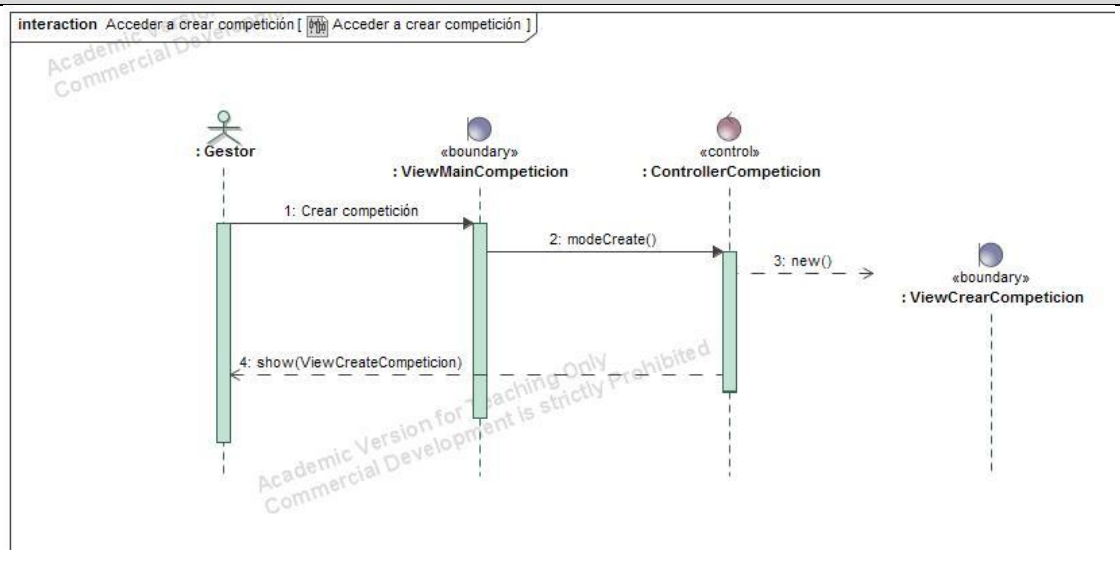
Nombre	Acceder al ámbito de una competición
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol **admin** o **gestor/invitado con permisos**). PRECOND3 (Existe alguna competición creada).
Escenario principal	
1- El usuario accede a la lista de competiciones. 2- El usuario selecciona la competición a la que desea acceder como ámbito de gestión. 3- El sistema informa al usuario del correcto acceso y actualiza su estado con la nueva configuración.	
Post-condiciones	POSCOND1 (El usuario está dentro del ámbito de la competición)
Escenarios alternativos	
Diagramas de secuencia	

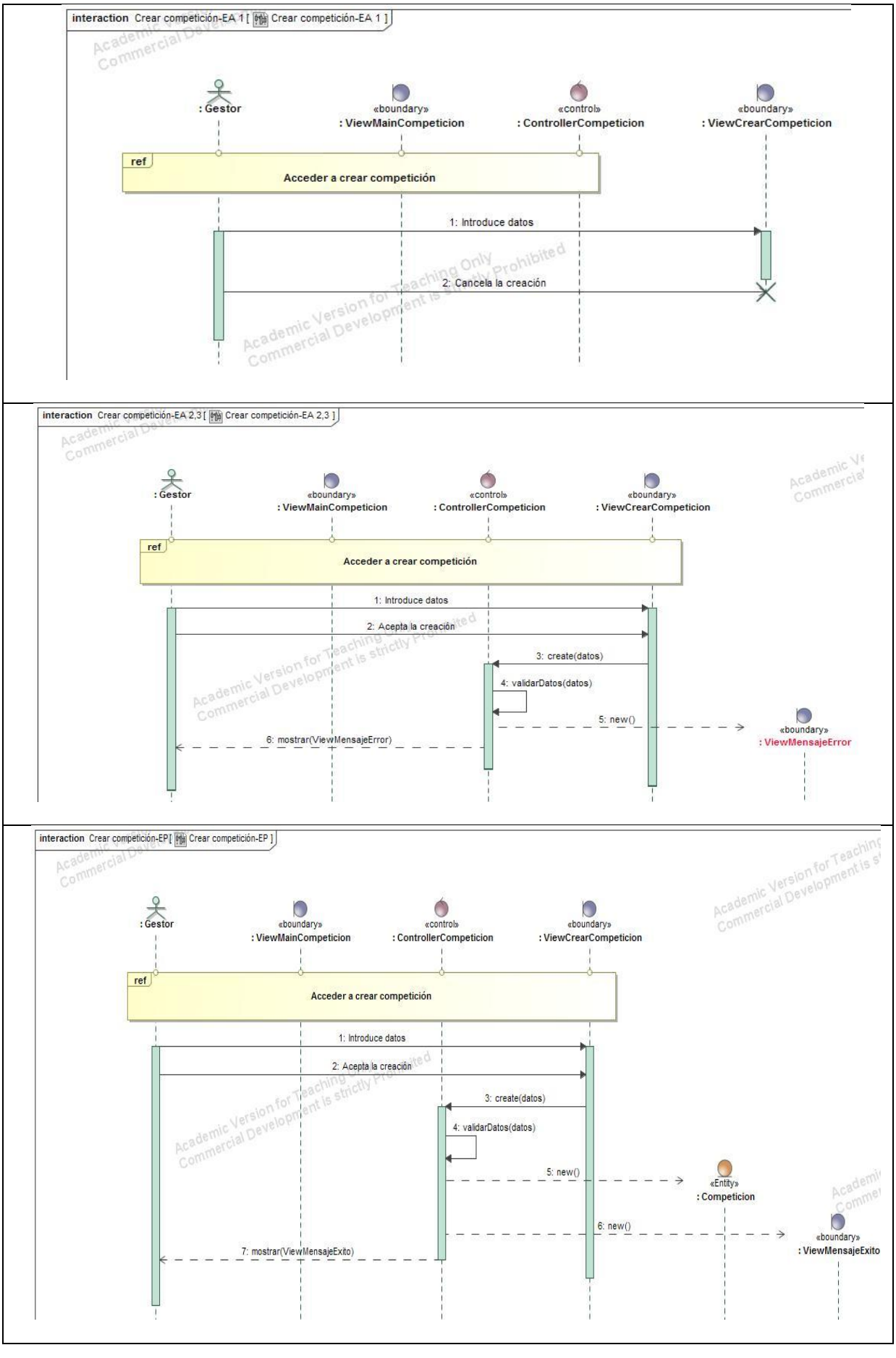




Mockups asociados

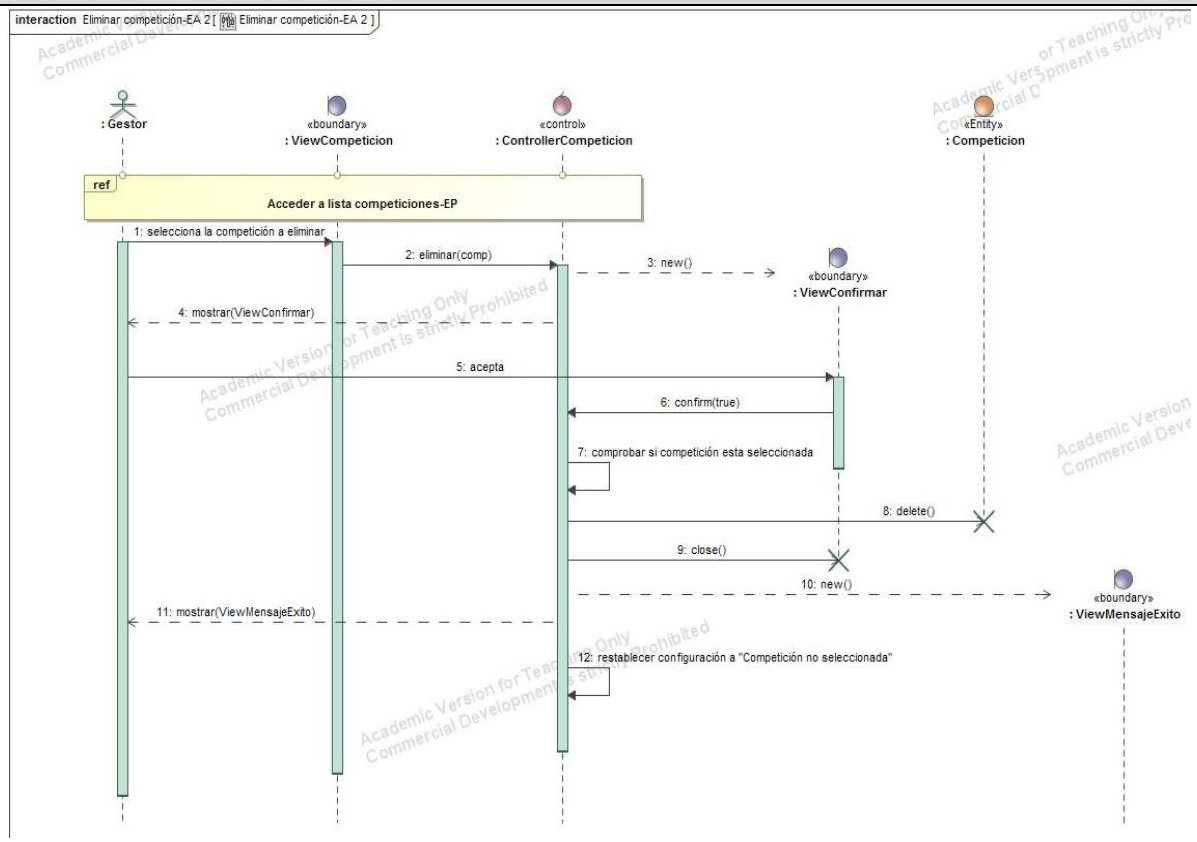

Competiciones
Lista de competiciones
Comp1
Comp2
Comp3
Comp4
Comp5
Comp6
Crear competición
Usuarios
Competiciones
Importar/Exportar datos
Cerrar sesión
Salir de la aplicación
Comp1
Grupos
Participantes
Equipos
Pruebas
Registros
Usuarios
Exportar en PDF
Importar/Exportar datos
Competiciones

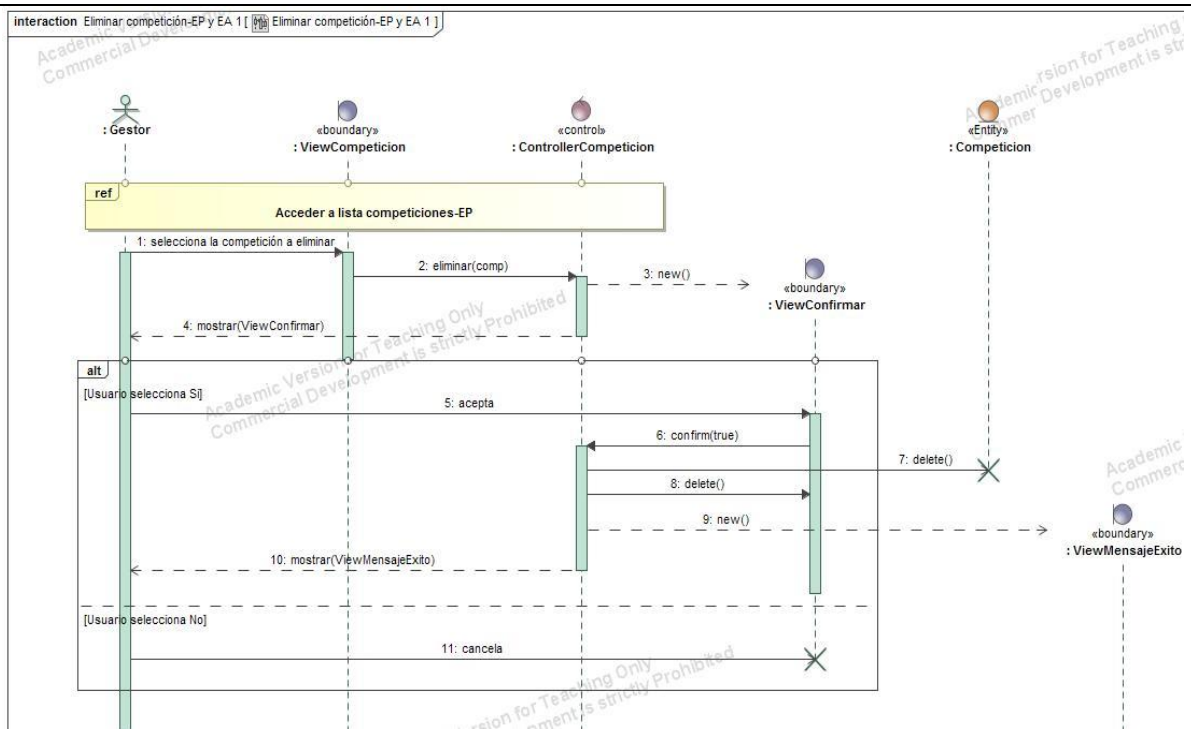
Nombre	Crear una competición
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol **admin** o **gestor**).
Escenario principal	
1- El usuario accede a "creación de una competición". 2- El usuario introduce los datos de la competición. 3- El usuario acepta la creación de la competición. 4- El sistema crea la nueva competición.	
Post-condiciones	POSCOND1 (La competición existe en la aplicación)
Escenarios alternativos	
3- El usuario cancela la creación de la competición. 4- El sistema no modifica nada y devuelve al usuario a una vista anterior.	
4- El sistema detecta que ya existe una competición con el mismo nombre y avisa al usuario.	
3- El sistema detecta que faltan campos necesarios por rellenar. 4- El sistema avisa al usuario del error.	
Diagramas de secuencia	
	


interaction Crear competición-EA 1 [Crear competición-EA 1]
: Gestor
«boundary»
: ViewMainCompeticion
«control»
: ControllerCompeticion
«boundary»
: ViewCrearCompeticion
ref
Acceder a crear competición
1: Introduce datos
2: Cancela la creación
interaction Crear competición-EA 2,3 [Crear competición-EA 2,3]
: Gestor
«boundary»
: ViewMainCompeticion
«control»
: ControllerCompeticion
«boundary»
: ViewCrearCompeticion
ref
Acceder a crear competición
1: Introduce datos
2: Acepta la creación
3: create(datos)
4: validarDatos(datos)
5: new()
«boundary»
: ViewMensajeError
6: mostrar(ViewMensajeError)
interaction Crear competición-EP[Crear competición-EP]
: Gestor
«boundary»
: ViewMainCompeticion
«control»
: ControllerCompeticion
«boundary»
: ViewCrearCompeticion
ref
Acceder a crear competición
1: Introduce datos
2: Acepta la creación
3: create(datos)
4: validarDatos(datos)
5: new()
«Entity»
: Competicion
6: new()
«boundary»
: ViewMensajeExito
7: mostrar(ViewMensajeExito)

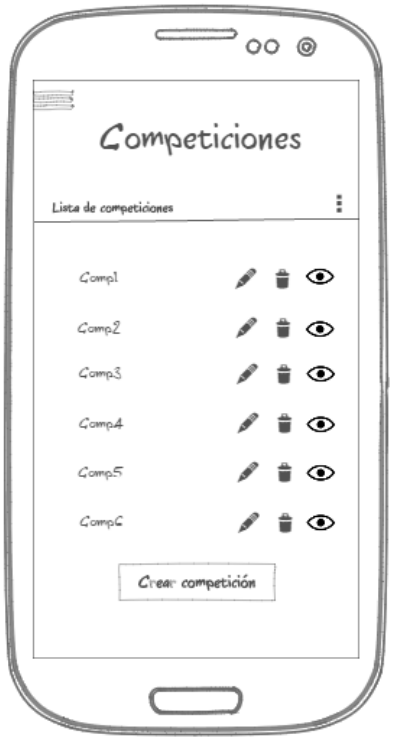
Mockups asociados


Nombre	Eliminar una competición
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol **admin** o **gestor**). PRECOND3 (Existe alguna competición creada).
Escenario principal	
1- El usuario accede a la lista de competiciones. 2- El usuario selecciona la competición a eliminar. 3- El sistema pide confirmación sobre la eliminación. 4- El usuario acepta eliminar la competición. 5- El sistema elimina la competición junto con toda la información relacionada con ella.	
Post-condiciones	POSCOND1 (la competición seleccionada no existe en la aplicación).

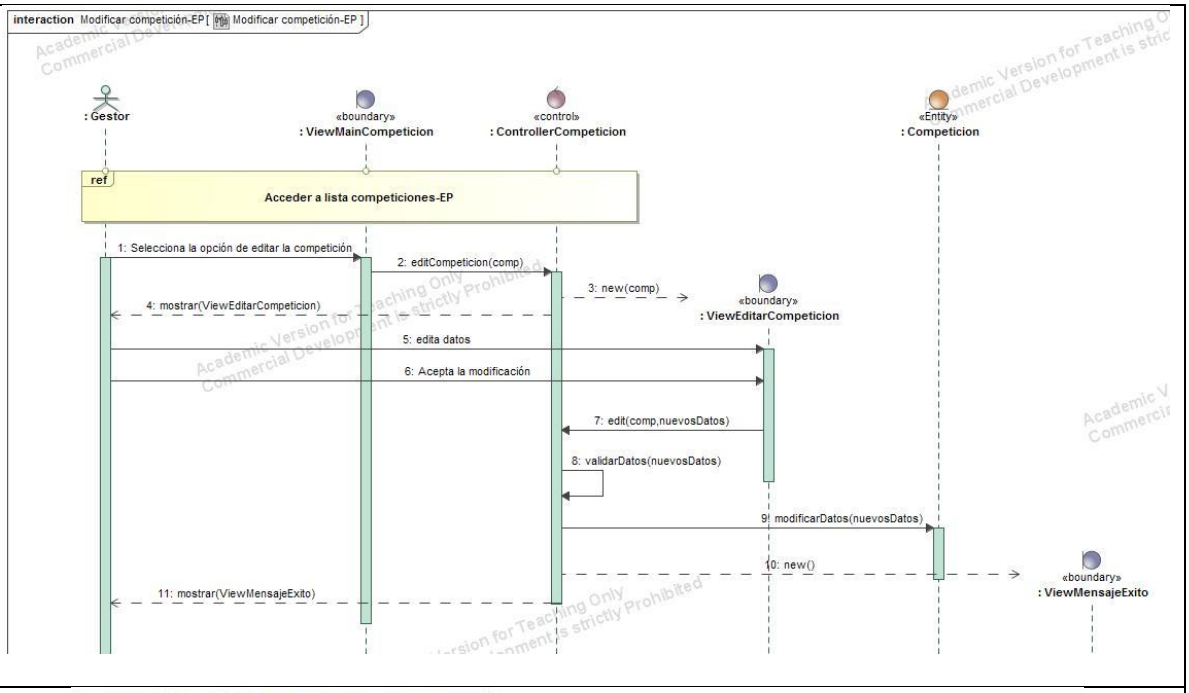
Escenarios alternativos
4- El usuario cancela la petición. 5- El sistema no modifica nada.
5- El sistema detecta que el usuario está dentro del ámbito de dicha competición. 6- El sistema elimina la competición junto con toda la información relacionada con ella y devuelve al usuario a un estado sin ámbito de competición.
Diagramas de secuencia


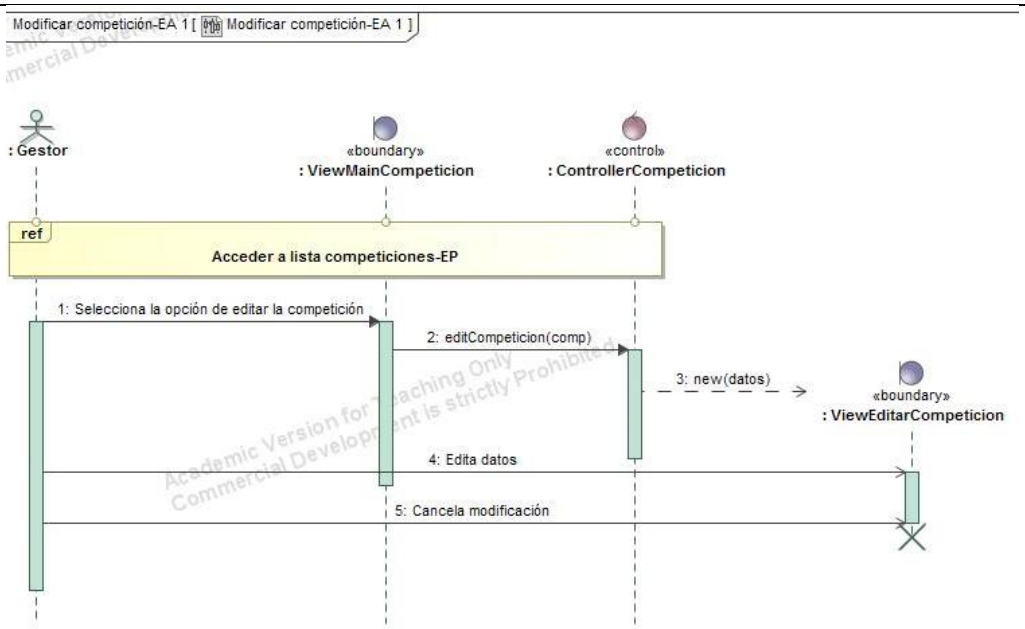


Mockups asociados



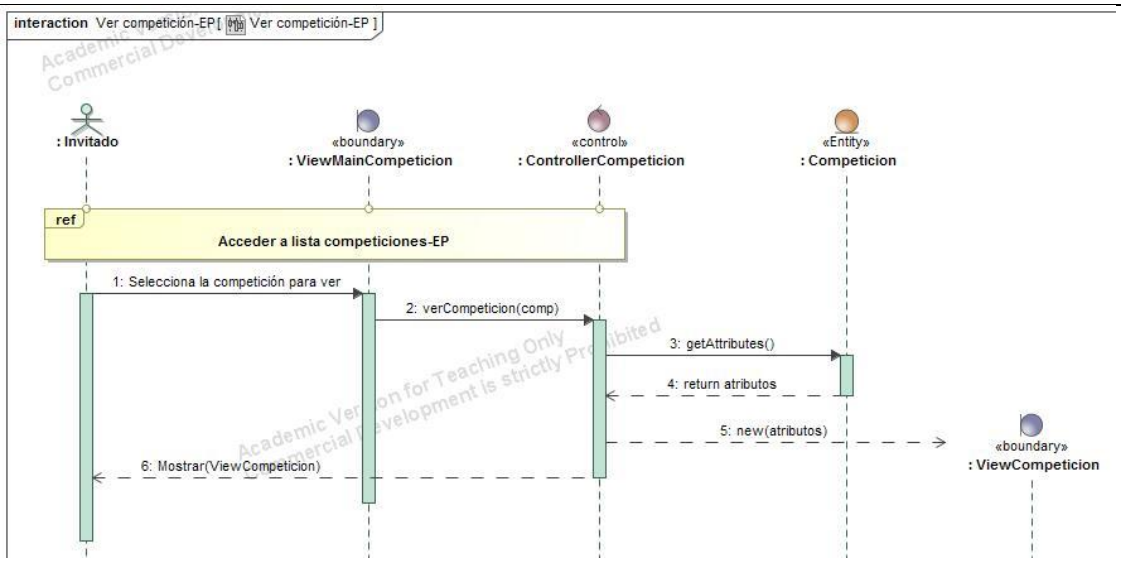
Nombre	Modificar los datos de una competición
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol **admin** o **gestor**). PRECOND3 (Existe alguna competición creada).
Escenario principal	
1- El usuario accede a la lista de competiciones. 2- El usuario selecciona la competición a modificar. 3- El usuario modifica los campos que necesite. 4- El usuario guarda la modificación. 5- El sistema guarda los nuevos datos de la competición.	
Post-condiciones	POSCOND1 (La competición está actualizada)
Escenarios alternativos	
4- El usuario cancela la modificación. 5- El sistema no modifica nada.	
3- El usuario modifica el campo "nombre". 4- El sistema detecta que ya existe una competición con el mismo nombre y avisa al usuario. 5- El usuario modifica el nombre escrito por otro diferente o cancela la creación. 6- Posteriormente si el nombre escrito es correcto y acepta el usuario, el sistema guarda la información de la nueva competición.	
4- El sistema detecta que faltan campos necesarios por rellenar. 5- El usuario rellena los datos o cancela la creación. 6- Posteriormente si el nombre escrito es correcto y acepta el usuario, el sistema guarda la información de la competición.	
Diagramas de secuencia	





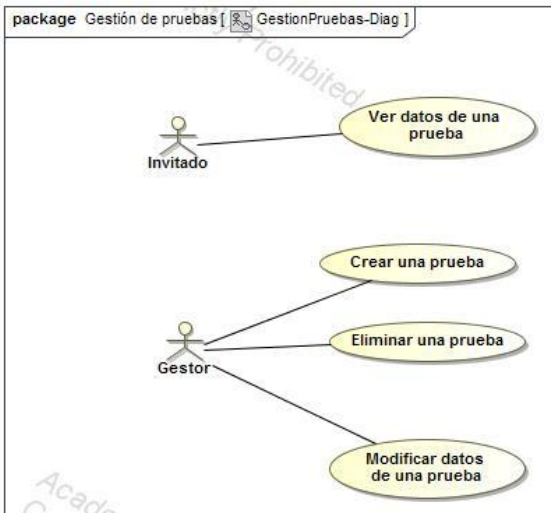
Mockups asociados

Nombre	Ver datos de una competición
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol **admin** o **gestor/invitado con permisos**). PRECOND3 (Existe alguna competición creada).
Escenario principal	
1- El usuario accede a la lista de competiciones. 2- El usuario selecciona la competición a visualizar. 3- El usuario puede ver todos los datos referentes a la competición seleccionada.	
Post-condiciones	
Escenarios alternativos	
Diagramas de secuencia	



Mockups asociados

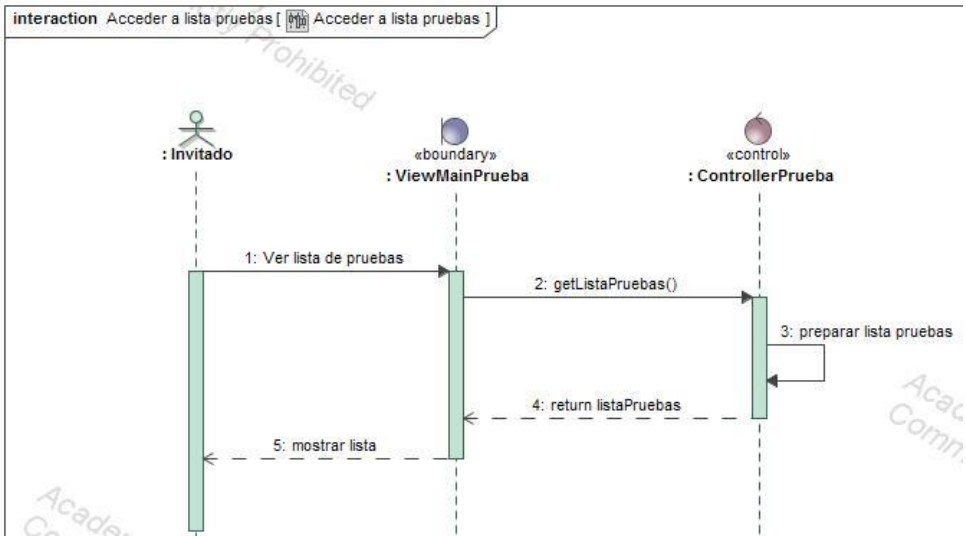
2. Gestión de pruebas

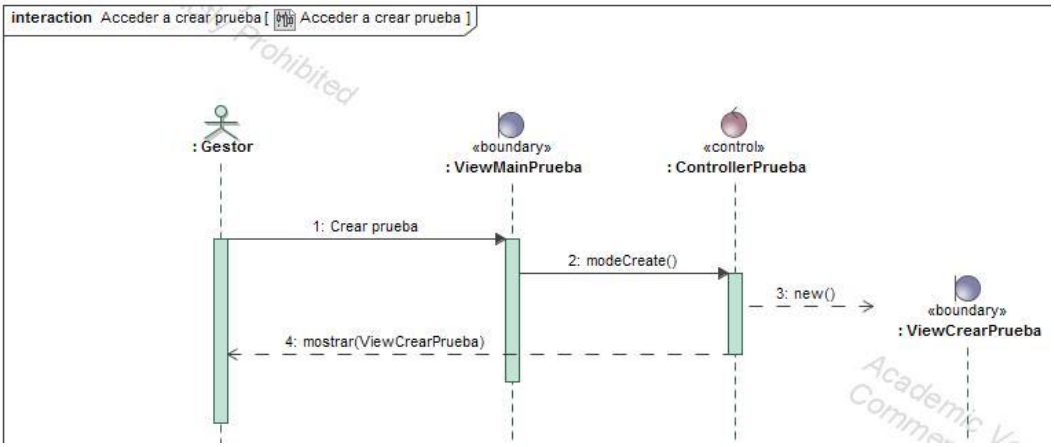


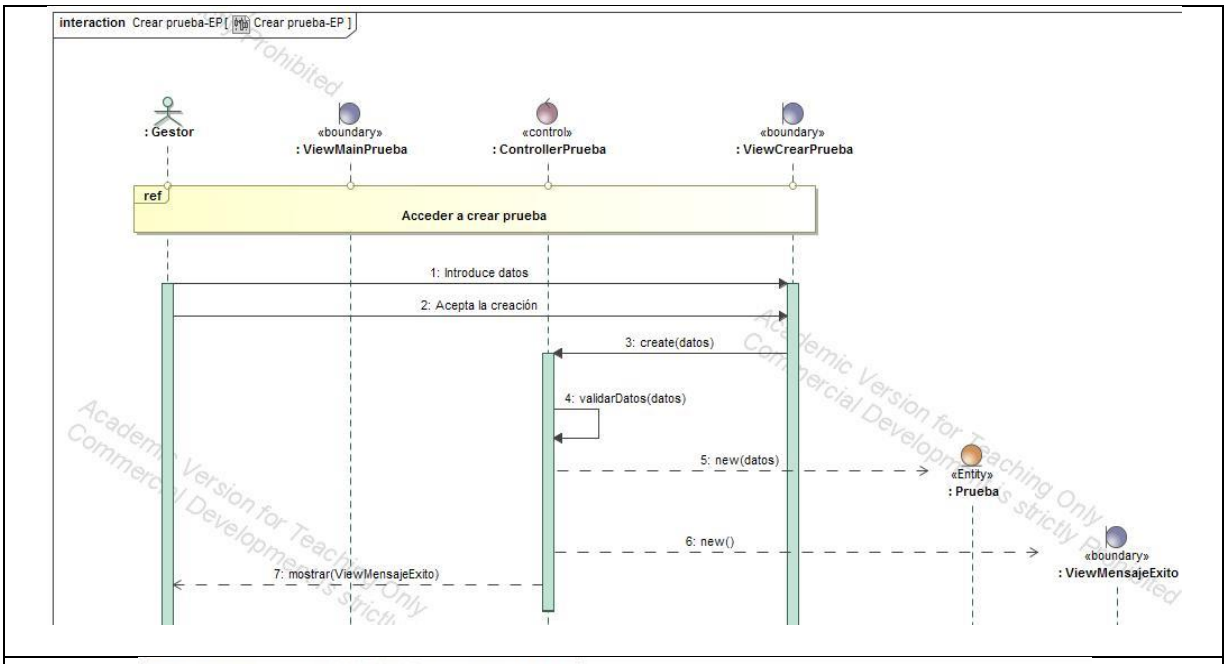
Nombre	Crear una prueba
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario selecciona "crear prueba". 2- El sistema muestra una vista con los campos a rellenar. 3- El usuario introduce los datos de la prueba. 4- El usuario acepta la creación. 5- El sistema crea la nueva prueba.	
Post-condiciones	POSCOND1 (la prueba existe en la competición)
Escenarios alternativos	
4- El usuario cancela la creación. 5- El sistema no modifica nada y devuelve a usuario a una vista anterior.	
4- El sistema detecta que ya existe una prueba con el mismo nombre y avisa al usuario. 5- El usuario cambia el nombre por otro o cancela la creación.	

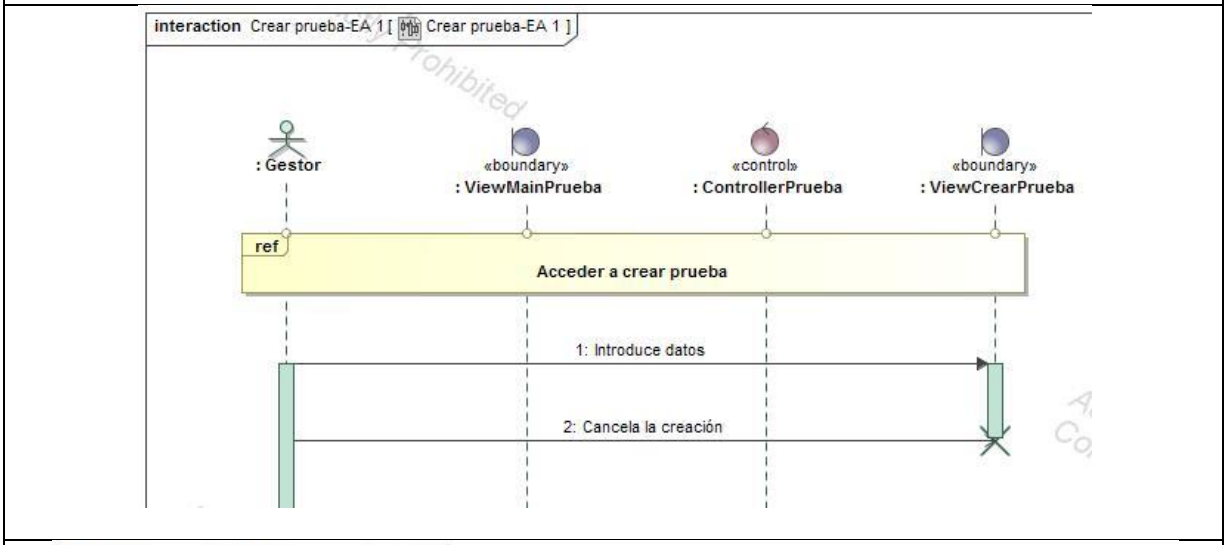
5- El sistema detecta que existen datos obligatorios sin rellenar.
6- El sistema informa al usuario de dicho error y cancela la operación.

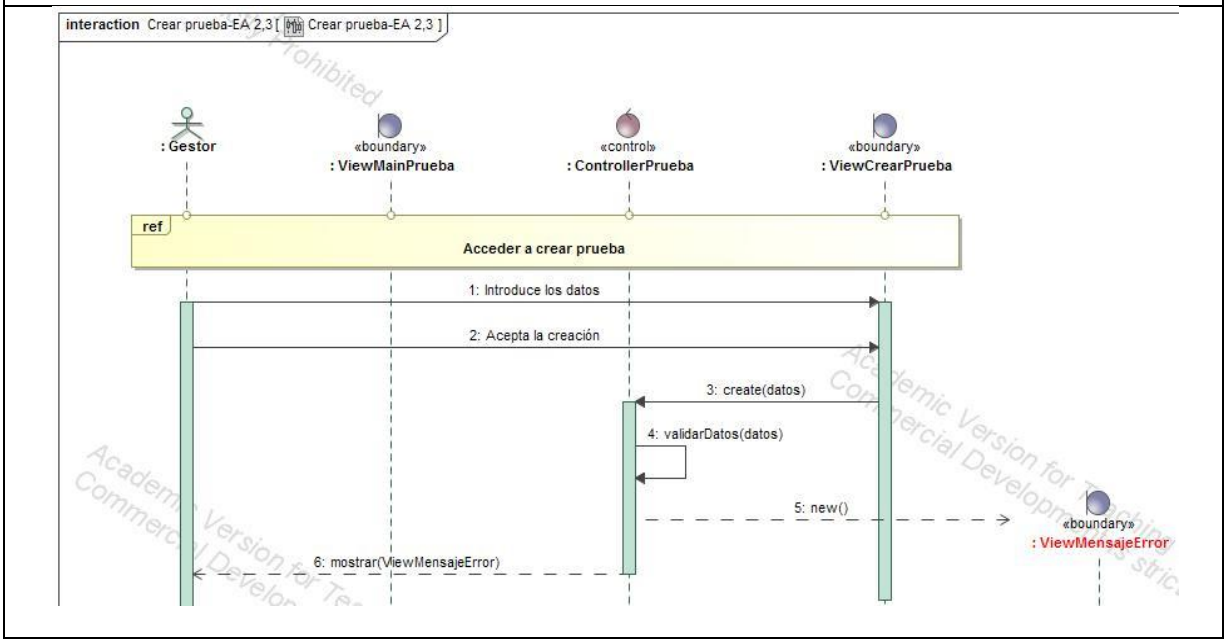
Diagramas de secuencia






interaction Crear prueba-EP [Crear prueba-EP]
: Gestor
«boundary»
: ViewMainPrueba
«control»
: ControllerPrueba
«boundary»
: ViewCrearPrueba
ref
Acceder a crear prueba
1: Introduce datos
2: Acepta la creación
3: create(datos)
4: validarDatos(datos)
5: new(datos)
«Entity»
: Prueba
6: new()
«boundary»
: ViewMensajeExito
7: mostrar(ViewMensajeExito)


interaction Crear prueba-EA 1 [Crear prueba-EA 1]
: Gestor
«boundary»
: ViewMainPrueba
«control»
: ControllerPrueba
«boundary»
: ViewCrearPrueba
ref
Acceder a crear prueba
1: Introduce datos
2: Cancela la creación


interaction Crear prueba-EA 2,3 [Crear prueba-EA 2,3]
: Gestor
«boundary»
: ViewMainPrueba
«control»
: ControllerPrueba
«boundary»
: ViewCrearPrueba
ref
Acceder a crear prueba
1: Introduce los datos
2: Acepta la creación
3: create(datos)
4: validarDatos(datos)
5: new()
«boundary»
: ViewMensajeError
6: mostrar(ViewMensajeError)

Mockups asociados

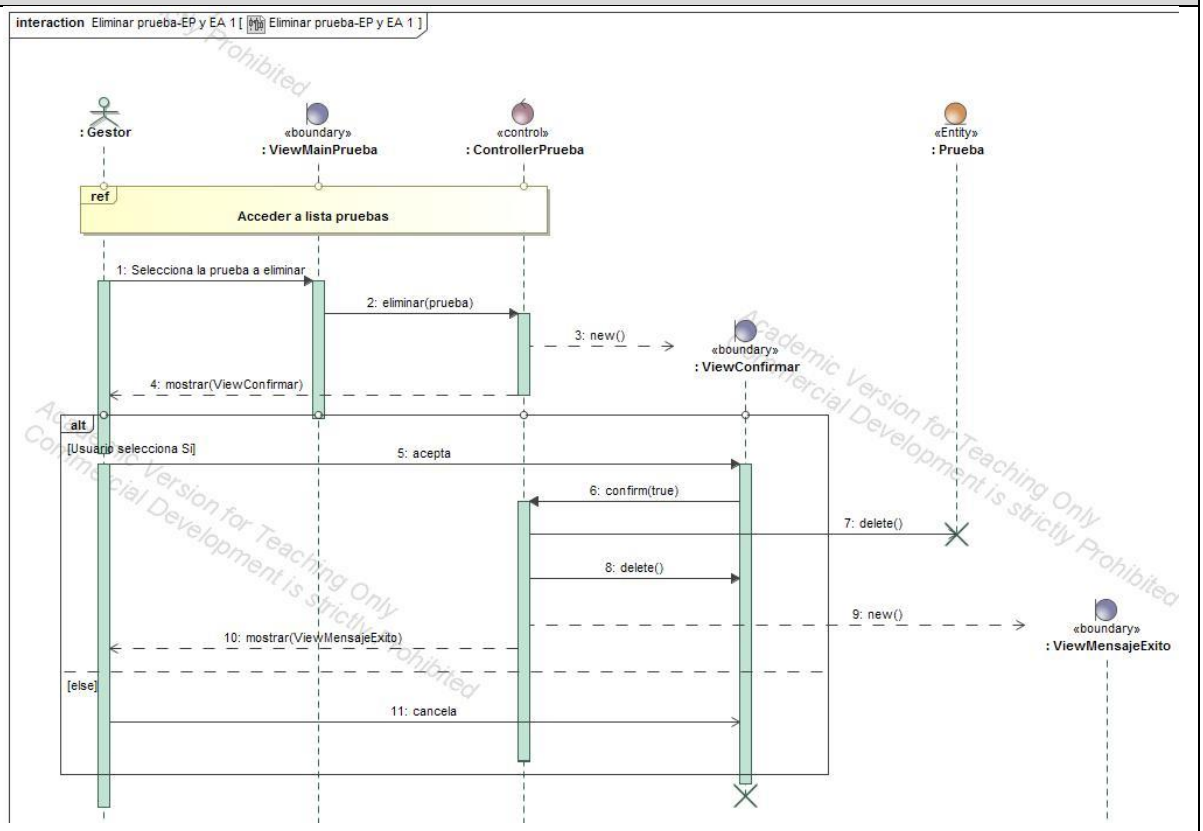

Nombre	Eliminar una prueba
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe alguna prueba en la competición).
Escenario principal	
1- El usuario selecciona la prueba a eliminar de la lista de pruebas. 2- El usuario selecciona "eliminar". 3- El sistema pide confirmación sobre la eliminación. 4- El usuario acepta eliminar la prueba. 5- El sistema elimina la prueba junto con toda la información relacionada con ella.	

Post-condiciones	POSCOND1 (la prueba no existe en la competición)

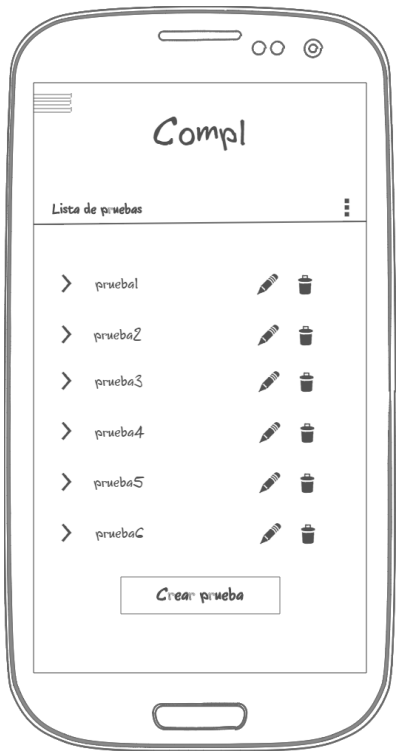
Escenarios alternativos

4- El usuario cancela la petición.
5- El sistema no modifica nada.

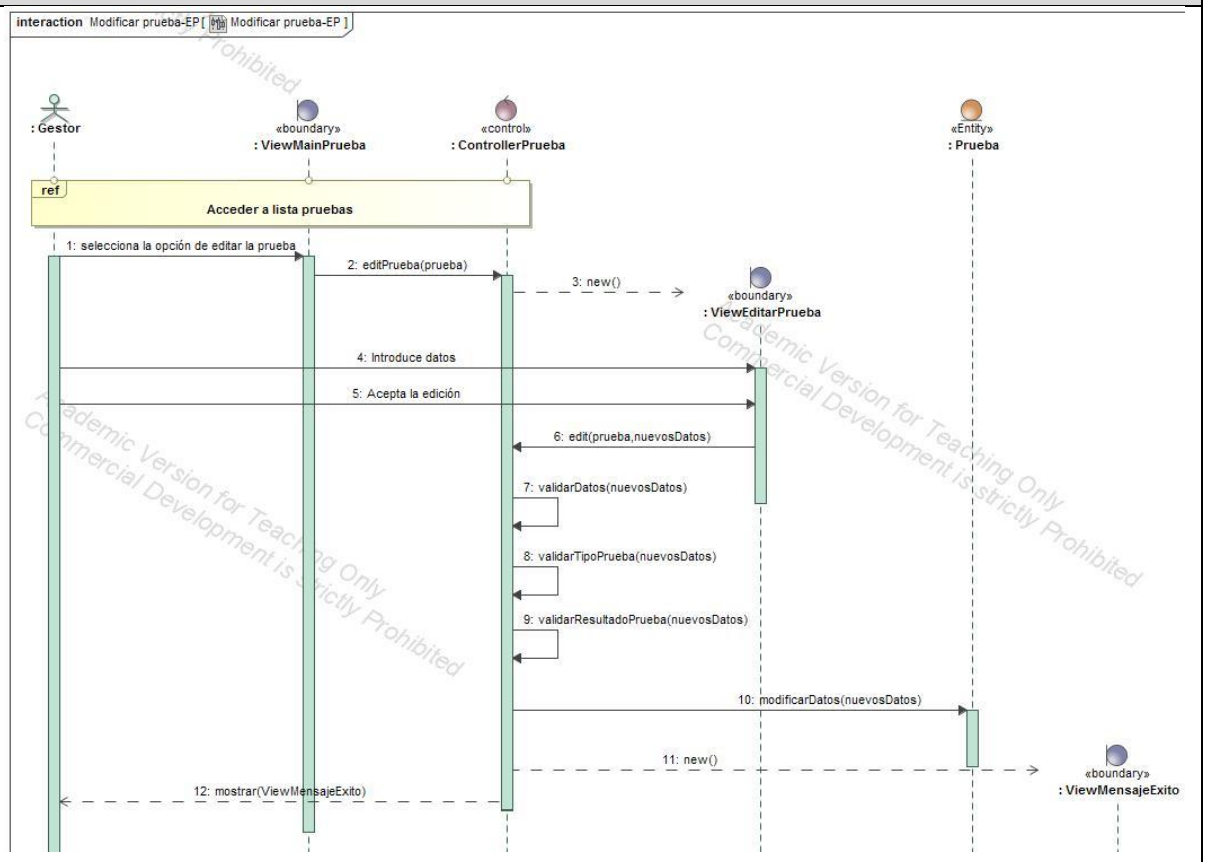
Diagramas de secuencia

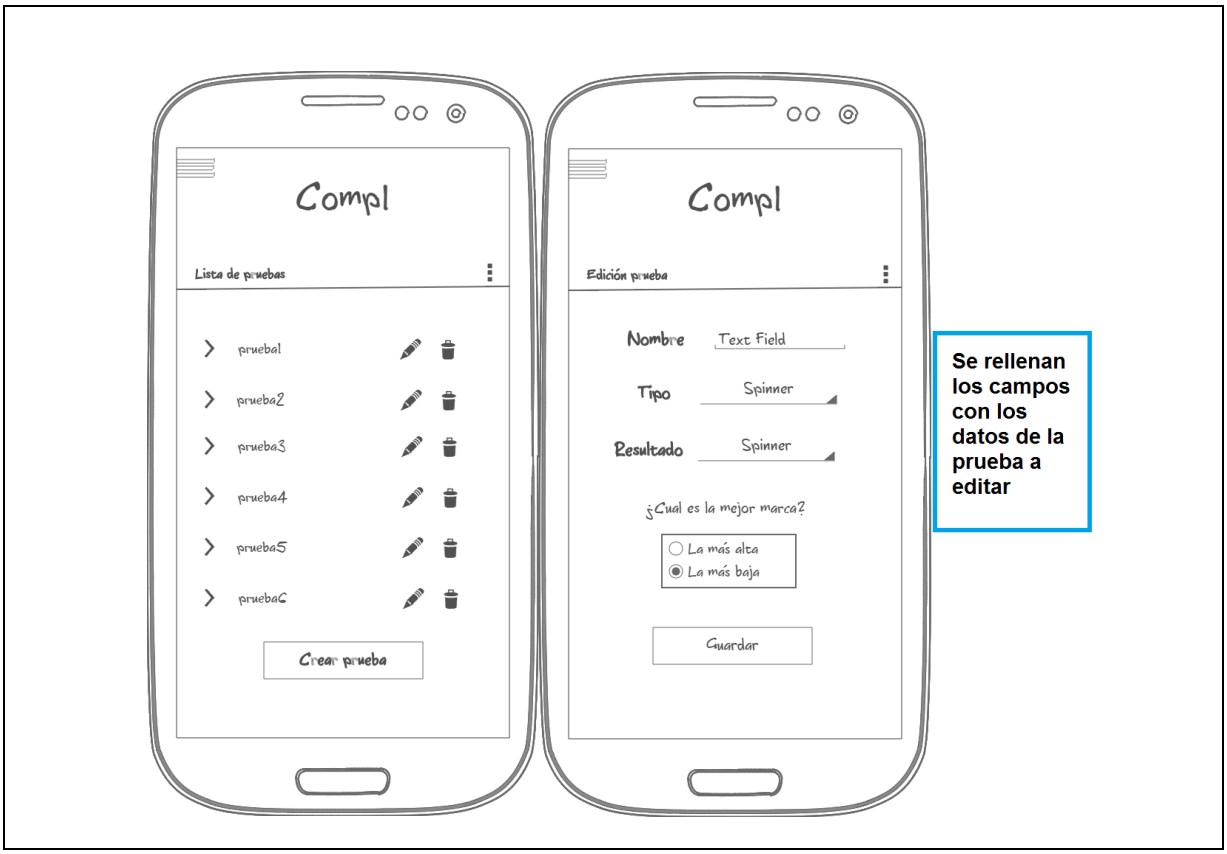


Mockups asociados



Nombre	Modificar datos de una prueba
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe alguna prueba en la competición).
Escenario principal	
1- El usuario accede al ámbito de una competición. 2- El usuario selecciona una prueba de la lista de pruebas. 3- El usuario selecciona "modificar prueba". 4- El usuario modifica los campos que necesite. 5- El usuario guarda la modificación. 6- El sistema guarda los nuevos datos de la prueba.	
Post-condiciones	POSCOND1 (la prueba esta actualizada)

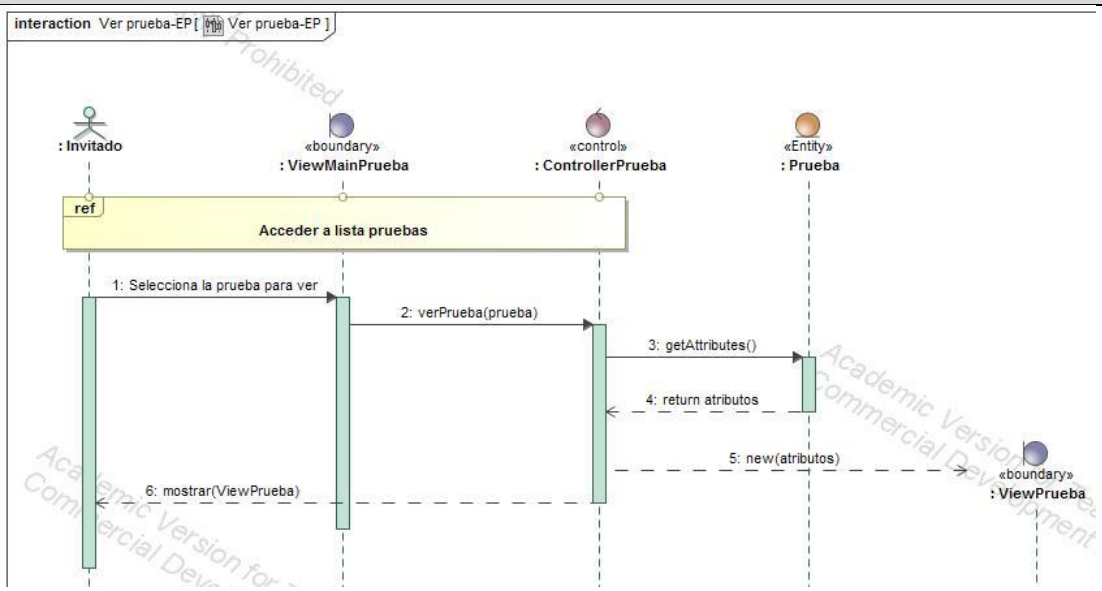
Escenarios alternativos
4- El usuario cancela la petición. 5- El sistema no modifica nada.
4- El sistema detecta que ya existe una prueba con el mismo nombre y avisa al usuario. 5- El usuario cambia el nombre por otro o cancela la modificación.
5- El sistema detecta que existen datos obligatorios sin rellenar. 6- El sistema informa al usuario de dicho error y cancela la operación.
2- El usuario intenta modificar el tipo de prueba. 3- El sistema detecta que la prueba contiene registros de participantes o equipos e informa al usuario impidiendo la edición de la prueba.
2- El usuario intenta modificar el tipo de resultado de prueba. 3- El sistema detecta que la prueba está asignada a algún participante/equipo e informa al usuario impidiendo la edición de la prueba.
Diagramas de secuencia

Mockups asociados



Nombre	Ver datos de una prueba
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor/invitado con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe alguna prueba en la competición).
Escenario principal	
1- El usuario accede al ámbito de la competición. 2- El usuario selecciona una prueba de la lista de pruebas. 3- El sistema muestra al usuario información sobre la prueba.	
Post-condiciones	

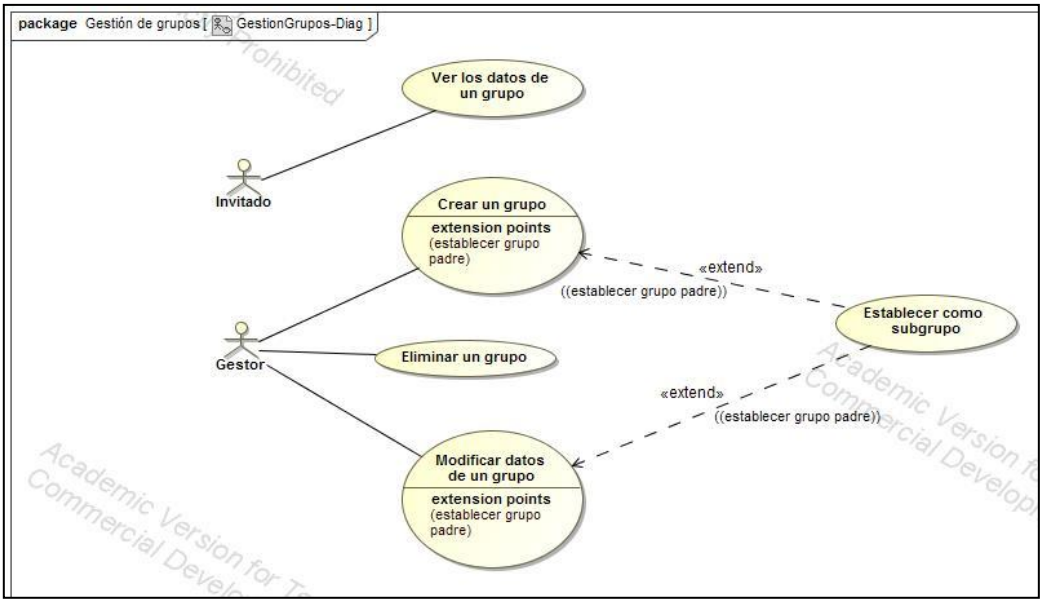
Escenarios alternativos

Diagramas de secuencia



Mockups asociados

3. Gestión de grupos

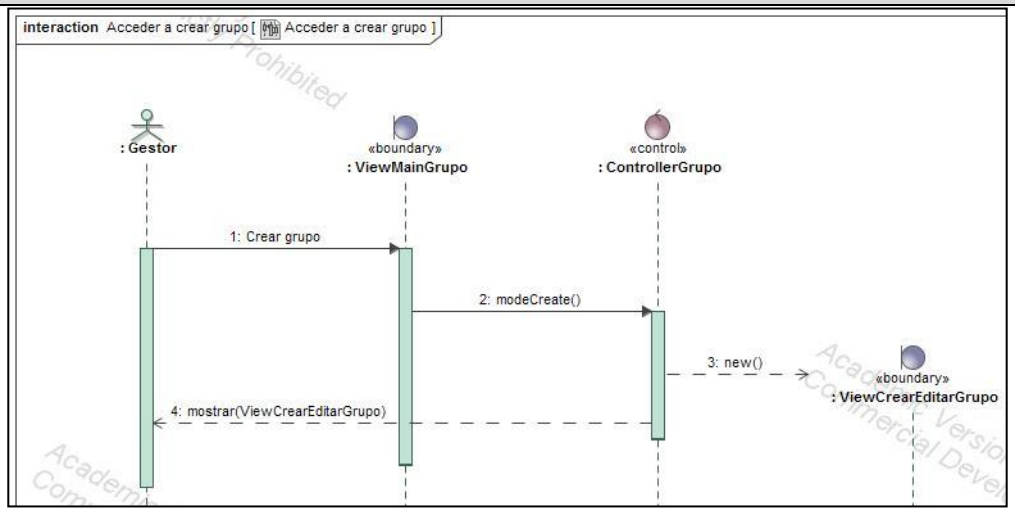


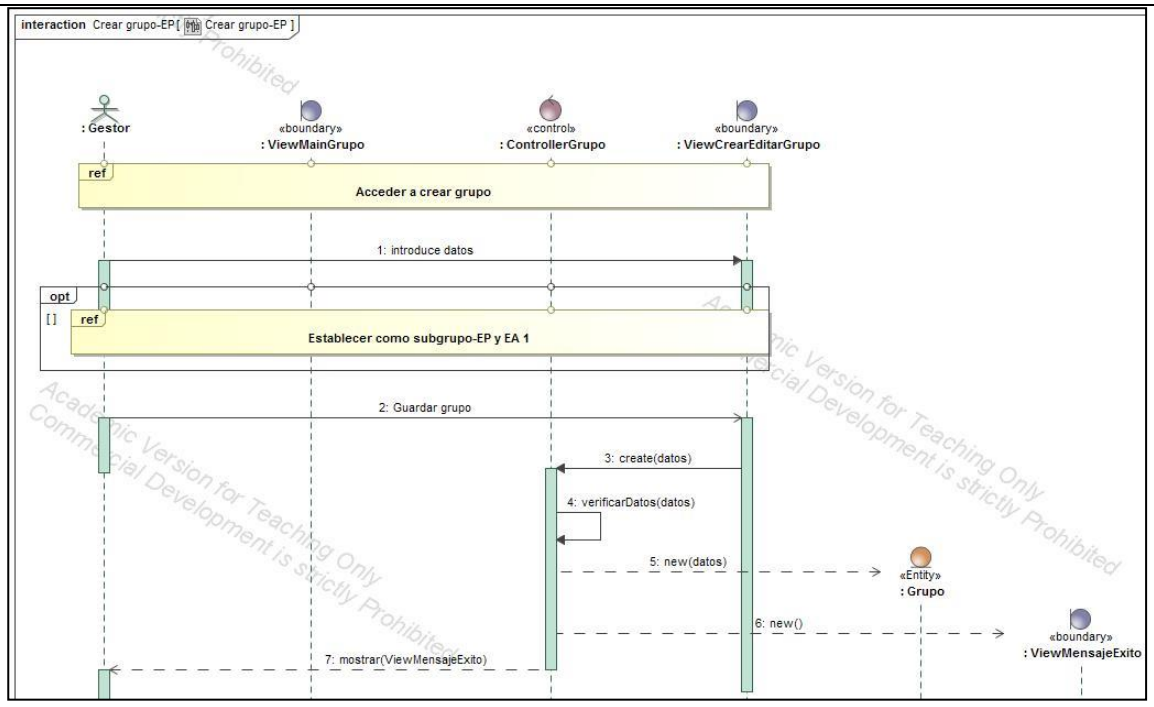
Nombre	Crear un grupo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario selecciona la opción que permite crear un nuevo grupo. 2- El sistema muestra una vista con los campos a rellenar. 3- El usuario introduce los datos del grupo. (establecer grupo padre). 4- El usuario selecciona la opción de guardar el grupo. 5- El sistema crea el nuevo grupo.	
Post-condiciones	POSCOND1 (el grupo existe en la competición)
Escenarios alternativos	
4- El usuario cancela la creación. 5- El sistema no modifica nada y devuelve a usuario a una vista anterior.	

4- El sistema detecta que ya existe un grupo con el mismo nombre y avisa al usuario.
5- El usuario cambia el nombre por otro o cancela la creación.

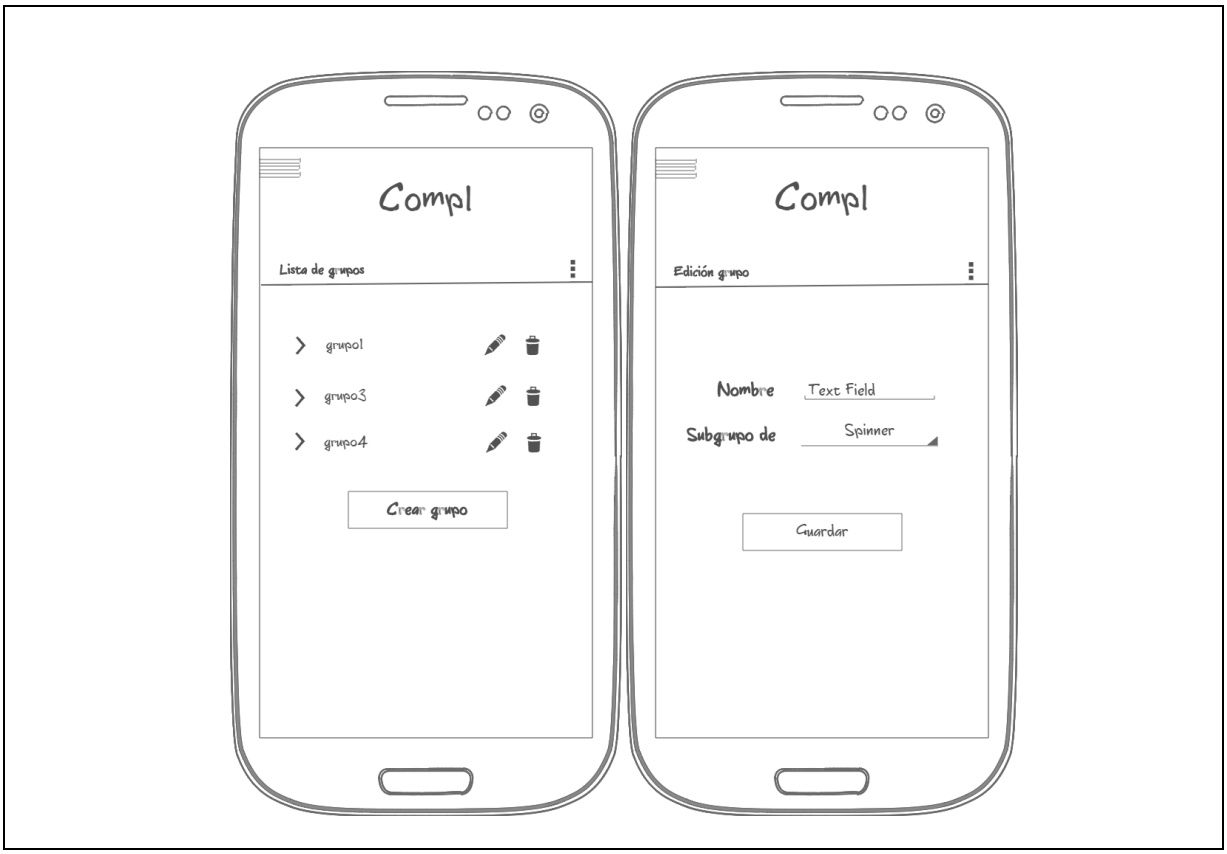
3- El sistema detecta que existen datos obligatorios sin rellenar.
4- El sistema informa al usuario de dicho error y cancela la operación.

Diagramas de secuencia





Mockups asociados



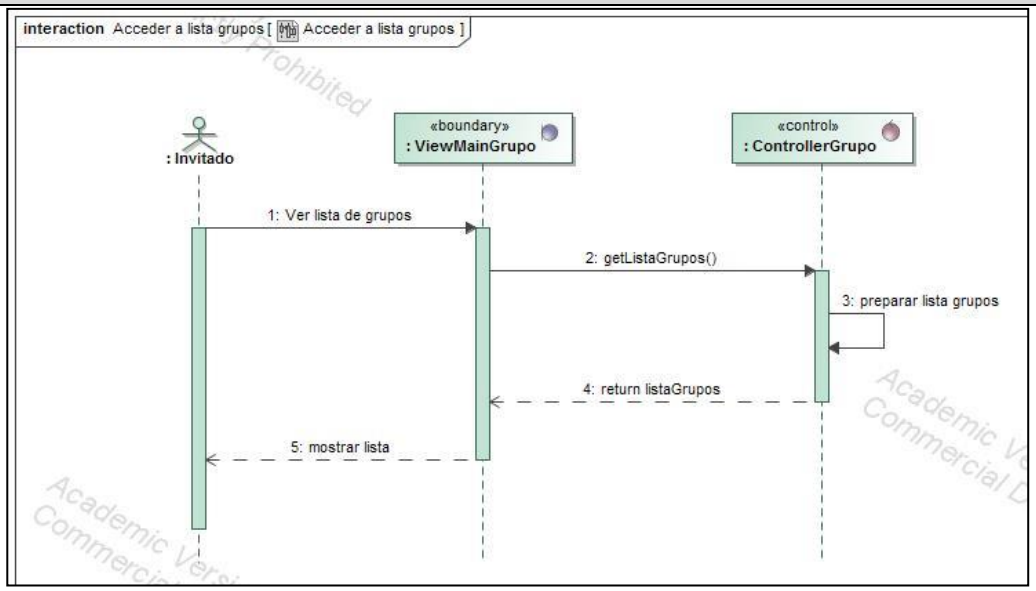
Nombre	Eliminar un grupo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe algún grupo en la competición)
Escenario principal	
1- El usuario selecciona el grupo a eliminar de la lista de grupos. 2- El usuario selecciona "eliminar". 3- El sistema pide confirmación sobre la eliminación. 4- El usuario acepta eliminar el grupo. 5- El sistema elimina el grupo junto con todos sus registros, participantes, equipos y subgrupos.	

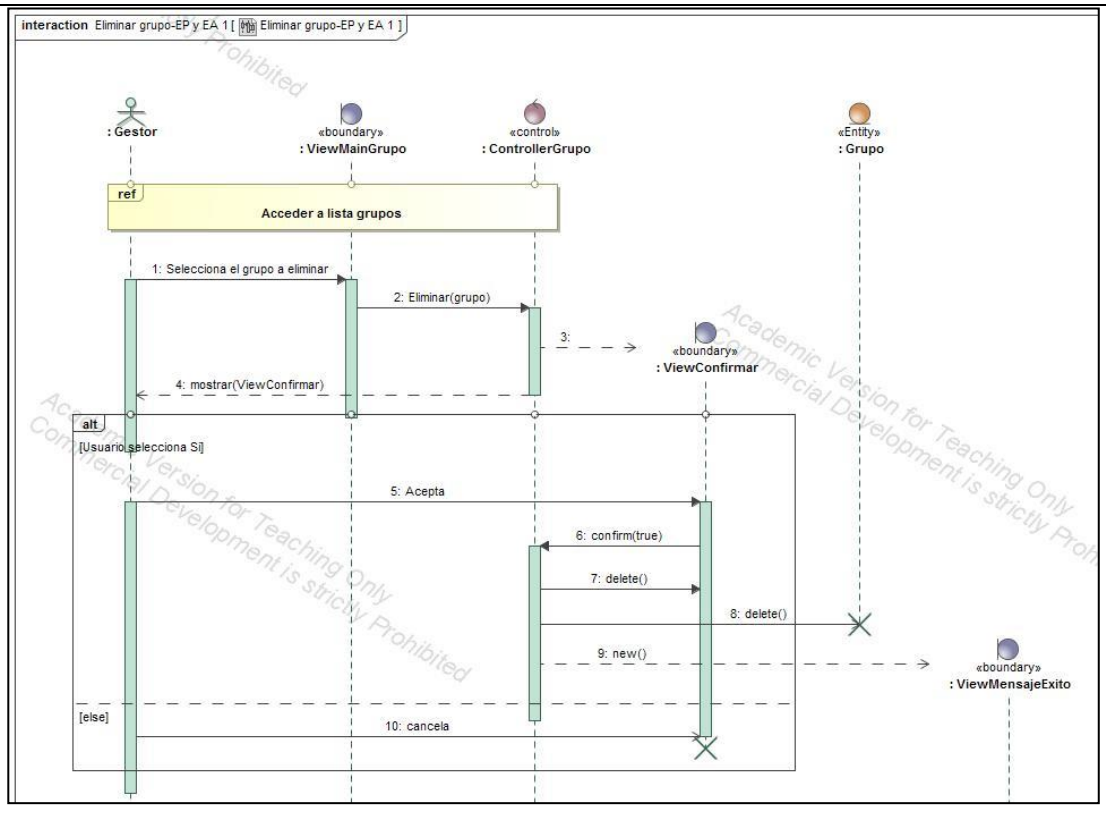
Post-condiciones	POSCOND1 (el grupo seleccionado junto con sus registros, participantes, equipos y subgrupos no existe en la competición).

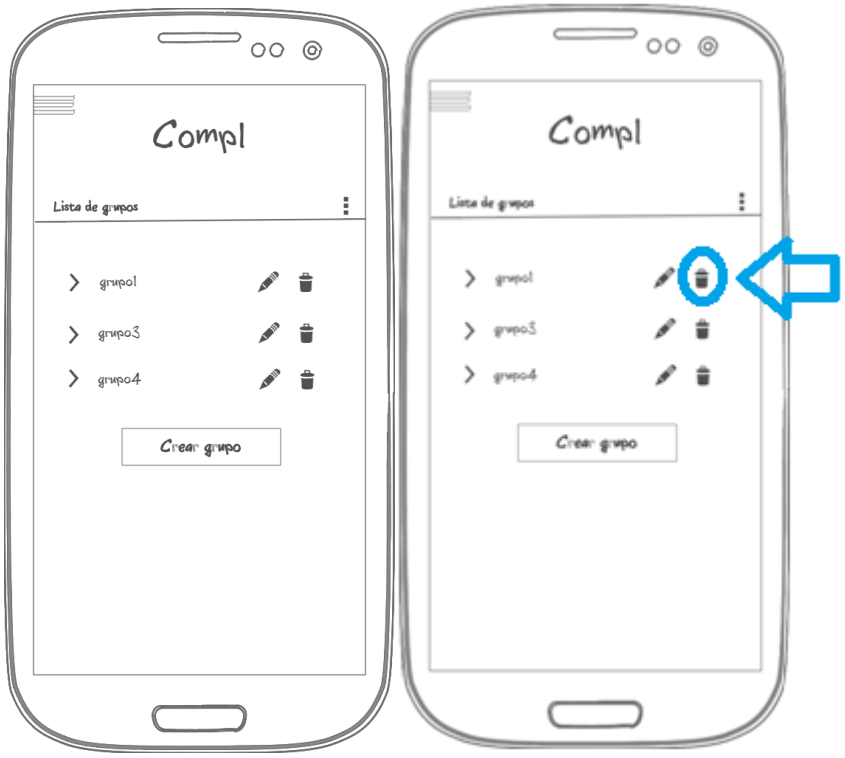
Escenarios alternativos

4- El usuario cancela la petición.
5- El sistema no modifica nada.

Diagramas de secuencia





Mockups asociados


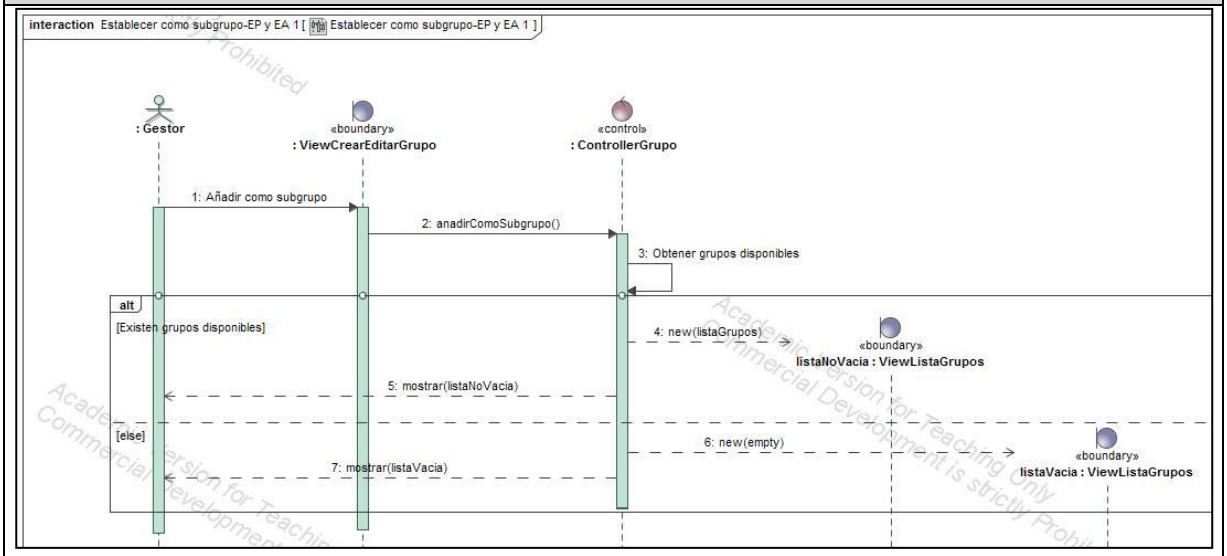
Nombre	Establecer como subgrupo de otro grupo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe algún grupo en la competición)
Escenario principal	
1- El usuario selecciona la opción de añadir el grupo como subgrupo de otro. 2- El sistema muestra una lista de grupos disponibles de los que el grupo puede formar parte como subgrupo. 3- El usuario selecciona un grupo de la lista.	

Post-condiciones	POSCOND1 (el grupo forma parte de los subgrupos del grupo señalado).

Escenarios alternativos

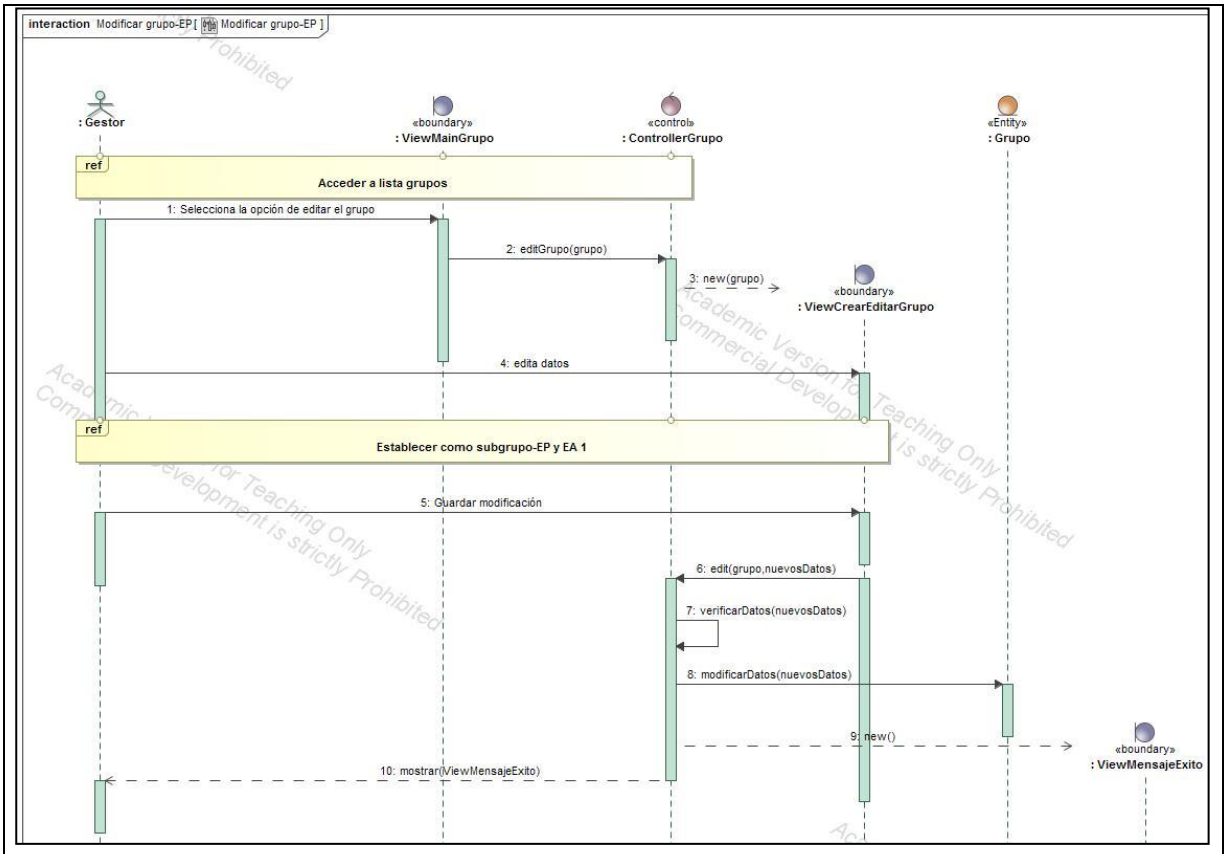
2- El sistema detecta que no existe ningún grupo disponible para que el grupo puede formar parte como subgrupo.
3- El sistema informa al usuario de dicha situación.

Diagramas de secuencia



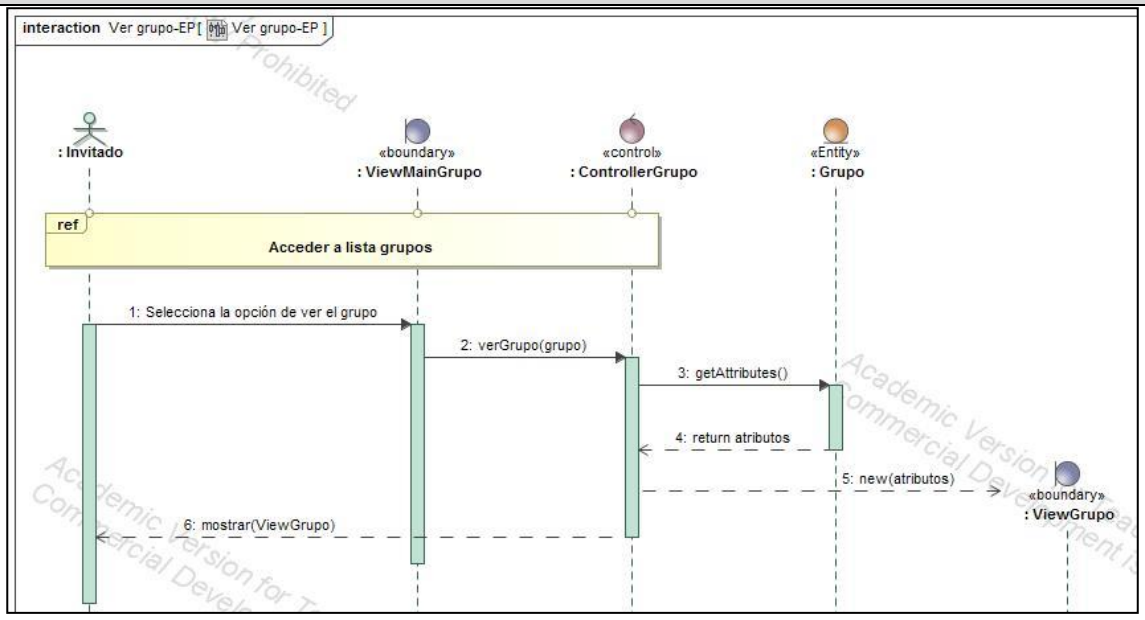
Mockups asociados

Nombre	Modificar datos de un grupo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe algún grupo en la competición)
Escenario principal	
1- El usuario selecciona un grupo de la lista de grupos. 2- El usuario selecciona la opción que permite modificar el grupo. 3- El usuario modifica los campos que necesite. (establecer grupo padre). 4- El usuario guarda la modificación. 5- El sistema guarda los nuevos datos del grupo.	
Post-condiciones	POSCOND1 (el grupo está actualizado).
Escenarios alternativos	
4- El usuario cancela la modificación. 5- El sistema no modifica nada.	
4- El sistema advierte que ya existe un grupo con el mismo nombre dentro de la competición. 5- El usuario cambia el nombre por otro o cancela la creación.	
3- El sistema detecta que existen datos obligatorios sin rellenar. 4- El sistema informa al usuario de dicho error y cancela la operación.	
Diagramas de secuencia	

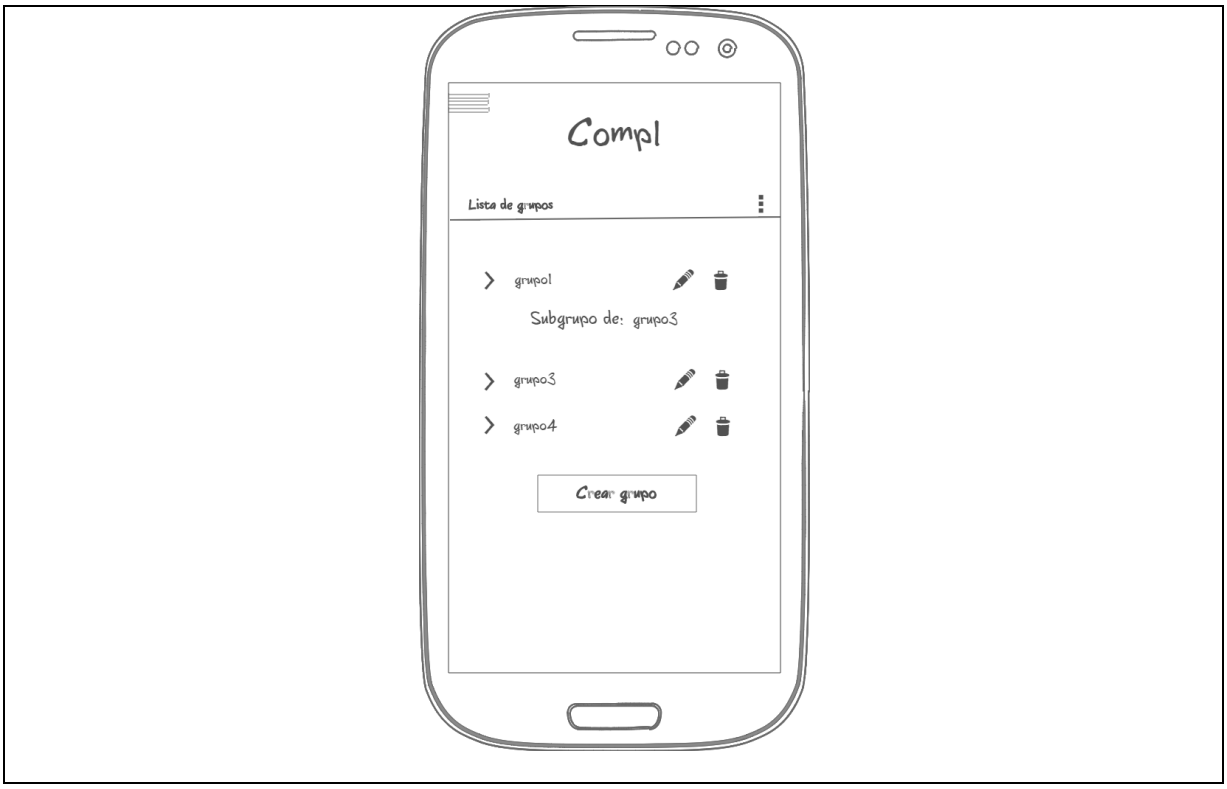


Mockups asociados

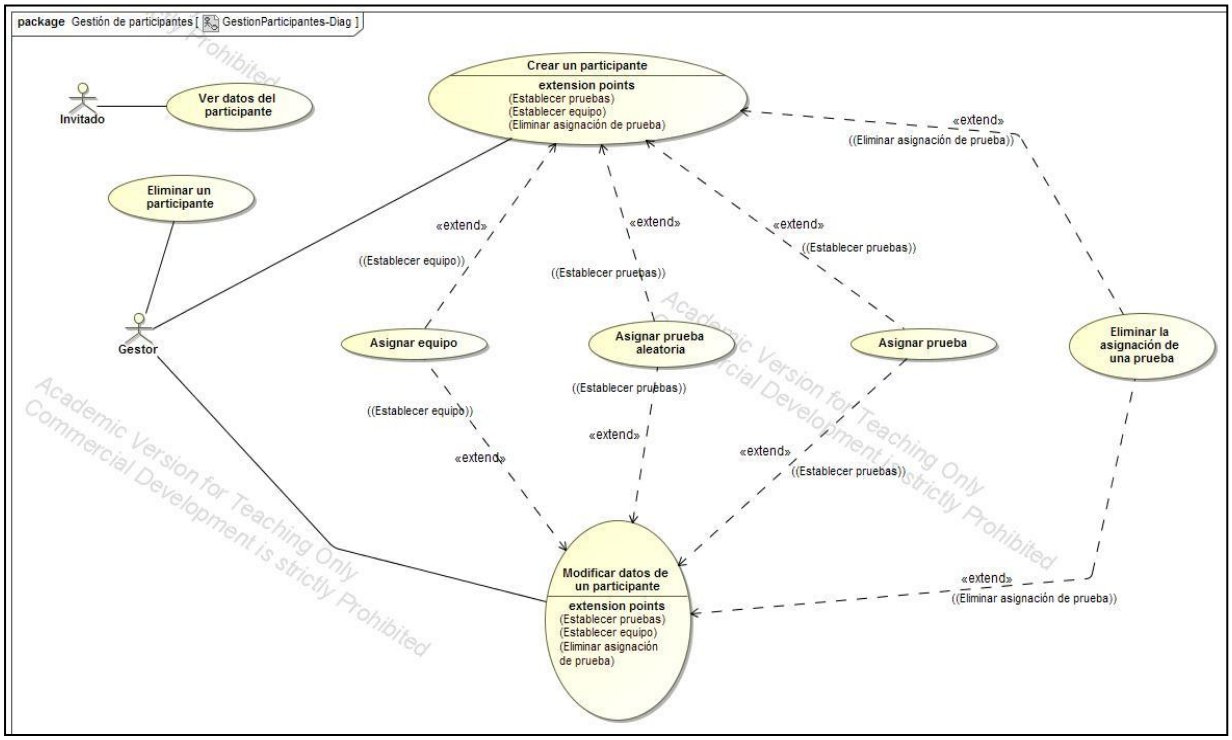
Nombre	Ver los datos de un grupo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor/invitado con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe algún grupo en la competición).
Escenario principal	
1- El usuario selecciona la opción que permite ver los datos de un grupo de la lista. 2- El sistema muestra al usuario información sobre el grupo.	
Post-condiciones	
Escenarios alternativos	
Diagramas de secuencia	



Mockups asociados



4. Gestión de participantes



Nombre	Crear un participante
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (Existe algún grupo creado en la competición). PRECOND4 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario selecciona "crear participante". 2- El sistema muestra una vista con los campos a rellenar. 3- El usuario introduce los datos del participante. (Establecer pruebas). 4- El usuario acepta la creación. 5- El sistema crea el nuevo participante.	
Post-condiciones	POSCOND1 (el participante existe en la competición).
Escenarios alternativos	
4- El usuario cancela la creación. 5- El sistema no modifica nada y devuelve a usuario a una vista anterior.	
4- El sistema detecta que ya existe un participante con el mismo dorsal y avisa al usuario. 5- El usuario cambia el dorsal por otro o cancela la creación.	
4- El sistema detecta que el equipo asignado no pertenece al mismo grupo que el establecido para el participante. 5- El sistema avisa al usuario y cambia automáticamente el grupo asignado al participante al mismo que el equipo asignado.	
4- El sistema detecta que ya existe un participante con el mismo dorsal y avisa al usuario. 5- El usuario cambia el dorsal por otro o cancela la creación.	
5- El sistema detecta que existen datos obligatorios sin rellenar. 6- El sistema informa al usuario de dicho error y cancela la operación.	
Diagramas de secuencia	

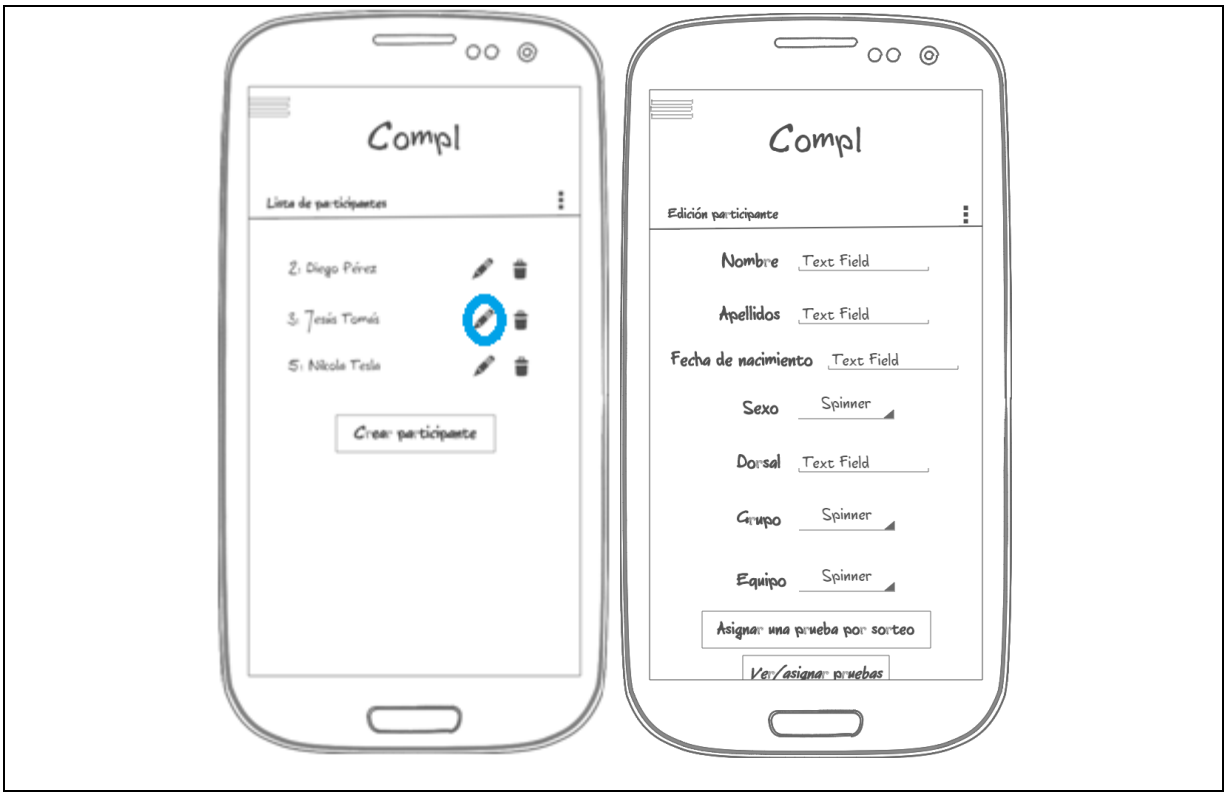
Mockups asociados


Nombre	Eliminar un participante
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario selecciona el participante a eliminar de la lista de participantes. 2- El usuario selecciona "eliminar". 3- El sistema pide confirmación sobre la eliminación. 4- El usuario acepta eliminar al participante. 5- El sistema elimina al participante junto con todos sus registros.	
Post-condiciones	POSCOND1 (el participante seleccionado y sus registros no existen en la competición).

Escenarios alternativos
4- El usuario cancela la petición. 5- El sistema no modifica nada y devuelve a usuario a una vista anterior.
Diagramas de secuencia
Mockups asociados


Nombre	Modificar datos de un participante
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	

1- El usuario selecciona un participante de la lista de participantes. 2- El usuario selecciona "modificar participante". 3- El usuario modifica los campos que necesite. (Establecer pruebas). 4- El usuario guarda la modificación. 5- El sistema guarda los nuevos datos del participante.	
Post-condiciones	POSCOND1 (el participante seleccionado y sus registros no existen en la competición).
Escenarios alternativos	
4- El usuario cancela la petición. 5- El sistema no modifica nada y devuelve a usuario a una vista anterior.	
4- El sistema detecta que ya existe un participante con el mismo dorsal y avisa al usuario. 5- El usuario cambia el dorsal por otro o cancela la creación.	
4- El sistema detecta que el equipo asignado no pertenece al mismo grupo que el establecido para el participante. 5- El sistema avisa al usuario y cambia automáticamente el grupo asignado al participante al mismo que el equipo asignado.	
5- El sistema detecta que existen datos obligatorios sin rellenar. 6- El sistema informa al usuario de dicho error y cancela la operación.	
Diagramas de secuencia	
Mockups asociados	

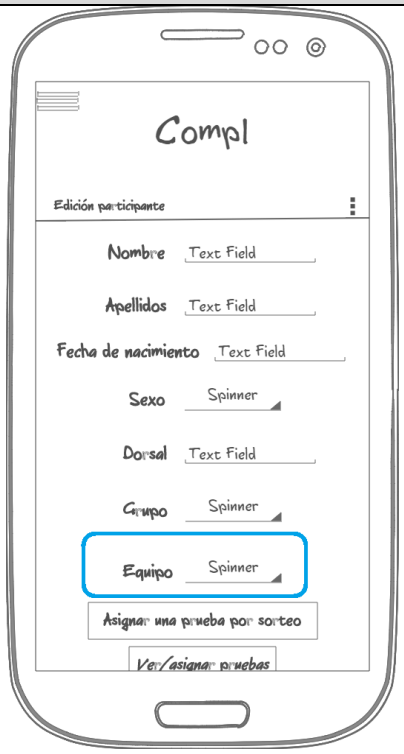


Nombre	Ver datos de un participante
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor/invitado con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario selecciona un participante de la lista de participantes. 2- El sistema muestra al usuario información sobre el participante y las pruebas asignadas.	
Post-condiciones	
Escenarios alternativos	
4- El usuario cancela la petición. 5- El sistema no modifica nada y devuelve a usuario a una vista anterior.	
Diagramas de secuencia	

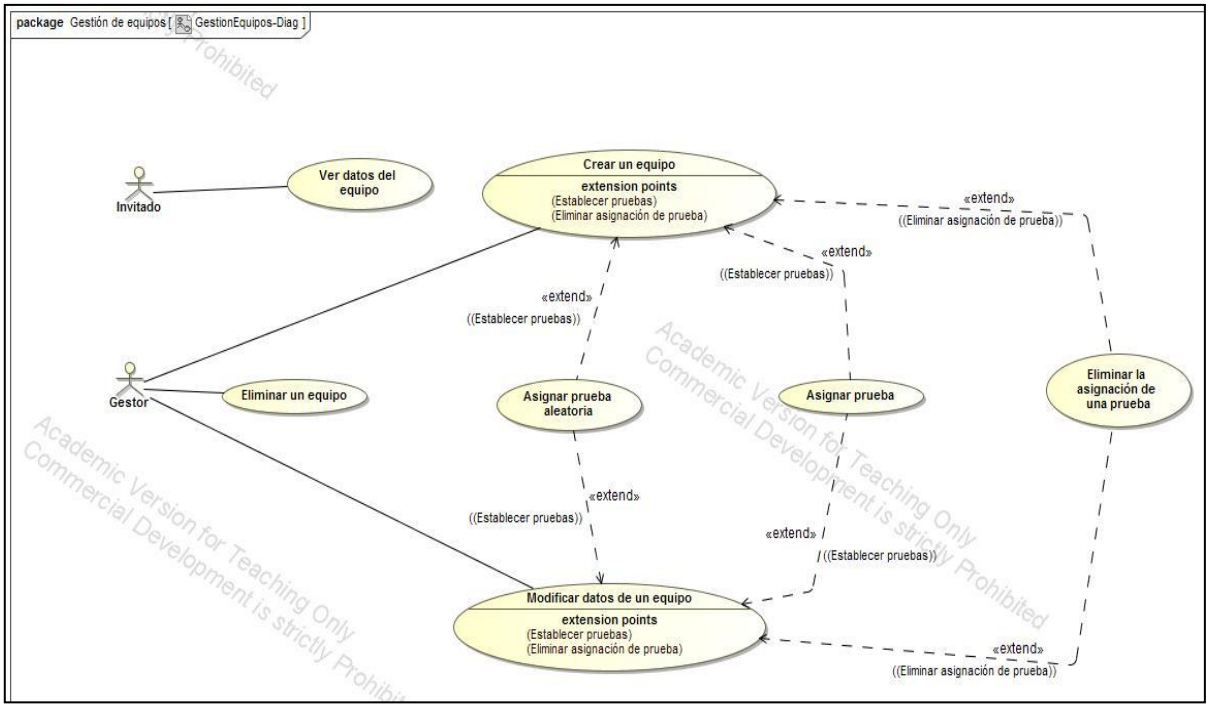
Mockups asociados


Nombre	Asignar equipo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (Existe algún grupo creado en la competición). PRECOND4 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El sistema obtiene el grupo seleccionado para el participante. 2- El sistema muestra una lista de equipos disponibles que pertenecen a ese grupo. 3- El usuario señala el equipo al que desea inscribir el participante. 4- El sistema guarda la petición.	

Post-condiciones	
Escenarios alternativos	
Diagramas de secuencia	
Mockups asociados	



5. Gestión de equipos



Nombre	Crear un equipo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (Existe algún grupo creado en la competición). PRECOND4 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario selecciona la opción de crear un equipo. 2- El sistema muestra una vista con los campos a rellenar. 3- El usuario introduce los datos del nuevo equipo junto con el grupo al que pertenece. (Establecer pruebas). 4- El usuario acepta la creación. 5- El sistema crea el nuevo equipo.	

Post-condiciones	POSCOND1 (el equipo existe en la competición)

Escenarios alternativos

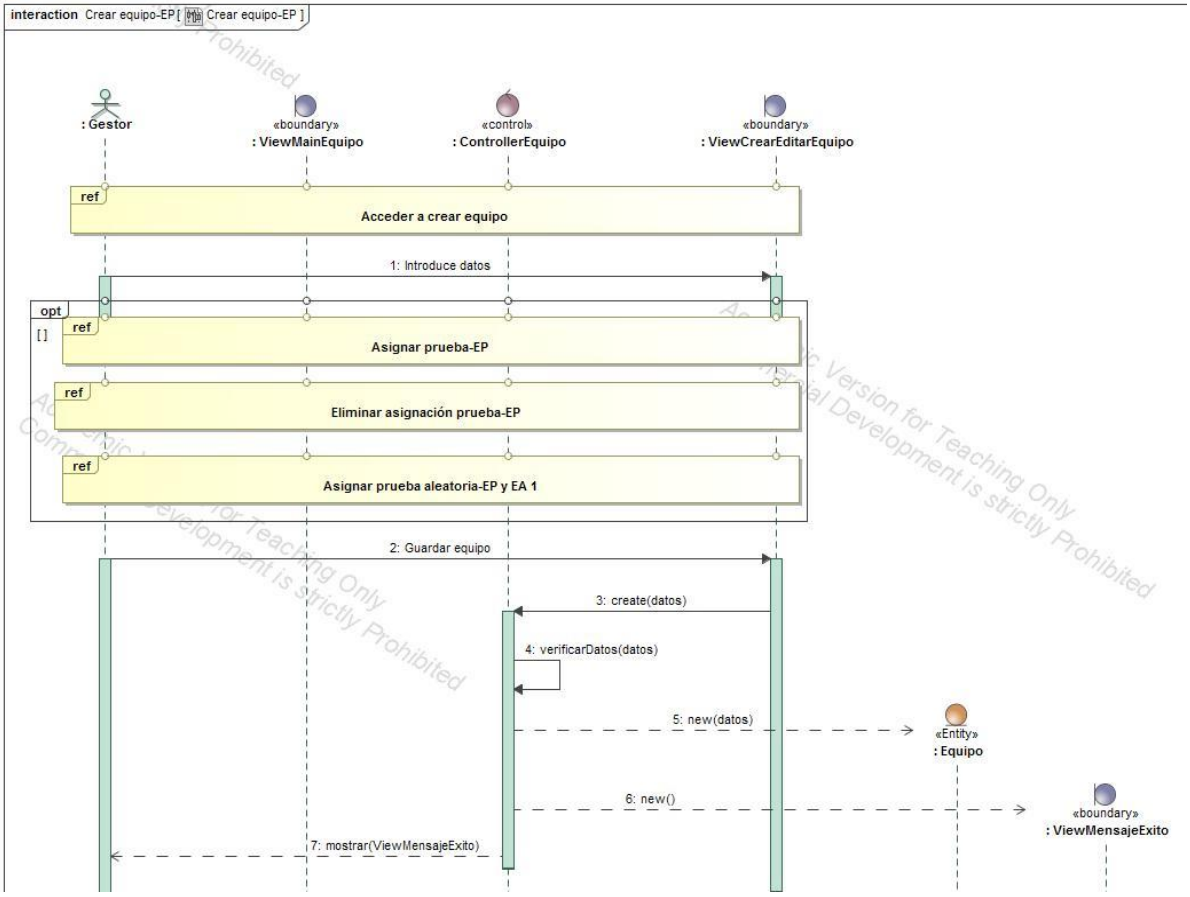
4- El usuario cancela la petición.
5- El sistema no modifica nada y devuelve a usuario a una vista anterior.

3- El usuario cancela la creación.
4- El sistema no modifica nada.

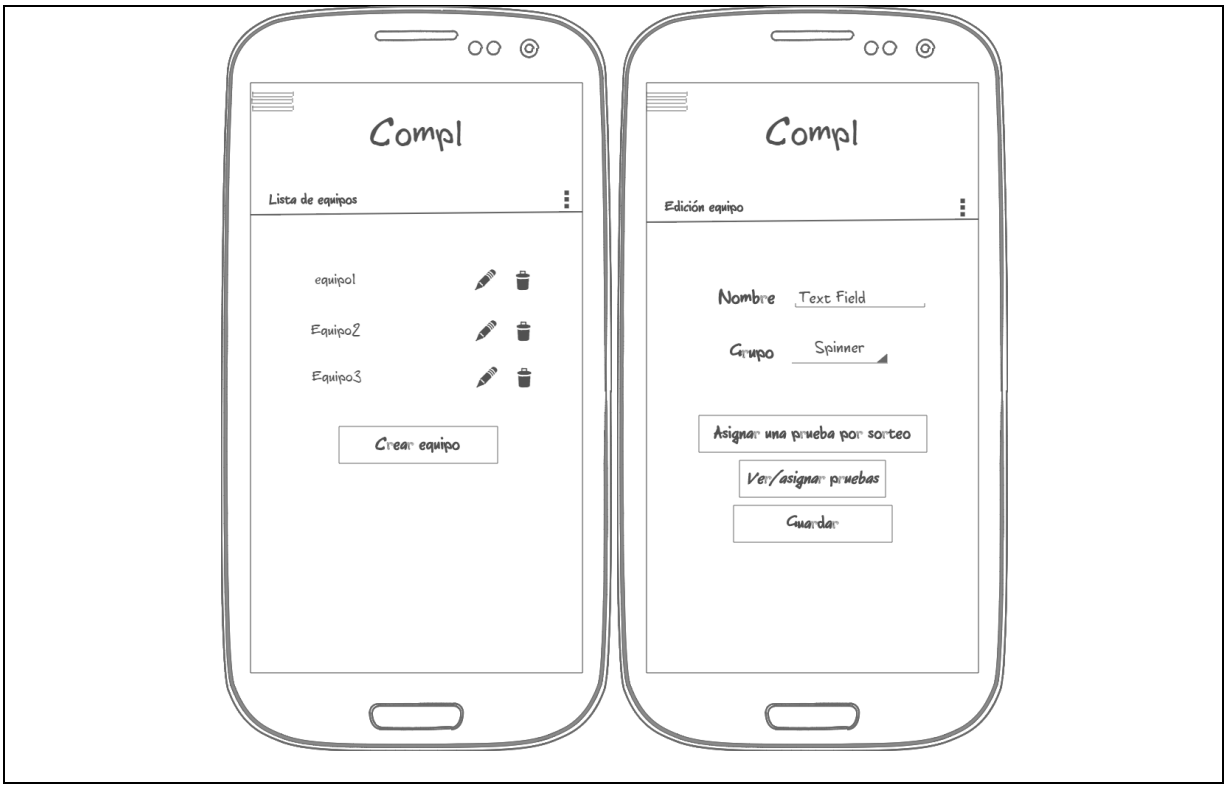
4- El sistema detecta que ya existe un equipo con el mismo nombre y avisa al usuario.
5- El usuario cambia el nombre por otro o cancela la creación.

3- El sistema detecta que existen datos obligatorios sin rellenar.
4- El sistema informa al usuario de dicho error y cancela la operación.

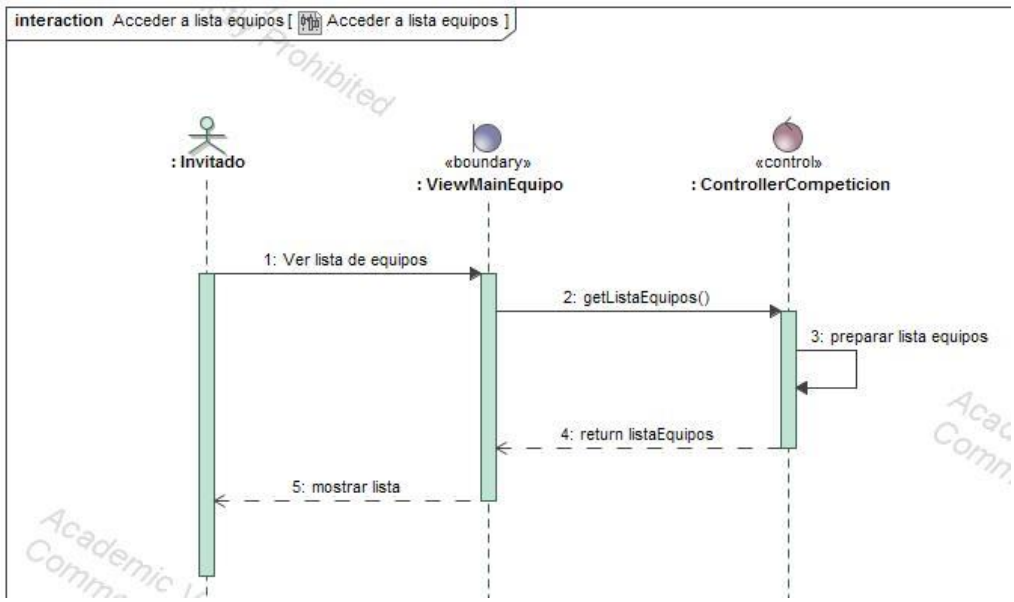
Diagramas de secuencia

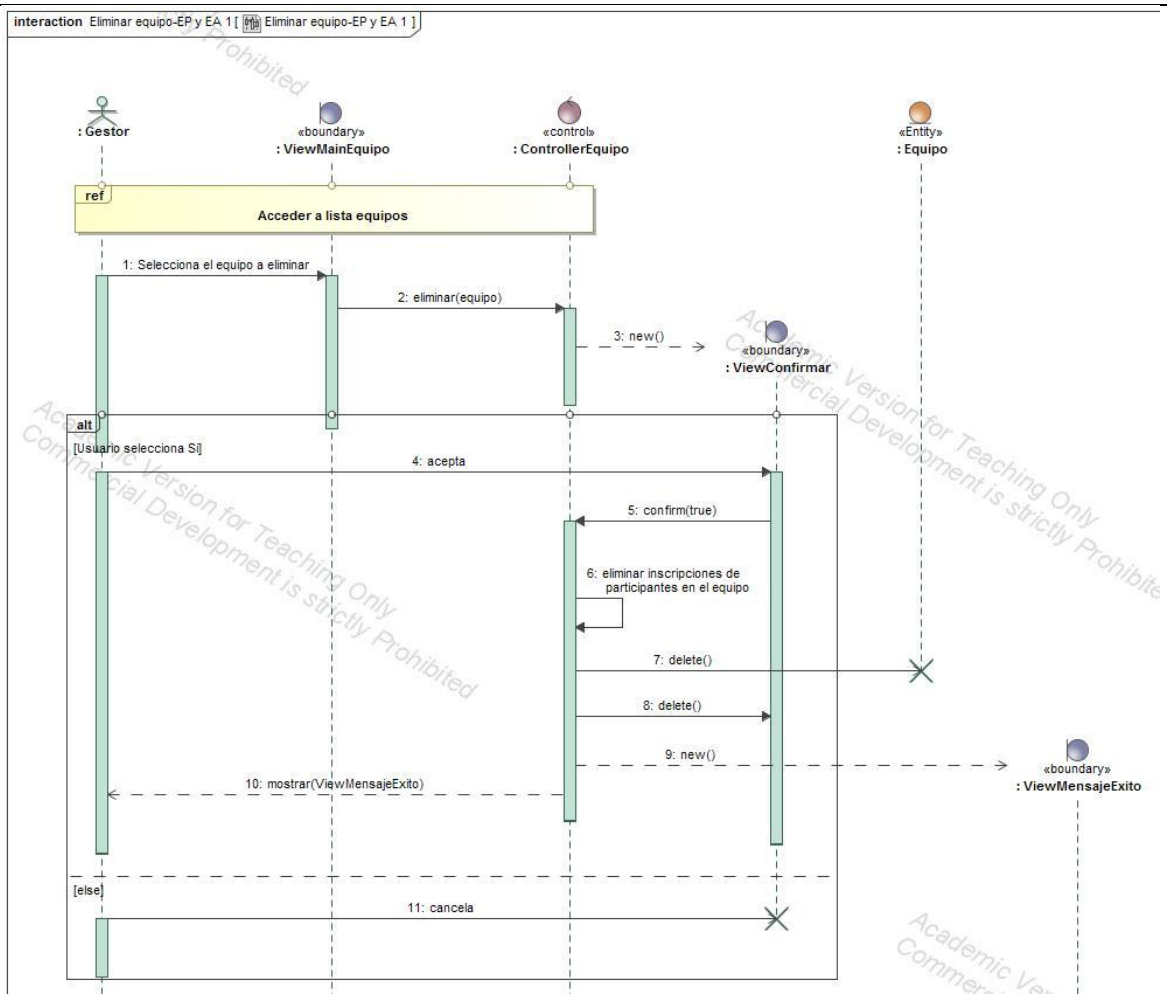


Mockups asociados

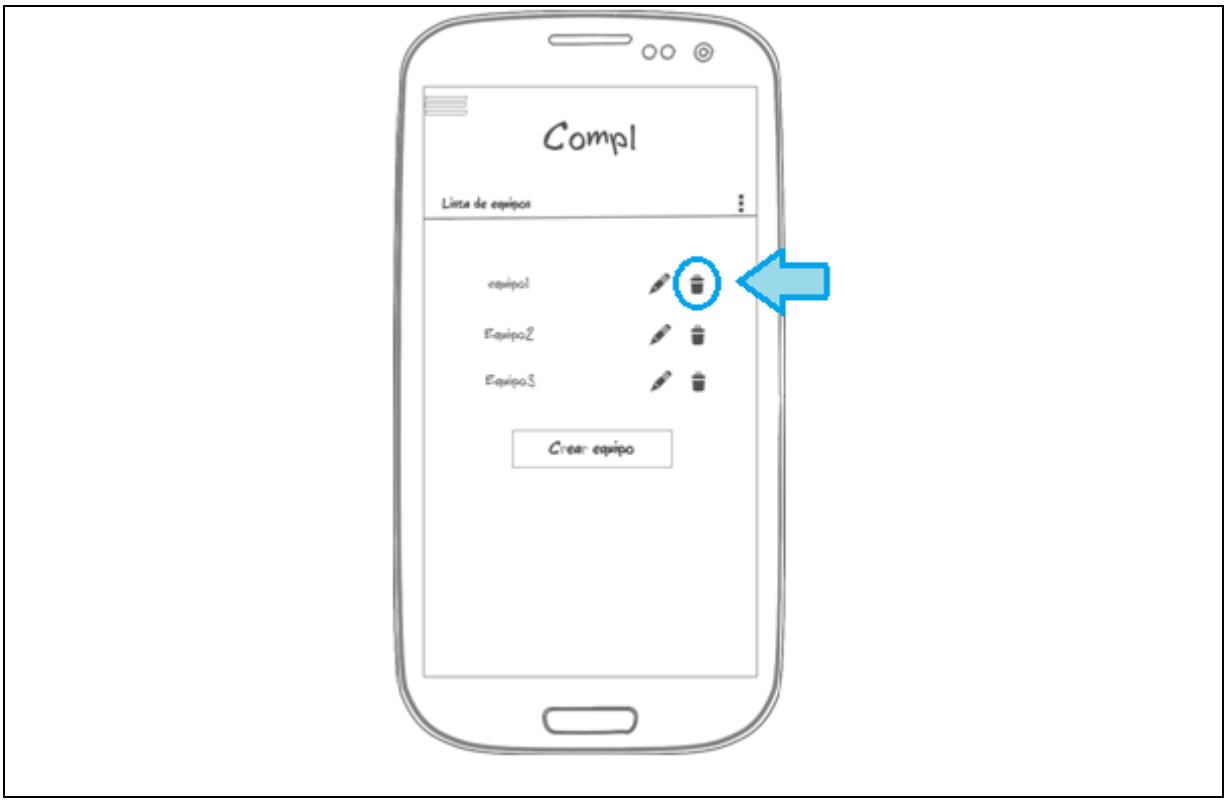


Nombre	Eliminar un equipo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (Existe algún grupo creado en la competición). PRECOND4 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario selecciona el equipo a eliminar de la lista de equipos a los que tiene acceso. 2- El usuario selecciona la opción que permite eliminarlo. 3- El sistema pide confirmación sobre la eliminación. 4- El usuario acepta eliminar. 5- El sistema elimina al equipo junto con todos sus registros.	
Post-condiciones	POSCOND1 (el equipo seleccionado no existe en la competición).

Escenarios alternativos
4- El usuario cancela la petición. 5- El sistema no modifica nada.
Diagramas de secuencia


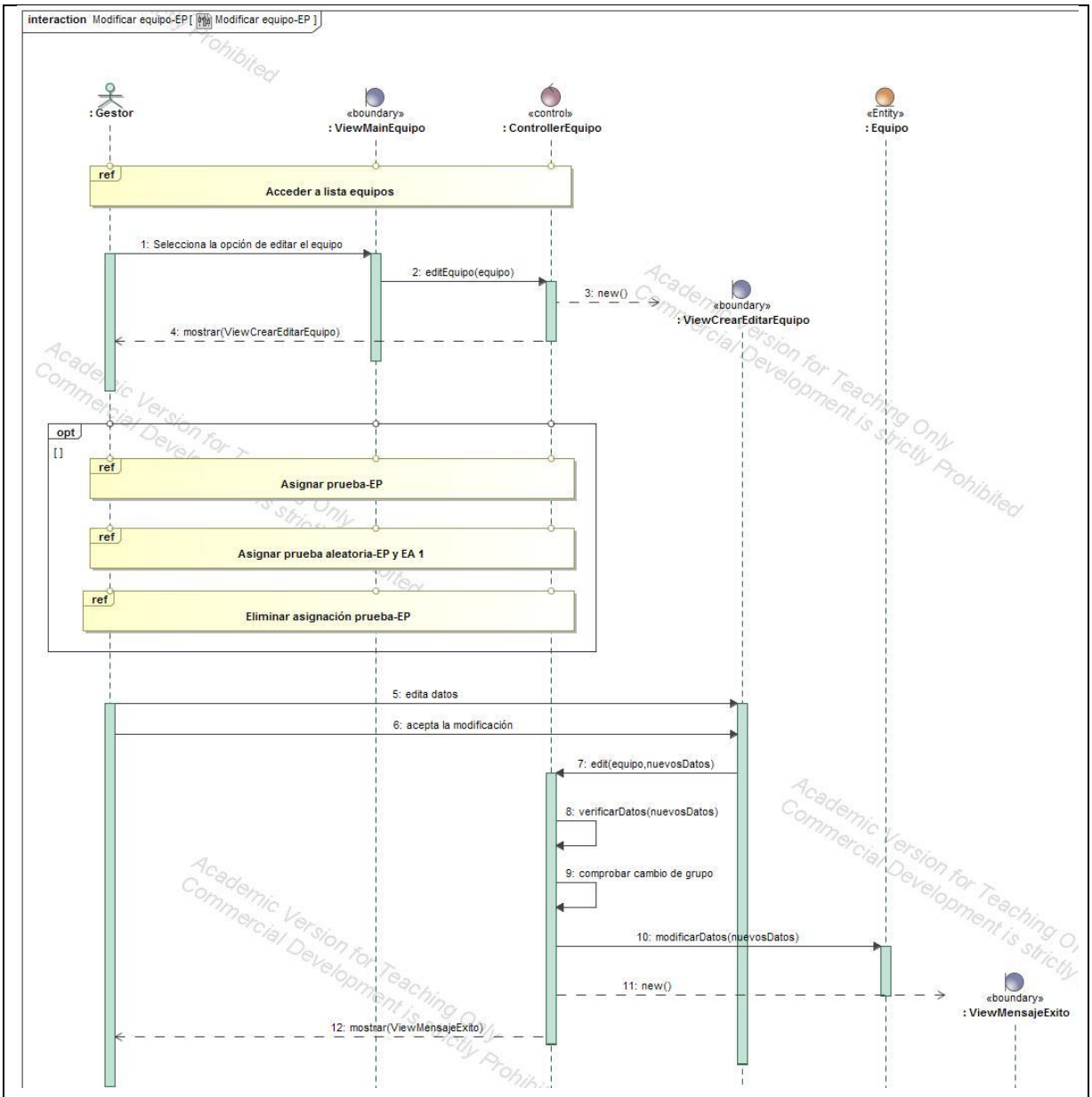


Mockups asociados

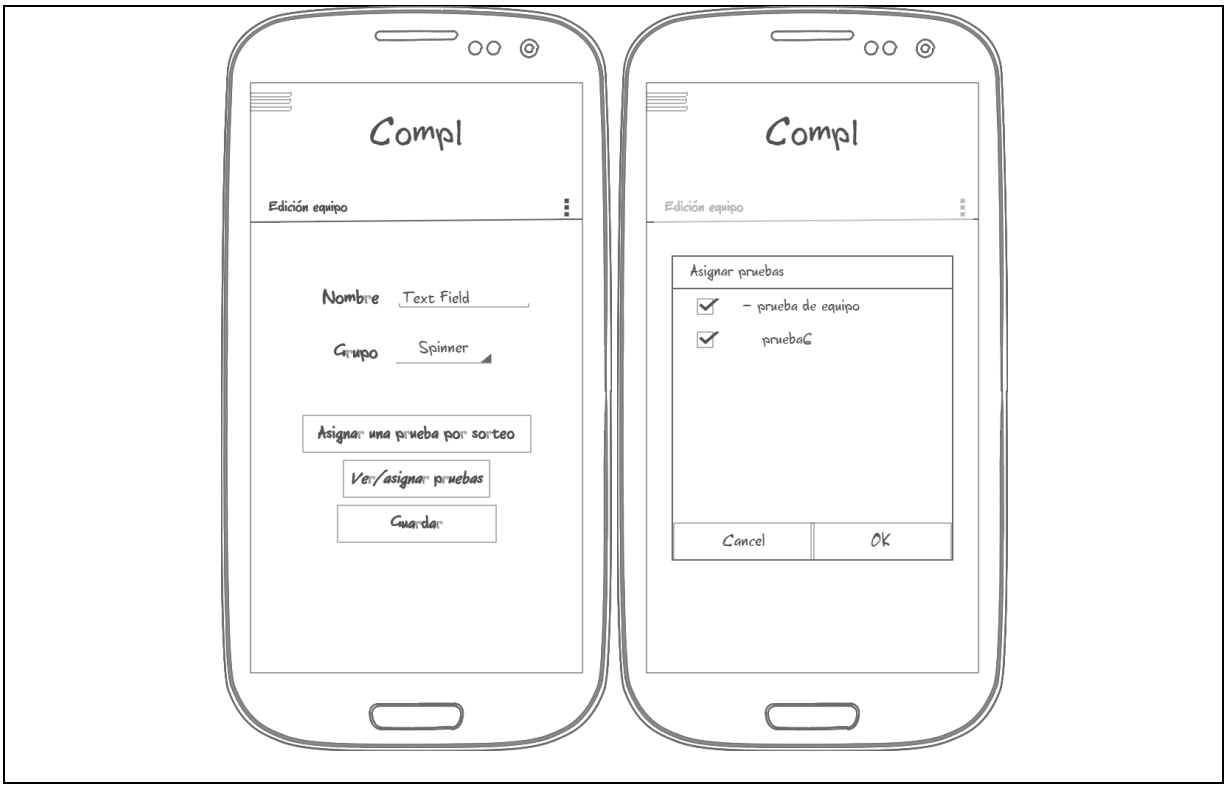


Nombre	Modificar datos de un equipo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (Existe algún grupo creado en la competición). PRECOND4 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario selecciona un equipo de la lista de equipos. 2- El usuario selecciona la opción que permite modificar el equipo. 3- El usuario modifica los campos que necesite o el grupo al que pertenece. (Establecer pruebas). 4- El usuario guarda la modificación. 5- El sistema guarda los nuevos datos del equipo.	
Post-condiciones	POSCOND1 (el equipo está actualizado en la competición).

Escenarios alternativos
4- El usuario cancela la petición. 5- El sistema no modifica nada.
4- El sistema detecta que ya existe un equipo con el mismo nombre y avisa al usuario. 5- El usuario cambia el nombre por otro o cancela la creación.
3- El usuario modifica el grupo al que pertenece. (Establecer pruebas). 4- El sistema detecta que existen participantes inscritos en el equipo. 5- El sistema informa al usuario de la imposibilidad de cambiar de grupo mientras haya participantes inscritos a él.
Diagramas de secuencia

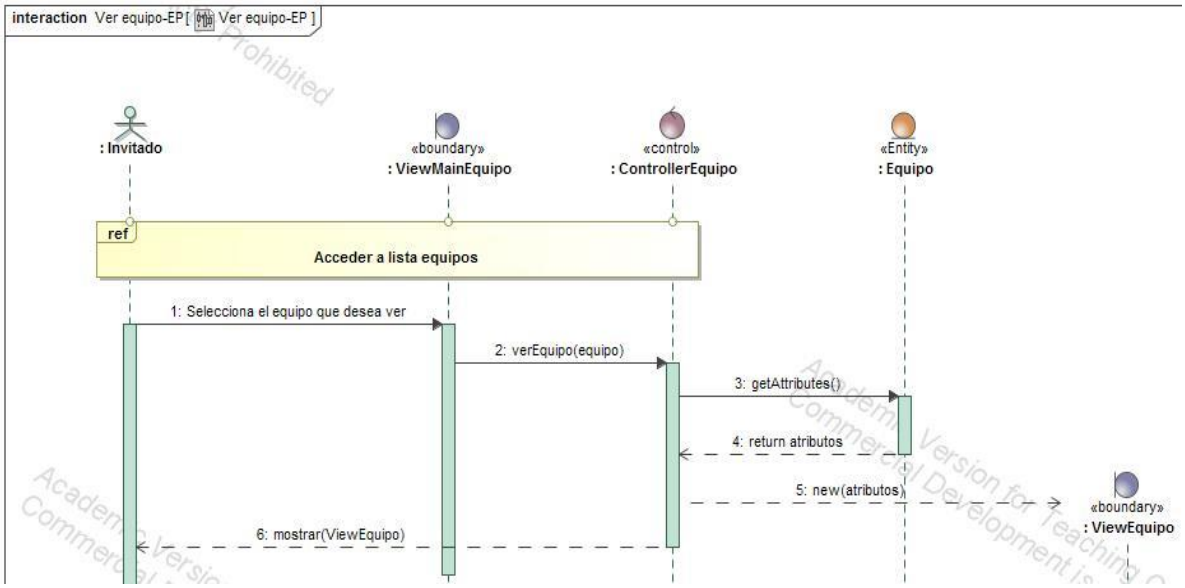


Mockups asociados

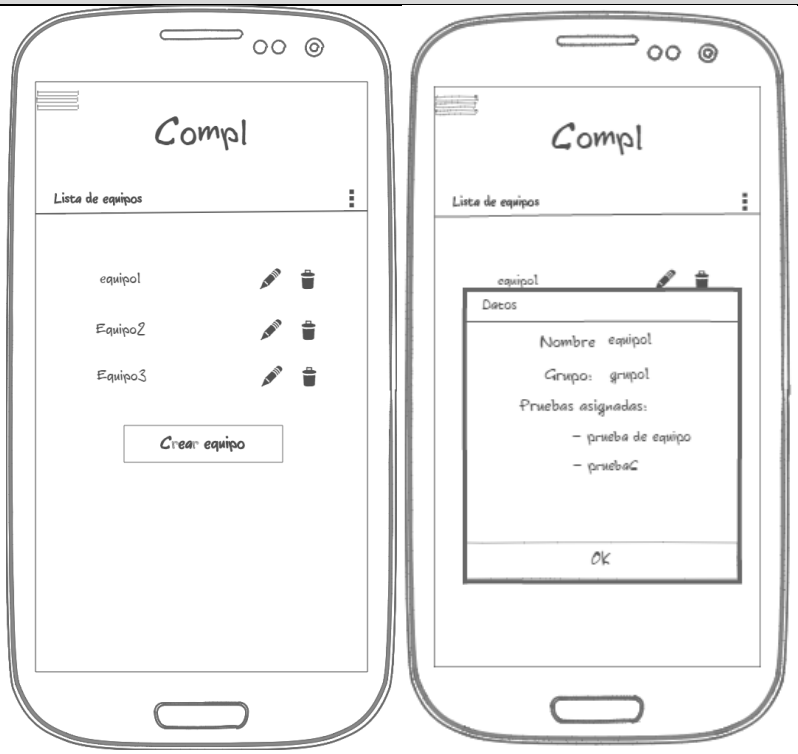


Nombre	Ver datos de un equipo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor/invitado con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe algún equipo en la competición).
Escenario principal	
1- El sistema muestra una lista de equipos. 2- El usuario selecciona un equipo de la lista de equipos. 3- El sistema muestra al usuario información sobre el equipo.	
Post-condiciones	
Escenarios alternativos	

Diagramas de secuencia

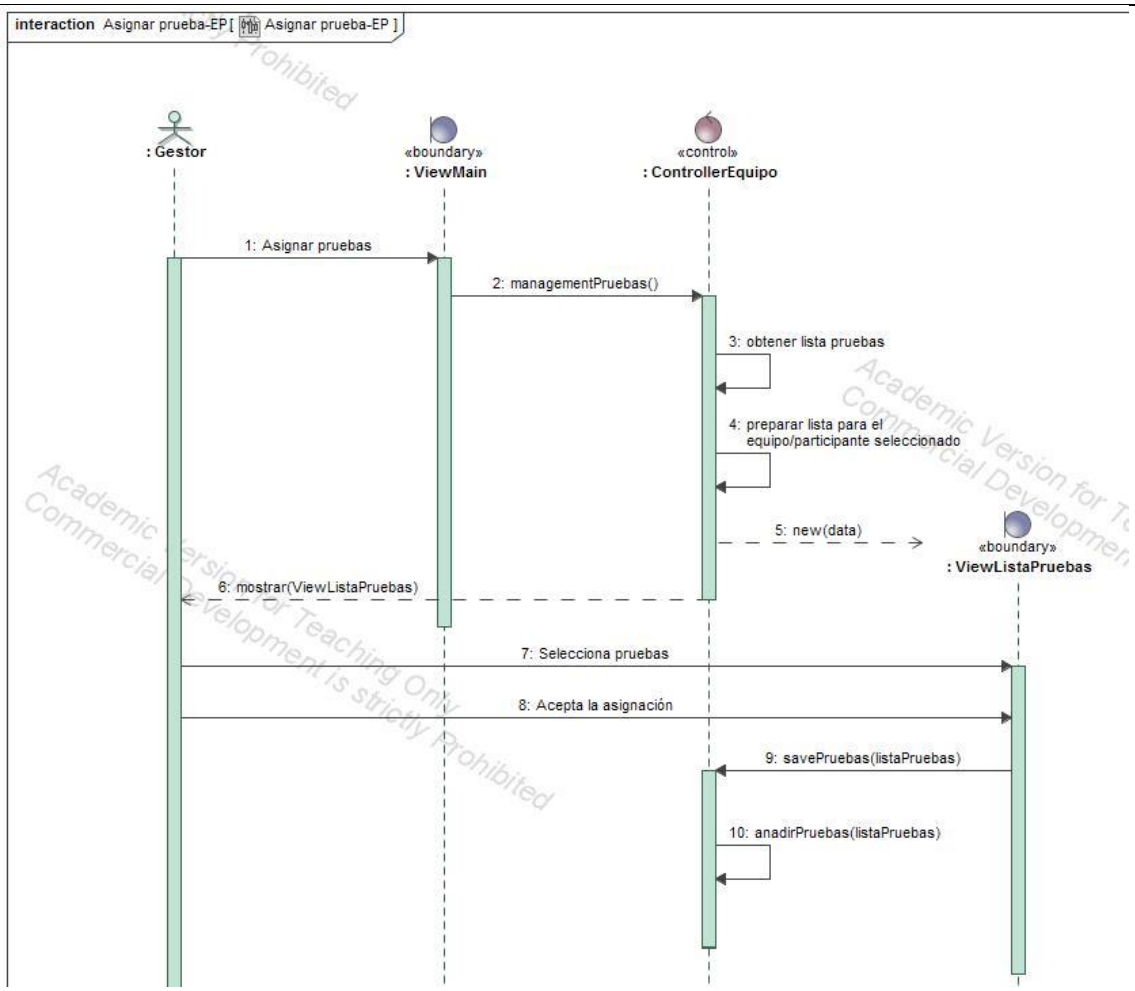


Mockups asociados



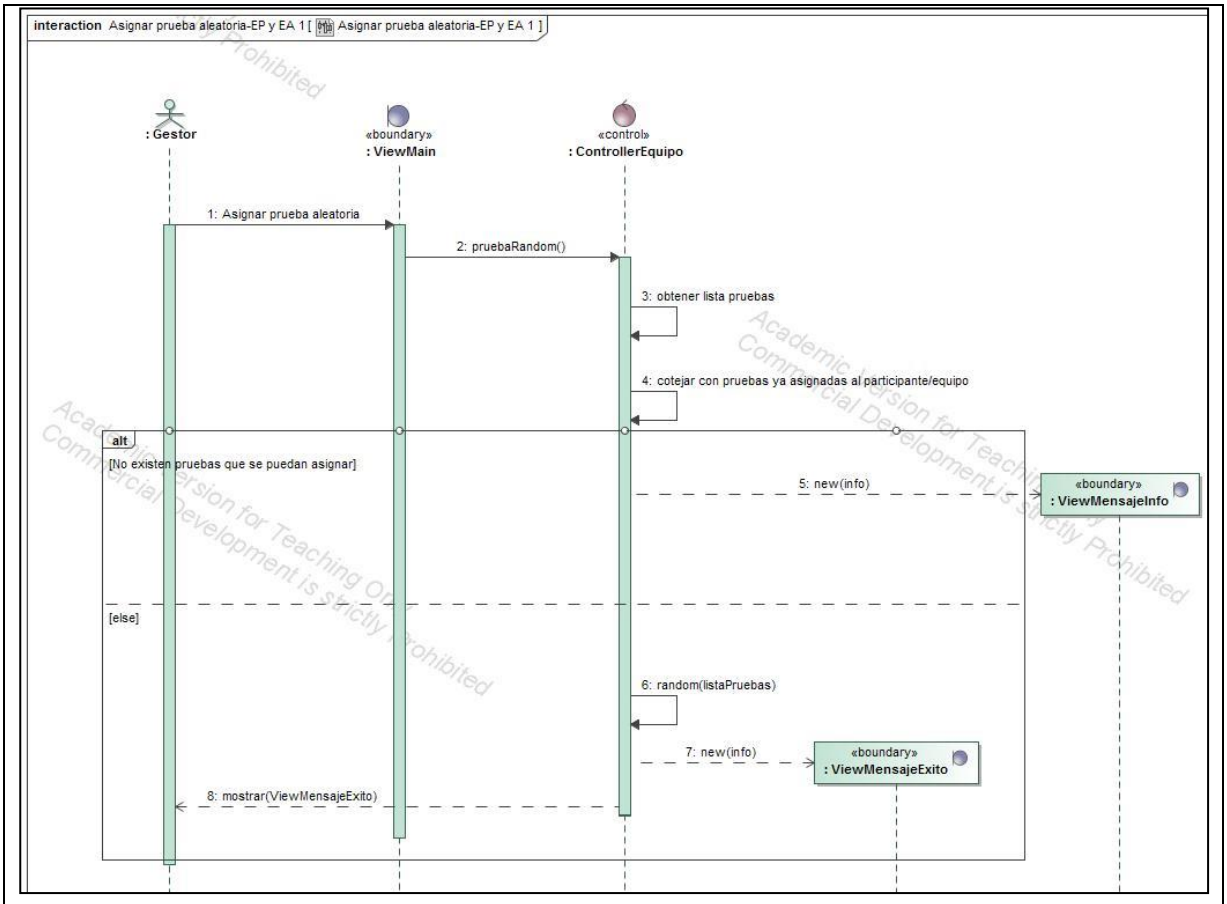
6. Casos de uso comunes a equipos y participantes

Nombre	Asignar prueba
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (Se está editando o creando un participante o equipo).
Escenario principal	
1- El usuario selecciona la opción de asignar pruebas. 2- El sistema muestra una vista con las pruebas que se pueden asignar. 3- El usuario selecciona la/s prueba/s que desea asignar al competidor/equipo. 4- El usuario acepta la/s asignación/es. Y 5- El sistema guarda la nueva configuración para el equipo/participante.	
Post-condiciones	
Escenarios alternativos	
4- El usuario cancela la operación. 5- El sistema devuelve al usuario a un estado anterior.	
2- El sistema detecta que no existen pruebas que se puedan asignar al participante/equipo. 3- El sistema informa al usuario de la situación.	
Diagramas de secuencia	



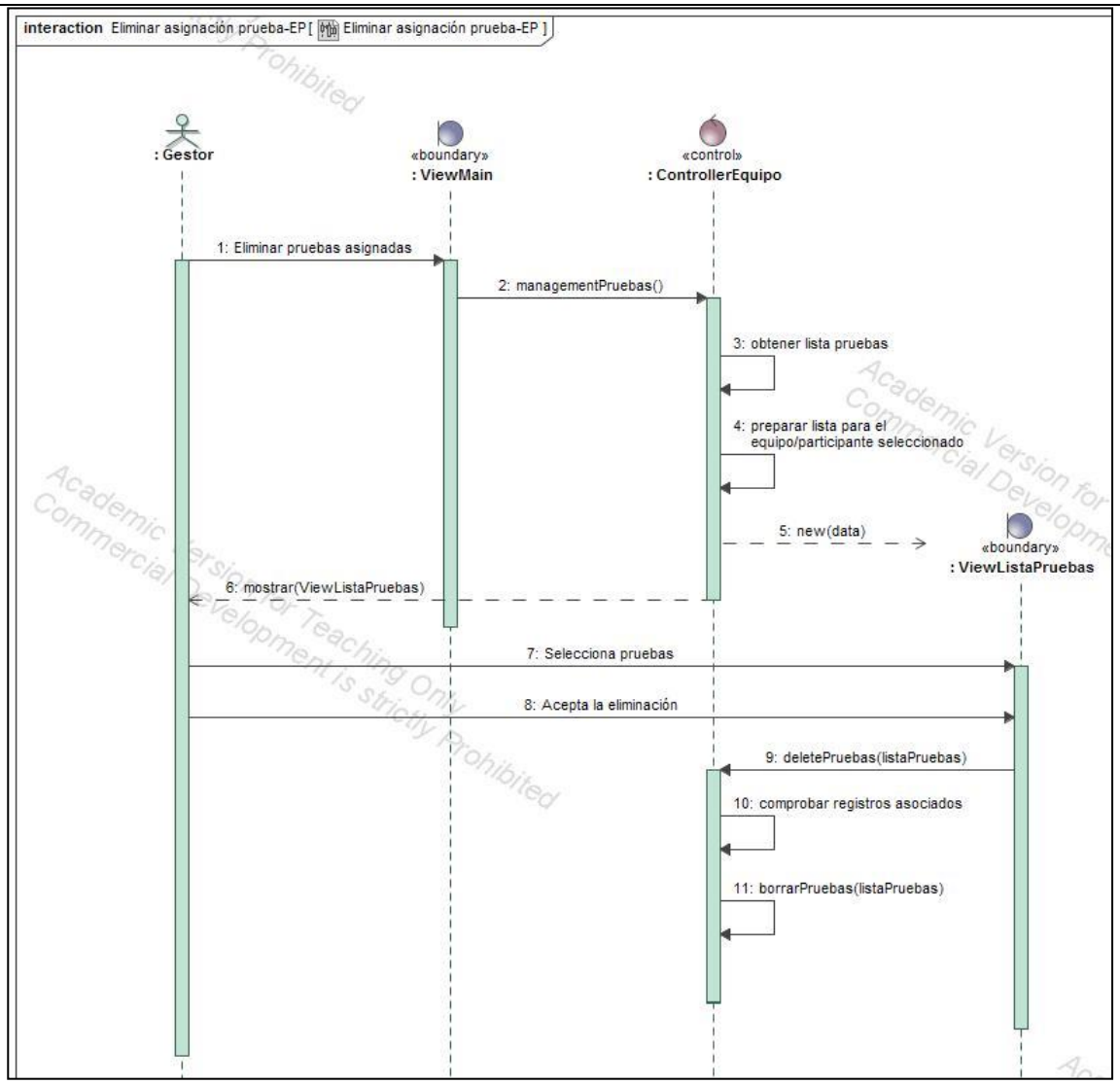
Mockups asociados

Nombre	Asignar prueba aleatoria
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (Se está editando o creando un participante o equipo).
Escenario principal	
1- El usuario selecciona la opción de asignar una prueba aleatoria. 2- El sistema selecciona una prueba aleatoria entre las disponibles. 3- El sistema informa al usuario de la prueba aleatoria que se ha escogido.	
Post-condiciones	
Escenarios alternativos	
2- El sistema detecta que no existen pruebas que se puedan asignar al participante/equipo. 3- El sistema informa al usuario de la situación.	
Diagramas de secuencia	

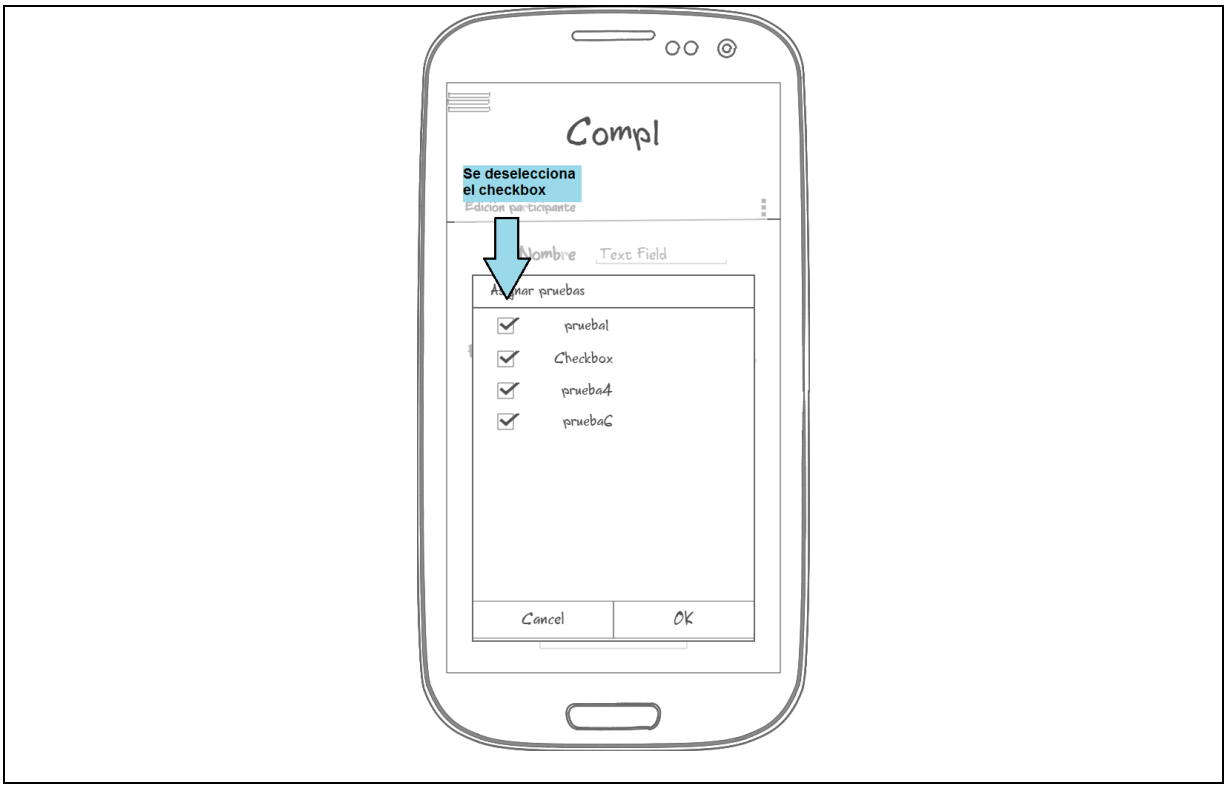


Mockups asociados

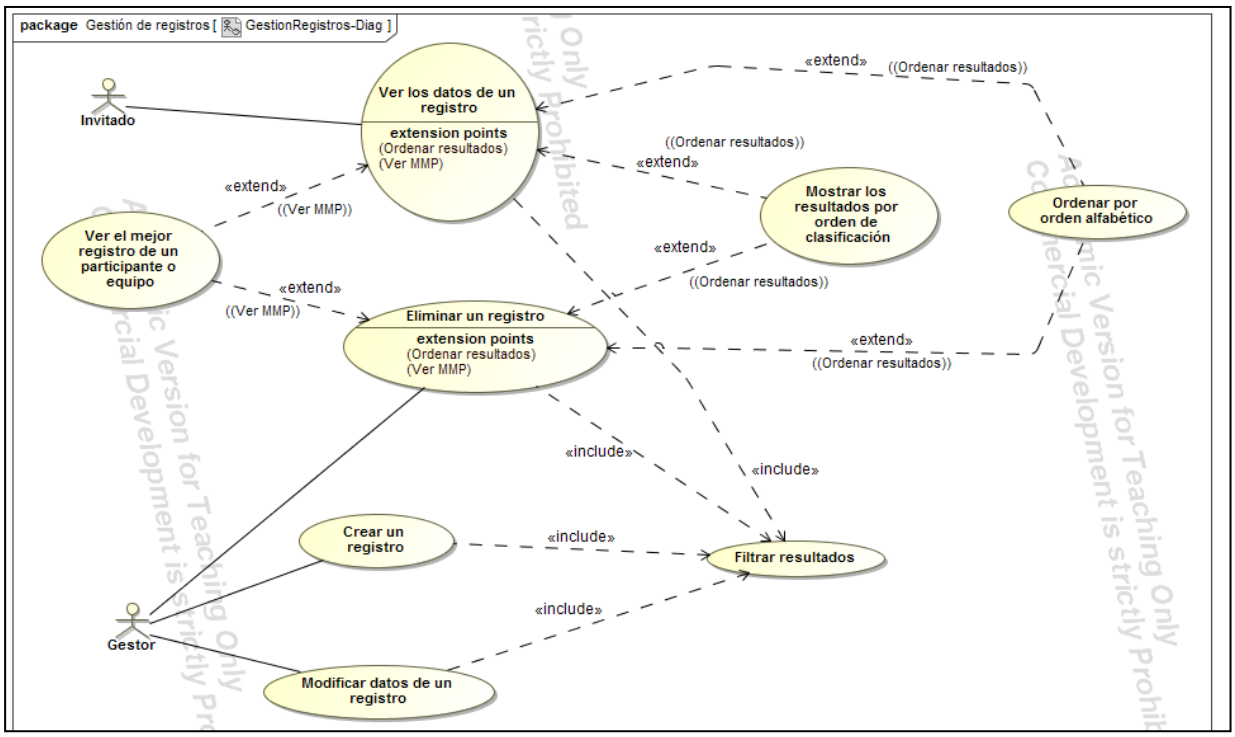
Nombre	Eliminar la asignación de una prueba
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (Se está editando o creando un participante o equipo).
Escenario principal	
1- El usuario selecciona la opción de eliminar una asignación de una prueba. 2- El sistema muestra una vista con la lista de pruebas asignadas. 3- El usuario selecciona la/s prueba/s que desea eliminar. 4- El usuario acepta los cambios. 5- El sistema guarda la nueva configuración para el equipo/participante.	
Post-condiciones	
Escenarios alternativos	
4- El usuario cancela la operación. 5- El sistema devuelve al usuario a un estado anterior.	
5- El sistema detecta que existen registros asociados a alguna de las pruebas de las que se desea eliminar su asignación. 6- El sistema informa al usuario del error.	
Diagramas de secuencia	



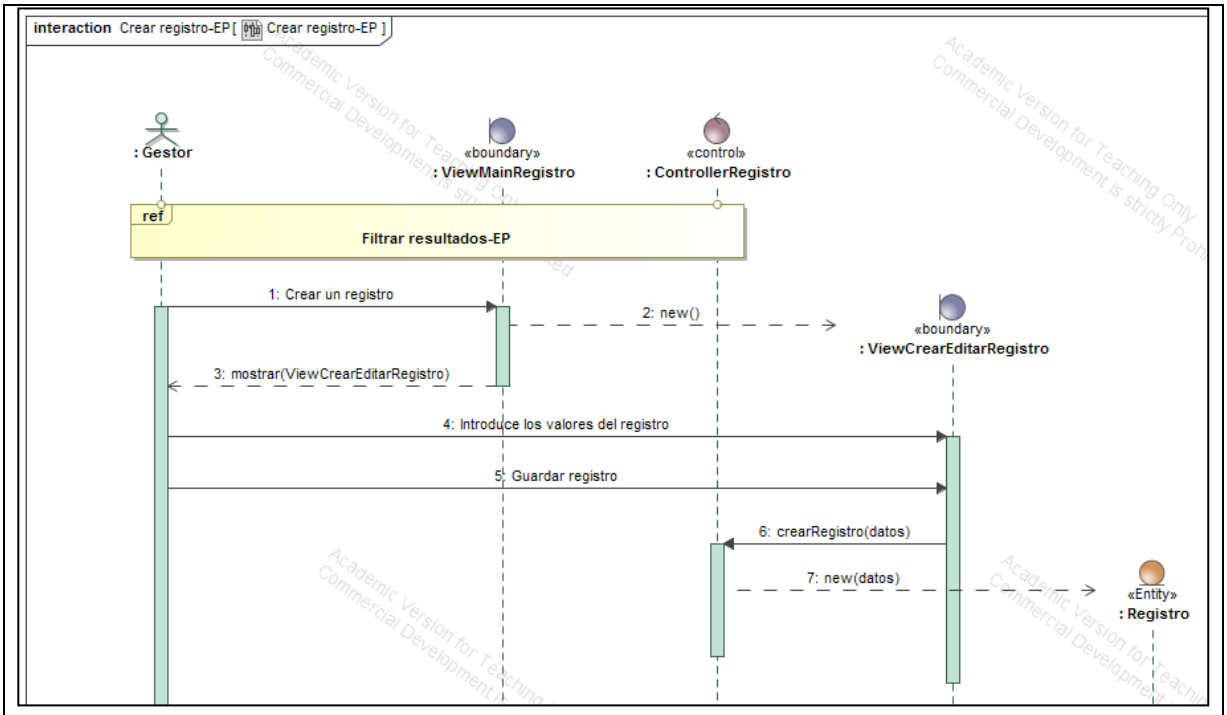
Mockups asociados



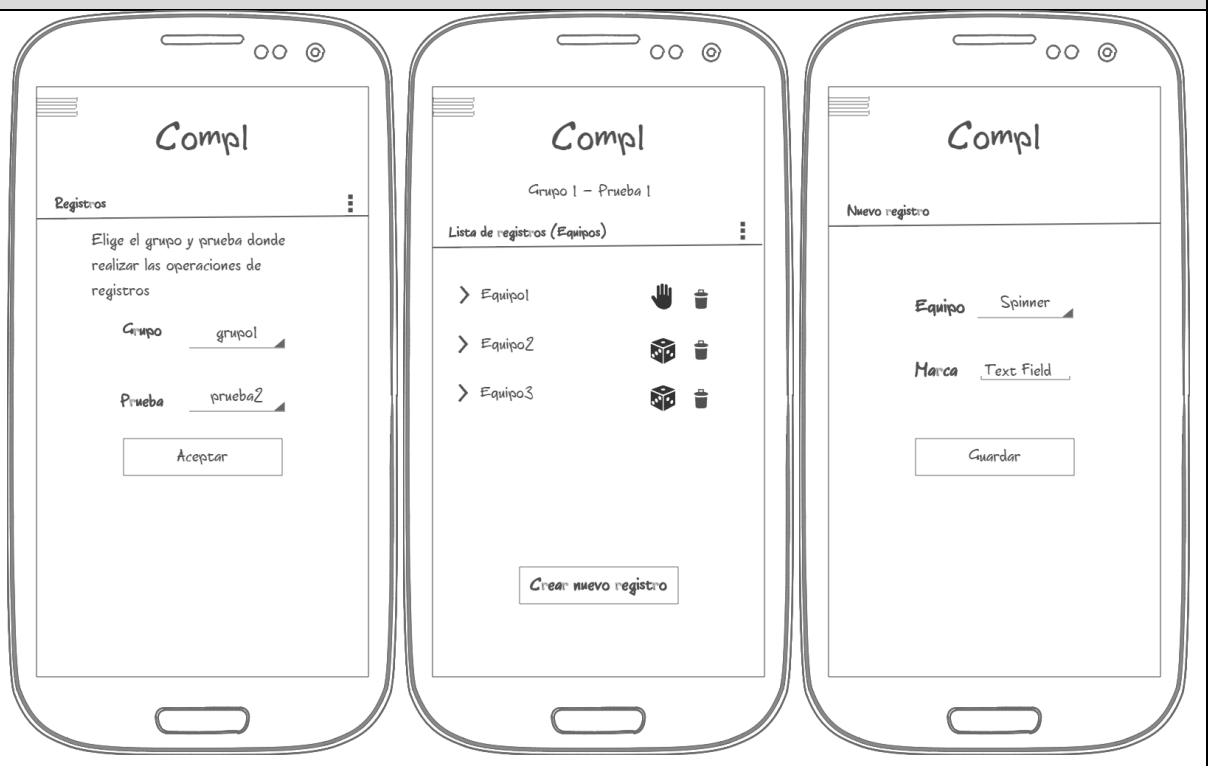
7. Gestión de registros



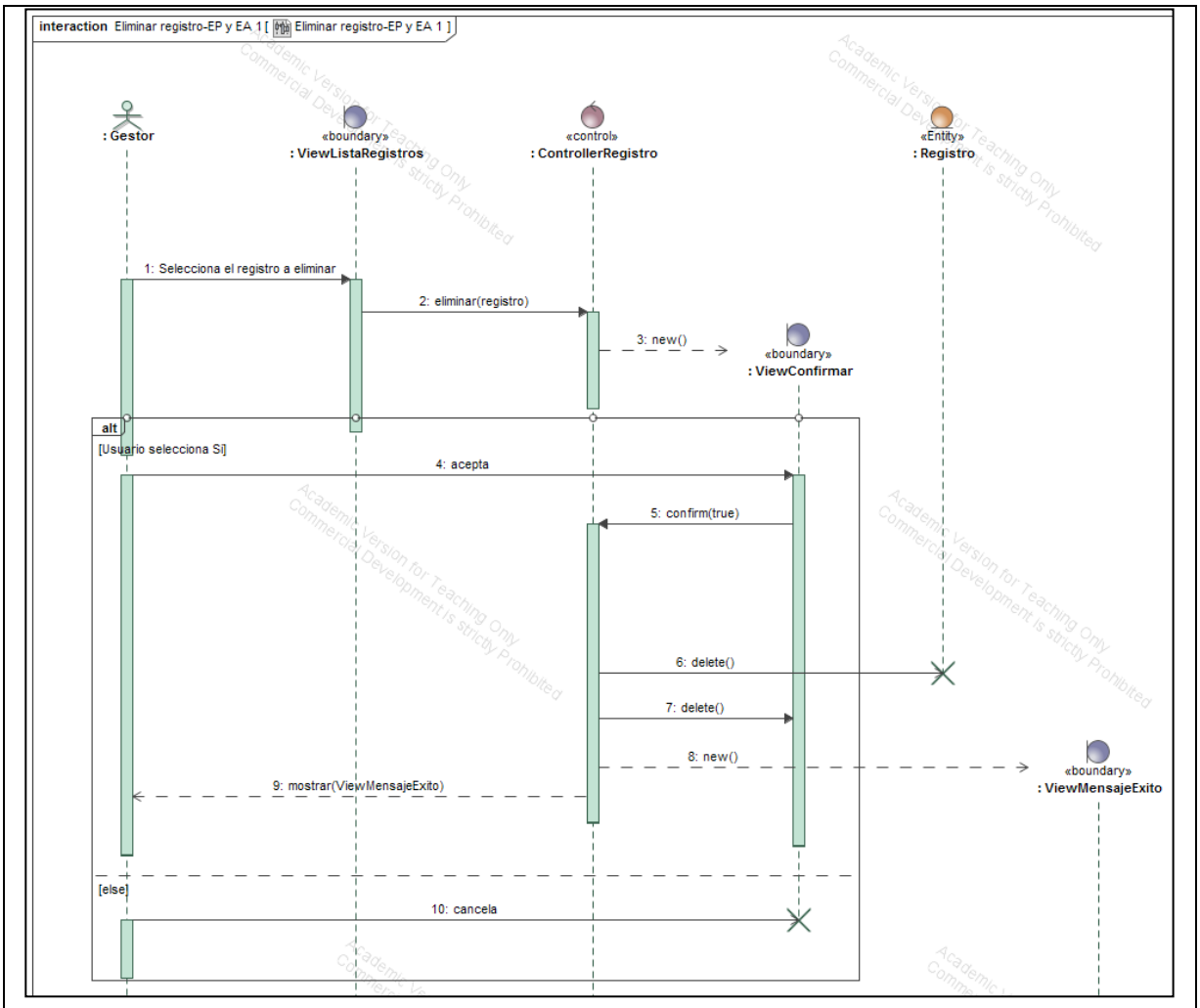
Nombre	Crear un registro
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe alguna prueba asignada a un participante o equipo).
Escenario principal	
incluir(Filtrar resultados) 1- El usuario selecciona la opción que permite crear un nuevo registro. 2- El sistema muestra una vista con la lista de participantes/equipos filtrados y los campos necesarios a rellenar. 3- El usuario selecciona el equipo/participante al que desea asignar la marca. 4- El usuario introduce el valor de la marca. 5- El usuario selecciona la opción de guardar el registro. 6- El sistema crea el nuevo registro.	
Post-condiciones	POSCOND1 (el registro existe en la competición)
Escenarios alternativos	
4- El usuario cancela la operación. 5- El sistema devuelve al usuario a un estado anterior.	
5- El sistema detecta que existen campos obligatorios por definir e informa al usuario del error.	
5- El sistema detecta que existen valores de campos incorrectos e informa al usuario del error.	
Diagramas de secuencia	



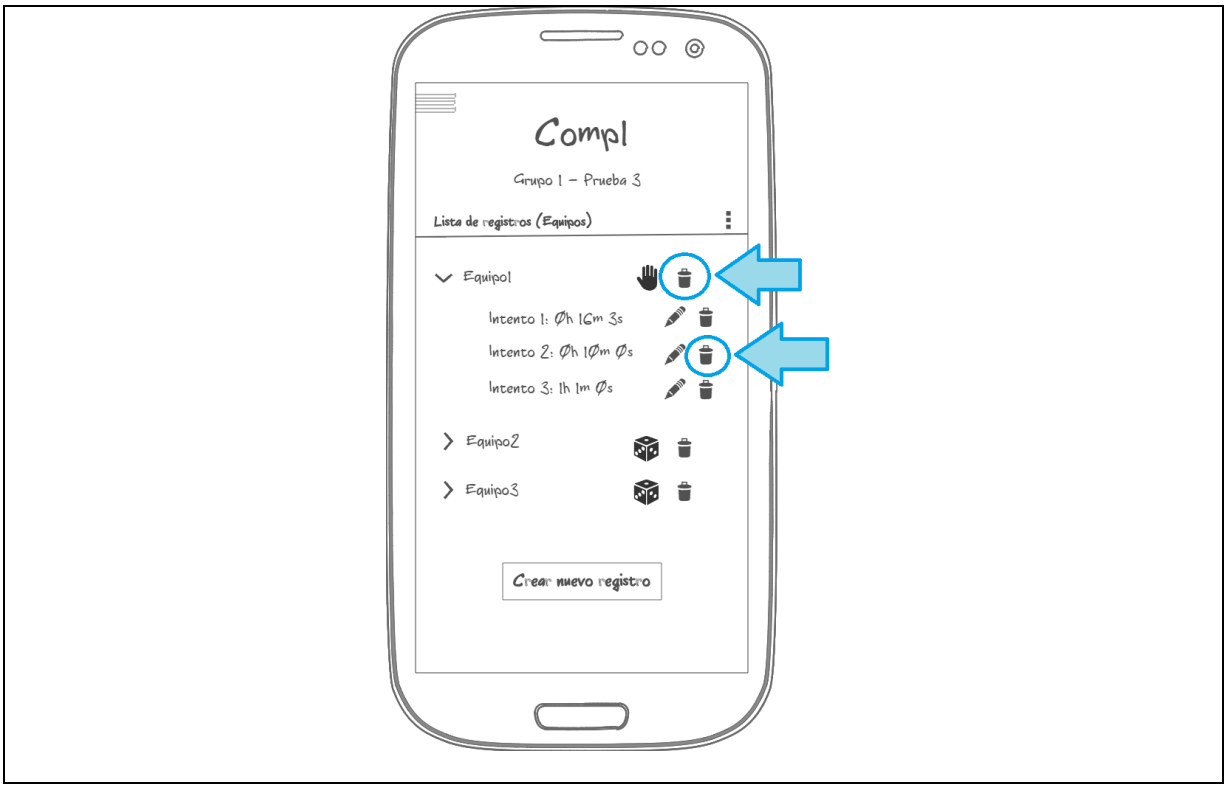
Mockups asociados



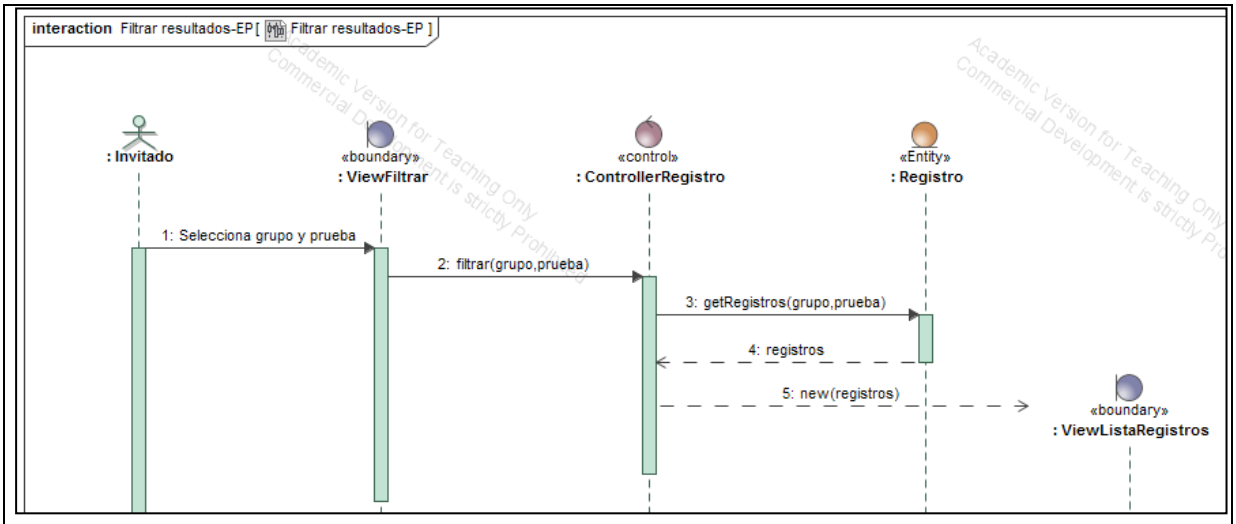
Nombre	Eliminar un registro
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
(Ordenar resultados). (Ver MMP). 1- El usuario selecciona el registro a eliminar de la lista de registros. 2- El usuario selecciona "eliminar". 3- El sistema pide confirmación sobre la eliminación. 4- El usuario acepta eliminar el registro. 5- El sistema elimina el registro.	
Post-condiciones	POSCOND1 (el registro no existe en la competición)
Escenarios alternativos	
4- El usuario cancela la operación. 5- El sistema no modifica nada.	
Diagramas de secuencia	



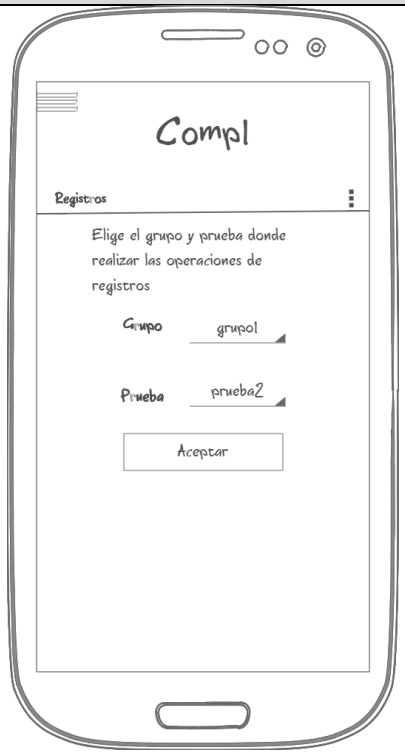
Mockups asociados



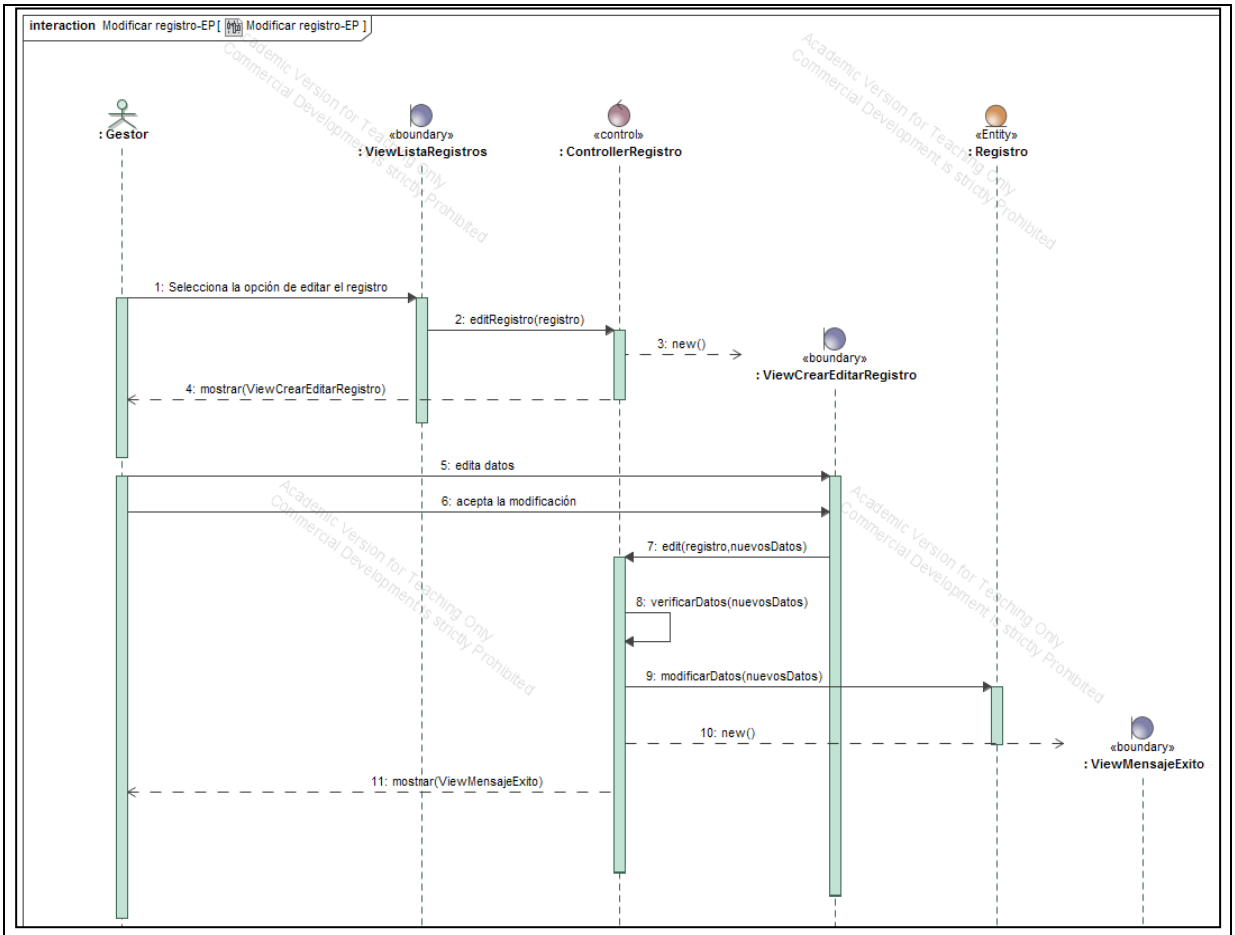
Nombre	Filtrar resultados
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El sistema muestra una vista para filtrar los registros. 2- El usuario selecciona la prueba y grupo que desea filtrar. 3- El usuario acepta la filtración. 4- El sistema muestra los registros filtrados.	
Post-condiciones	
Escenarios alternativos	
Diagramas de secuencia	



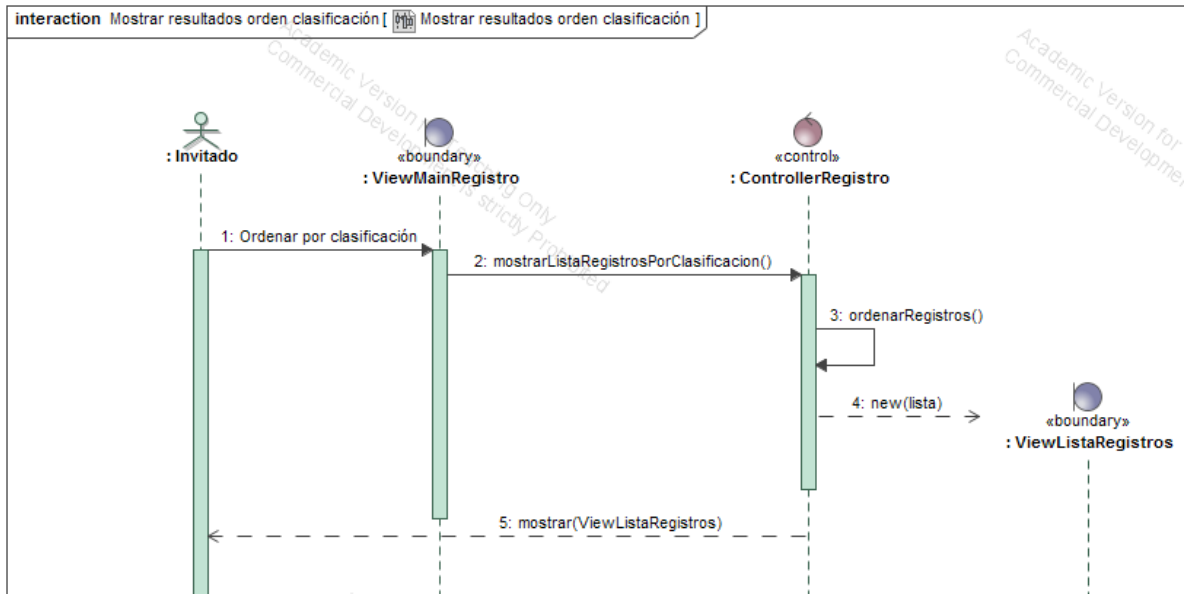
Mockups asociados

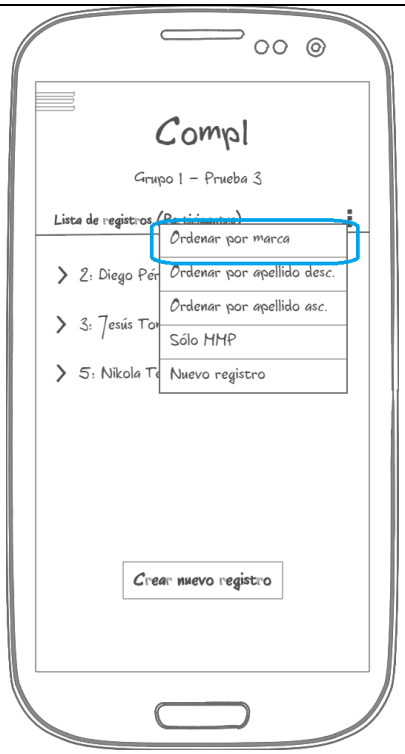


Nombre	Modificar datos de un registro
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario accede a la opción de gestionar registros. incluir(Filtrar resultados). 2- El usuario selecciona el registro a modificar. 3- El sistema muestra una vista con los datos del registro editables. 4- El usuario modifica los campos que necesite. 5- El usuario guarda la modificación. 6- El sistema guarda los nuevos datos del registro.	
Post-condiciones	POSCOND1 (el registro está actualizado)
Escenarios alternativos	
5- El usuario cancela la modificación. 6- El sistema no modifica nada.	
5- El sistema detecta que existen valores incorrectos en los campos editados. 6- El sistema informa al usuario del error.	
5- El sistema detecta que existen campos obligatorios sin rellenar. 6- El sistema informa al usuario del error.	
Diagramas de secuencia	



Mockups asociados

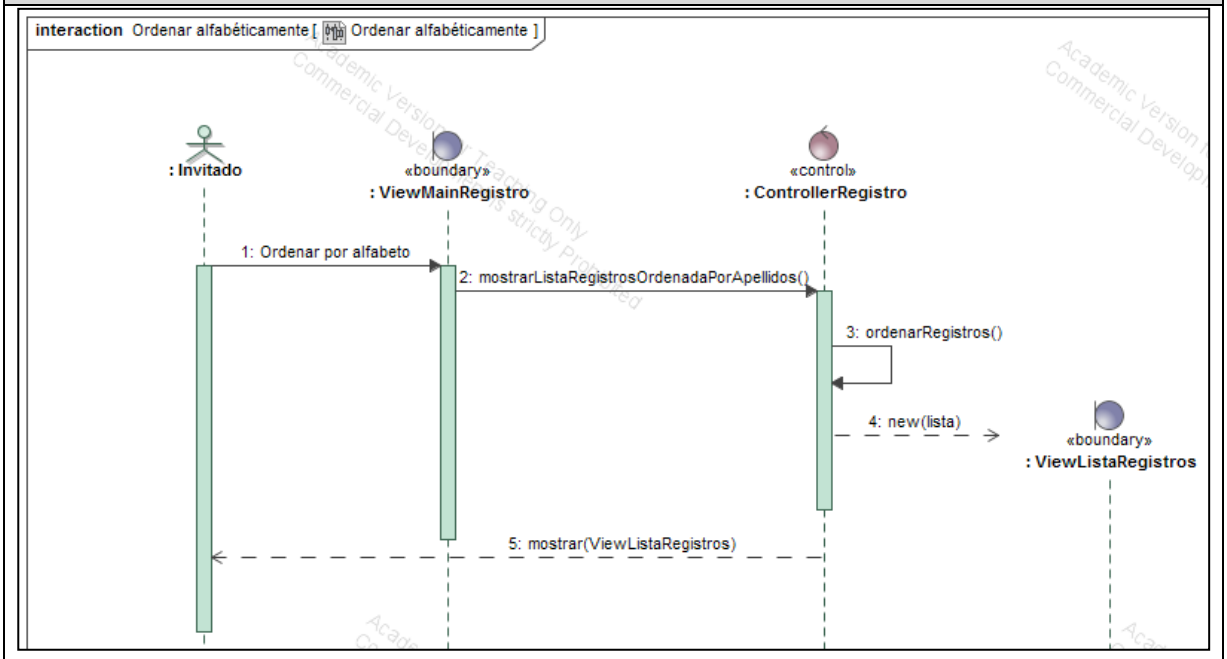
Nombre	Mostrar los resultados por orden de clasificación
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor/invitado con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe algun registro en la competición).
Escenario principal	
1- El usuario selecciona la opción que permite mostrar los resultados por orden de clasificación. 2- El sistema ordena la lista y la muestra.	
Post-condiciones	
Escenarios alternativos	
Diagramas de secuencia	
	
Mockups asociados	



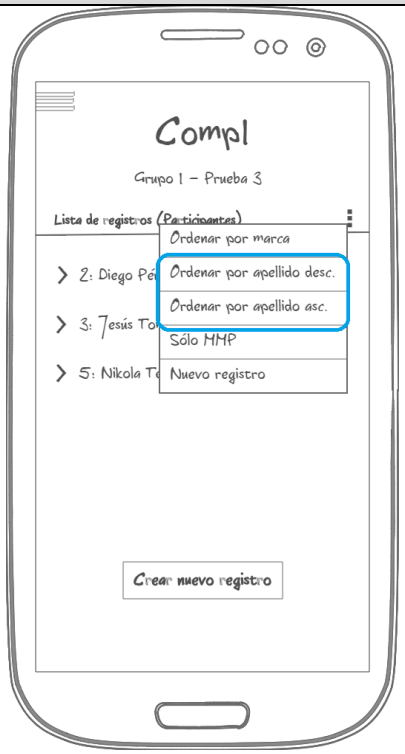
Nombre	Ordenar por orden alfabético
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor/invitado con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición). PRECOND4 (Existe algun registro en la competición). PRECOND5 (El tipo de prueba de los registros es "individual").
Escenario principal	
1- El usuario selecciona la opción que permite ordenar de mayor a menor los participantes según los apellidos. 2- El sistema muestra el nuevo orden.	
Post-condiciones	
Escenarios alternativos	

1- El usuario selecciona la opción que permite ordenar de menor a mayor los participantes según los apellidos.
2- El sistema muestra el nuevo orden.

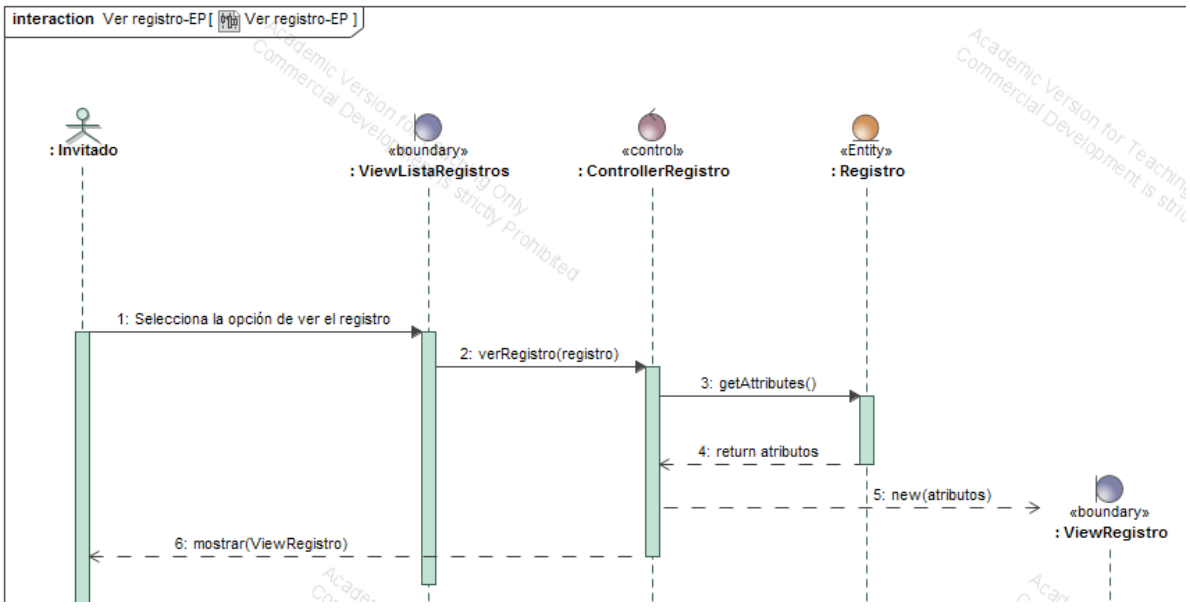
Diagramas de secuencia



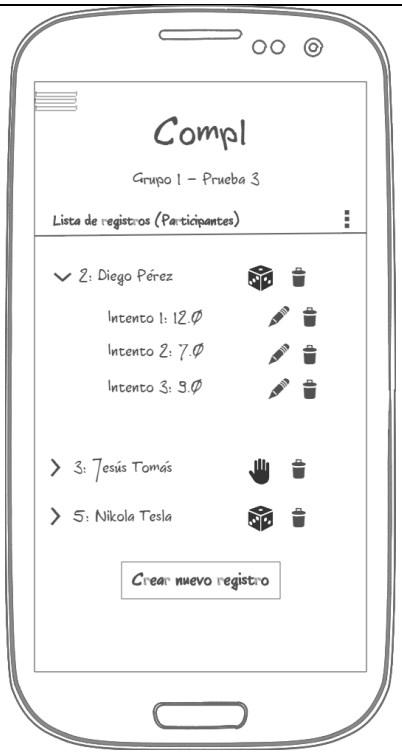
Mockups asociados



Nombre	Ver los datos de un registro
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor/invitado con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario accede a la lista de registros. (Ordenar resultados). (Ver MMP). 2- El usuario selecciona un registro de la lista de registros. 3- El sistema muestra al usuario información sobre el registro.	
Post-condiciones	
Escenarios alternativos	
Diagramas de secuencia	

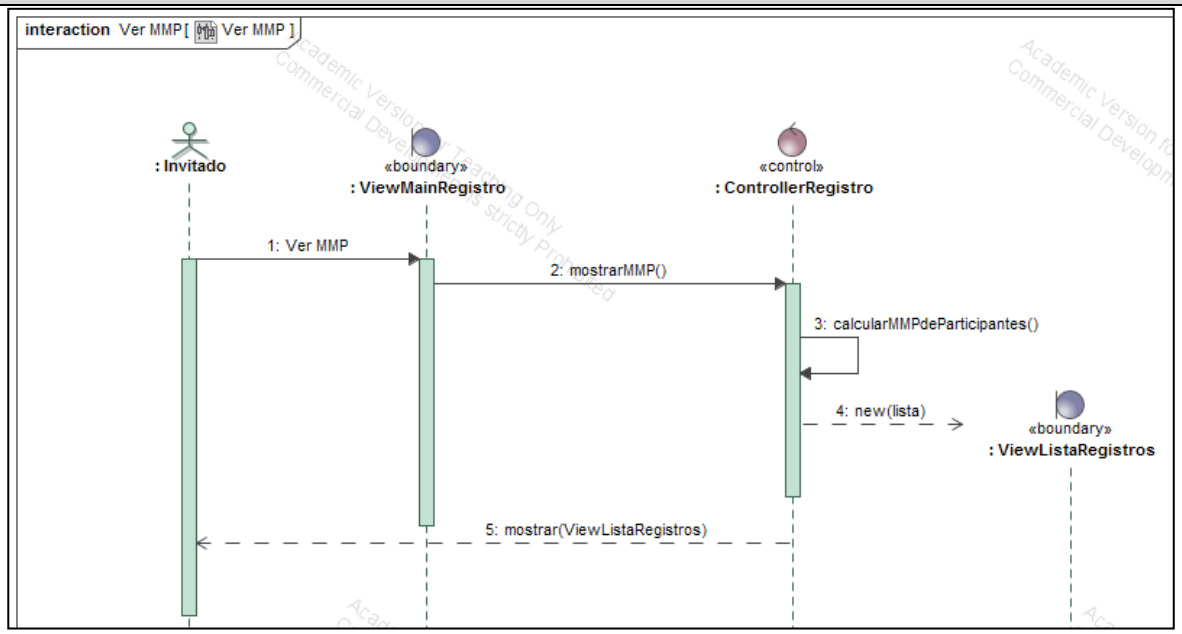


Mockups asociados

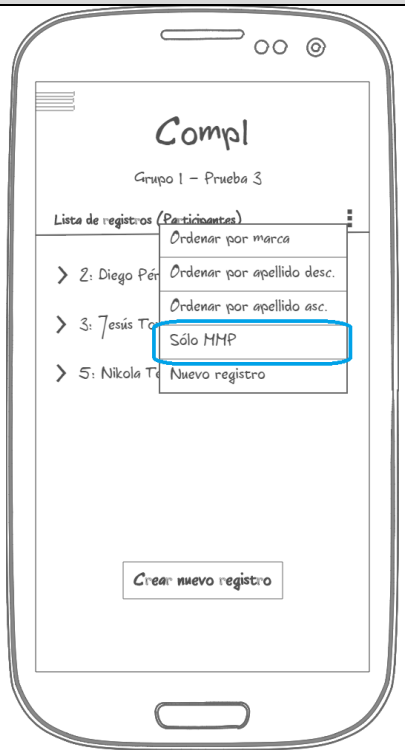


Nombre	Ver el mejor registro de un participante o equipo
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor/invitado con permisos). PRECOND3 (El usuario ha accedido al ámbito de alguna competición).
Escenario principal	
1- El usuario selecciona la opción que permite ver el MMP. 2- El sistema filtra las marcas de cada participante/equipo. 3- El sistema muestra al usuario los resultados.	
Post-condiciones	
Escenarios alternativos	
2- El sistema detecta que no existen registros en la competición para esa prueba (y grupo filtrado). 3- El sistema muestra una lista vacía.	

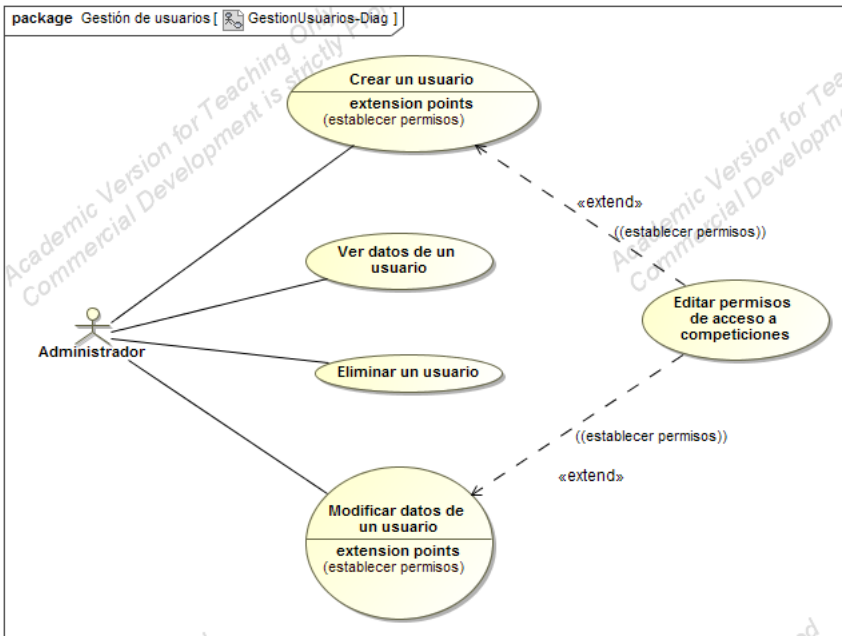
Diagramas de secuencia



Mockups asociados



8. Gestión de usuarios

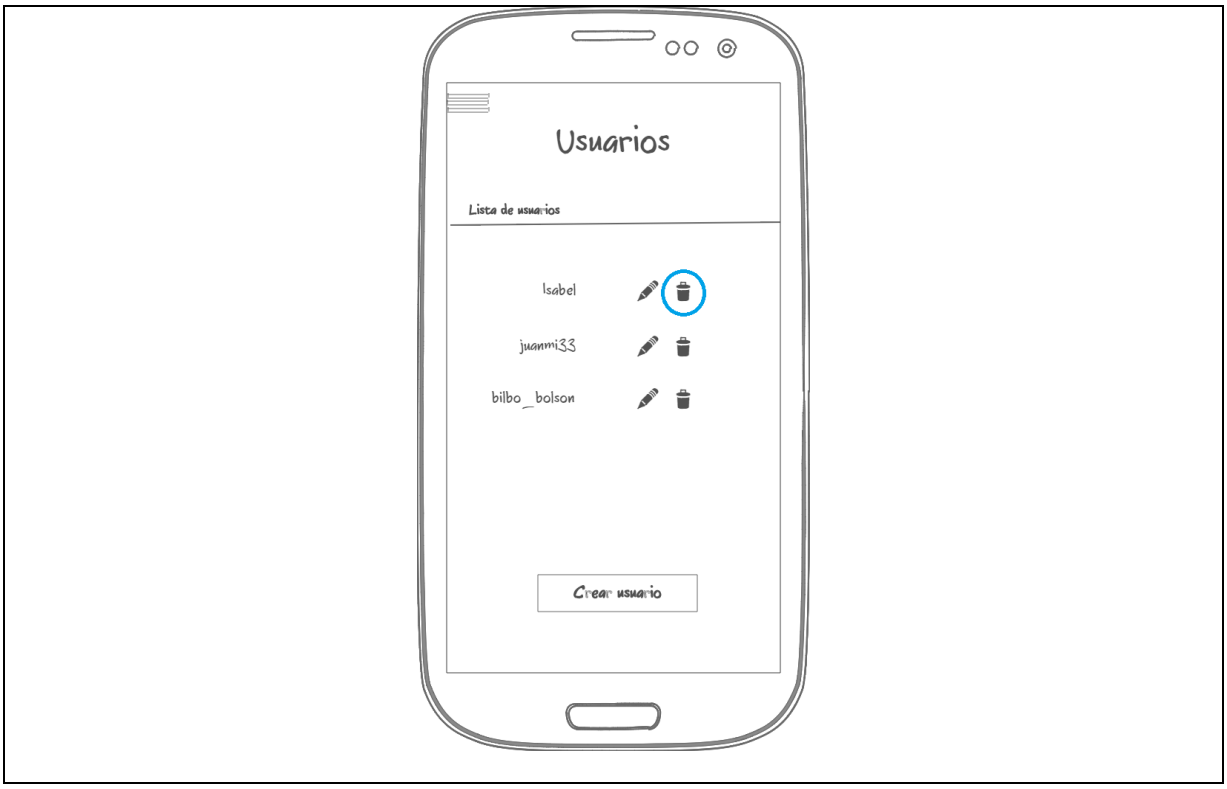


Nombre	Crear un usuario
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado como admin).
Escenario principal	
1- El administrador accede al sistema de administración de usuarios. 2- El administrador selecciona la opción que permite crear un nuevo usuario. 3- El sistema muestra una vista con los campos a rellenar. (establecer permisos) 4- El administrador introduce los datos del usuario. 5- El administrador acepta la creación. 6- El sistema crea el nuevo usuario.	
Post-condiciones	POSCOND1 (el usuario existe en la aplicación)
Escenarios alternativos	

5- El administrador cancela la creación. 6- El sistema no modifica nada y devuelve al administrador a una vista anterior.
5- El sistema detecta que ya existe un usuario con el mismo nick y avisa al administrador. 6- El administrador cambia el nick por otro o cancela la creación.
Diagramas de secuencia
Mockups asociados


Nombre	Eliminar un usuario
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado como admin).
Escenario principal	

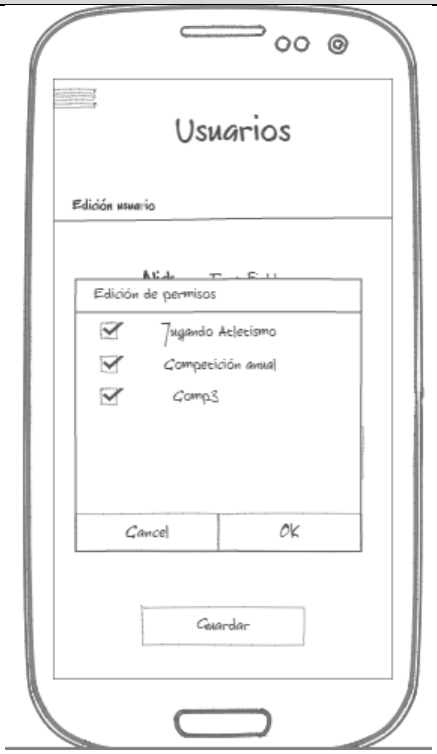
1- El administrador accede al sistema de administración de usuarios. 2- El administrador selecciona el usuario que desea eliminar. 3- El administrador selecciona la opción que permite borrar al usuario. 4- El sistema pide confirmación sobre la eliminación. 5- El administrador acepta borrar al usuario. 6- El sistema elimina al usuario junto con todos sus registros.	
Post-condiciones	POSCOND1 (el usuario seleccionado no existe en la aplicación)
Escenarios alternativos	
5- El administrador cancela la eliminación. 6- El sistema no modifica nada y devuelve al administrador a una vista anterior.	
5- El sistema detecta que ya existe un usuario con el mismo nick y avisa al administrador. 6- El administrador cambia el nick por otro o cancela la creación.	
4- El sistema detecta que el usuario que desea eliminar es él mismo y lo avisa. 5- El usuario acepta el aviso. 6- El sistema no modifica nada.	
Diagramas de secuencia	
Mockups asociados	



Nombre	Modificar datos de un usuario
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado como admin).
Escenario principal	
1- El administrador accede al sistema de administración de usuarios. 2- El administrador selecciona un usuario de la lista de usuarios. 3- El administrador selecciona "modificar usuario". 4- El administrador modifica los campos que necesite. (establecer permisos) 5- El administrador guarda la modificación. 6- El sistema guarda los nuevos datos del usuario.	
Post-condiciones	POSCOND1 (el usuario seleccionado no existe en la aplicación)
Escenarios alternativos	
5- El administrador cancela la operación. 6- El sistema no modifica nada y devuelve al administrador a una vista anterior.	

5- El sistema detecta que ya existe un usuario con el mismo nick y avisa al administrador. 6- El administrador cambia el nick por otro o cancela la creación.
Diagramas de secuencia
Mockups asociados

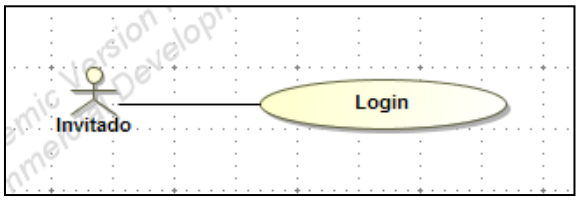

Nombre	Editar permisos de acceso a competiciones
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado como admin). PRECOND2 (El usuario está en modo creación o edición de un usuario).
Escenario principal	
1- El administrador selecciona "modificar permisos". 2- El administrador selecciona o elimina los permisos que desee. 3- El administrador guarda la modificación.	

Post-condiciones	
Escenarios alternativos	
5- El administrador cancela la operación. 6- El sistema no modifica nada y devuelve al administrador a una vista anterior.	
Diagramas de secuencia	
Mockups asociados	
	

Nombre	Ver datos de un usuario
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado como admin).
Escenario principal	
1- El administrador accede al sistema de administración de usuarios. 2- El administrador selecciona un usuario de la lista de usuarios. 3- El sistema muestra al usuario información sobre el participante.	

Post-condiciones	
Escenarios alternativos	
Diagramas de secuencia	
Mockups asociados	

9. Login



Nombre	Login
Autores	David Doña Corrales
Pre-condiciones	

Escenario principal

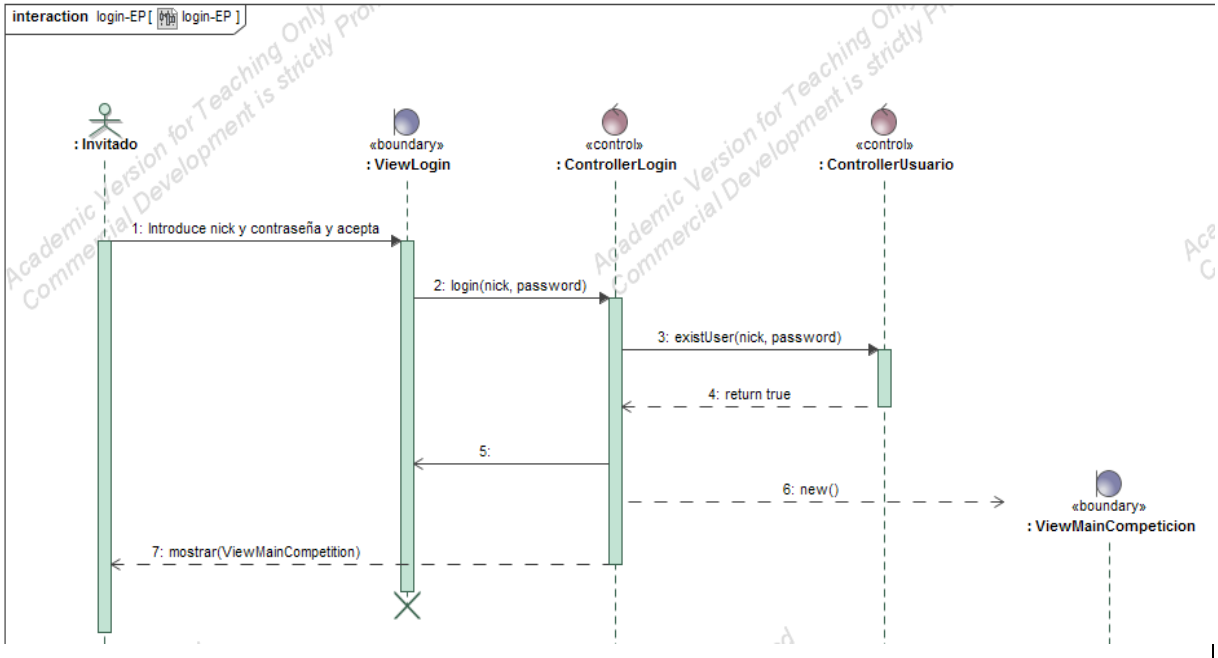
1- El usuario abre la aplicación.
2- El sistema muestra una vista con los datos a introducir por el usuario para el loguin.
3- El usuario introduce los datos.
4- El usuario pulsa la opción que permite iniciar la sesión.
5- El sistema comprueba los datos y muestra el "main" de la aplicación.

Post-condiciones	

Escenarios alternativos

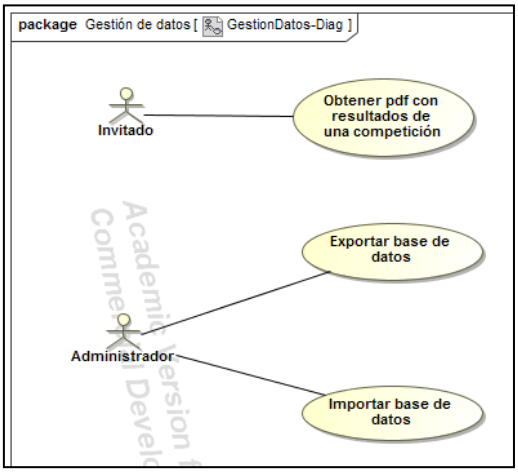
5- El sistema comprueba los datos y detecta que los datos son incorrectos.
6- El sistema informa al usuario del error.

Diagramas de secuencia



Mockups asociados

1. Gestión de datos



Nombre	Obtener pdf con resultados de una competición
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin o gestor/invitado con permisos).

Escenario principal

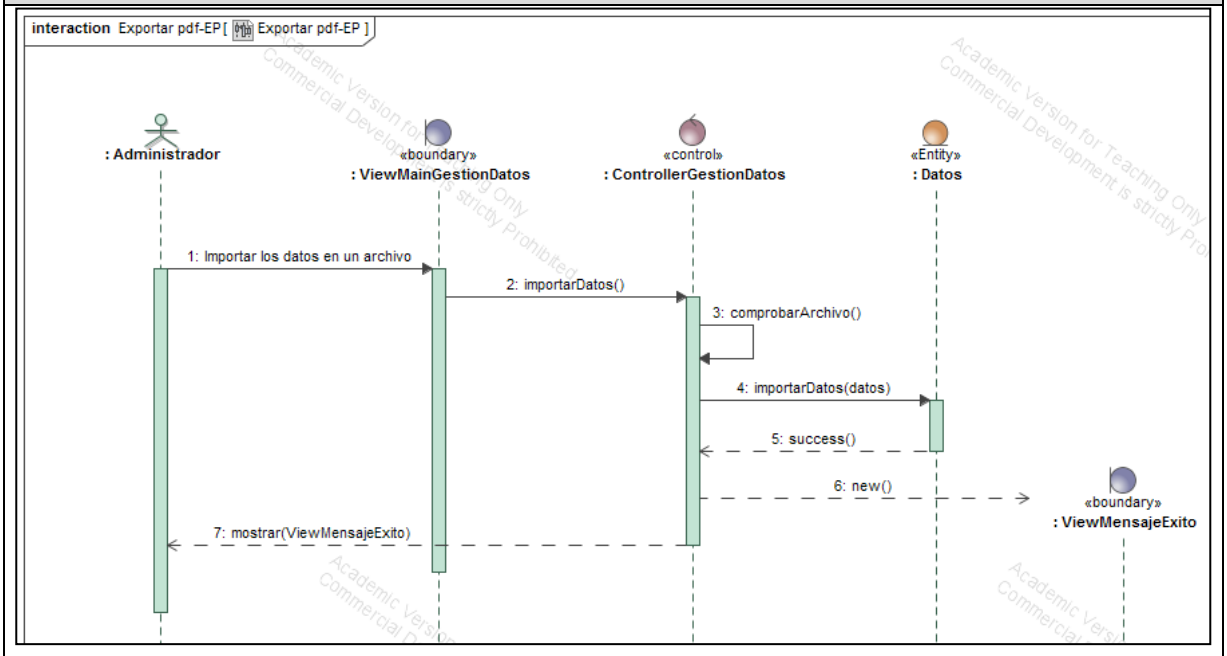
1- El usuario selecciona la opción que permite exportar un fichero pdf.
2- El sistema muestra al usuario una vista con las pruebas y grupos a seleccionar que existen en la competición.
3- El usuario selecciona las pruebas y grupos que desea.
4- El usuario acepta generar el pdf.
5- El sistema genera el pdf.

Post-condiciones	

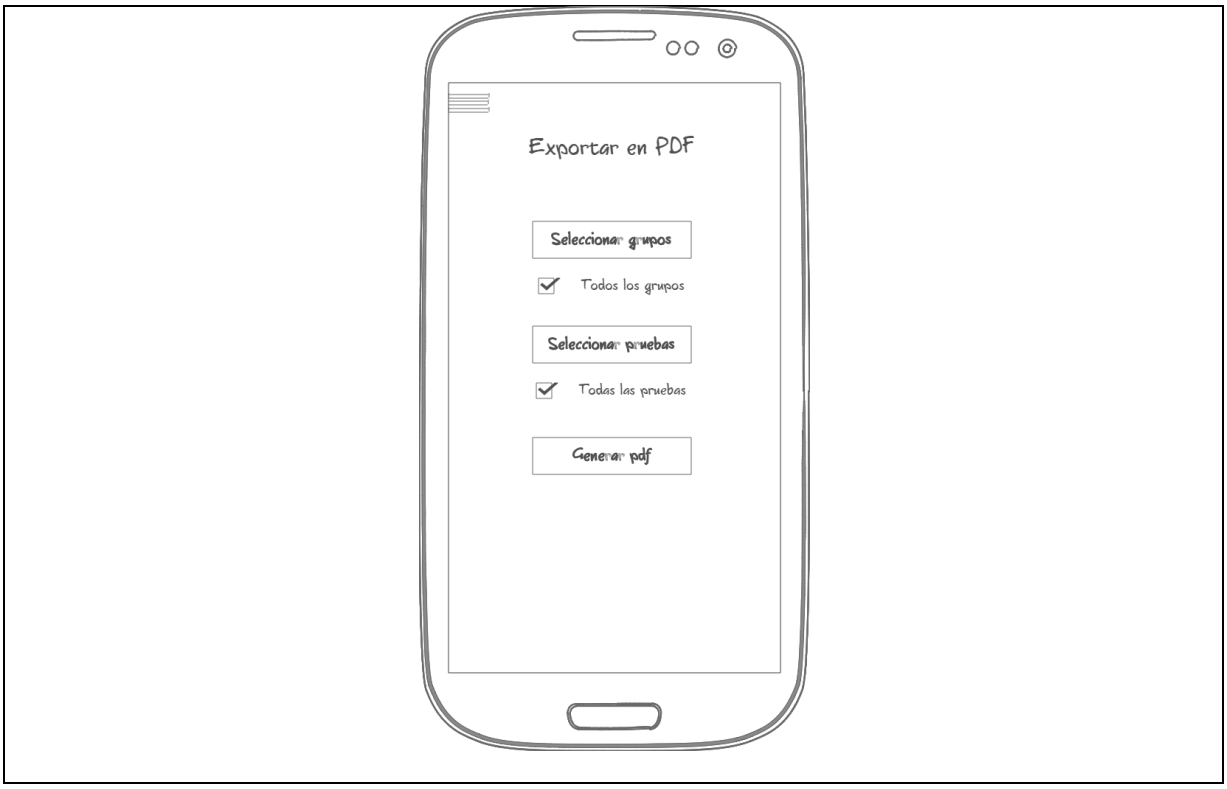
Escenarios alternativos

4- El usuario cancela la petición.
5- El sistema no genera nada.

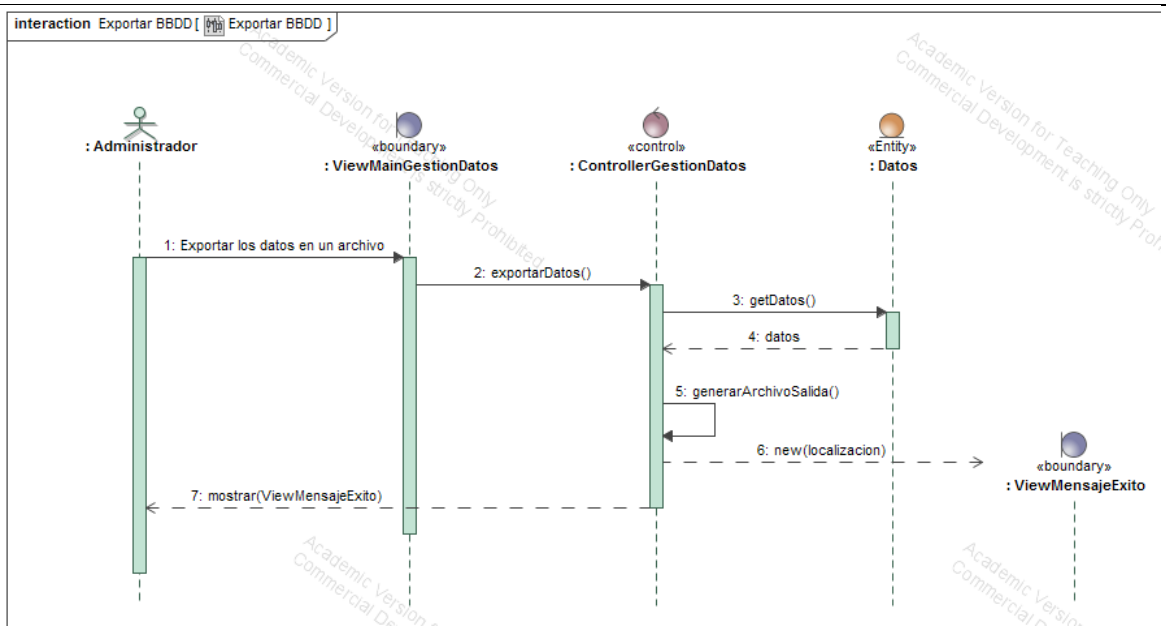
Diagramas de secuencia



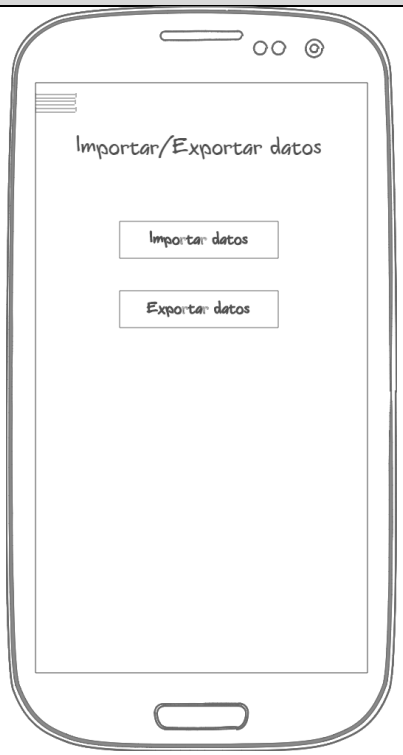
Mockups asociados

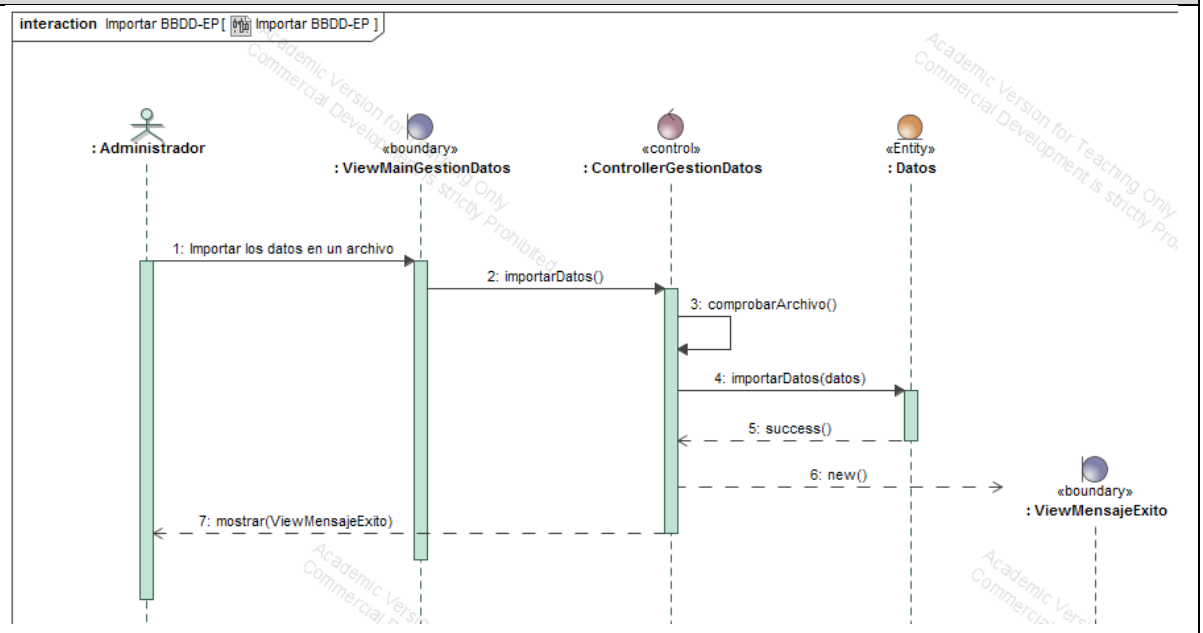


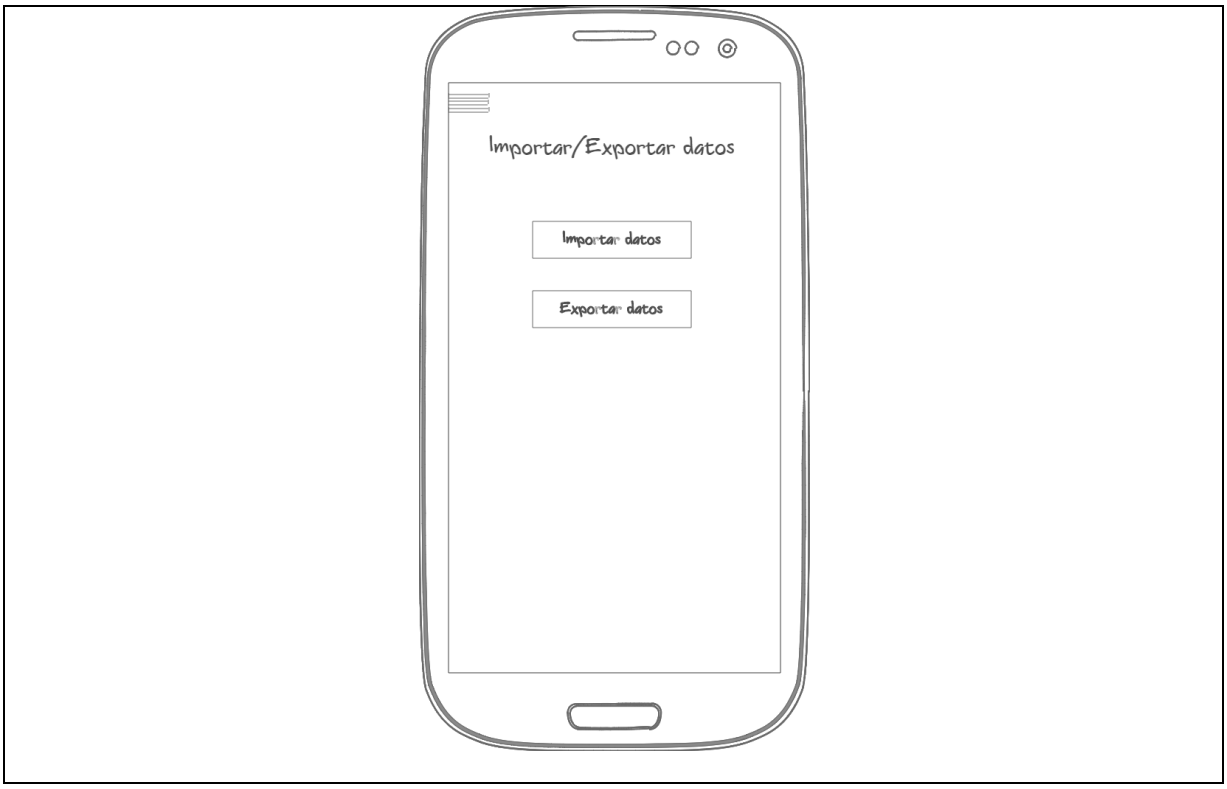
Nombre	Exportar base de datos
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin).
Escenario principal	
1- El administrador selecciona la opción que permite exportar los datos de la aplicación. 2- El sistema genera un fichero de copia de seguridad 3- El sistema guarda el fichero en el directorio de aplicación.	
Post-condiciones	
Escenarios alternativos	
2- El sistema detecta que existe ya un fichero con el mismo nombre que el fichero backup. 3- El sistema sobrescribe el fichero con el generado.	
Diagramas de secuencia	



Mockups asociados



Nombre	Importar base de datos
Autores	David Doña Corrales
Pre-condiciones	PRECOND1 (El usuario está logueado). PRECOND2 (El usuario tiene rol admin).
Escenario principal	
1- El administrador selecciona la opción que permite importar datos a la aplicación. 2- El sistema pide confirmación al usuario de la operación 3- El usuario acepta la operación 4- El sistema toma los datos del fichero alojado en el directorio de aplicación. 5- El sistema elimina los datos internos de la aplicación y restaura los del fichero.	
Post-condiciones	
Escenarios alternativos	
4- El sistema no detecta el fichero en la ubicación determinada. 5- El sistema informa al usuario del error.	
Diagramas de secuencia	
	
Mockups asociados	


Importar/Exportar datos
Importar datos
Exportar datos