

UNIVERSIDAD DE POLITÉCNICA DE MADRID



POLITÉCNICA

**Escuela Técnica Superior de Ingeniería de
Sistemas Informáticos**

Máster en Ingeniería Web

Proyecto Fin de Máster

TOP GAMES

Jorge Lillo Cobacho

Junio / 2015

**UNIVERSIDAD DE POLITÉCNICA DE
MADRID**

**Escuela Técnica Superior de Ingeniería de Sistemas
Informáticos**

MASTER EN INGENIERÍA WEB

Proyecto Fin de Máster
TOP GAMES

Autor: Jorge Lillo Cobacho

Director: Luis Fernando de Mingo López

ÍNDICE RESUMIDO

1. INTRODUCCIÓN	8
2. ALCANCE DEL PROYECTO.....	16
3. DESARROLLO DE LA SOLUCIÓN	23
4. CONCLUSIONES.....	69
5. BIBLIOGRAFÍA E INFOGRAFÍA	69
6. APÉNDICES	71

ÍNDICE DETALLADO

1. INTRODUCCIÓN	8
2. ALCANCE DEL PROYECTO.....	16
2.1. CONCEPTOS BÁSICOS	16
(a) <i>Framework Web</i>	16
(b) <i>Patrón MVC</i>	18
(c) <i>API REST</i>	19
(d) <i>ORM</i>	21
3. DESARROLLO DE LA SOLUCIÓN	23
3.1. FASE PREVIA	23
(a) <i>Modelo de datos</i>	23
(b) <i>Mockups</i>	24
(c) <i>Diagramas</i>	29
(d) <i>Planificación</i>	32
3.2. FASE UNO: DESARROLLO WEB.....	32
(a) <i>Symfony 2</i>	32
(b) <i>Comentarios del código</i>	40
(c) <i>Manual de usuario</i>	43
3.3. FASE DOS: API REST.....	55
3.4. FASE TRES: APP MÓVIL	63
(a) <i>Ionic Framework</i>	63
(b) <i>Comentarios del código</i>	64
4. CONCLUSIONES.....	69
4.1. CONCLUSIÓN Y LÍNEAS FUTURAS	69
5. BIBLIOGRAFÍA E INFOGRAFÍA	70
6. APÉNDICES	71
6.1. ANEXO A: MANUAL DE INSTALACIÓN TOP GAMES WEB	71
(a) <i>Creación nuevo proyecto</i> :	72
(b) <i>Importar TopGames web</i> :.....	74
(c) <i>Composer añadir bundles</i> :	77
6.2. ANEXO B: MANUAL DE IONIC FRAMEWORK	79
(a) <i>Creación nuevo proyecto y comandos útiles</i> :.....	83
(b) <i>Importar Top Games ionic</i> :	88

ÍNDICE DE FIGURAS

ILUSTRACIÓN 1 TOP GAMES WEB USUARIO NO REGISTRADO.....	8
ILUSTRACIÓN 2 TOP GAMES WEB USUARIO REGISTRADO	9
ILUSTRACIÓN 3 TOP GAMES WEB ADMINISTRADOR	9
ILUSTRACIÓN 4 TOP GAMES WEB DETALLE LISTA	10
ILUSTRACIÓN 5 API REST DOCUMENTACIÓN	11
ILUSTRACIÓN 6 APP MÓVIL LOGIN.....	12
ILUSTRACIÓN 7 APP MÓVIL HOME	12
ILUSTRACIÓN 8 APP MÓVIL DETALLES JUEGO	13
ILUSTRACIÓN 9 APP MÓVIL LISTAS.....	14
ILUSTRACIÓN 10 APP MÓVIL ACERCA DE.....	14
ILUSTRACIÓN 11 APP MÓVIL ACERCA DE 2.....	15
ILUSTRACIÓN 12 CONCEPTOS BÁSICOS MVC	18
ILUSTRACIÓN 13 REST VS SOAP	20
ILUSTRACIÓN 14 MODELO DE DATOS	23
ILUSTRACIÓN 15 MOCKUP HOME	24
ILUSTRACIÓN 16 MOCKUP SEARCH.....	25
ILUSTRACIÓN 17 MOCKUP GAME DETAIL	25
ILUSTRACIÓN 18 MOCKUP LIST DETAIL.....	26
ILUSTRACIÓN 19 MOCKUP APP LOGIN	26
ILUSTRACIÓN 20 MOCKUP APP GAMES	27
ILUSTRACIÓN 21 MOCKUPS APP LIST	27
ILUSTRACIÓN 22 MOCKUP APP GAME DETAILS	28
ILUSTRACIÓN 23 MOCKUP APP ABOUT	28
ILUSTRACIÓN 24 CASOS DE USO USUARIO NO REGISTRADO.....	29
ILUSTRACIÓN 25 CASOS DE USO USUARIOREGISTRADO.....	30
ILUSTRACIÓN 26 CASOS DE USO ADMIN	31
ILUSTRACIÓN 27 DIAGRAMA DE ESTADO BUSCAR	31
ILUSTRACIÓN 28 MANUAL USUARIO TopGAMES HOME.....	43
ILUSTRACIÓN 29 MANUAL USUARIO BUSCAR JUEGO	43
ILUSTRACIÓN 30 MANUAL USUARIO DETALLE JUEGO	44
ILUSTRACIÓN 31 MANUAL USUARIO BUSCAR LISTAS.....	44
ILUSTRACIÓN 32 MANUAL USUARIO DETALLE LISTA.....	45
ILUSTRACIÓN 33 MANUAL USUARIO LOGIN CABECERA.....	45
ILUSTRACIÓN 34 MANUAL USUARIO BAD CREDENTIALS	45
ILUSTRACIÓN 35 MANUAL USUARIO CABECERA REGISTRADO.....	46
ILUSTRACIÓN 36 MANUAL USUARIO MIS LISTAS.....	46
ILUSTRACIÓN 37 MANUAL USUARIO NUEVA LISTA	46
ILUSTRACIÓN 38 MANUAL USUARIO EDITAR LISTA 1.....	47
ILUSTRACIÓN 39 MANUAL USUARIO EDITAR LISTA 2.....	47
ILUSTRACIÓN 40 MANUAL USUARIO BORRAR.....	47
ILUSTRACIÓN 41 MANUAL USUARIO AÑADIR JUEGO	48
ILUSTRACIÓN 42 MANUAL USUARIO ELIMINAR JUEGO	48
ILUSTRACIÓN 43 MANUAL USUARIO LISTA OTRO USUARIO	49
ILUSTRACIÓN 44 MANUAL USUARIO LISTA CLONADA.....	49
ILUSTRACIÓN 45 MANUAL USUARIO HOME ADMIN	50

ILUSTRACIÓN 46 MANUAL USUARIO LISTAR USUARIOS	50
ILUSTRACIÓN 47 MANUAL USUARIO NUEVO USUARIO	50
ILUSTRACIÓN 48 MANUAL USUARIO LISTADO JUEGOS	51
ILUSTRACIÓN 49 MANUAL USUARIO NUEVO JUEGO	51
ILUSTRACIÓN 50 MANUAL USUARIO EDITAR JUEGO	52
ILUSTRACIÓN 51 MANUAL USUARIO DETALLES JUEGO	52
ILUSTRACIÓN 52 MANUAL USUARIO GESTIÓN DE PLATAFORMAS	53
ILUSTRACIÓN 53 MANUAL USUARIO LISTAS ADMIN.....	53
ILUSTRACIÓN 54 MANUAL USUARIO GESTIÓN LISTAS USUARIOS.....	54
ILUSTRACIÓN 55 MANUAL USUARIO GESTION LISTA OTRO USUARIO	54
ILUSTRACIÓN 56 MANUAL USUARIO ELIMINAR JUEGO DE OTRA LISTA	55
ILUSTRACIÓN 57 API REST GET ALL GAMES	55
ILUSTRACIÓN 58 API REST GET GAME	56
ILUSTRACIÓN 59 API REST GAMES FROM LIST.....	56
ILUSTRACIÓN 60 API REST BUSCAR JUEGO	56
ILUSTRACIÓN 61 API REST GET LISTS	57
ILUSTRACIÓN 62 API REST GET LIST	57
ILUSTRACIÓN 63 API REST LISTAS DE USUARIO	57
ILUSTRACIÓN 64 API REST BUSCAR LISTA	58
ILUSTRACIÓN 65 API REST MÁS JUEGOS	58
ILUSTRACIÓN 66 API REST GET USERS.....	58
ILUSTRACIÓN 67 API REST LOGIN.....	59
ILUSTRACIÓN 68 API REST GET USER.....	59
ILUSTRACIÓN 69 API REST EJEMPLO RESPUESTA JSON	60
ILUSTRACIÓN 70 API REST EJEMPLO RESPUESTA XML.....	61
ILUSTRACIÓN 71 NOTEPAD++ & LIGHT EXPLORER.....	65
ILUSTRACIÓN 72 IONIC APP JS	66
ILUSTRACIÓN 73 IONIC CONTROLLER.JS	67
ILUSTRACIÓN 74 IONIC SERVICE.JS	67
ILUSTRACIÓN 75 IONIC HTML TEMPLATE	68
ILUSTRACIÓN 76 IONIC ANDROID O IOS TEMPLATE.....	68
ILUSTRACIÓN 77 ANEXO I INSTALACIÓN SYMFONY: NETBEANS DESCARGA.....	71
ILUSTRACIÓN 78 ANEXO I INSTALACIÓN SYMFONY: NETBEANS NEW PROJECT	72
ILUSTRACIÓN 79 ANEXO I INSTALACIÓN SYMFONY: NETBEANS NEW PROJECT 2.....	72
ILUSTRACIÓN 80 ANEXO I INSTALACIÓN SYMFONY: NETBEANS NEW PROJECT 3.....	73
ILUSTRACIÓN 81 ANEXO I INSTALACIÓN SYMFONY: NETBEANS NEW PROJECT 4.....	73
ILUSTRACIÓN 82 ANEXO I IMPORTAR TOP GAMES 1.....	74
ILUSTRACIÓN 83 ANEXO I IMPORTAR TOP GAMES 2.....	74
ILUSTRACIÓN 84 ANEXO I IMPORTAR TOP GAMES 3.....	75
ILUSTRACIÓN 85 ANEXO I IMPORTAR TOP GAMES 4.....	75
ILUSTRACIÓN 86 ANEXO I IMPORTAR TOP GAMES 5.....	76
ILUSTRACIÓN 87 ANEXO I IMPORTAR TOP GAMES 6.....	76
ILUSTRACIÓN 88 ANEXO I COMPOSER	77
ILUSTRACIÓN 89 ANEXO 2: INSTALACIÓN IONIC JAVA	79
ILUSTRACIÓN 90 ANEXO 2: INSTALACIÓN IONIC NODE.....	79
ILUSTRACIÓN 91 INSTALACIÓN IONIC CORDOVA.....	80
ILUSTRACIÓN 92 INSTALACIÓN IONIC IONIC	80
ILUSTRACIÓN 93 INSTALACIÓN IONIC CONFIGURACIÓN AVANZADA	80
ILUSTRACIÓN 94 INSTALACIÓN IONIC VARIABLES DE ENTORNO	81

ILUSTRACIÓN 95 INSTALACIÓN IONIC VARIABLES DE ENTORNO JAVA_HOME.....	81
ILUSTRACIÓN 96 INSTALACIÓN IONIC VARIABLES DE ENTORNO PATH	82
ILUSTRACIÓN 97 IONIC PUESTA EN MARCHA 1	83
ILUSTRACIÓN 98 IONIC PUESTA EN MARCHA 2	84
ILUSTRACIÓN 99 IONIC PUESTA EN MARCHA 3	84
ILUSTRACIÓN 100 IONIC PUESTA EN MARCHA 4	85
ILUSTRACIÓN 101 IONIC PUESTA EN MARCHA 5	85
ILUSTRACIÓN 102 IONIC PUESTA EN MARCHA 6	87
ILUSTRACIÓN 103 IONIC PUESTA EN MARCHA 7	87
ILUSTRACIÓN 104 IONIC TOPGAMES SERVE	88
ILUSTRACIÓN 105 IONIC TOPGAMES SERVE --LAB	88

Resumen

Top Games es una aplicación web realizada en Symfony 2 que permite a los usuarios registrados crear listas de sus juegos favoritos con el fin de tenerlas guardadas en un mismo sitio.

Además tenemos la opción de conectarnos con nuestros móviles desde las apps de ios y android desarrolladas con el framework ionic.

Todo ello, usando todos los conocimientos adquiridos a lo largo de este año en el Master Universitario en Ingeniería Web.

Summary

Top Games is a web application developed in Symfony 2 which allows registered users to create lists of their favorite games in order to have them stored in one place.

We also have the option to connect with our smartphones from our ios and android apps which are developed with ionic framework.

We do all that using the knowledge acquired during this year's Master's Degree in Web Engineering.

Palabras clave:

Desarrollo web, Symfony 2, php, api rest, ionic framework, aplicación híbrida. angular js, android, ios

Keywords:

Web development, Symfony 2, php, api rest, ionic framework, hybrid application. angular js, android, ios

1. INTRODUCCIÓN

Top Games se trata de una aplicación web mediante la cual podemos crear nuestras listas de videojuegos favoritos. Cuenta con dos versiones: la versión web, accesible desde cualquier navegador y la versión móvil (app ios y android).

Para poder acceder a los datos de la web con las versiones móviles, se ha preparado una api rest pública que nos devuelva los datos.

Versión web:

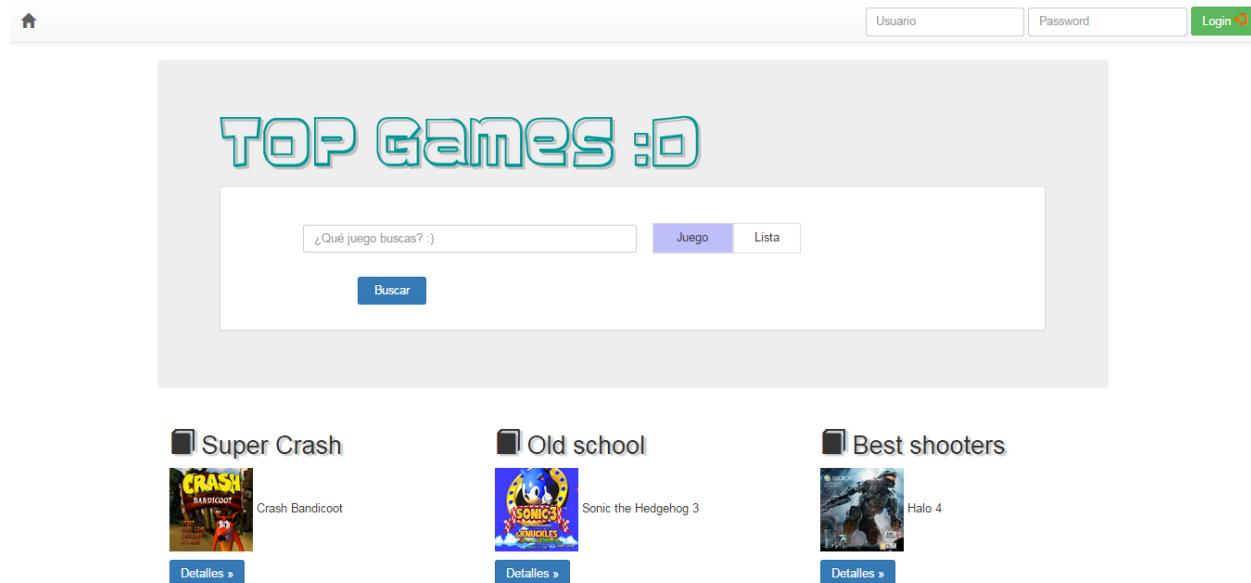


Ilustración 1 Top Games web usuario no registrado

Al entrar en la web, nos encontramos una página sencilla en la que podemos observar un formulario para iniciar sesión en la parte superior derecha, un formulario para realizar búsquedas de juegos o listas y unas listas recomendadas con uno de sus juegos que salen de forma aleatoria.

Existen tres tipos de usuarios en la aplicación:

- Usuarios no registrados: podrán buscar juegos/listas y ver los detalles de cada uno de ellos.
- Usuarios registrados: podrán crear, editar y eliminar listas de usuario, a las cuales podrán añadirlo o quitarlo de ellas, tras encontrar un juego de su gusto (ya sea mediante el buscador o entrando desde los detalles de otra lista).

Además, si les gusta alguna lista de otro usuario, se podrá clonar la lista con los juegos incluidos.

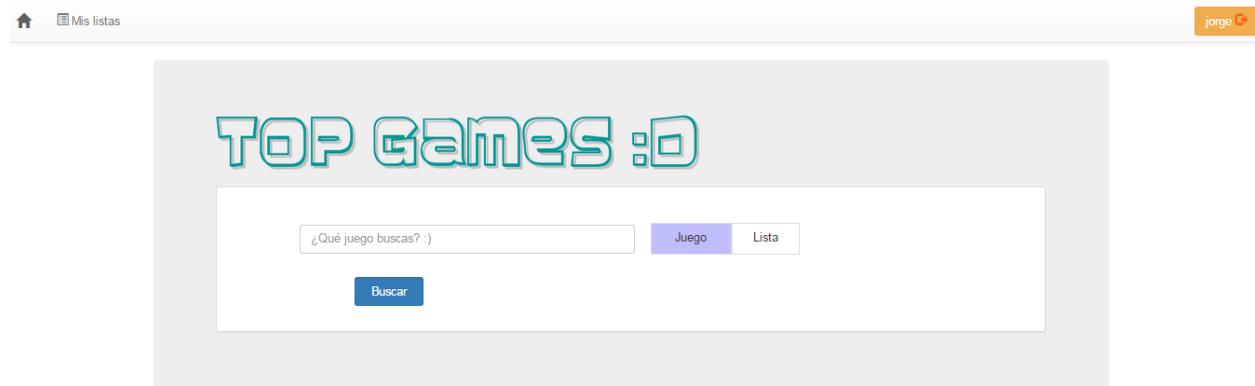


Ilustración 2 Top Games web usuario registrado

- Administradores: además de poder realizar todo lo que un usuario registrado pueda hacer, podrán gestionar:
 - Juegos: creación, edición, borrado, añadir plataforma, quitar plataforma.
 - Listas: creación, edición, borrado (ambas opciones para sus propias listas y las de todos los usuarios)
 - Usuarios: dar de alta, modificar datos, eliminar usuarios.



Ilustración 3 Top Games web administrador

El objetivo de Top Games es poder tener las listas de nuestras sagas favoritas o juegos de época similar por el criterio que más nos guste:

Top Games Web - Juegos - Listas

Listas: detalles lista

Id	6
Nombre	Final Fantasy List
Descripción	Lista de final fantasy
Autor Original	u1
Propietario de lista	jorge
Juegos en lista	7

[clonar lista](#)

- [Volver a listas](#)
- [Editar](#)
- [Eliminar](#)

Listado de juegos:

Id	Título	Plataformas	Opciones
10	Final Fantasy I	• PSX • PC	eliminar juego
11	Final Fantasy II	• PSX • PC	eliminar juego
12	Final Fantasy III	• Nintendo DS • PSX • PC	eliminar juego
13	Final Fantasy IV	• PSX • PC	eliminar juego
14	Final Fantasy V	• PSX • Nintendo DS	eliminar juego
15	Final Fantasy VI	• PSX • PC	eliminar juego
26	Final Fantasy X	• PS2 • PSP	eliminar juego

Ilustración 4 Top Games Web detalle lista

Asimismo, podemos buscar listas para poder, posteriormente, clonar y mejorarlas nosotros quitando o añadiendo más juegos. En cuanto al resto de las secciones de la web, se detallarán en los siguientes apartados:

Api Rest:

La web también cuenta con una api rest pública a la que podemos acceder desde: <http://127.0.0.1:8000/api>

Para poder facilitar la tarea a desarrolladores externos que quieran obtener nuestros datos, hemos preparado una documentación:

The screenshot shows a detailed API documentation page. At the top, there's a green header bar with the text "API documentation" on the left and "body format: Form Data ▾ request format: json ▾" on the right. Below the header, the main content area has a light blue background. It starts with a section for the endpoint "/api/games". This section includes a "Documentation" tab (selected) and a "Sandbox" tab. The "Documentation" tab contains a "Requirements" table with one row: "Name" (id), "Requirement" (json|xml|html), "Type" (empty), and "Description" (empty). Below this is a "GET /api/games.{_format}" entry with a "Returns a game" note. Further down, there's another entry for "/api/games/fromList/{idList}" with a "Search for games by list" note. Finally, there's an entry for "/api/games/search/{search}" with a "Search for games" note.

Ilustración 5 Api rest documentación

Gracias a esta documentación, se podrán enterar de qué opciones tiene la api disponible para poderla usar. Debido a la falta de tiempo, la app móvil consistirá en un simple visor de la web (no podrá insertar o borrar datos). Por ello, la api contiene las peticiones GET necesarias para construir el visor y un único POST para realizar el login. En futuras versiones, este será uno de los puntos más importante que se deberá mejorar: que la app móvil pueda hacer lo mismo que la web. Por otro lado, se detallarán las llamadas disponibles de la api en el apartado de desarrollo correspondiente.

App móvil:

Para poder aportar algo más al usuario, se ha creado una aplicación híbrida que nos permite entrar en el mercado móvil teniendo nuestra aplicación en Google Play y en la Apple Store. Nuestra aplicación, como hemos mencionado en apartados anteriores, se encuentra limitada permitiéndonos únicamente ser un mero visor de la web. Para poder usar la app, el usuario debe estar registrado, dado que la primera pantalla es un login que nos comprueba con la api si el usuario existe.

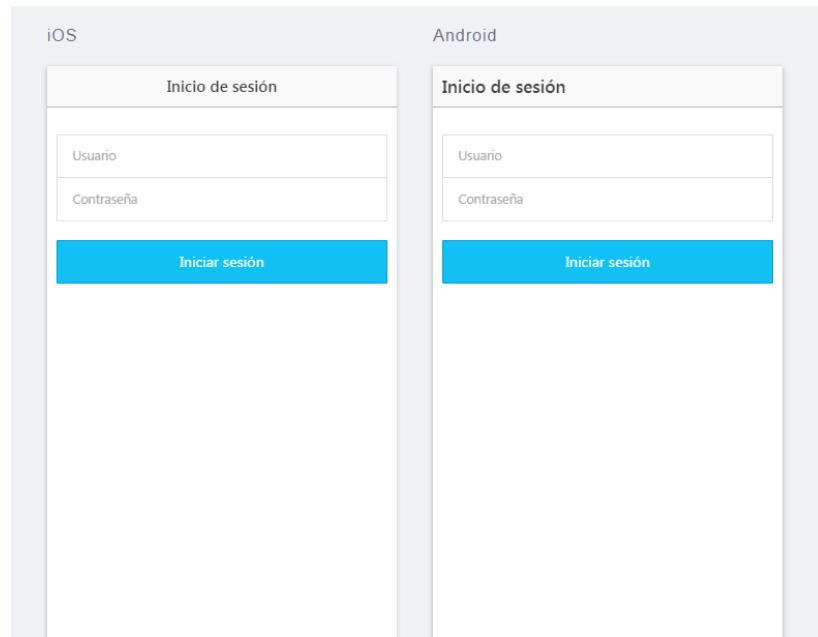


Ilustración 6 APP Móvil login

Una vez loggeados, entramos a ver la aplicación, la cual tiene tres pestañas:

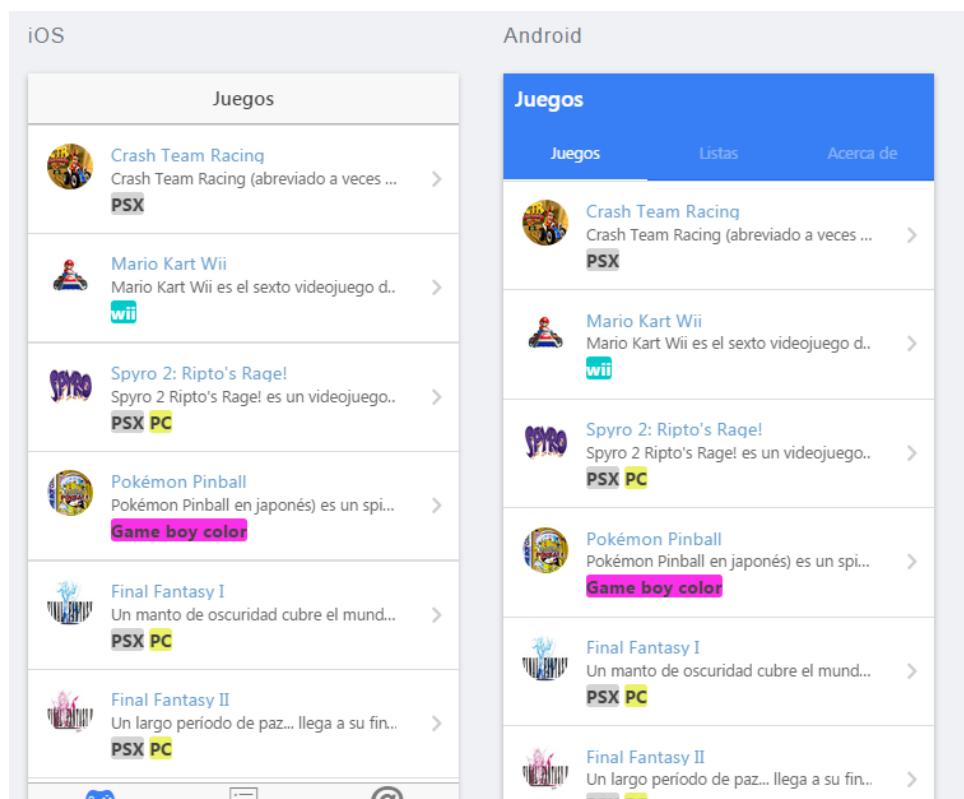


Ilustración 7 APP Móvil home

Observamos en la imagen la potencia de ionic que, con el mismo código, nos genera las versiones de ios y android. Cada una con el estilo que espera el usuario de cada plataforma.

La app tiene tres pestañas:

- La primera de ellas nos permite ver un listado de los juegos de la web. La carga es dinámica; según va bajando el usuario, el scroll carga más juegos. Pinchando sobre uno de los juegos, se nos muestran los detalles del juego al igual que en la web.

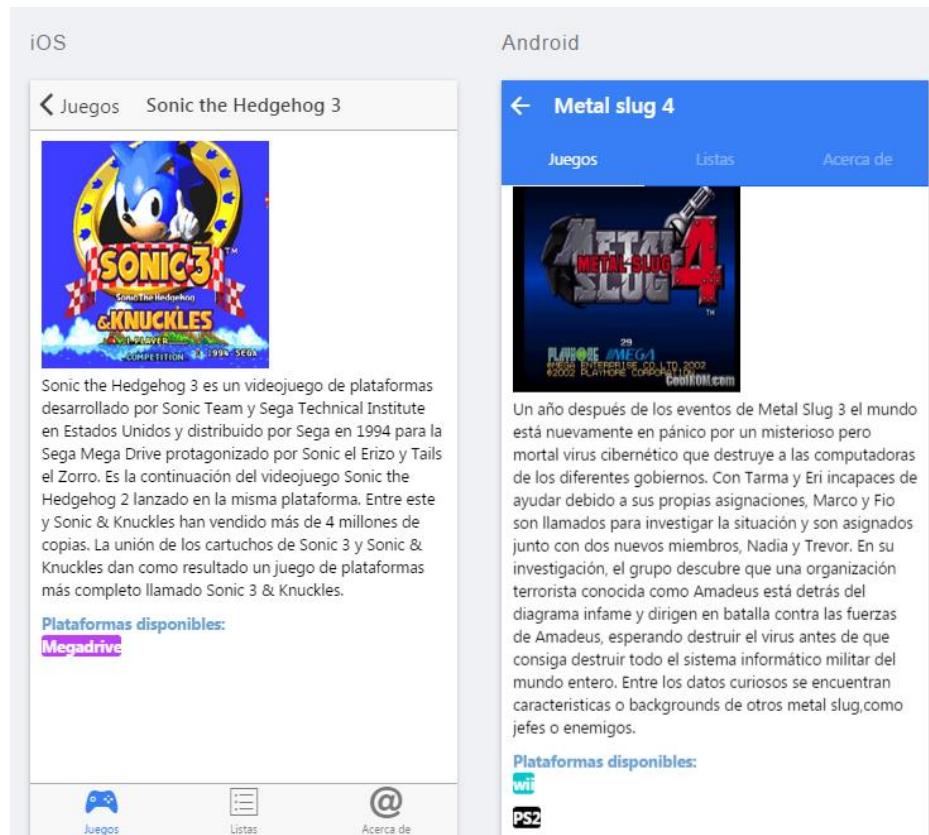


Ilustración 8 App Móvil detalles juego

- En la segunda de las pestaña, se nos muestran las listas del usuario que inició sesión. Desde esta pestaña, podemos entrar a ver qué juegos contiene y después ver los detalles del mismo. Cabe destacar que, en cada pestaña, se mantiene la navegación y se puede volver hacia atrás en todo momento.

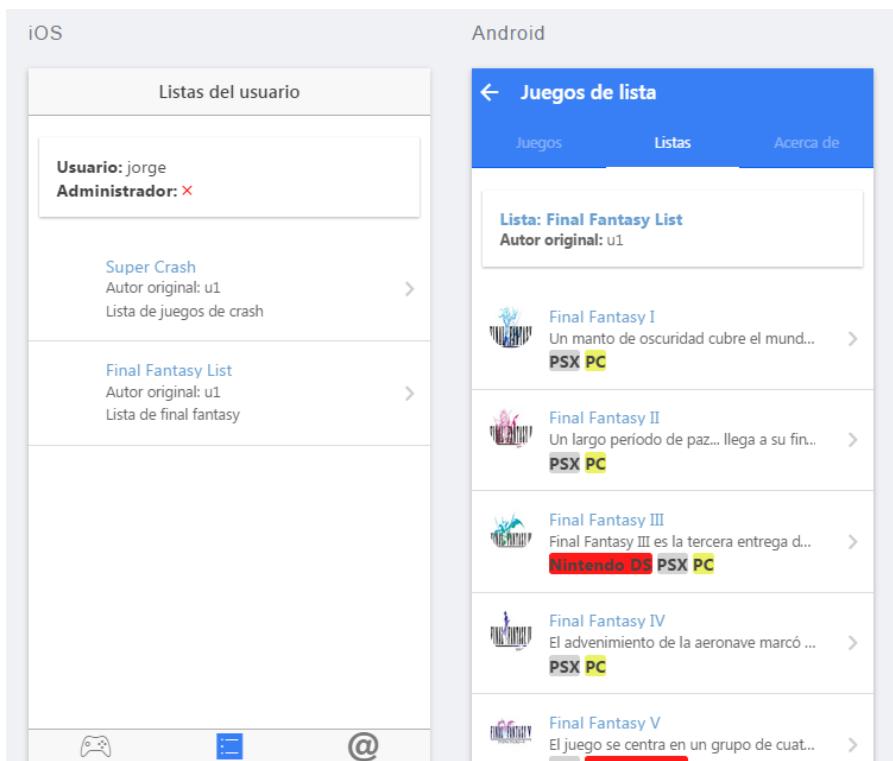


Ilustración 9 App Móvil listas

- En la última de las pestañas, nos encontramos con una tabla con datos de contacto (email y cuenta de github) y enlaces a la web de TopGames, así como a la documentación de la api:

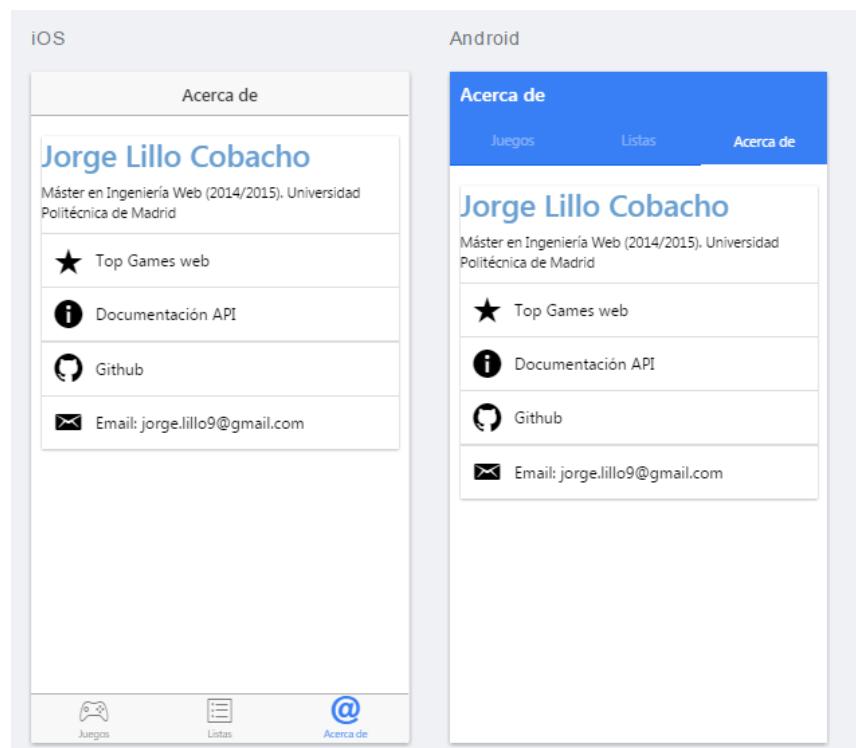


Ilustración 10 App móvil acerca de

Si pinchamos desde la versión móvil a los enlaces a la web, las abrirá desde el navegador:

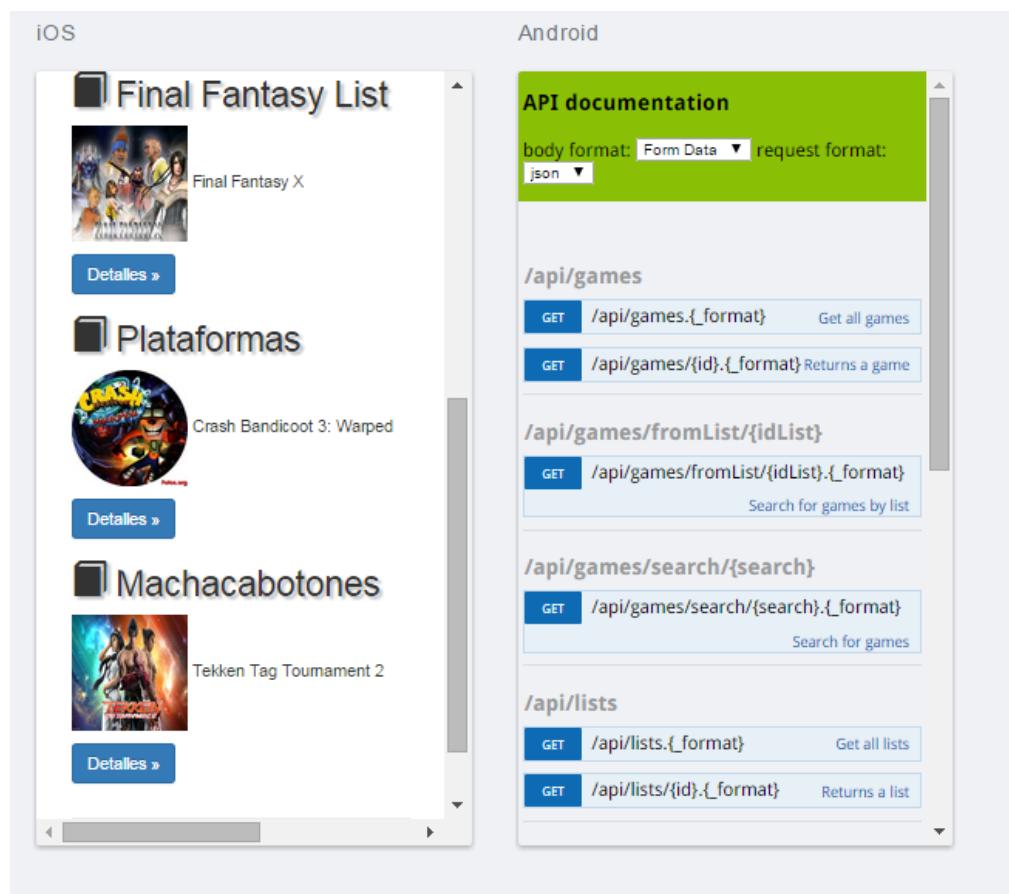


Ilustración 11 App móvil acerca de 2

2. ALCANCE DEL PROYECTO

El objetivo del proyecto es demostrar todo lo aprendido en el Máster, poder desarrollar una web y suministrar su contenido mediante una api rest a una aplicación móvil. Para compensar la falta de tiempo, se decidió que la app móvil sería un visor más sencillo; por ello, en vez de hacerla nativa con el fin de aprender algo nuevo, se tomó la decisión de usar el **framework ionic** para que la app fuese híbrida.

Esta memoria refleja todos los pasos seguidos para la realización de los tres módulos ya mencionados. Está orientada a gente con conocimientos técnicos como: framework web, patrón MVC... La mayoría de ellos se explicarán de forma breve con el fin de poder llegar a un mayor número de personas.

El proyecto se ha dividido en tres **fases**:

- Fase uno: desarrollo de la web
- Fase dos: creación de api pública.
- Fase tres: aplicación móvil.

* Fase previa: modelo de datos, diseño de mockups de las posibles pantallas, planificación de tiempo.

Cada fase se detallará en el siguiente apartado.

2.1. Conceptos básicos

(a) Framework Web

En el desarrollo de software, un framework es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado. Un framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

Por framework nos estamos refiriendo a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework se puede considerar como “una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta”. Se trata de extensiones de un lenguaje que, a través de una o más jerarquías de clases, implementan una funcionalidad y pueden ser extendidas de manera opcional.

Características básicas de los frameworks web:

- Abstracción de URLs y sesiones: no es necesario manipular directamente las URLs ni las sesiones. El framework ya se encarga de hacerlo.

- Acceso a datos: incluyen las herramientas e interfaces necesarias para integrarse con herramientas de acceso a datos, en BBDD, XML, etc...
- Controladores: la mayoría de frameworks implementan una serie de controladores para gestionar eventos como una introducción de datos mediante un formulario o el acceso a una página. Estos controladores suelen ser fácilmente adaptables a las necesidades de un proyecto concreto.
- Autentificación y control de acceso: incluyen mecanismos para la identificación de usuarios mediante login y password y permiten restringir el acceso a determinadas páginas y a determinados usuarios.
- Internacionalización
- Separación entre diseño y contenido.

El objetivo de los frameworks es hacer que nos centremos en el verdadero problema y no preocuparnos por implementar funcionalidades que son de uso común en muchas aplicaciones, como podría ser el proceso de login de usuarios o establecer la conexión con la base de datos. Por ello, cuando usamos frameworks, nuestra mente ha de centrarse en el verdadero centro del problema y hacer fluir todos los detalles “menores”, ya que seguramente el framework nos dará una solución para ellos.

Ventajas de los frameworks:

Una de las principales ventajas es la estructuración de directorios, dado que, si conocemos las reglas de estructuración, cuando necesitemos tocar algo en el código, sabremos rápidamente dónde localizar dicho código porque tendremos montada una buena jerarquía de directorios, mucho mejor de la que podríamos crearnos manualmente si no utilizáramos frameworks. No obstante, algunos frameworks dan facilidades para poder crear jerarquías propias.

Por otro lado, el código de las librerías base que aportan los frameworks está muy probado. Sería posible desarrollar de forma manual una librería con las mismas funcionalidades, pero sería complejo poder alcanzar un nivel de testeo tan grande como el de los framework. (no es necesario reinventar la rueda).

Además, hay que tener en cuenta las grandes comunidades (unas más grandes que otras) de usuarios. En estas comunidades de usuarios, algunos de ellos desarrollan módulos o extensiones para el framework y los distribuyen gratuitamente.

En cuanto al trabajo en equipo, es importante mencionar que permite que este sea más sencillo. Si todo el equipo conoce el funcionamiento del framework, todos sabrán, por ejemplo, la estructura de directorios de la aplicación y sabrán localizar con facilidad el fichero de código fuente con el que requieran trabajar. Además, si, en un futuro, es necesario incorporar más personal a la plantilla, con que conozcan el framework será suficiente.

Desventajas de los frameworks:

La principal desventaja del uso de frameworks es la curva de aprendizaje. La afirmación de que el uso de framework hace que las aplicaciones se desarrollen en menos tiempo es cierta, siempre y cuando se conozca y se tenga experiencia en el uso del framework.

En la primera aplicación que se desarrolle con un framework nuevo, seguramente será necesario invertir más tiempo que si se desarrollase sin usar ningún framework, pero, a medio/largo plazo y una vez comprendido el framework, se podrá ahorrar más tiempo.

Una vez conocido el framework, permitirá un mantenimiento de la aplicación mucho más ágil, además de que la aplicación, desde sus cimientos, será mucho más robusta que si se desarrollara sin ningún framework.

(b) Patrón MVC

Symfony está basado en un patrón clásico del diseño web conocido como arquitectura **MVC**, que está formado por tres niveles:

- El **Modelo** representa la información con la que trabaja la aplicación, es decir, su lógica de negocio.
- La **Vista** transforma el modelo en una página web que permite al usuario interactuar con ella.
- El **Controlador** se encarga de procesar las interacciones del usuario y realiza los cambios apropiados en el modelo o en la vista.

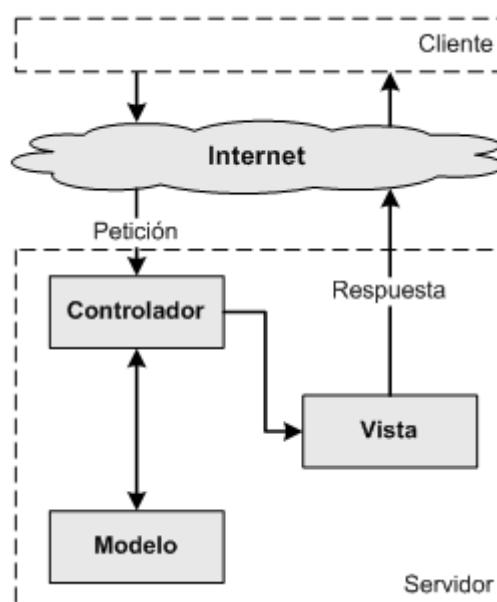


Ilustración 12 Conceptos básicos MVC

La arquitectura **MVC** separa la lógica de negocio (el modelo) y la presentación (la vista), por lo que se consigue un mantenimiento más sencillo de las aplicaciones. Si, por ejemplo, una misma aplicación debe ejecutarse tanto en un navegador estándar, como en un navegador de un dispositivo móvil, solamente es necesario crear una vista nueva para cada dispositivo; manteniendo el controlador y el modelo original. El controlador se encarga de aislar al modelo a la vista de los detalles del protocolo utilizado para las peticiones (HTTP, consola de comandos, email, etc.).

El modelo se encarga de la abstracción de la lógica relacionada con los datos, haciendo que la vista y las acciones sean independientes de, por ejemplo, el tipo de gestor de bases de datos utilizado por la aplicación.

(c) API REST

REST deriva de "**R**Epresentational State Transfer" que, traducido, vendría a ser "transferencia de representación de estado. Un servicio REST no tiene estado (es *stateless*), esto quiere decir que, entre dos llamadas cualesquiera, el servicio pierde todos sus datos, es decir, no se puede llamar a un servicio REST y pasárle unos datos (por ejemplo: un usuario y una contraseña) y esperar a que "nos recuerde" en la siguiente petición. De ahí el nombre: el estado lo mantiene el cliente y, por lo tanto, es el cliente quien debe pasar el estado en cada llamada. Si queremos que un servicio REST nos recuerde, debemos pasárle quién somos en cada llamada. Eso puede ser un usuario y una contraseña, un *token* o cualquier otro tipo de credenciales, pero tenemos que pasárlas en cada llamada. Lo mismo se aplica para el resto de información.

El no tener estado es una desventaja clara, pues tener que pasar el estado en cada llamada es, como mínimo, tedioso, pero la contrapartida es clara: escalabilidad. Para mantener un estado, se requiere algún sitio (generalmente memoria) donde guardar todos los estados de todos los clientes. A más clientes, más memoria hasta que, al final, podemos llegar a no poder admitir más clientes, no por falta de CPU, sino por falta de memoria.

SOAP VS REST

SOAP es el acrónimo de "**S**imple **O**bject **A**ccess **P**rotocol". Es el protocolo que se oculta tras la tecnología que comúnmente denominamos "Web Services" o servicios web. SOAP es un protocolo extraordinariamente complejo pensado para dar soluciones a casi cualquier necesidad en lo que a comunicaciones se refiere, incluyendo aspectos avanzados de seguridad, transaccionalidad, mensajería asegurada y demás. Cuando salió SOAP, se vivió una época dorada de los servicios web. Aunque las primeras implementaciones eran lo que se llamaban WS1.0 y no soportaban casi ningún escenario avanzado, todo el mundo pagaba el precio de usar SOAP, ya que parecía claro que era el estándar que dominaría el futuro. Con el tiempo, salieron las especificaciones WS-* que daban soluciones avanzadas, pero a la vez que crecían las capacidades de SOAP, crecía su complejidad. Al final, los servicios web SOAP terminan siendo un monstruo con muchas capacidades pero que, en la mayoría de los casos, no necesitamos.

Por su parte, REST es simple. REST no quiere dar soluciones para todo y, por lo tanto, no pagamos con demasiada complejidad una potencia que quizá no vamos a necesitar.

Una diferencia fundamental entre un servicio web clásico (SOAP) y un servicio REST es que el primero está orientado a RPC; es decir, a invocar métodos sobre un servicio remoto, mientras que el segundo está orientado a recursos; es decir, operamos sobre recursos, no sobre servicios.

En una API REST la idea de “servicio” como tal desaparece. Lo que tenemos son recursos, accesibles por identificadores (URIs). Sobre esos recursos, podemos realizar acciones generalmente diferenciadas a través de verbos HTTP distintos.

Así, en un servicio web clásico (SOAP) tendríamos un servicio llamado TestService, que tendría un método llamado GetAll, que nos devolvería todas las test. La idea, independientemente de la tecnología usada para consumir el servicio web, es que se llama a un método (GetAll) de un servicio remoto (TestService). Del mismo modo, para obtener un test en concreto llamaríamos al método GetById() pasándole el id de la test. De ahí que se diga que están orientados a RPC (Remote Procedure Call – Llamada a método remoto).

Por su parte en un servicio REST la propia idea de servicio se desvanece. En su lugar, nos queda la idea de un “recurso”, llamémosle “Colección de test” que tiene una URI que lo identifica, p. ej. /Tests. Así, si invocamos dicha URI, debemos obtener una representación de dicho recurso, es decir, es necesario obtener el listado de todas los test.

REST	SOAP
REpresentational State Transfer	Service Oriented Architecture Protocol
Architecture Style	Protocol
Uses simple HTTP protocol	Uses SOAP envelop and then HTTP (or FTP/SMTP etc) to transfer the data
Supports many different data format like JSON, XML, YAML etc	Supports only XML format
Performance & Scalability, Caching	Slower performance & scalability is bit complex, Caching not possible
Widely & Frequently used	Used where REST is not possible. Provides WS-* features

Ilustración 13 REST VS SOAP

(d) ORM

Las bases de datos siguen una estructura relacional, mientras que PHP y **Symfony** están orientados a objetos. Por este motivo, para acceder a la base de datos como si fuera orientada a objetos, es necesario una interfaz que traduzca la lógica de los objetos a la lógica relacional. Esta interfaz se denomina "mapeo de objetos a bases de datos" (ORM, de sus siglas en inglés "object-relational mapping"). Un **ORM** consiste en una serie de objetos que permiten acceder a los datos y que contienen en su interior cierta lógica de negocio.

Una de las ventajas de utilizar estas capas de abstracción de objetos/relacional es que evita utilizar una sintaxis específica de un sistema de bases de datos concreto. Esta capa transforma automáticamente las llamadas a los objetos en consultas SQL optimizadas para el sistema gestor de bases de datos que se está utilizando en cada momento.

De esta forma, es muy sencillo cambiar a otro sistema de bases de datos completamente diferente en mitad del desarrollo de un proyecto. Estas técnicas son útiles por ejemplo cuando se debe desarrollar un prototipo rápido de una aplicación y el cliente aún no ha decidido el sistema de bases de datos que más le conviene. El prototipo se puede realizar utilizando SQLite y después se puede cambiar fácilmente a MySQL, PostgreSQL u Oracle cuando el cliente se haya decidido. El cambio se puede realizar modificando solamente una línea en un archivo de configuración.

La capa de abstracción utilizada encapsula toda la lógica de los datos. El resto de la aplicación no tiene que preocuparse por las consultas. Los desarrolladores especializados en la programación con bases de datos pueden localizar fácilmente el código.

Utilizar objetos en vez de registros y clases en vez de tablas tiene otra ventaja: se pueden definir nuevos métodos de acceso a las tablas; por ejemplo, si se dispone de una tabla llamada 'Cliente' con dos campos: Nombre y Apellido, puede que sea necesario acceder directamente al nombre completo (NombreCompleto). Con la programación orientada a objetos, este problema se resuelve añadiendo un nuevo método de acceso a la clase 'Cliente' de la siguiente forma:

```
public function getNombreCompleto()
{
    return $this->getNombre() . ' ' . $this->getApellido();
}
```

Todas las funciones comunes de acceso a los datos y toda la lógica de negocio relacionada con los datos se pueden mantener dentro de ese tipo de objetos, por ejemplo, la siguiente clase ‘CarritoCompra’ almacena los productos (que son objetos). Para obtener el precio total de los productos del carrito y así realizar el pago, se puede añadir un método llamado `getTotal()` de la siguiente forma:

```
public function getTotal()
{
    $total = 0;
    foreach ($this->getProductos() as $producto)
    {
        $total += $producto->getPrecio() * $item->getCantidad();
    }
    return $total;
}
```

Gracias a este método, es posible controlar los valores devueltos desde el propio objeto. Si más adelante se quiere aplicar un descuento que afecta al precio total, las modificaciones necesarias se pueden añadir directamente en el método `getTotal()` o incluso en los métodos `getPrecio()` de cada producto.

Symfony soporta los dos ORM libres más populares de PHP: Propel y Doctrine. Symfony se integra perfectamente con los dos. Lo único que se debe hacer es elegir o Propel o Doctrine al crear un nuevo proyecto Symfony.

3. DESARROLLO DE LA SOLUCIÓN

3.1. Fase previa

En esta fase trataremos todos los cimientos de la aplicación, desde el modelo de datos hasta los primeros mockups de la web y la app.

(a) Modelo de datos

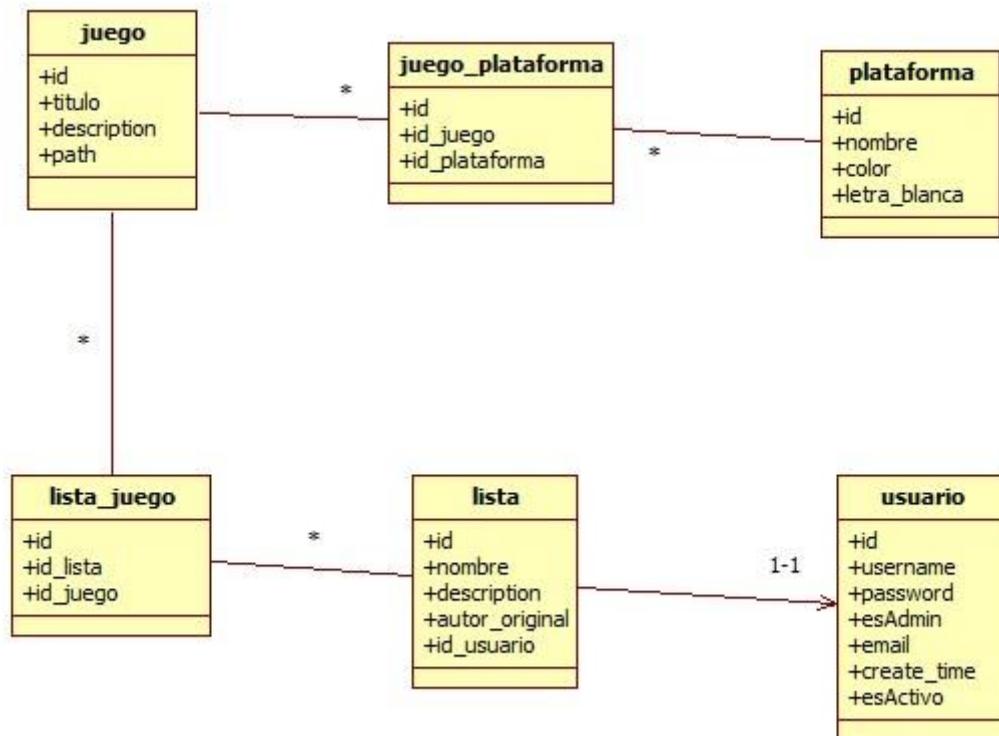


Ilustración 14 Modelo de datos

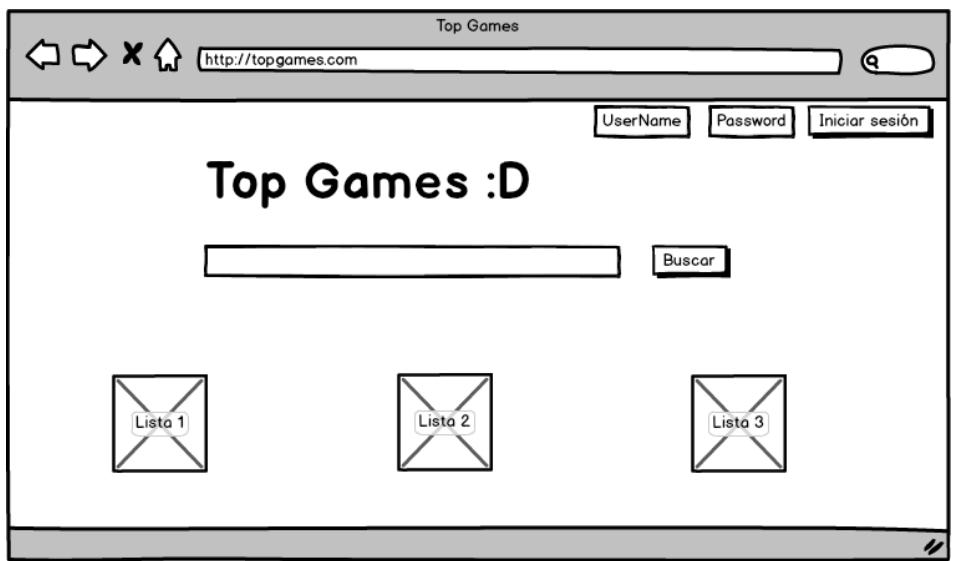
Contamos con seis tablas en la base de datos:

- **Juego**: contiene el título del juego y la descripción. Además, guarda el path de la imagen que le asociamos. Con el nombre que guardamos, podemos cargar la imagen que le asociamos al juego. (Guardamos las imágenes en una carpeta concreta del proyecto web\uploads\documents)

- **Lista:** contiene nombre, descripción y nombre del usuario original que la creó. Además, contiene el id como clave ajena a la tabla usuario con una relación 1x1 (una lista pertenece a un usuario)
- **Plataforma:** tiene el nombre, el color en hexadecimal para poner de fondo y, si queremos, color de la letra blanca.
- **Usuario:** tiene nombre de usuario, password (codificado), si es o no administrador, el email, la fecha de creación y si están activos o no.
- Tablas para relacionar elementos:
 - **ListaJuego:** enlaza listas con sus respectivos juegos, relación NxM, cada lista puede tener N juegos
 - **JuegoPlataforma:** enlaza a un juego sus plataformas relación NxM.

(b) Mockups

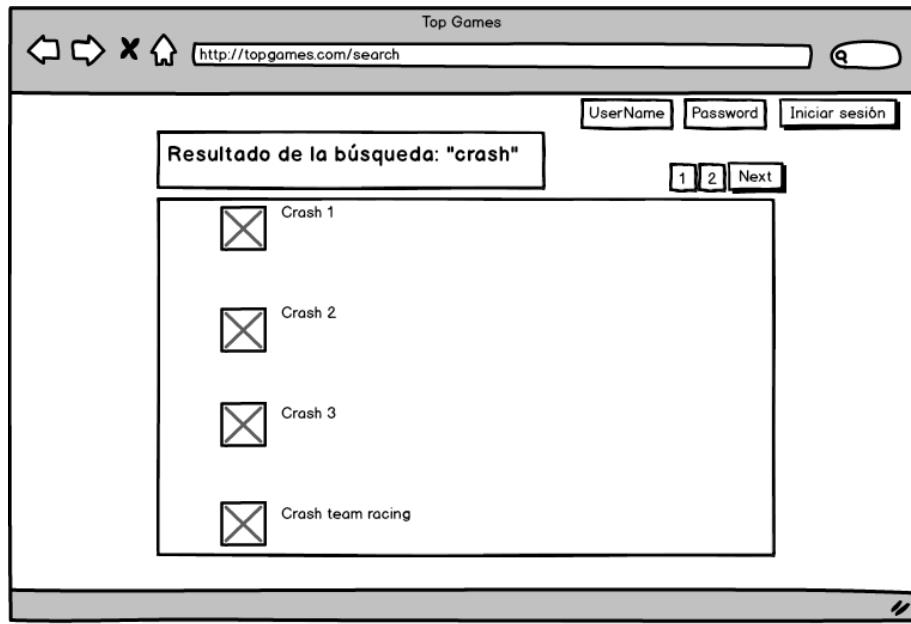
- **Web:**
 - **Home:** pantalla de inicio, formulario de login, formulario de búsquedas, listas aleatorias con un juego aleatorio.



Created with Balsamiq - www.balsamiq.com

Ilustración 15 Mockup Home

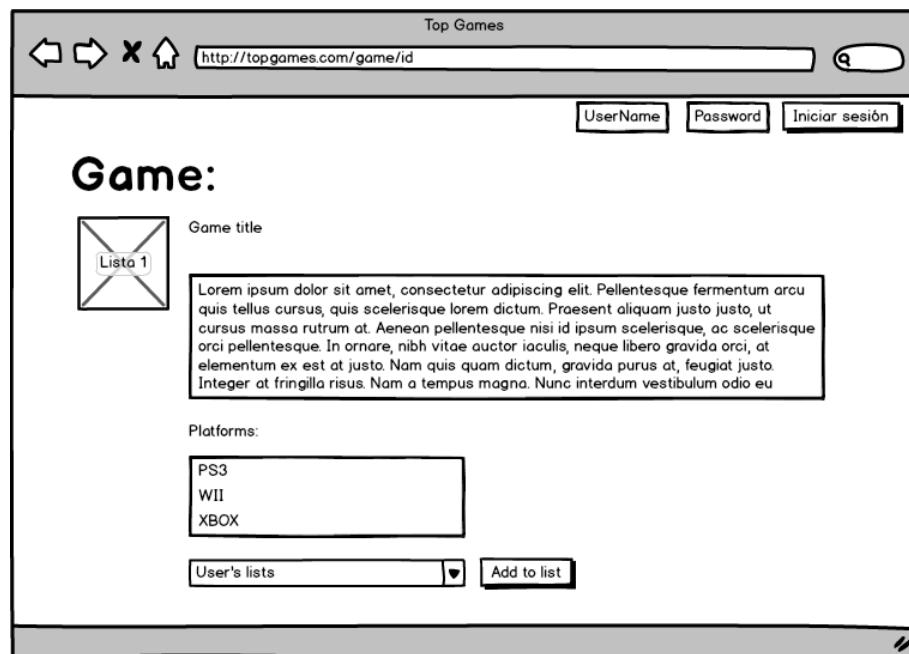
- **Buscador:** buscador de listas y juegos. Si se encuentran más de diez resultados, se mostrarán los resultados paginados.



Created with Balsamiq - www.balsamiq.com

Ilustración 16 Mockup search

- **Detalles de juego:** se muestra la portada del juego, su descripción y plataformas asociadas. Además, si el usuario se encuentra registrado, podrá añadirlo a alguna de sus listas.



Created with Balsamiq - www.balsamiq.com

Ilustración 17 Mockup game detail

- **Detalles lista:** nos muestra los detalles de la lista, así como los juegos asociados a ella. Además, si se trata de una lista que no es del propio usuario que inició sesión, se da la posibilidad de clonarla para que quede asociada al usuario.

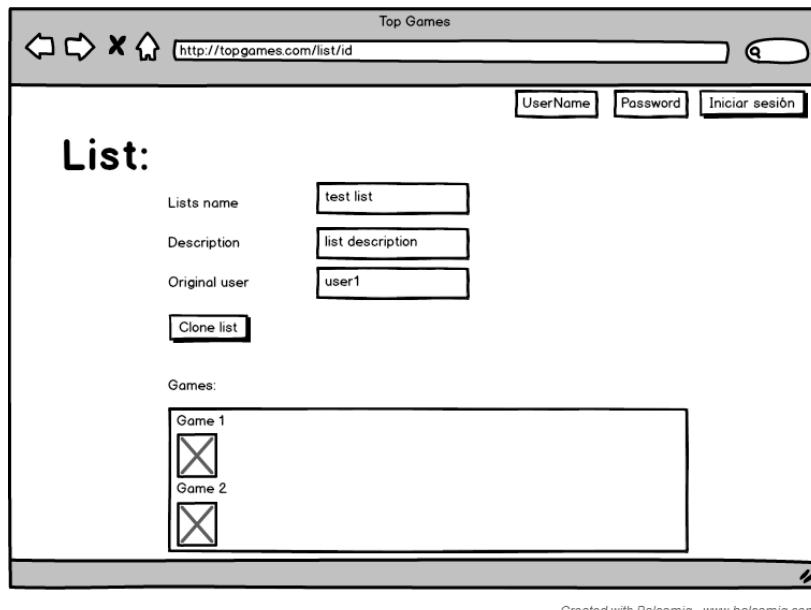


Ilustración 18 Mockup list detail

El resto de pantallas serán las típicas de administración, creación, edición y borrado de cada uno de los elementos de la aplicación. (Usuarios, Juegos y Listas)

- **App:**
 - **Login:** pantalla de inicio de sesión

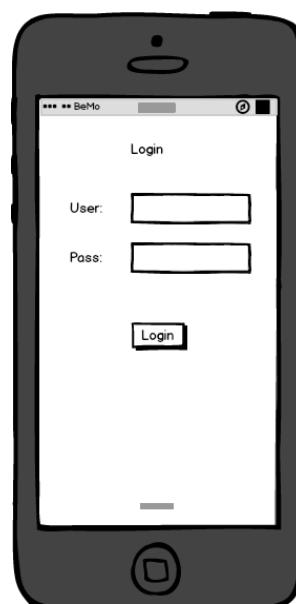
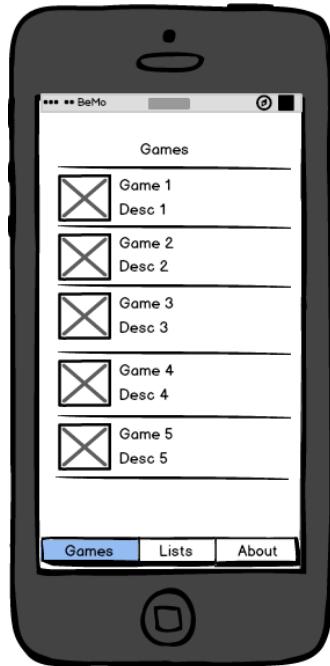


Ilustración 19 Mockup app login

- **Home (Games):** primera pestaña, listado de juegos. Al hacer scroll, solicita más a la api y los carga de manera dinámica:

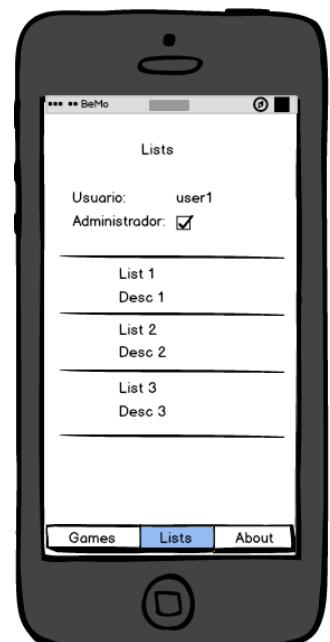


Created with Balsamiq - www.balsamiq.com

Ilustración 20 Mockup app games

Pichando sobre uno de los juegos, llegamos a la vista de detalle de juego.

- **Detalle listas:** muestra el usuario, si es administrado o no, y un listado de las listas asociadas al usuario.

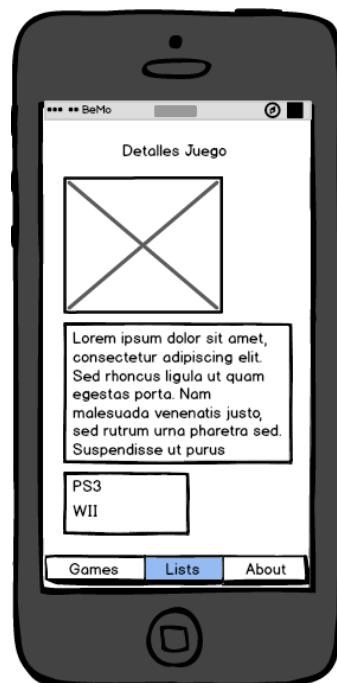


Created with Balsamiq - www.balsamiq.com

Ilustración 21 Mockups app list

Pinchando sobre una de las listas, nos muestra un listado de los juegos similar al de la primera pantalla de Juegos. Al pinchar sobre uno de ellos, nos lleva a la pantalla de detalles de juego.

- **Detalles de juego:** pantalla de detalles de juego.



Created with Balsamiq - www.balsamiq.com

Ilustración 22 Mockup app game details

- **Acerca de:**



Created with Balsamiq - www.balsamiq.com

Ilustración 23 Mockup app about

(c) Diagramas

- Casos de uso:

- **Usuarios no registrados** pueden:

- Buscar lista.
 - Buscar juego.
 - Detalles de listas.
 - Detalles de juego.

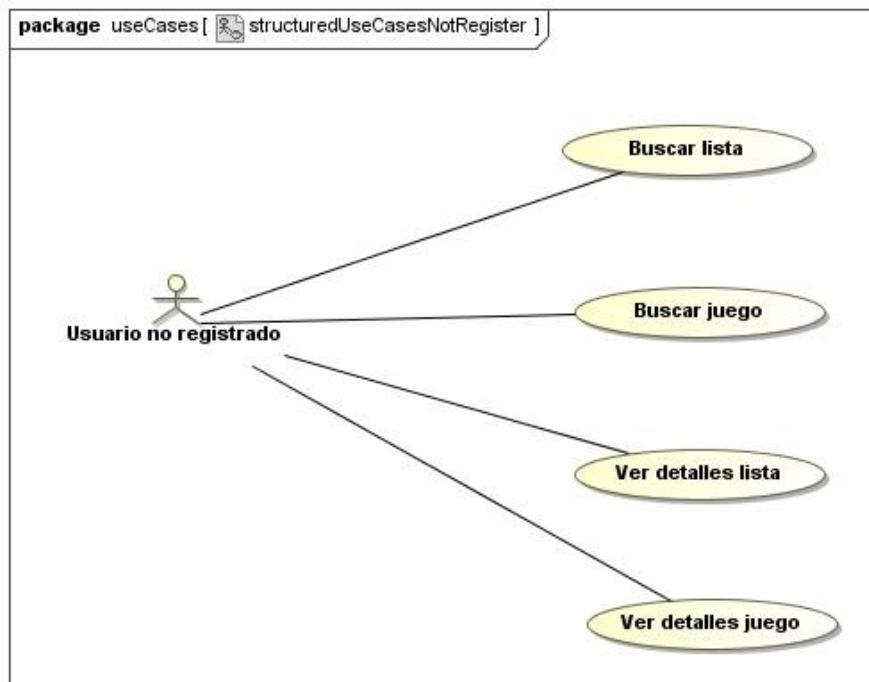


Ilustración 24 Casos de uso usuario no registrado

- **Usuarios registrados** pueden:

- Lo mismo que uno no registrado.
 - Añadir juegos a lista.
 - Eliminar juegos de lista.
 - Crear listas.
 - Editar listas.
 - Eliminar listas.
 - Clonar listas: si no es su propia lista.

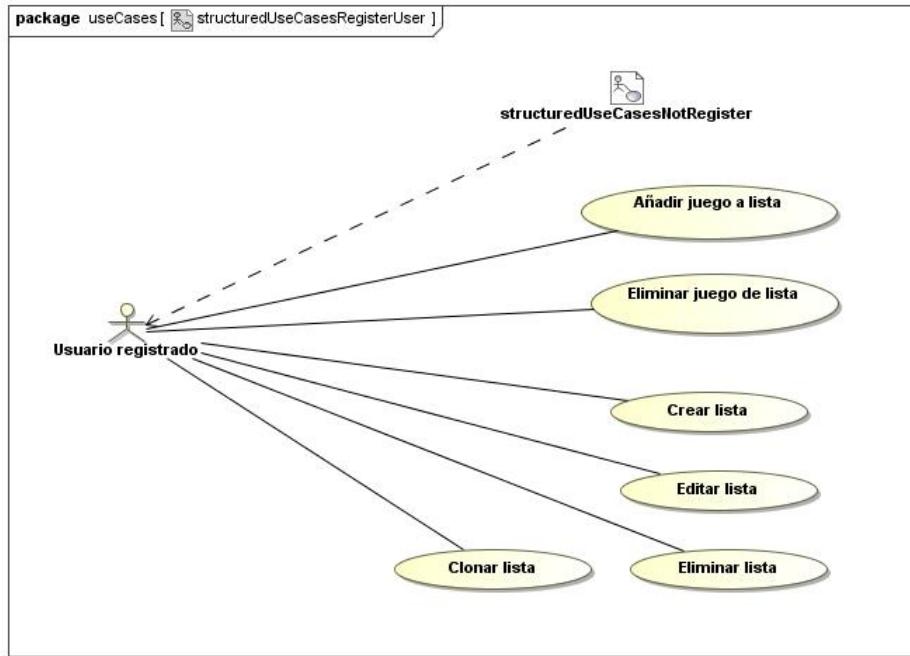


Ilustración 25 Casos de uso usuario registrado

- **Administradores** pueden:

- Lo mismo que un usuario registrado.
- Crear juego.
- Editar juego.
- Eliminar juego.
- Añadir plataforma a juego.
- Quitar plataforma a juego.
- Añadir usuario.
- Eliminar usuario.
- Modificar usuario.

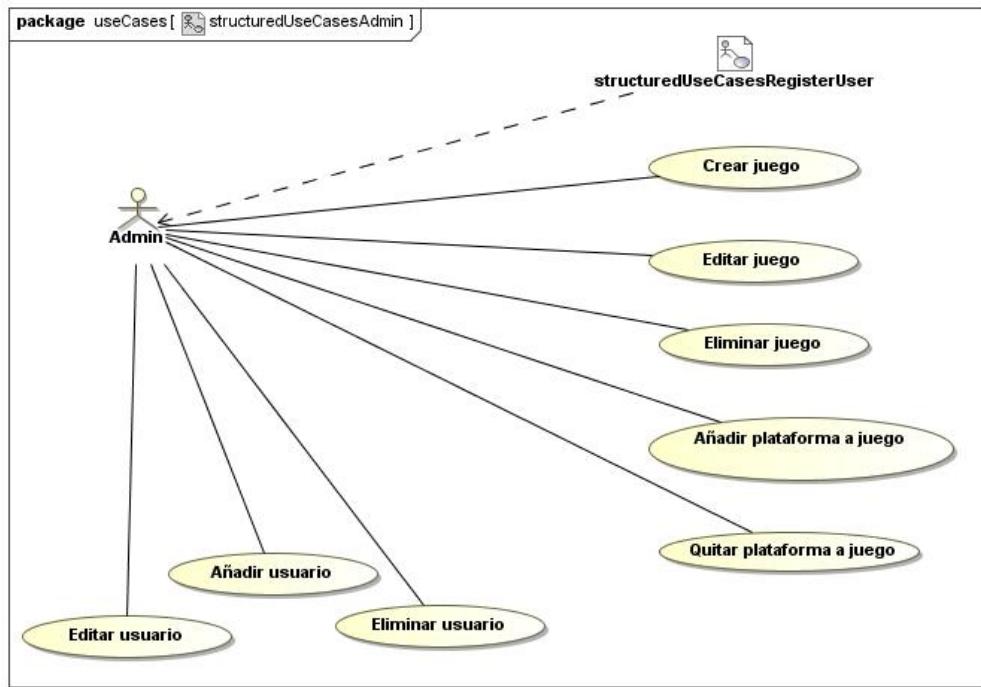


Ilustración 26 Casos de uso admin

- Diagramas de estado:
 - BuscarSpecification: diagrama de estado del caso de uso buscar

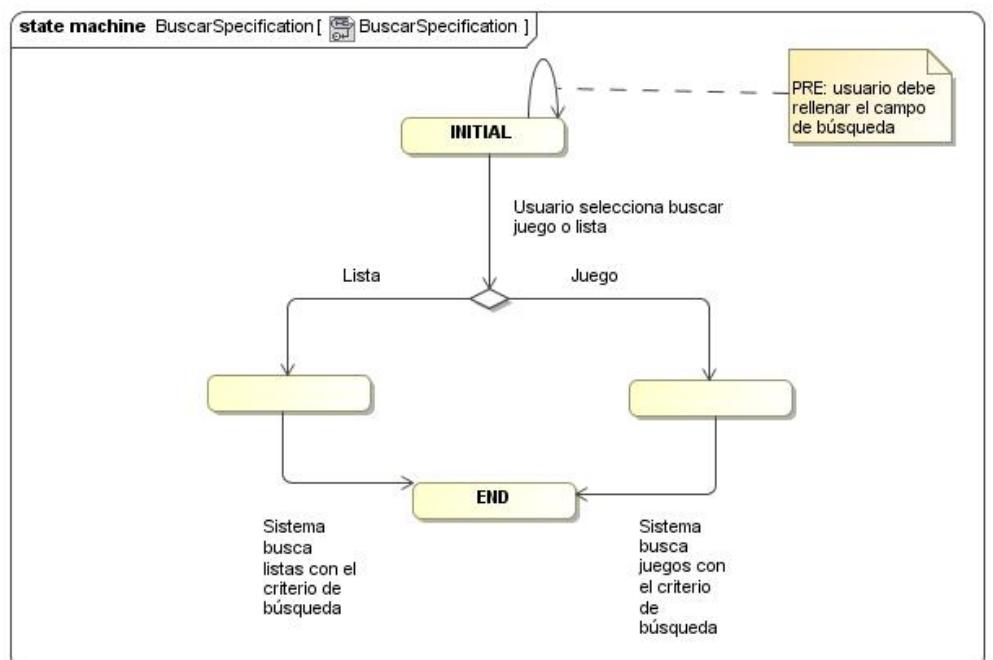


Ilustración 27 Diagrama de estado buscar

(d) Planificación

El inicio del proyecto: **15 de mayo**

Fin del proyecto (máximo): **2 de julio**

Mayo

- 4- 10: planificación del proyecto, modelo de datos, requisitos...
- 11-17: web: proyecto base, subida a github, sistema de usuarios y crud de juegos
- 18 - 24: web: imágenes para juegos, asociación con plataformas
- 25- 31: web: buscador en home, crud listas, añadir juegos a listas, clonar

Junio

- 1-7: pantalla home, revisión bugs
- 8-14: desarrollo de la api + mirar ionic framework
- 15-21: desarrollo de app
- 22-28: desarrollo de app + juntar datos para memoria

Julio

- 29- 2: revisión y entrega

3.2. Fase uno: desarrollo web

(a) Symfony 2

La tecnología usada para la construcción de la web es **Symfony 2** framework, tecnología estudiada en el máster. **Symfony** es un completo framework PHP diseñado para optimizar el desarrollo de las aplicaciones web. Está basado en el patrón Modelo Vista Controlador y separa la lógica de negocio, la lógica de servidor y la presentación de la aplicación web. Además, proporciona varias herramientas y clases encaminadas a reducir el tiempo de desarrollo de una aplicación web compleja. Por otro lado, automatiza las tareas más comunes, permitiendo al desarrollador dedicarse por completo a los aspectos específicos de cada aplicación. El resultado de todas estas ventajas es que no se debe reinventar la rueda cada vez que se crea una nueva aplicación web.

Symfony fue diseñado para ajustarse a los siguientes requisitos:

- Es fácil de instalar y configurar en la mayoría de plataformas (y con la garantía de que funciona correctamente en los sistemas Windows y *nix estándares).
- Es independiente del sistema gestor de bases de datos. Su capa de abstracción y el uso de ORM (**Doctrine 2**, Propel) permiten cambiar con facilidad de SGBD en cualquier fase del proyecto.
- Utiliza programación orientada a objetos y características como los espacios de nombres, de ahí que sea imprescindible **PHP 5.3**.
- Es sencillo de usar en la mayoría de casos, aunque es preferible para el desarrollo de grandes aplicaciones Web más que para pequeños proyectos.

- Aunque utiliza **MVC** (Modelo Vista Controlador), tiene su propia forma de trabajo con variantes del MVC clásico como la capa de abstracción de base de datos, el controlador frontal y las acciones.
- Está basado en la premisa de “convenir en vez de configurar”, en la que el desarrollador sólo debe configurar aquello que no es convencional.
- Sigue la mayoría de las mejores prácticas y patrones de diseño para la web.
- Está preparado para aplicaciones empresariales y se adapta a las políticas y arquitecturas propias de cada empresa, además de ser lo suficientemente estable como para desarrollar aplicaciones a largo plazo.
- Su código es fácil de leer, ya que incluye comentarios de phpDocumentor y permite un mantenimiento muy sencillo.
- Es fácil de extender, lo que permite su integración con las bibliotecas de otros fabricantes.
- Consta de una potente línea de comandos que facilita la generación de código, lo cual contribuye a ahorrar tiempo de trabajo.

Características para el desarrollo automatizado de proyectos web

- Permite la internacionalización para la traducción del texto de la interfaz, los datos y el contenido de localización.
- La presentación usa templates y layouts que pueden ser construidos por diseñadores de HTML que no posean conocimientos del framework.
- Los formularios soportan la validación automática, lo cual asegura una mejor calidad de los datos en la base de datos y una mejor experiencia para el usuario.
- El manejo de caché reduce el uso de banda ancha y la carga del servidor.
- La facilidad de soportar autenticación y credenciales facilita la creación de áreas restringidas y el manejo de seguridad de los usuarios.
- El enrutamiento y las URLs inteligentes hacen ‘amigable’ las direcciones de las páginas de la aplicación.
- Las listas son más ‘amigables’, ya que permiten la paginación, clasificación y filtraje automáticos.
- Los pluggins proveen un alto nivel de extensibilidad.
- La interacción con AJAX es mucho más sencilla.

Inconvenientes:

Como no todo son ventajas, hay que tener presente que la curva de aprendizaje no es trivial. Tendremos que estar familiarizados con patrones y conceptos como: Programación orientada a objetos (OOP), Modelo Vista Controlador (MVC), Inyección de dependencias (DI), Mapeador de objetos relacional (ORM), herramientas como Composer

Trabajar con Doctrine como ORM es muy productivo (desde el punto de vista del desarrollador). El poder acceder a las entidades hijas/padre de cualquier entidad y que se carguen solo cuando se accede a ellas, que se actualicen o borren en cascada, sin escribir nada de código (solo configurando los archivos .orm) es una maravilla, pero, para que funcione correctamente, se requiere un gran esfuerzo hasta que se consiguen definir las relaciones 1 a N, y N a M (y dónde utilizar mappedBy e inverseBy).

Además, si el modelo de base de datos ya lo tenemos definido antes de empezar, nos podemos encontrar con que a Doctrine no le gusta cómo se han creado las claves primarias o las relaciones entre entidades y nos obligará a hacer cambios en el modelo.

En el entorno de producción, exige que tengamos instalado APC, OPCache, o algún otro acelerador de código PHP. Esto es bueno para nuestra aplicación, puesto que nos está obligando a tener una máquina donde va a rodar rápido, pero, por contra, la instalación se hace más compleja y necesitamos tener cierto control sobre el Hosting donde la vamos a hospedar. Esto hay que tenerlo en cuenta antes de contratar uno.

Por otro lado, si preveemos que vamos a tener un tráfico elevado, lo mejor será que tengamos presente desde el principio el cacheado en memoria de las variables de sesión, consultas frecuentes a base de datos con baja variabilidad, renderizado de páginas enteras, etc. La integración de Symfony con Memcached y Redis es buena, pero, al igual que ocurría en el caso anterior, necesitamos que nuestro Hosting lo tenga disponible.

[Ver anexo 1 manual de instalación symfony](#)

Seguridad que aporta Symfony:

Autorización y autenticación:

En este apartado vamos a conocer con más detalle las ventajas que nos proporciona el componente **Security** que viene con Symfony2, el cual nos ofrece un completo sistema de seguridad web con distintas funcionalidades para la autenticación de usuarios a través de autenticación HTTP, a través de formularios o empleando certificados. Asimismo, es extensible para poder implementar nuestras propias estrategias de autenticación. Además de la autenticación, nos propone un concepto de autorización basado en roles y un sistema avanzado de ACL (Listas de control de Acceso).

La seguridad se basa en un proceso de dos etapas básicas: la **autenticación** y la **autorización**.

- En la **primera etapa**, el sistema de seguridad debe identificar quién es el usuario que accede a la aplicación web, obligando a presentar algún tipo de identificación, como puede ser un formulario web, una autenticación HTTP, un certificado x509 o un método personalizado. Una vez que el sistema sabe de quién se trata, se pasa a la siguiente etapa.
- En la **segunda etapa**, se valida que el usuario que se ha identificado tenga los privilegios suficientes para acceder al recurso seleccionado. Symfony propone esta autorización a través de URL protegidas por roles, protección de objetos y métodos o empleando listas de control.

Para este proceso, debemos conocer el uso de los firewall. El firewall de symfony actúa justo en el proceso de autenticación. Este cortafuegos determina si la URL solicitada está protegida como una zona segura, comprobando si el usuario está autenticado para mandar la petición al proceso de autorización. (Symfony posee un fichero de configuración en **app/config/security.yml** donde se puede configurar todos estos conceptos).

En el apartado firewalls, podremos determinar áreas protegidas, a través de un patrón que represente qué URLs solicitan un proceso de autenticación además de determinar qué tipo de autenticación se solicitará. En este ejemplo, se solicita una autenticación http básica:

```
security:  
    firewalls:  
        secured_area:  
            pattern:  ^/  
            anonymous: ~  
            http_basic:  
                realm: "Secured Demo Area"
```

Sin embargo, podríamos haber configurado esta zona con autenticación con formulario, determinando cuál es la path de login y el controlador que se encarga de la autenticación.

```
secured_area:  
    pattern:  ^/  
    anonymous: ~  
    form_login:  
        login_path: login  
        check_path: login_check
```

Ambas configuraciones protegerían todos los recursos, ya que el patrón determina que se protegerían todos los recursos que cuelgan de la url raíz /. Una vez que este usuario esté autenticado, el cortafuego pasaría las peticiones directamente sin solicitar continuamente la autenticación. Además, se permite el acceso de usuarios anónimos a través del parámetro anonymous para que las zonas no protegidas puedan ser accedidas por éstos sin autenticación.

Una vez ha pasado el cortafuego, se realiza el proceso de autorización. Si un usuario solicita un recurso, el sistema de seguridad debe acceder al apartado de **access_control** el cual determina qué patrones de URL son permitidos para qué rol de usuario. Los roles de usuario son la gran base para todo el proceso de autorización. En este ejemplo, podemos ver una regla de control de acceso:

```
access_control:  
    - { path: ^/admin, roles: ROLE_ADMIN }
```

Todas las rutas que cuelguen directamente de /admin estarían protegidas sólo para usuarios autenticados que posean el rol de ROLE_ADMIN

SQL Injection

Symfony2 viene, por defecto, integrado con el ORM **Doctrine**, una librería cuyo objetivo es dar herramientas para facilitar la persistencia de los datos de la aplicación a través de un mapeo entre los objetos y la estructura relacional. La gran ventaja es que está desacoplado de Symfony y su empleo es

opcional. Además, permite el cambio de base de datos totalmente transparente, pudiendo cambiar el SGBD de mysql a postgres o a sql server de una manera transparente para la aplicación; ya que el framework viene integrado por defecto con esta herramienta, vamos a conocer cómo ésta se protege frente a los posibles ataques y qué conceptos de seguridad maneja.

La base de datos proporciona diversas formas de consulta. Una de ellas es una API llamada **QueryBuilder**, que permite la construcción de las consultas con una API sencilla PHP. También podemos construir las consultas con un lenguaje propio parecido a SQL, llamado **DQL**, Doctrine Query Language, el cual nos permite hacer las queries como si de objetos se trataran. La tercera manera de consultar es de una manera nativa, empleando la conexión a la base de datos proporcionándonos la posibilidad de crear consultas preparadas.

Como hemos visto en la asignatura, el problema más peligroso respecto a la base de datos es la posibilidad de realizar inyecciones SQL que permitan acceder a contenido no permitido. Por ello, Doctrine recomienda unas buenas prácticas para evitar este tipo de ataques:

Doctrine propone una parte de su api la cual garantiza que se encuentra segura, salvo ataques de SQL Inyection, aunque, en general, no se debería de asumir para las entradas de los usuarios.

Doctrine propone la siguiente parte de su API, que es totalmente segura:

```
Doctrine\DBAL\Connection#insert($table, $values, $types)
Doctrine\DBAL\Connection#update($table, $values, $where, $types)
Doctrine\DBAL\Connection#delete($table, $where, $types)
Doctrine\DBAL\Query\QueryBuilder#setFirstResult($offset)
Doctrine\DBAL\Query\QueryBuilder#setMaxResults($limit)
Doctrine\DBAL\Platforms\AbstractPlatform#modifyLimitQuery($sql, $limit, $offset)
```

Sin embargo, no todos sus métodos son seguros, por lo que propone validar los valores de entradas de los usuarios antes de lanzarlos contra su API. Por ejemplo, el ORM propone evitar la concatenación de strings, tanto en las consultas SQL como consultas con su lenguaje DQL, ya que un atacante podría inyectar cualquier valor en una variable pasada por get. Esta técnica se recomienda totalmente con SQL y parcialmente con DQL, puesto que no garantiza, a pesar de tener ciertas implicaciones de seguridad, un 100% de fiabilidad al construir las consultas DQL. Por ejemplo, propone la siguiente consulta en DQL que sería vulnerable:

```
$dql = "SELECT u FROM User u WHERE u.username = " . $_GET['username'] . "";
```

En este escenario un atacante podría inyectar una inyección SQL, que creará una construcción DQL válida. Aunque DQL tiene sus métodos de protección, no garantiza que un atacante pueda añadir literales válidos que no sean detectados por el parseador del lenguaje, por lo que deja la responsabilidad total al programador

Otra recomendación de Doctrine es el empleo de sentencias preparadas. Doctrine propone un sistema de preparación de sentencias tanto en SQL como en DQL. En lugar de emplear la concatenación de parámetros, esta API propone especificar marcadores tanto de posición como por nombre para pasar los parámetros a nuestras sentencias.

Validación de datos y formularios:

El empleo de formularios HTML en cualquier aplicación web es una tarea común y una de las tareas más importantes para un desarrollador web. Symfony 2 nos proporciona el componente **Form**, que nos facilita el empleo de la construcción de estos formularios. Symfony2 nos proporciona la creación de distintos tipos de campos que, por defecto, poseen validaciones concretas para cada tipo y, lo más importante, podemos construir nuestras propias validaciones y tipos de campos.

Por defecto, podemos emplear campos de texto para texto plano, para campos que representan emails, valores enteros, valores monetarios, números, contraseñas, porcentajes, urls.... Todos estos campos nos proporcionan su propia validación, impidiendo pasar datos que no correspondan con los patrones comunes. Además, también tenemos campos para representar fechas, horas, idiomas.... La gran ventaja de estos tipos de campos es que nos proporciona validación y filtrado para evitar mandar en los formularios cualquier tipo de ataque malicioso.

Además de los tipos de campos descritos para cualquier propiedad del formulario, podemos añadir distintos tipos de validaciones predefinidas o creadas por el propio usuario. El componente **Validation** nos proporciona esta funcionalidad integrado con el componente Form, aunque se puede emplear independientemente para cualquier dato o clase. El formulario siempre procesa el submit y analiza todas estas restricciones con la función **\$form->isValid()**.

Es importante la validación de los datos enviados, ya que después pueden ser escritos en base de datos, en el propio HTML o mandándose a través de un web service. Esta comprobación de la validación provee una implementación sencilla y transparente.

Por defecto, Symfony nos proporciona validaciones para comprobar si los datos enviados están vacíos, si son nulos, si son valores booleanos, si el valor corresponde con una IP, una URL, la longitud del dato, si cumple una expresión regular, si se encuentra en un rango numérico, validaciones de comparación, validaciones de fechas, validaciones para ficheros, validaciones numéricas para monedas, cuentas bancarias o identificadores como el ISBN.

Protección csrf

Cross Site Request Forgery es un método por el cual un usuario malicioso intenta que un usuario legítimo, sin saberlo, realice una acción sin su intención. Por ejemplo, enviar un formulario que realice una acción concreta.

Una solución para validar los formularios es el empleo de un token único por cada creación de formulario. **Symfony** nos proporciona esta solución, ya implementada, para proteger nuestros formularios. Por defecto, en todos los formularios creados a través del componente Form incluyen este campo oculto CSRF token para validar el formulario. Esto es una gran ventaja, puesto que automáticamente incluye y valida este campo sin tener que hacer nada.

Sistema Bundle:

Un bundle es un concepto similar al de los plugins en otras aplicaciones, pero todavía mejor. La diferencia clave es que en **Symfony2** todo es un bundle, incluyendo tanto la funcionalidad básica de la plataforma, como el código escrito para tu aplicación.

Los bundles son la parte más importante de **Symfony2**. Permiten utilizar funcionalidades construidas por terceros o empaquetar nuestras propias funcionalidades para distribuirlas y reutilizarlas en otros proyectos. Además, facilitan mucho la activación o desactivación de determinadas características dentro de una aplicación.

Un **bundle** simplemente es un conjunto estructurado de archivos que se encuentran en un directorio y que implementan una sola característica. Podemos crear, por ejemplo, un BlogBundle, un ForoBundle o un bundle para gestionar usuarios (muchos de ellos ya existen como bundles de software libre). Cada directorio contiene todo lo relacionado con esa característica, incluyendo archivos PHP, plantillas, hojas de estilo, archivos Javascript, tests y cualquier otra cosa necesaria.

Las aplicaciones Symfony se componen de bundles, tal como se define en el método `registerBundles()` de la clase AppKernel:

```
// app/AppKernel.php

public function registerBundles()
{
    $bundles = array(
        new Symfony\Bundle\FrameworkBundle\FrameworkBundle(),
        new Symfony\Bundle\SecurityBundle\SecurityBundle(),
        new Symfony\Bundle\TwigBundle\TwigBundle(),
        new Symfony\Bundle\MonologBundle\MonologBundle(),
        new Symfony\Bundle\SwiftmailerBundle\SwiftmailerBundle(),
        new Symfony\Bundle\DoctrineBundle\DoctrineBundle(),
        new Symfony\Bundle\AsseticBundle\AsseticBundle(),
        new Sensio\Bundle\FrameworkExtraBundle\SensioFrameworkExtraBundle(),
    );
}
```

```

);

if (in_array($this->getEnvironment(), array('dev', 'test'))) {

    $bundles[] = new Acme\DemoBundle\AcmeDemoBundle();

    $bundles[] = new Symfony\Bundle\WebProfilerBundle\WebProfilerBundle();

    $bundles[] = new
Sensio\Bundle\DistributionBundle\SensioDistributionBundle();

    $bundles[] = new Sensio\Bundle\GeneratorBundle\SensioGeneratorBundle();

}

return $bundles;
}

```

Con el método **registerBundles()**, podemos controlar completamente los bundles que utiliza nuestra aplicación, incluso aquellos bundles que forman el núcleo del framework.

Algunos de los **bundles** que usaremos en nuestra web son:

- KnpPaginatorBundle: permite realizar paginación de forma sencilla.
- FOSRestBundle: herramientas para desarrollar api rest de forma rápida
- NelmioApiDocBundle: permite generar documentación de las apis de forma dinámica mediante anotaciones.
- NelmioCorsBundle: permite configurar la aplicación para permitir peticiones de dominio cruzado.

[Ver anexo 1: instalación de bundles](#) para más detalle.

(b) Comentarios del código

Primero generamos las tablas en base de datos a través de **phpMyAdmin**. Una vez creadas, para facilitar el trabajo, usamos el comando de symfony:

```
doctrine:mapping:import
```

Este comando nos genera los archivos .orm.yml que contienen las relaciones y nuestra estructura de datos. Una vez creados, podemos ejecutar el comando:

```
doctrine:generate:entities
```

El cual nos genera los objetos del dominio de nuestra solución. En nuestro caso, Lista, Juego, Plataforma, Usuario... Una vez que tenemos los modelos, symfony nos permite generar un crud automático por defecto de cada una de nuestras entidades.

```
doctrine:generate:crud
```

Todo el código generado nos ayuda a avanzar de forma muy rápida, pero después debemos retocarlo para acercarlo a nuestros objetivos y optimizarlos.

Top Games contiene dos bundles:

- TopGamesBundle: bundle que incluye las clases del dominio y toda la lógica de la aplicación web.
- TopGamesRestBundle: bundle que usa FOSRestBundle para crear la lógica de nuestra api.

Todo el código de **TopGames** trata de métodos sencillos de administración; en este caso de juegos y listas. Destacamos la forma del guardado de imágenes para los juegos:

En base de datos, únicamente guardamos el path: nombre de la foto subida, pero ésta no se llega a guardar en base de datos, por lo que guardamos todas en un mismo directorio: web/uploads/documents

Cuando guardamos un juego, tras validar el formulario y ver que todos los datos son correctos, la imagen la guardamos en ese directorio.

Método **upload** de la entidad Juego:

```
// src/JorgeLillo/TopGamesBundle/Entity/Juego.php

public function upload() {
    // the file property can be empty if the field is not required
    if (null === $this->getFile()) {
        return;
    }
```

```

    // use the original file name here but you should
    // sanitize it at least to avoid any security issues
    // move takes the target directory and then the
    // target filename to move to
    $this->getFile()->move(
        $this->getUploadRootDir(),
        $this->getFile()->getClientOriginalName()
    );

    // set the path property to the filename where you've saved the file
    $this->path = $this->getFile()->getClientOriginalName();

    // clean up the file property as you won't need it anymore
    $this->file = null;
}

```

Para mostrar la imagen de la aplicación usamos los métodos:

```

// src/JorgeLillo/TopGamesBundle/Entity/Juego.php

public function getWebPath() {
    return null === $this->path ? null : $this->getUploadDir() . '/' . $this->path;
}

protected function getUploadDir() {
    // get rid of the __DIR__ so it doesn't screw up
    // when displaying uploaded doc/image in the view.
    return 'uploads/documents';
}

```

Con ello, conseguimos mostrar las imágenes de forma correcta en la web, pero, al no guardar la imagen en base de datos, ¿cómo la podemos mandar a la app móvil?

La solución que hemos optado es, tras obtener todos los juegos en cada una de las llamadas de la api, recorremos el listado y actualizamos el valor de la variable imageBytes ('transient', no se guarda en bbdd), abriendo la imagen con la funciton fopen y codificando la imagen en base 64. De esta forma, como el json devuelve los bytes de la imagen, no nos es difícil pintarla en la app móvil.

GetImageBytes:

```
// src/JorgeLillo/TopGamesRestBundle/Controller/JuegoRestController.php

public function getImageBytes($juego) {
    if ($juego->getPath() != null) {
        $filename = $juego->getAbsolutePath();
        $file = fopen($filename, "rb");
        $contents = fread($file, filesize($filename));
        fclose($file);

        return base64_encode($contents);
    }
}
```

(c) Manual de usuario

Al entrar a la web nos encontramos con la siguiente pantalla:

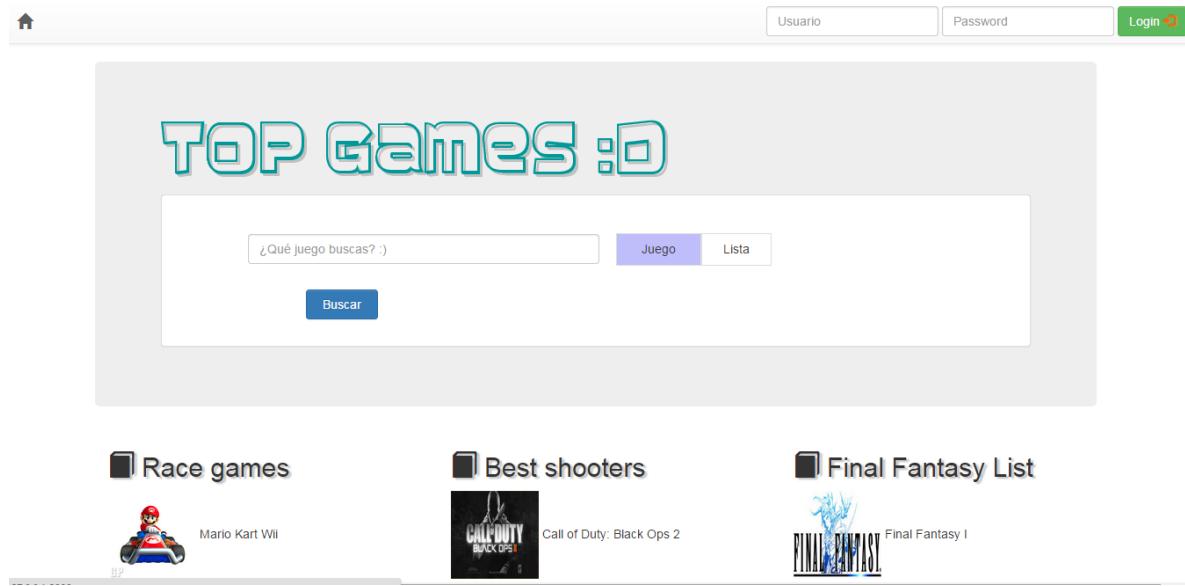


Ilustración 28 Manual usuario TopGames home

En la home, nos recomiendan tres listas de forma aleatoria, cada una de ellas nos muestra uno de sus juegos también de forma aleatoria. Como hemos comentado en otros apartados, un usuario no registrado puede ver detalles de listas, juegos, buscar elementos e iniciar sesión.

Al buscar un juego, nos muestra en otra pantalla un listado. Si el listado sobrepasa los 10 elementos, se pagina el resultado para facilitar su carga y mejorar su apariencia:

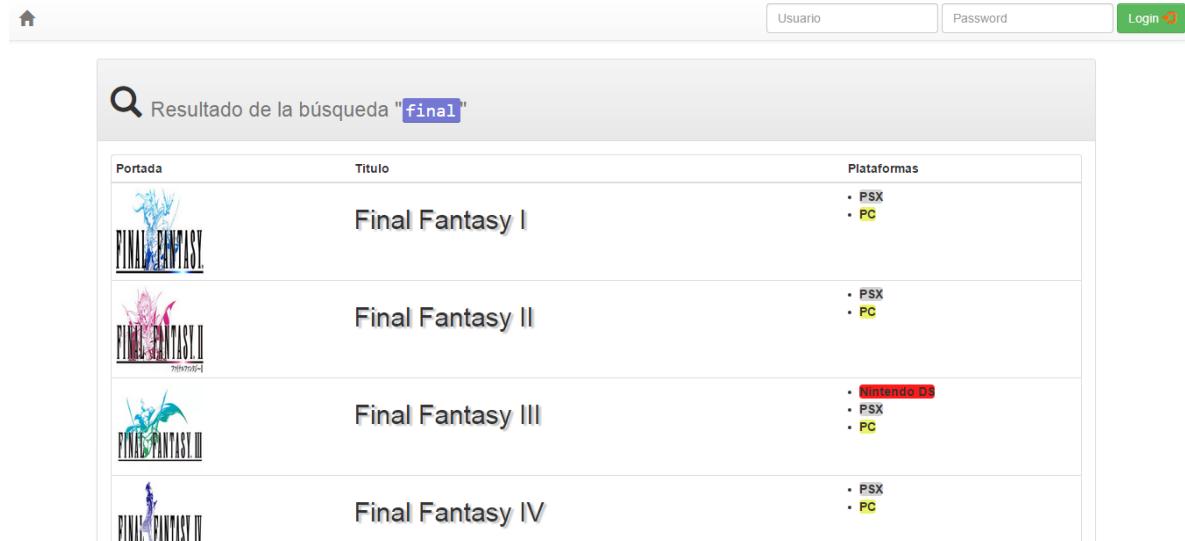


Ilustración 29 Manual usuario buscar juego

Al elegir un juego (o seleccionando uno de las listas aleatorias), nos muestra la pantalla de detalle:

Id	Titulo	Descripcion	Plataformas
15	Final Fantasy VI	Final Fantasy VI se inicia con la Guerra de los Magi, en la que se enfrentaron humanos y espers, criaturas que poseían poderes mágicos. Esta guerra se saldó dejando el mundo reducido a cenizas, y humanos y espers viviendo por separado. Mil años más tarde, cuando el poder de la magia parecía haber caído en el olvido, la fuerza expansionista conocida como el Imperio comienza una conquista exhaustiva con el fin primordial de desentrañar los secretos de los espers y de la magia, que introduce en humanos mediante técnicas de ingeniería genética. Una noche, una pequeña unidad formada por una mujer, cuya voluntad está sometida al Imperio a través de un dispositivo en forma de diadema, y dos soldados, entra en la ciudad-estado de Narshe, donde se rumorea que habita un esper en las profundidades de las minas.	• PSX • PC gestión de plataformas

Para añadir juegos a listas debe estar registrado.

Ilustración 30 Manual usuario detalle juego

En la pantalla de detalle, observamos la portada del juego, la descripción y las plataformas asociadas a él. Como no estamos registrados, no podemos añadir el juego a ninguna lista.

Cuando elegimos una búsqueda en la home de listas, llegamos a la pantalla de búsqueda de listas (misma pantalla que cuando seleccionamos 'buscar juegos'):

Nombre	Descripción	Propietario	
Machacabtones	Juegos de lucha	u1	ver Lista
Best shooters	Juegos de disparos	u1	ver Lista
Race games	Juegos de carreras	u1	ver Lista
Total elementos: 3			

Ilustración 31 Manual usuario buscar listas

Al igual que con los juegos, si el resultado sobrepasa los 10 elementos, se paginan los resultados.

Seleccionando una o entrando a una de las listas de la home, llegamos al detalle de la lista:

Id	Título	Plataformas
23	Star Wars: Battlefront II	• PS4 • PC
24	Halo 4	• Xbox 360

Ilustración 32 Manual usuario detalle lista

Como no se trata de nuestra lista, no podemos interactuar con ella.

Desde la cabecera, en todo momento podemos iniciar sesión, si estamos registrados. Actualmente, sólo un administrador puede crear nuevos usuarios. En futuras versiones, se liberará el registro en la web.

Ilustración 33 Manual usuario login cabecera

Si fallamos el login, nos muestra la pantalla de inicio de sesión para volver a intentarlo:

Ilustración 34 Manual usuario bad credentials

Accediendo de forma correcta iniciamos sesión y nos modifica la cabecera:



Ilustración 35 Manual usuario cabecera registrado

Para cerrar sesión, con pulsar sobre el botón con el nombre de usuario es suficiente.

Al estar registrado, al usuario se le habilita la opción de Mis listas.

A screenshot of a web application interface titled 'Listas: listas dadas de alta en el sistema'. It displays a table with two rows of data. The columns are 'Id', 'Nombre', 'Descripción', 'Autor Original', 'Propietario', and 'Actions'. Row 4: Id 4, Nombre 'Super Crash', Descripción 'Lista de juegos de crash', Autor Original 'u1', Propietario 'jorge', Actions (details, editar, borrar). Row 6: Id 6, Nombre 'Final Fantasy List', Descripción 'Lista de final fantasy', Autor Original 'u1', Propietario 'jorge', Actions (details, editar, borrar). At the bottom left is a blue button labeled 'Nueva lista'.

Ilustración 36 Manual usuario mis listas

Dentro de mis listas nos permite:

- Crear listas:

A screenshot of a 'nueva lista' creation form. It has two input fields: 'Nombre' containing 'test' and 'Descripción' containing 'test'. Below the fields is a 'Crear' button and a link '• Volver a listas'.

Ilustración 37 Manual usuario nueva lista

- Modificar listas:

Listas: editar lista

Nombre: test modificado
Descripción: test

Editar

- Volver a listas
- Eliminar

Ilustración 38 Manual usuario editar lista 1

Listas: listas dadas de alta en el sistema

ID	Nombre	Descripción	Autor Original	Propietario	Actions
4	Super Crash	Lista de juegos de crash	u1	jorge	<ul style="list-style-type: none"> • detalles • editar • borrar
6	Final Fantasy List	Lista de final fantasy	u1	jorge	<ul style="list-style-type: none"> • detalles • editar • borrar
23	test modificado	test	jorge	jorge	<ul style="list-style-type: none"> • detalles • editar • borrar

Nueva lista

Ilustración 39 Manual usuario editar lista 2

- Borrar listas:

Listas: listas dadas de alta en el sistema

ID	Nombre	Descripción	Autor Original	Propietario	Actions
4	Super Crash	Lista de juegos de crash	u1	jorge	<ul style="list-style-type: none"> • detalles • editar • borrar
6	Final Fantasy List	Lista de final fantasy	u1	jorge	<ul style="list-style-type: none"> • detalles • editar • borrar
23	test modificado	test	jorge	jorge	<ul style="list-style-type: none"> • detalles • editar • borrar

Nueva lista

Ilustración 40 Manual usuario borrar

Además de la posibilidad de añadir juegos a listas, quitar juegos y clonar listas de otros usuarios.

The screenshot shows a user interface for managing game lists. At the top, there are navigation links for 'Home' and 'Mis listas' (My Lists), and a user profile 'jorge'. Below this is a header 'Juegos: detalles juego' (Games: game details). The main content area displays a table with game information:

ID	Título	Descripción	Plataformas
24	Warcraft III: Reign of Chaos	Warcraft III: Reign of Chaos es un videojuego de estrategia en tiempo real creado por Blizzard Entertainment y es la tercera parte de la serie Warcraft. Además de continuar la historia del mundo épico medieval de Warcraft se distingue de sus predecesores por incorporar dos importantes cambios: el paso a los gráficos 3D y la aparición de dos nuevas razas. El juego consiste básicamente en administrar los recursos disponibles (oro, madera y alimento) para producir unidades militares y desarrollar un ejército que dirigir en contra de los oponentes hasta destruir todos sus edificios. El juego provee varias estrategias de ataque o defensa, y se ejecutan las tácticas de combate y producción a partir de cuatro diferentes tipos de civilizaciones, llamadas «razas», que protagonizan el juego: humanos, orcos, elfos nocturnos y muertos vivientes. Cada una de estas razas es comandada a su vez por tres clases de héroes que encabezan y apoyan significativamente las batallas ante sus adversarios. U	• PC gestión de plataformas

Below the table is a form for adding the game to a list:

Añadir a lista:

Ilustración 41 Manual usuario añadir juego

Al entrar en una de las listas, encontramos lo siguiente (detalle):

The screenshot shows a user interface for managing game lists. At the top, there are navigation links for 'Home' and 'Mis listas' (My Lists), and a user profile 'jorge'. Below this is a header 'Listas: detalles lista' (Lists: list details). The main content area displays a table with list information:

ID	Nombre	Descripción	Autor Original	Propietario de lista
23	test modificado	test	jorge	jorge

Below the table is a section for 'Juegos en lista' (Games in list):

- Volver a listas
- Editar
- Eliminar

Below this is a 'Listado de juegos:' (Game list) table:

ID	Título	Plataformas
24	Warcraft III: Reign of Chaos	• PC eliminar juego

Ilustración 42 Manual usuario eliminar juego

Al tratarse de nuestra lista, podemos eliminar el juego que acabamos de añadir. Si entramos en la lista de otro usuario, nos habilita la opción de clonar la lista para que sea nuestra. Observamos que no podemos borrar sus juegos:

The screenshot shows a user profile interface. At the top, there are navigation links: 'Mis listas' and a user icon labeled 'jorge'. Below this, a section titled 'Listas: detalles lista' displays a list named 'Machacabtones'. The list details are:

Id	19
Nombre	Machacabtones
Descripción	Juegos de lucha
Autor Original	u1
Propietario de lista	u1
Juegos en lista	2

Below the details is a button labeled 'clonar lista' (clone list). A dropdown menu next to it contains three options: 'Volver a listas', 'Editar', and 'Eliminar' (Delete). The 'Eliminar' option is highlighted with a red box.

Under the heading 'Listado de juegos:' is a table showing one game entry:

ID	Título	Plataformas
33	Dragon Ball Xenoverse	• PS3 • PS4 • XBOX • XBOX 360

Ilustración 43 Manual usuario lista otro usuario

Si clonamos la lista, nos encontramos con una lista igual, pero asociada al usuario actual:

The screenshot shows a user profile interface. At the top, there are navigation links: 'Mis listas' and a user icon labeled 'jorge'. Below this, a section titled 'Listas: detalles lista' displays a list named 'Machacabtones'. The list details are identical to the original:

Id	24
Nombre	Machacabtones
Descripción	Juegos de lucha
Autor Original	u1
Propietario de lista	jorge
Juegos en lista	2

Below the details is a button labeled 'clonar lista' (clone list). A dropdown menu next to it contains three options: 'Volver a listas', 'Editar', and 'Eliminar' (Delete). The 'Eliminar' option is highlighted with a red box.

Under the heading 'Listado de juegos:' is a table showing one game entry:

ID	Título	Plataformas
33	Dragon Ball Xenoverse	• PS3 • PS4 • XBOX • XBOX 360 • PC

Ilustración 44 Manual usuario lista clonada

Como la lista es ya del usuario, podemos editarla, eliminar juegos, añadirle más etc... Se mantiene una referencia al usuario original de la lista para futuras modificaciones.

Por último, los administradores, además de toda la funcionalidad anteriormente descrita, pueden gestionar a todos los demás elementos de la web.

test modificado
Warcraft III: Reign of Chaos

Best shooters
Call of Duty: Black Ops 2

Machacabotones
Dragon Ball Xenoverse

Ilustración 45 Manual usuario home admin

- **Usuarios:**

Id	Usuario	¿Administrador?	¿Activo?	Email	Fecha de alta	Acciones
4	u1	✓	✓	jorge@jorge.com	2015-01-03 11:44:57	
10	jorge	✗	✓	jorge@jorge.com	2015-01-04 23:09:00	
11	patri	✓	✓	patri@patri.com	2015-05-15 17:41:00	

Ilustración 46 Manual usuario listar usuarios

Los administradores pueden ver a todos los usuarios, editar sus datos, borrarlos o desactivarlos.

Un usuario desactivado no puede iniciar sesión. Además de gestionar los usuarios, puede crear nuevos:

Desde aquí puede añadir un nuevo usuario:

Usuario: patri
Contraseña:
¿Administrador?:
¿Activo?:
Correo Electrónico: patri@patri.com
Fecha de alta:
2015 ▾ Jun ▾ 15 ▾
17 ▾ 41 ▾
Nuevo Usuario

Ilustración 47 Manual usuario nuevo usuario

- **Juegos:** al igual que con los usuarios se puede gestionar los datos de los juegos:

Portada	Título	Descripción	Plataformas	Actions
	Crash Team Racing	Crash Team Racing (abreviado a veces en forma de siglas, CTR) es un juego de carreras para la consola PlayStation de la serie de videojuegos Crash Bandicoot, desarrollado por la compañía Naughty Dog. Fue lanzado al mercado en el año 1999, posiblemente en respuesta al gran éxito de la serie Mario Kart de Nintendo. Fue el último juego de Crash Bandicoot creado por Naughty Dog.	• PSX	<ul style="list-style-type: none"> • detalles • edit • plataformas
	Mario Kart Wii	Mario Kart Wii es el sexto videojuego de la serie Mario Kart, segundo en usar la conexión Wi-Fi de Nintendo y el primero en aparecer en la consola Wii.	• WII	<ul style="list-style-type: none"> • detalles • edit • plataformas
	Spyro 2: Ripto's Rage!	Spyro 2: Ripto's Rage! es un videojuego de plataformas para la consola PlayStation. Fue lanzado en Norteamérica bajo este título el 31 de octubre de 1999, en Europa como Spyro 2: Gateway To Glimmer (Spyro 2: En busca de los talismanes en España) el 5 de noviembre de 1999, y en Japón el 16 de marzo de 2000 como Spyro X Sparx Tondemo Tours. Spyro 2 es el segundo juego de la serie Spyro the Dragon, que comenzó con Spyro the Dragon en 1998. El protagonista de la serie, Spyro, recorre la	• PSX • PC	<ul style="list-style-type: none"> • detalles • edit • plataformas

Ilustración 48 Manual usuario listado juegos

Juegos: nuevo juego

Titulo:

Descripción:

File

Ningún archivo seleccionado

[• Volver a lista de juegos](#)

Ilustración 49 Manual usuario nuevo juego

Juegos: editar juego

Título: Crash Team Racing

Crash Team Racing (abreviado a veces en forma de siglas, CTR) es un juego de carreras para la consola PlayStation de la serie de videojuegos Crash Bandicoot, desarrollado por la compañía Naughty Dog. Fue lanzado al mercado en el año 1999, posiblemente en respuesta al gran éxito de la serie Mario Kart de Nintendo. Fue el último juego de Crash Bandicoot creado por Naughty Dog.

Descripción:

File:

Seleccionar archivo Ningún archivo seleccionado

Actualizar

Volver a lista de juegos

Ilustración 50 Manual usuario editar juego

Juegos: detalles juego

ID	Título	Descripción	Plataformas
7	Mario Kart Wii	Mario Kart Wii es el sexto videojuego de la serie Mario Kart, segundo en usar la conexión Wi-Fi de Nintendo y el primero en aparecer en la consola Wii.	wii gestión de plataformas

Añadir a lista: Old school ▾

Añadir a lista

Volver a lista de juegos
Editar
Borrar

Ilustración 51 Manual usuario detalles juego

Además, para añadir plataformas a los juegos, debemos acceder desde el listado principal a plataformas para asociárselas o desde la pantalla de detalles(gestión de plataformas) para añadírselas

The screenshot shows a user interface for managing game lists. At the top, there are navigation links: Home, Usuarios, Juegos, Listas, and a user icon. Below the header, the title "Juegos: Super Mario Sunshine: gestión de plataformas" is displayed. A sidebar on the left lists various platforms with heart icons: Gamecube, PSX, PS2, PS3, PS4, xbox, xbox 360, wii, Game boy color, PC, Nintendo DS, Megadrive, and Nintendo 64. At the bottom of the sidebar is a link to "Volver a lista de juegos".

Ilustración 52 Manual usuario gestión de plataformas

- **Listas:** la gestión de listas para un administrador es igual que la de un usuario registrado con las listas propias, pero el administrador además de poder gestionar sus listas (es un usuario también), puede gestionar las de todos los usuarios.

The screenshot shows a user interface for managing game lists. At the top, there are navigation links: Home, Usuarios, Juegos, Listas, and a user icon. A dropdown menu from the "Listas" link shows options: "Gestión de Listas" and "Mis listas". Below the header, the title "TOP Games :)" is displayed. A search bar contains the placeholder "¿Qué juego buscas? :)". Below the search bar are two buttons: "Juego" and "Lista". A "Buscar" button is located below the search bar. At the bottom, there are three categories: "Machacabotones" (Dragon Ball Xenoverse), "Old school" (Gran Turismo 2), and "Best shooters" (Halo 4). Each category has a thumbnail image, the category name, the game title, and a "Detalles »" button.

Ilustración 53 Manual usuario listas admin

Gestión de listas de todos los usuarios:

Listas: listas dadas de alta en el sistema

Id	Nombre	Descripción	Autor Original	Propietario	Actions
4	Super Crash	Lista de juegos de crash	u1	jorge	<ul style="list-style-type: none"> detalles editar borrar
6	Final Fantasy List	Lista de final fantasy	u1	jorge	<ul style="list-style-type: none"> detalles editar borrar
7	Old school	Juegos de la infancia	u1	u1	<ul style="list-style-type: none"> detalles editar borrar
19	Machacabotones	Juegos de lucha	u1	u1	<ul style="list-style-type: none"> detalles editar borrar
20	Best shooters	Juegos de disparos	u1	u1	<ul style="list-style-type: none"> detalles editar borrar
21	Plataformas	Juegos de plataformas	u1	u1	<ul style="list-style-type: none"> detalles editar borrar
22	Race games	Juegos de carreras	u1	u1	<ul style="list-style-type: none"> detalles

Ilustración 54 Manual usuario gestión listas usuarios

Listas: detalles lista

Id	23
Nombre	test modificado
Descripción	test
Autor Original	jorge
Propietario de lista	jorge
Juegos en lista	1

clonar lista

- Volver a listas
- Editar
- Eliminar**

Listado de juegos:

Id	Título	Plataformas
24	Warcraft III: Reign of Chaos	• PC

Ilustración 55 Manual usuario gestion lista otro usuario

Observamos que el administrador puede eliminar el juego de la lista del usuario registrado que había creado con anterioridad.

Ilustración 56 Manual usuario eliminar juego de otra lista

3.3. Fase dos: api rest

La api rest está desarrollada gracias al bundle:

FOSRestBundle: <https://github.com/FriendsOfSymfony/FOSRestBundle>

Este bundle nos permite crear Api Rest de forma rápida y sencilla, gracias a las herramientas que nos proporciona.

[Ver anexo de instalación de bundles.](#)

- **Get juegos:** devuelve la lista de todos los juegos

Ilustración 57 Api rest get all games

- **Get juego:** devuelve un juego en base a un id

GET /api/games/{id}.{format}

Documentation Sandbox

Documentation
Get a games by id form application, by default it will return a json object.

Requirements

Name	Requirement	Type	Description
id	\d+		
_format	json xml html		

Parameters

Parameter	Type	Required?	Format	Description
{id}	integer	true		game id

Ilustración 58 Api rest get game

- **Juegos de lista:** devuelve listado de juegos de una lista

/api/games/fromList/{idList}

GET /api/games/fromList/{idList}.{format}

Search for games by list

Documentation Sandbox

Documentation
Return the list's list of games, by default it will return a json object.

Requirements

Name	Requirement	Type	Description
id	\d+		
_format	json xml html		
idList			

Ilustración 59 Api rest games from list

- **Buscar juegos:** busca juegos y devuelve el resultado

/api/games/search/{search}

GET /api/games/search/{search}.{format}

Search for games

Documentation Sandbox

Documentation
Search for games given a criteria, by default it will return a json object.

Requirements

Name	Requirement	Type	Description
id	\d+		
_format	json xml html		
search			

Ilustración 60 Api rest buscar juego

- **Get listas:** devuelve todas las listas

/api/lists

GET /api/lists.{_format} Get all lists

Documentation Sandbox

Documentation
Get all the list form application, by default it will return a json object.

Requirements

Name	Requirement	Type	Description
_format	json xml html		

Ilustración 61 Api rest get lists

- **Get lista:** devuelve una lista por id:

GET /api/lists/{id}.{_format} Returns a list

Documentation Sandbox

Documentation
Get a list by id form application, by default it will return a json object.

Requirements

Name	Requirement	Type	Description
id	\d+		
_format	json xml html		

Parameters

Parameter	Type	Required?	Format	Description
(id)	integer	true		list.id

Ilustración 62 Api rest get list

- **Listas de usuario:** devuelve todas las listas de un usuario concreto por id de usuario:

/api/lists/fromUser/{idUser}

GET /api/lists/fromUser/{idUser}.{_format} Get user's lists

Documentation Sandbox

Documentation
Return the user's lists, by default it will return a json object.

Requirements

Name	Requirement	Type	Description
id	\d+		
_format	json xml html		
idUser			

Ilustración 63 Api rest listas de usuario

- **Buscar listas:** busca las listas y devuelve el resultado:

`/api/lists/search/{search}`

GET `/api/lists/search/{search}.{format}` Search for lists

Documentation Sandbox

Documentation
Search for lists given a criteria, by default it will return a json object.

Requirements

Name	Requirement	Type	Description
<code>id</code>	<code>id+</code>		
<code>_format</code>	<code>json xml html</code>		
<code>search</code>			

Ilustración 64 Api rest buscar lista

- **Más juegos:** devuelve de 3 en 3 los juegos basándose en un offset, sirve para implementar el “scroll infinito” en la app de ionic:

`/api/moreGames/{offset}`

GET `/api/moreGames/{offset}.{format}` Get more games from an offset

Documentation Sandbox

Documentation
Get more games form application from an offset, by default it will return a json object.

Requirements

Name	Requirement	Type	Description
<code>id</code>	<code>id+</code>		
<code>_format</code>	<code>json xml html</code>		
<code>offset</code>			

Ilustración 65 Api rest más juegos

- **Get users:** devuelve todos los usuarios

`/api/users`

GET `/api/users.{format}` Get all users from application

Documentation Sandbox

Documentation
Get all the users form application, by default it will return a json object.

Requirements

Name	Requirement	Type	Description
<code>_format</code>	<code>json xml html</code>		Get all users from application

Ilustración 66 Api rest get users

- **Login:** inicia sesión con los parámetros que se le pasan a la llamada. Devuelve “true”, si se autenticó de forma correcta. “False”, si no lo hizo bien.

The screenshot shows a REST API documentation page for a 'Login method'. The URL is `POST /api/users/login.{_format}`. The page includes sections for Requirements and Parameters. Requirements show a parameter `_format` with requirement `json|xml|html`. Parameters show two parameters: `user` (string, required) and `pass` (string, required).

Name	Requirement	Type	Description
<code>_format</code>	<code>json xml html</code>		

Parameter	Type	Required?	Format	Description
<code>user</code>	string	true		user name
<code>pass</code>	string	true		pass of the account

Ilustración 67 Api rest login

- **Get user:** devuelve los datos de un usuario, basándose en un id:

The screenshot shows a REST API documentation page for a 'Returns an user' endpoint. The URL is `GET /api/users/{id}.{_format}`. The page includes sections for Requirements and Parameters. Requirements show a parameter `id` with requirement `\d+`. Parameters show a parameter `{id}` (integer, required).

Name	Requirement	Type	Description
<code>id</code>	<code>\d+</code>		

Parameter	Type	Required?	Format	Description
<code>{id}</code>	integer	true		list id

Ilustración 68 Api rest get user

Todas las llamadas pueden probarse arrancando el proyecto y entrando desde el navegador a alguna de las url de la documentación, por ejemplo:

<http://127.0.0.1:8000/api/lists>

Nos devuelve las listas en formato **json** por defecto:

```

    {
      "listas": [
        {
          "id": 4,
          "nombre": "Super Crash",
          "descripcion": "Lista de juegos de crash",
          "autor_original": "u1",
          "id_usuario": 10
        },
        {
          "id": 6,
          "nombre": "Final Fantasy List",
          "descripcion": "Lista de final fantasy",
          "autor_original": "u1",
          "id_usuario": 10
        },
        {
          "id": 7,
          "nombre": "Old school",
          "descripcion": "Juegos de la infancia",
          "autor_original": "u1",
          "id_usuario": 4
        },
        {
          "id": 19,
          "nombre": "Machacabotones",
          "descripcion": "Juegos de lucha",
          "autor_original": "u1",
          "id_usuario": 4
        },
        {
          "id": 20,
          "nombre": "Best shooters",
          "descripcion": "Juegos de disparos",
          "autor_original": "u1",
          "id_usuario": 4
        }
      ]
    }
  
```

Ilustración 69 Api rest ejemplo respuesta json

Pero, si entramos a <http://127.0.0.1:8000/api/lists.xml>, todas nuestras llamadas devuelven xml en vez de json. Esto es de especial utilidad para desarrolladores externos que se decaten por un formato mayor que por otro.

This XML file does not appear to have any style information associated with it. The document tree is shown below.

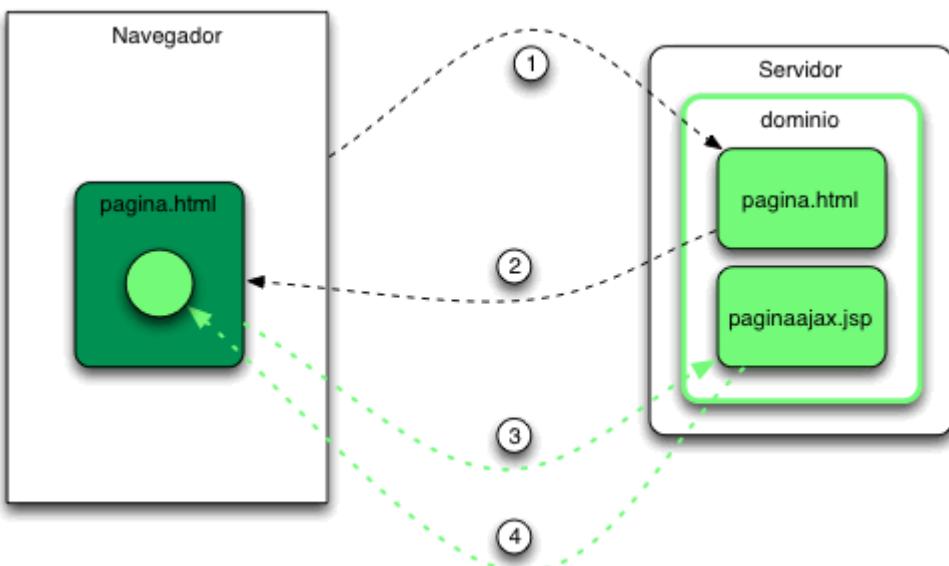
```
<?xml version="1.0" encoding="UTF-8"?>
<result>
    <entry>
        <id>4</id>
        <nombre>
            <![CDATA[ Super Crash ]]>
        </nombre>
        <descripcion>
            <![CDATA[ Lista de juegos de crash ]]>
        </descripcion>
        <autor_original>
            <![CDATA[ u1 ]]>
        </autor_original>
        <id_usuario>10</id_usuario>
    </entry>
    <entry>
        <id>6</id>
        <nombre>
            <![CDATA[ Final Fantasy List ]]>
        </nombre>
        <descripcion>
            <![CDATA[ Lista de final fantasy ]]>
        </descripcion>
        <autor_original>
            <![CDATA[ u1 ]]>
        </autor_original>
        <id_usuario>10</id_usuario>
    </entry>
</result>
```

Ilustración 70 Api rest ejemplo respuesta xml

Cross origin resource sharing (CORS)

Como en la app móvil estaremos probándola desde el navegador en local en un puesto y la aplicación web (y la api) se encuentra en local, pero en otro puerto, nos encontraremos un problema de **CORS**.

Para realizar peticiones javascript, éstas deben pertenecer al mismo dominio. En el caso de que esto no sea así, la aplicación no podrá cargar los datos por limitaciones de seguridad. Los problemas comienzan cuando nosotros tenemos aplicaciones que necesitan acceder a esos datos, pero no se encuentran bajo el mismo dominio.



Para solucionar esto, hemos optado por usar el bundle: NelmioCorsBundle. Gracias a él, después de instalarlo, lo configuraremos de la siguiente manera:

```
// app/config/config.yml

nelmio_cors:

    defaults:

        allow_credentials: false
        allow_origin: []
        allow_headers: []
        allow_methods: []
        expose_headers: []
        max_age: 0
        hosts: []

    paths:
        '^/api':
            allow_credentials: true
            allow_origin: ['*']
            allow_headers: ['*']
            allow_methods: ['POST', 'PUT', 'GET', 'DELETE']
            max_age: 3600
```

De esta forma, permitimos desde cualquier dominio, todas las peticiones, evitando el error CORS inicial.

3.4. Fase tres: app móvil

(a) Ionic Framework

Para llegar a un mayor número de usuarios, decidimos que nuestra app sería para iOS y android (los que dominan el sector), pero, dada la falta de tiempo y la imposibilidad de contar de un ordenador de Apple, nuestra solución fue una aplicación híbrida.

Las ventajas son que, de un mismo proyecto, obtenemos las 2 versiones. La desventaja, que perdemos rendimiento al no ser aplicación nativa, pero, al ser una aplicación sencilla, no presenta demasiados inconvenientes. Usamos Ionic framework para realizar nuestra aplicación híbrida:

NOTA: para poder generar la aplicación en ios, es necesario un mac con sus respectivos drivers o será imposible de generar. Usaremos el comando ionic serve -lab para poder emular en el navegador el resultado en ambos dispositivos.

Ionic es una herramienta, gratuita y open source, para el desarrollo de aplicaciones híbridas basadas en HTML5, CSS y JS. Está construido con Sass y optimizado con AngularJS.

Características:

- Alto rendimiento: la velocidad es importante; tan importante que sólo se nota cuando no está en tu app. Ionic está construido para ser rápido, gracias a la mínima manipulación del DOM, con cero jQuery y con aceleraciones de transiciones por hardware.

- AngularJS & Ionic; Ionic utiliza AngularJS con el fin de crear un marco más adecuado para desarrollar aplicaciones ricas y robustas. Ionic no sólo se ve bien, sino que su arquitectura central es robusta y sería para el desarrollo de aplicaciones. Trabaja perfectamente con AngularJS.

- Centro nativo Ionic se inspira en las SDK de desarrollo móviles nativos más populares, por lo que es fácil de entender para cualquier persona que ha construido una aplicación nativa para iOS o Android. Lo interesante, como es conocido, es que se desarrolla una vez, y se compila para varios.

- Bonito diseño: limpio, sencillo y funcional. Ionic ha sido diseñado para poder trabajar con todos los dispositivos móviles actuales. Cuenta con muchos componentes usados en móviles, tipografía, elementos interactivos, etc.

- Un potente CLI: con un sólo comando podrás crear, construir, probar y compilar tus aplicaciones en cualquier plataforma.

Ionic trabaja bajo Cordova, por lo que podremos utilizar todos los plugins que estén desarrollados para el framework e incluso desarrollar los nuestros propios. En esta url, se pueden ver todos los que actualmente están en el repositorio de Apache Cordova: <http://plugins.cordova.io/> Tan solo hay que añadirlos mediante línea de comando, tal como se instalan con Cordova, en caso de compilar de forma local.

Además, todas nuestras apps desarrolladas con Ionic pueden compilarse de forma remota utilizando <https://build.phonegap.com/apps>, que es un servicio de Adobe que nos permite compilar nuestra app

desde la nube. De esta forma no necesitamos un Mac para compilar para iOS o un Windows para compilar para WP.

La licencia Ionic es de código abierto, publicada bajo una licencia MIT. Esto significa que se puede utilizar en nuestros proyectos personales o comerciales, de forma gratuita. MIT es la misma licencia que utilizan otros proyectos populares como jQuery y Ruby on Rails.

Futuro: en los planes de ionic entra dar soporte a windows phone y firefoxOs

[Ver Anexo 3 instalacion de ionic](#)

(b)Comentarios del código

Para el desarrollo del código de TopGamesIonic se ha realizado sobre notepad++:

NotePad++: <https://notepad-plus-plus.org/download/v6.7.9.2.html>

Para tener una vista por carpetas, recomendamos instalar el plugin: Light explorer

Plugins->Plugins Manager ->Show Plugin Manager, buscarlo e instalar.

Tras reiniciar el notepad++, si pulsamos ALT + A, nos aparecerá una ventana para poder navegar entre archivos con mayor comodidad.

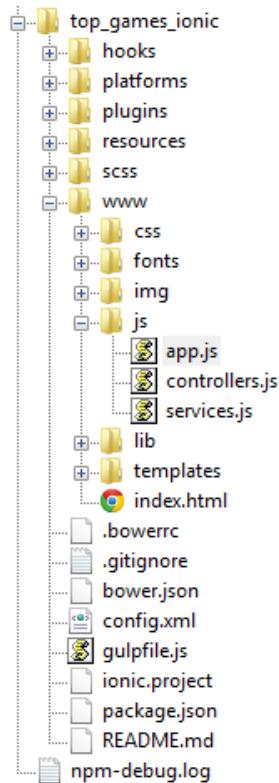


Ilustración 71 Notepad++ & Light explorer

Como vemos en la imagen, ahora nos permite navegar entre todas las carpetas del proyecto y abrirlas con mayor facilidad.

Los archivos más importantes del proyecto son:

- **App.js**: configuración del proyecto, en nuestro caso tenemos los estados de navegación necesarios para la aplicación con tres pestañas:

```

|.config(function($stateProvider, $urlRouterProvider) {

    // Ionic uses AngularUI Router which uses the concept of states
    // Learn more here: https://github.com/angular-ui/ui-router
    // Set up the various states which the app can be in.
    // Each state's controller can be found in controllers.js
    $stateProvider

        // setup an abstract state for the tabs directive
        .state('tab', {
            url: "/tab",
            abstract: true,
            templateUrl: function() {
                if (ionic.Platform.isAndroid()) {
                    return "templates/tabs-android.html";
                }
                return "templates/tabs-ios.html";
            }
        })

        // Each tab has its own nav history stack:

        .state('tab.game', {
            url: '/game',
            views: {
                'tab-game': {
                    templateUrl: 'templates/tab-game.html',
                    controller: 'GameCtrl',
                    resolve: {
                        firstgames: function(Games) {
                            return Games.firstGames();
                        }
                    }
                }
            }
        })
})

```

Ilustración 72 Ionic app js

Tras instalarlo, lo configuramos de la siguiente manera:

```

// if none of the above states are matched, use this as the fallback

$urlRouterProvider.otherwise('/login');

```

De esta forma, la primera vez que arranque, el proyecto nos cargará la pantalla de login.

- **Controller.js:** archivo que contiene la lógica de la aplicación, por ejemplo, todo lo que se debe realizar al intentar iniciar sesión:

```
.controller('LoginCtrl', function($scope, LoginService, $ionicPopup, $state, $http) {
  $scope.data = {};

  $scope.login = function() {
    LoginService.loginUser($scope.data.username, $scope.data.password, $http).success(function(data) {
      $state.go('tab.game');
    }).error(function(data) {
      var alertPopup = $ionicPopup.alert({
        title: 'Error al iniciar sesión',
        template: 'Por favor revise sus credenciales'
      });
    });
  }
})
```

Ilustración 73 Ionic controller.js

- **Service.js:** capa de proveedor de datos. Normalmente se usa para llamar al backend

```
.factory('Games', function($http, $q) {
var games = [];
return {
  firstGames: function(){
    var dfd = $q.defer();
    $http.get(SERVER_URL + "moreGames/" + 0).then(function(response){
      games = response.data.juegos;
      dfd.resolve(games);
    });
    return dfd.promise;
  },
  formList: function(listId){
    var gamesFromList = [];
    var dfd = $q.defer();
    $http.get(SERVER_URL + "games/fromList/" + listId).then(function(response){
      gamesFromList = response.data;
      dfd.resolve(gamesFromList);
    });
    return dfd.promise;
  },
  get: function(gameId) {
    var game = [];
    var dfd = $q.defer();
    $http.get(SERVER_URL + "games/" + gameId).then(function(response){
      game = response.data.juego
      dfd.resolve(game);
    });
    return dfd.promise;
  }
}
})
```

Ilustración 74 Ionic service.js

- Todas las pantallas de la aplicación se guardan dentro de www/templates, ficheros html apoyados con angular y las etiquetas propias de ionic:

Ejemplo login.html:

```
<!--
Create tabs with an icon and label, using the tabs-positive style.
Each tab's child <ion-nav-view> directive will have its own
navigation history that also transitions its views in and out.
-->
<ion-view view-title="Inicio de sesión" name="login-view">
  <ion-content class="padding">
    <div class="list list-inset">
      <label class="item item-input">
        <input type="text" placeholder="Usuario" ng-model="data.username">
      </label>
      <label class="item item-input">
        <input type="password" placeholder="Contraseña" ng-model="data.password">
      </label>
    </div>
    <button class="button button-block button-calm" ng-click="login()>Iniciar sesión</button>
  </ion-content>
</ion-view>
```

Ilustración 75 ionic html template

Cuando se cargue en android, saque las pestañas de color azul propias del sistema, pero, cuando sea iOs, cargue las tab bar. Hemos modificado el estado “tabs” en app.js para que, dependiendo del sistema, cargue una plantilla html u otra:

```
// setup an abstract state for the tabs directive
.state('tab', {
  url: "/tab",
  abstract: true,
  templateUrl: function() {
    if (ionic.Platform.isAndroid()) {
      return "templates/tabs-android.html";
    }
    return "templates/tabs-ios.html";
  }
})
```

Ilustración 76 ionic android o ios template

4. CONCLUSIONES

4.1. Conclusión y líneas futuras

Hoy en día vivimos en el mundo de la web, por lo que acceder a Internet desde nuestros ordenadores, tablets o smartphones es algo cotidiano. Por ello, el desarrollo de aplicaciones web que puedan ser compatibles con todo tipo de dispositivos, está en pleno auge. El desarrollo web tiene muchas ventajas que son explotadas a su favor, como no requerir de software especial para los clientes, información centralizada, copias de seguridad...

Para realizar aplicaciones web, podemos decidirnos por distintos lenguajes de programación, que nos ofrecen diferentes aproximaciones para los problemas que nos pueda plantear el desarrollo de aplicaciones. No hay una solución perfecta ni mejor ni peor; se trata de saber elegir la solución que mejor se adapte a nuestros problemas y esto no siempre es equivalente a la opción más famosa o extendida.

La idea de este proyecto era coger todo lo aprendido del máster y juntarlo todo para crear algo con los conocimientos aprendidos. Gracias al framework symphony, hemos podido crear una web en un plazo ajustado, siguiendo el patrón mvc y las buenas prácticas del framework, así como la aplicación móvil multiplataforma gracias a ionic.

Todo el código desarrollado se encuentra en los repositorios:

- https://github.com/jorgelillo7/top_games
- https://github.com/jorgelillo7/top_games_ionic

Debido al corto plazo para el desarrollo, la web queda como un mero prototipo, pero, en futuras versiones de Top Games, se mejorarán las prestaciones que tiene actualmente la web, añadiendo listas compartidas de usuarios, comentarios y sistema de puntos con ranking de usuarios. La api se mejorará, permitiendo introducir y modificar datos de la web. Además, la app móvil permitirá hacer en ella todas las funciones que se hagan desde la web.

5. BIBLIOGRAFÍA E INFOGRAFÍA

Bibliografía:

Dunglas, Kévin (2013), *Persistence in PHP with Doctrine ORM*.

Noback , Matthias (2013), *A Year With Symfony: Writing healthy, reusable Symfony2*.

Ravulavaru, Arvind (2015), *Learning Ionic*.

Infografía:

<http://blog.micayael.com/category/php/>

<http://dev.mysql.com/doc/refman/5.7/en/index.html>

<http://ionicframework.com/docs/>

<http://ionicframework.com/getting-started/>

<http://knpbundles.com/>

https://librosweb.es/libro/symfony_2_x/

<https://symfony.com/doc/current/index.html>

<http://symfony.es/>

<http://symfony.es/libro/>

http://es.wikipedia.org/wiki/Modelo_Vista_Controlador

<http://www.php.net/manual/es/>

6. APÉNDICES

6.1. Anexo A: Manual de instalación Top Games Web

Documentación oficial: <https://symfony.com/doc/current/index.html>

El siguiente manual tiene como función explicar el proceso de instalación de Symfony 2 en Windows. Pasos que se deben seguir:

- **Composer:** Composer es una herramienta imprescindible para programar aplicaciones Symfony 2.1. Pronto será también imprescindible para muchas otras aplicaciones y frameworks PHP, ya que muchos programadores lo consideran la versión actualizada y mejorada de PEAR. Los proyectos PHP grandes, como, por ejemplo, las aplicaciones Symfony2, dependen a su vez de muchos otros proyectos. Cuando se envía, por ejemplo, un email, Symfony2 utiliza una librería externa llamada SwiftMailer. Para que la aplicación funcione bien, Symfony2 necesita que todas esas librerías externas (llamadas *dependencias*) se instalen correctamente. La opción más sencilla para intalarlo y configurar las variables de entorno es acceder a: <https://getcomposer.org/download/> y descargar el archivo [Composer-Setup.exe](#). Con ello, todo se configurará de forma automática.
- **Entorno de trabajo:** Netbeans 7.4 <https://netbeans.org/downloads/7.4/>

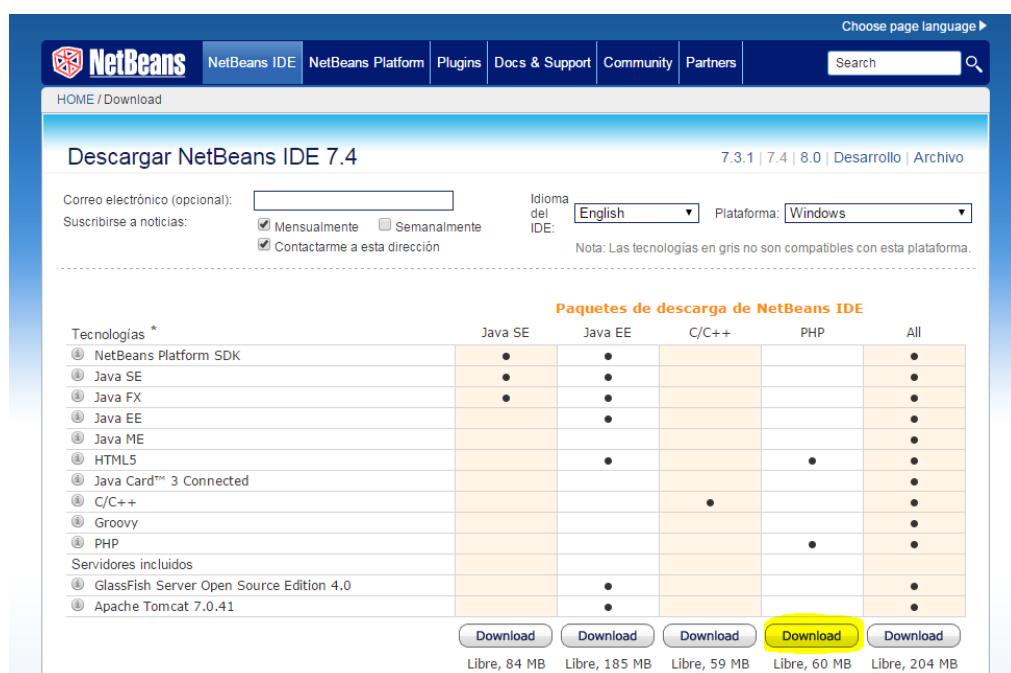


Ilustración 77 Anexo I Instalación Symfony: Netbeans descarga

Para el proyecto, descargamos la versión PHP desde la página de descarga de netbeans.

(a) Creación nuevo proyecto:

Creación de nuevo proyecto:

Una vez instalado, seleccionamos nuevo proyecto PHP:

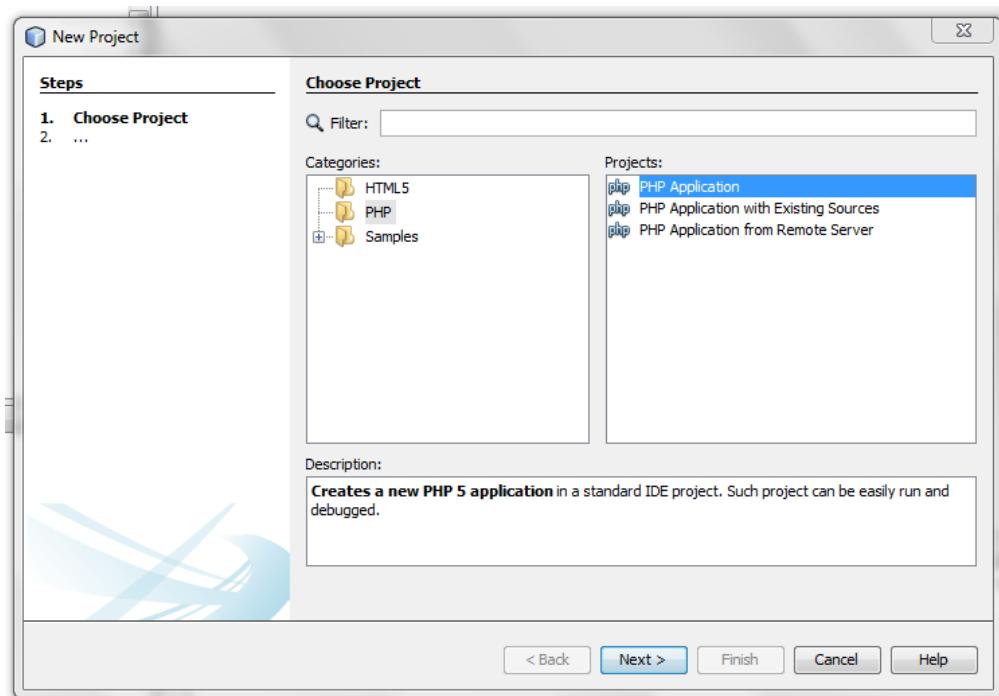


Ilustración 78 Anexo I Instalación Symfony: Netbeans new project

Elegimos el nombre del proyecto, la versión de PHP y la codificación.

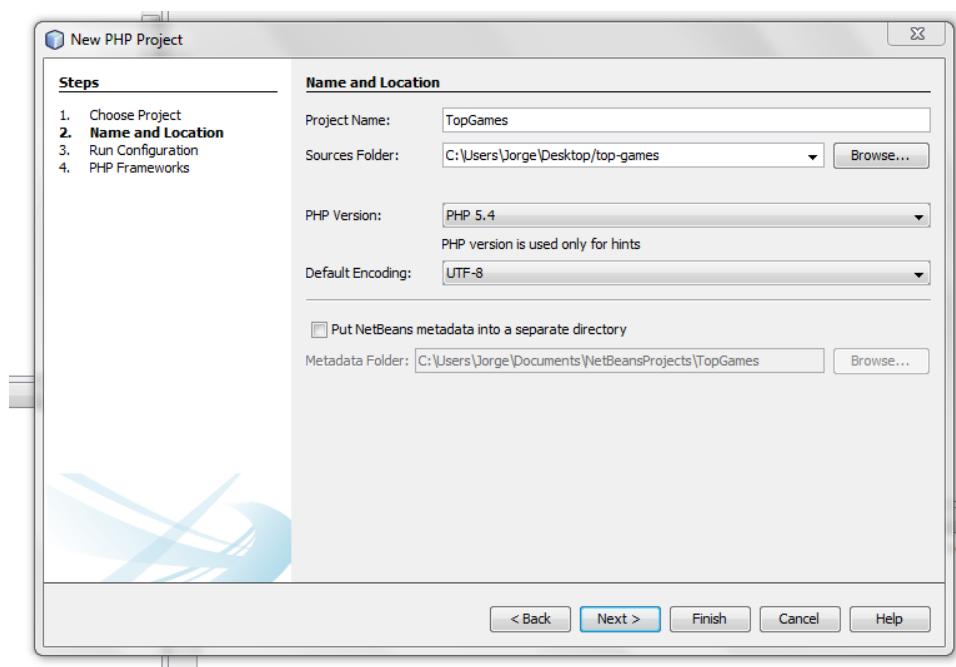


Ilustración 79 Anexo I Instalación Symfony: Netbeans new project 2

Configuramos que se desplegará en nuestro servidor local.

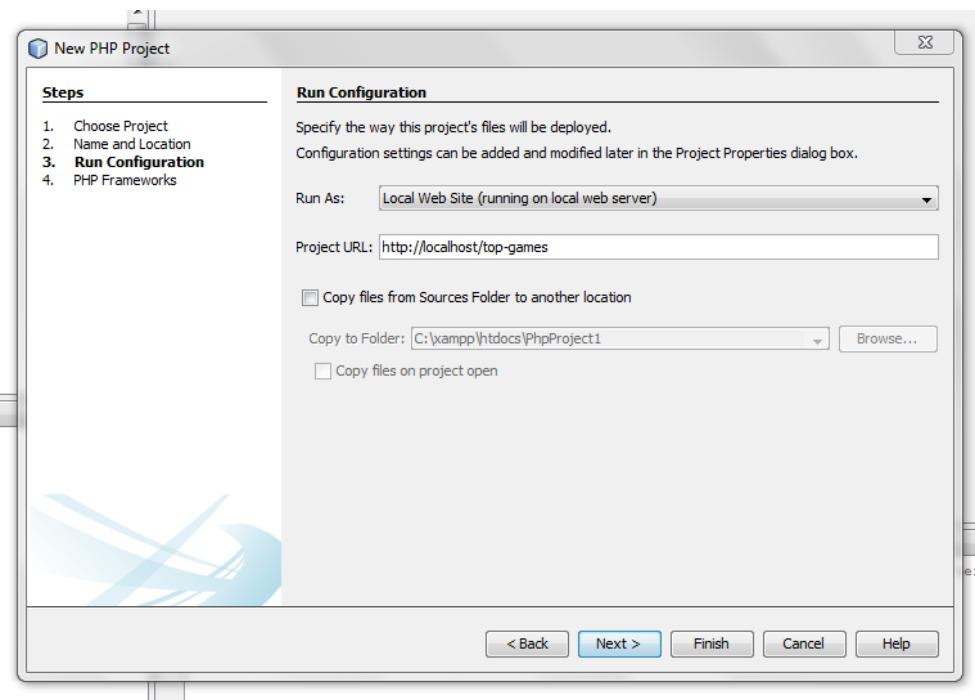


Ilustración 80 Anexo I Instalación Symfony: Netbeans new project 3

Y le añadimos el framework Symfony 2:

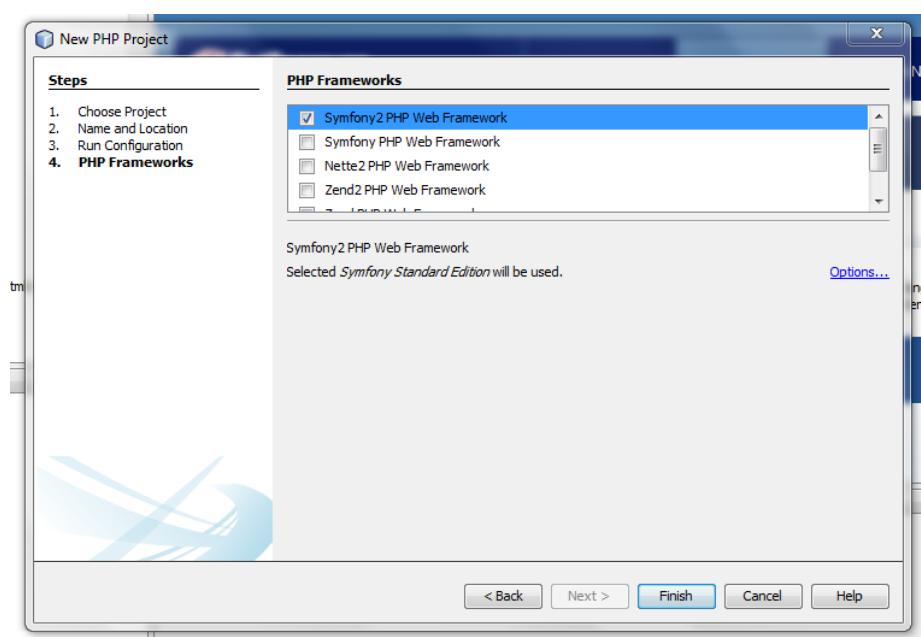


Ilustración 81 Anexo I Instalación Symfony: Netbeans new project 4

Con esto, nos crearía un proyecto nuevo con algunas clases de demo.

(b) Importar TopGames web:

Descargamos los archivos del proyecto desde el cd incluido con la memoria, o descargamos los ficheros desde https://github.com/jorgelillo7/top_games

Para añadir el proyecto a NetBeans, le damos a abrir proyecto y seleccionamos el proyecto:

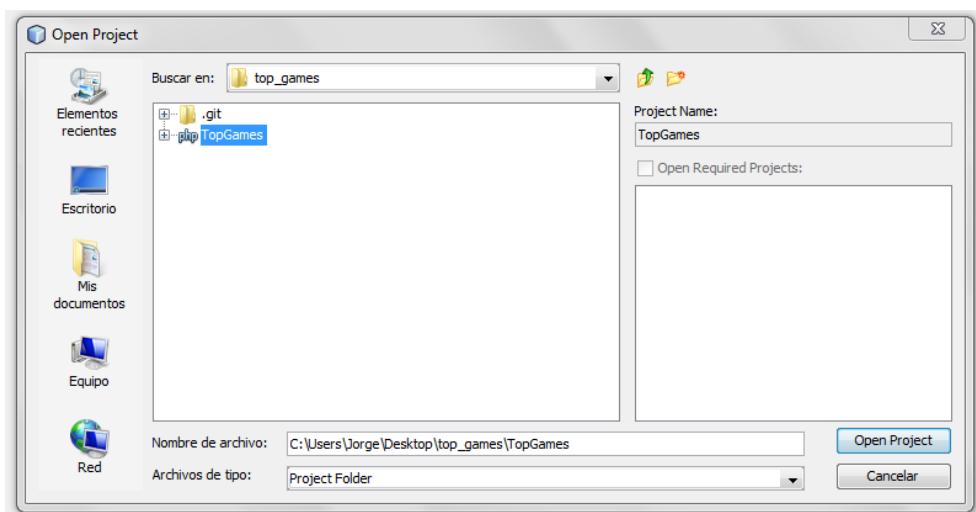


Ilustración 82 Anexo I Importar Top Games 1

Una vez importado, debemos ejecutar el comando de composer:

- Desde el propio Netbeans: Composer-> Install
- Desde consola: composer install

Gracias a este comando, se instalarán todas las dependencias que tiene el proyecto que no han sido incluidas ni en el cd ni en el repositorio de github para que el proyecto no sea tan pesado.

Una vez configurado el proyecto, nos faltaría tener la base de datos a la que se intenta conectar el proyecto, hemos usado para TopGames **xampp**. Podemos descargarlo desde <https://www.apachefriends.org/es/download.html>. Una vez instalado y arrancado el servidor de mysql:

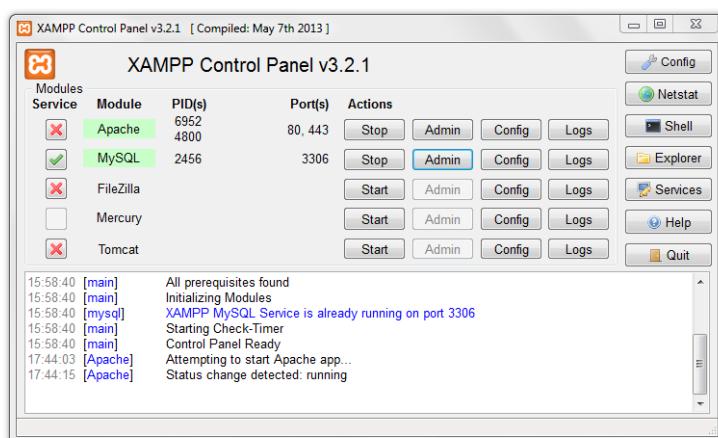


Ilustración 83 Anexo I Importar Top Games 2

Seleccionamos Admin de MySQL y accedemos a phpMyAdmin. Si elegimos ‘importar’, añadimos el fichero .sql incluido en el cd, que nos generará las tablas de base de datos con los datos básicos de prueba.



Importando al servidor actual

Archivo a importar:

El archivo puede ser comprimido (gzip, bzip2, zip) o descomprimido.
Un archivo comprimido tiene que terminar en **.[formato].[compresión]**. Por ejemplo: **.sql.zip**

Buscar en su ordenador: top_games (2015-06-27).sql (Máximo: 2,048KB)

Conjunto de caracteres del archivo:

Ilustración 84 Anexo I Importar Top Games 3

Para completar la instalación, debemos añadir un usuario con permisos al esquema top_games:

Agregar usuario

A screenshot of the 'Agregar usuario' (Add user) form in phpMyAdmin. The form is titled 'Información de la cuenta' (Account information). It contains five fields: 'Nombre de usuario:' with a dropdown 'top_games', 'Servidor:' with a dropdown 'localhost', 'Contraseña:' with a dropdown and a masked password field, 'Debe volver a escribir:' with a masked password field, and 'Generar contraseña:' with a 'Generar' button and an empty password field.

Ilustración 85 Anexo I Importar Top Games 4

Podemos cambiar y crear otro usuario siempre y cuando cambiemos el archivo de configuración del proyecto parameters.yml: app/config/parameters.yml

```

parameters:
    database_driver: pdo_mysql
    database_host: 127.0.0.1
    database_port: null
    database_name: top_games
    database_user: top_games
    database_password: '*top_games*'
    mailer_transport: smtp
    mailer_host: 127.0.0.1
    mailer_user: null
    mailer_password: null
    locale: en
    secret: TopGamesSymfony

```

Ilustración 86 Anexo I Importar Top Games 5

Por último, si todo ha ido bien, ejecutamos el comando de symfony server:run con los parámetros --env=prod --no-debug para ejecutarlo sin modo debug y nos arrancará el proyecto en: <http://127.0.0.1:8000>

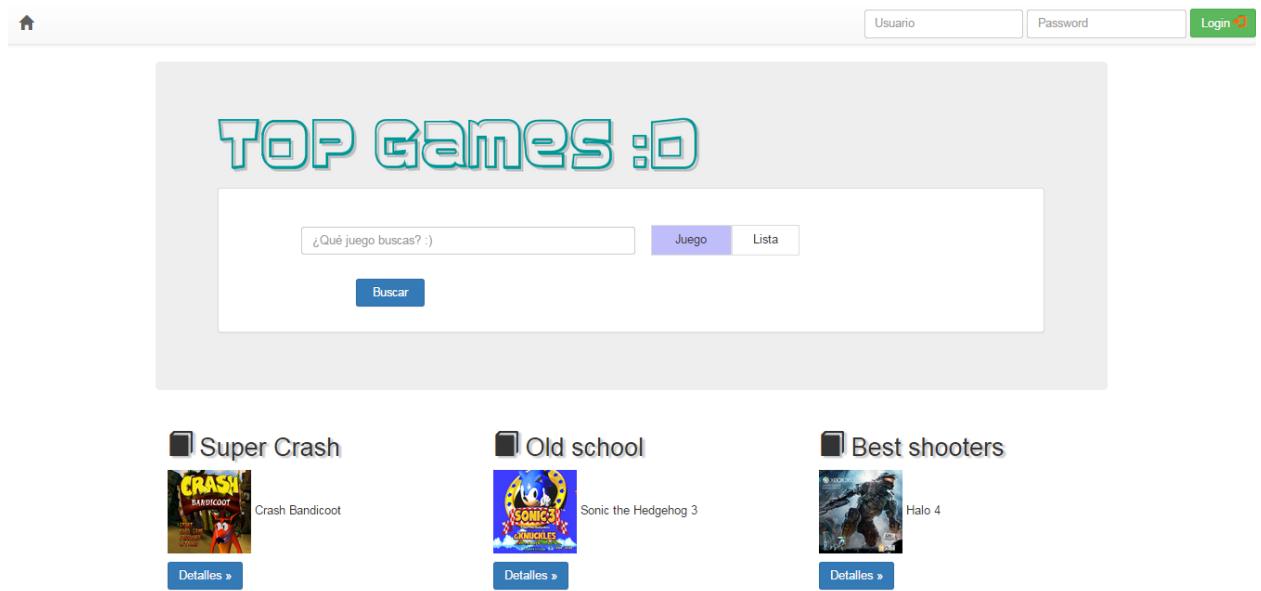


Ilustración 87 Anexo I Importar Top Games 6

Para iniciar sesión y poder gestionar los datos, se incluye en el volcado de base de datos un administrador:

Usuario: u1

Password: *u1*

(c) Composer añadir bundles:

Para ampliar la funcionalidad del proyecto usamos varios bundles adicionales como:

- KnpPaginatorBundle: permite realizar paginación de forma sencilla.
 - <https://github.com/KnpLabs/KnpPaginatorBundle>
 - composer require knplabs/knp-paginator-bundle
- FOSRestBundle: herramientas para desarrollar api rest de forma rápida
 - <https://github.com/FriendsOfSymfony/FOSRestBundle>
 - composer require friendsofsymfony/rest-bundle
- NelmioApiDocBundle: permite generar documentación de las apis de forma dinámica mediante anotaciones.
 - <https://github.com/nelmio/NelmioApiDocBundle>
 - composer require nelmio/api-doc-bundle
- NelmioCorsBundle: permite configurar la aplicación para dar acceso a peticiones de domino cruzado (Cross-Origin Resource Sharing)
 - <https://github.com/nelmio/NelmioCorsBundle>
 - composer require nelmio/cors-bundle

Podemos ejecutar los comandos para instalar las dependencias o desde el propio Netbeans, seleccionando con el botón derecho Composer -> Add dependency y buscando el nombre del bundle que queremos:

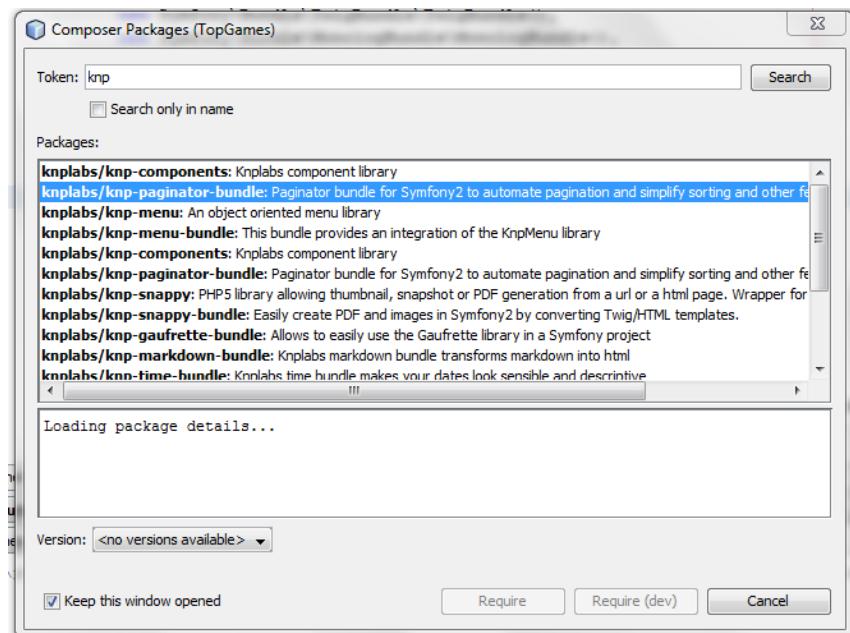


Ilustración 88 Anexo I Composer

Una vez instalado el bundle que queramos, debemos habilitar el bundle para nuestro proyecto. Por ello, en el AppKernel (app/AppKernel.php) debemos añadirlo: Ej:

```
// app/AppKernel.php
public function registerBundles()
{
    return array(
        // ...
        new Nelmio\ApiDocBundle\NelmioApiDocBundle(),
    );
}
```

Después, cada bundle necesitará una configuración específica. Para ello, deberemos leer la documentación de cada uno de ellos. Por lo general, habrá que añadir algún parámetro de configuración en el archivo config.yml (app/config/config.yml)

Ej:

```
// app/config.yml
knpPaginator:
    page_range: 5          # default page range used in pagination control
    default_options:
        page_name: page      # page query parameter name
        sort_field_name: sort  # sort field query parameter name
        sort_direction_name: direction # sort direction query parameter name
        distinct: true        # ensure distinct results, useful when ORM queries use GROUP BY
    template:
        pagination: KnpPaginatorBundle:Pagination:twitter_bootstrap_v3_pagination.html.twig
        sortable: KnpPaginatorBundle:Pagination:sortable_link.html.twig # sort link template
```

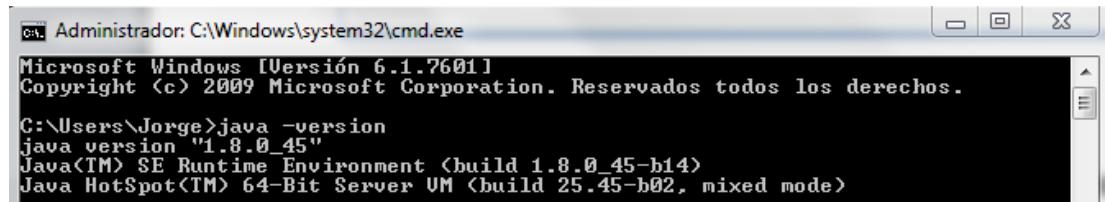
6.2. Anexo B: Manual de Ionic Framework

Documentación oficial: <http://ionicframework.com/docs/>

Pasos que hay que seguir para una correcta instalación de ionic en Windows:

- **Java:**

- <https://java.com/en/download/>
- Path recomendado: C:\Program Files\Java
- Comprobar instalación completa (consola): java -version



A screenshot of a Windows Command Prompt window titled "Administrador: C:\Windows\system32\cmd.exe". The window shows the following text:
Microsoft Windows [Versión 6.1.7601]
Copyright © 2009 Microsoft Corporation. Reservados todos los derechos.
C:\Users\Jorge>java -version
java version "1.8.0_45"
Java(TM) SE Runtime Environment (build 1.8.0_45-b14)
Java HotSpot(TM) 64-Bit Server VM (build 25.45-b02, mixed mode)

Ilustración 89 Anexo 2: instalación ionic java

- **Apache ant:**

- <http://ant.apache.org/bindownload.cgi>
- Path recomendado: C:\apache-ant-1.9.5
- Descargar zip y extraerlo en la ruta.

- **Node js:**

- <https://nodejs.org/>
- Path recomendado: C:\Program Files\nodejs
- Comprobar instalación completa (consola): node -v



A screenshot of a Windows Command Prompt window titled "C:\Users\Jorge>node -v". The window shows the following text:
v0.12.4

Ilustración 90 Anexo 2: instalación ionic node

- **Android SDK**

- <http://developer.android.com/sdk/index.html> (Descargar sólo el SDK)
- C:\Program Files\Android
- Una vez descargado, entrar en android-sdk y ejecutar como administrador el SDK manager y descargamos:
 - Android SDK tools
 - Android SDK platform-Tools

- Android SDK build-Tool for Rev 19,19.1,19.01.1 (19 and greater than 19 Rev)
 - Android 5.1.1(API 22)
 - Android 4.4W.2(API 20)
 - Android 4.4.2(API 19)
 - **Extra:** Intel x86 Emulator Accelerator (HAXM installer)
- **Cordova:**
 - *npm install -g cordova*
 - Comprobar la instalación completa (consola): cordova -v
- C:\Users\Jorge>cordova -v
5.0.0**
- **Ionic:**
 - *npm install -g ionic*
 - Comprobar la instalación completa (consola): ionic -v
- C:\Users\Jorge>ionic -v
1.5.0**

Ilustración 91 instalación ionic cordova

Ilustración 92 instalación ionic ionic

Una vez instalado todo, tenemos que configurar las variables de entorno para que funcione todo de forma correcta:

Equipo botón derecho propiedades -> Propiedades -> Configuración avanzada del sistema:

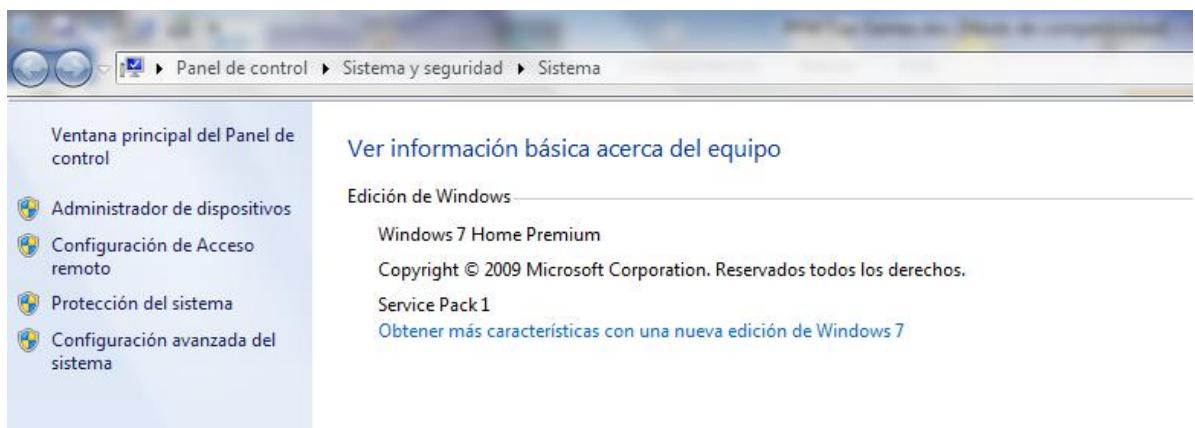


Ilustración 93 instalación ionic configuración avanzada

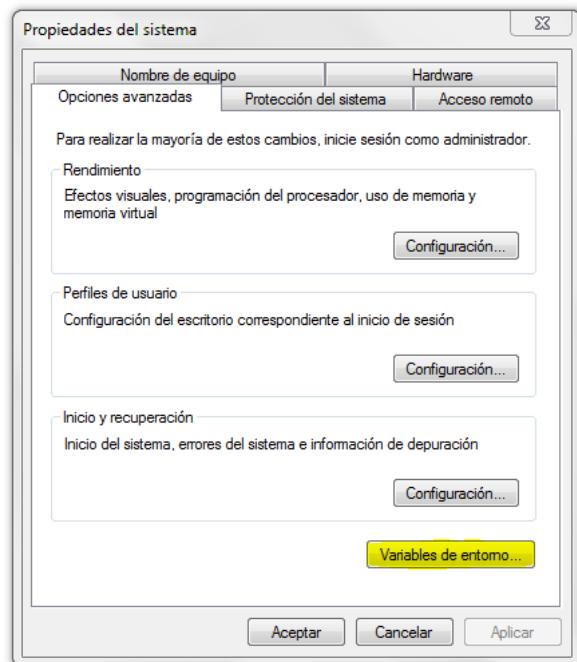


Ilustración 94 instalación ionic variables de entorno

JAVA_HOME: ponemos la ruta en el lugar donde tenemos java instalado. Si no existe, la creamos:

- Path: C:\Program Files\Java\jdk1.8.0_40

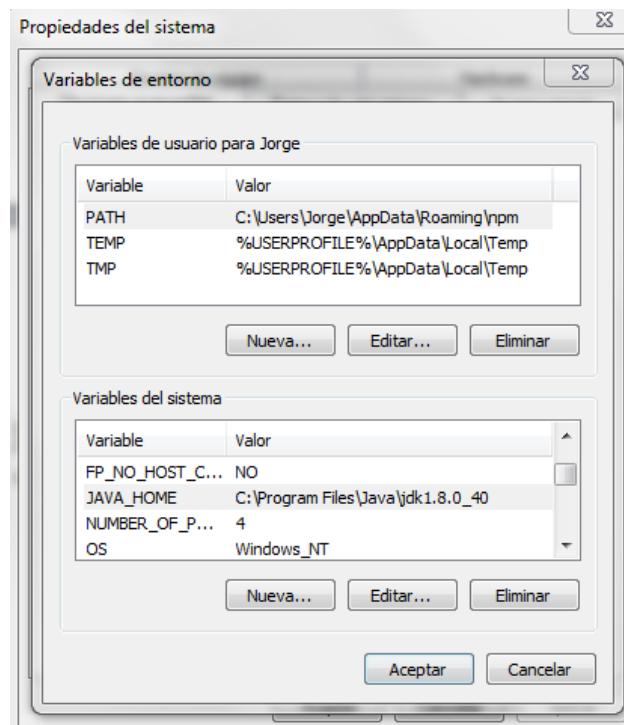


Ilustración 95 instalación ionic variables de entorno java_home

- **PATH:** (Variables del sistema):

- Debemos añadir a continuación de lo que tengamos:

```
// Variable de sistema Path
C:\Program Files\nodejs;C:\Program Files\nodejs\node_modules\npm;C:\Program
Files\Android;
C:\Program Files\Android\android-sdk;C:\Program Files\Android\android-sdk\platforms;
C:\Program Files\Android\android-sdk\platform-tools;C:\Program
Files\Android\android-sdk\tools;
C:\Program Files\Android\android-sdk\platforms\android-19;C:\Program
Files\Android\android-sdk\platforms;
C:\Program Files\Android\android-sdk\platform-tools;C:\Program
Files\Java\jdk1.8.0_40\bin;
C:\Program Files\Android\android-sdk\tools;C:\apache-ant-1.9.5\bin;
```

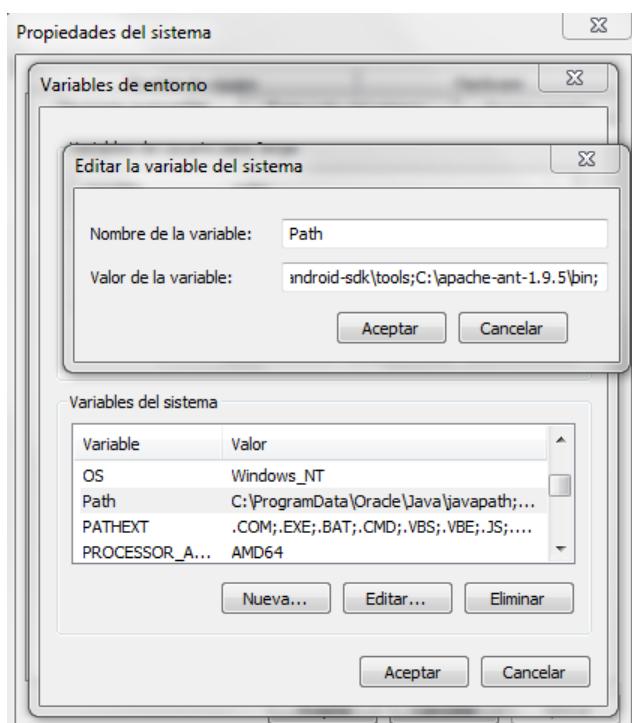


Ilustración 96 instalación ionic variables de entorno Path

** Nota: si tenemos algún valor en la variable Path, antes de copiar las nuevas dependencias, debemos incluir antes un ";" (sin comillas) que actúa como separador.

Para realizar pruebas en dispositivos virtuales, podemos configurar AVD Manager para tener un dispositivo de prueba. Para nuestro proyecto, usaremos directamente la opción de checkear desde el navegador:

```
// C:\Program Files\Android\android-sdk  
AVD Manager -> android virtual device
```

Device definition

*Device:Nexus S 480*800*

Target:Android 4.4.2-API level 19

CPU/ABI:ARM(armeabi-v7a)

Select:hardware keyboard present

RAM:512

Emulation option: select USE host GPU

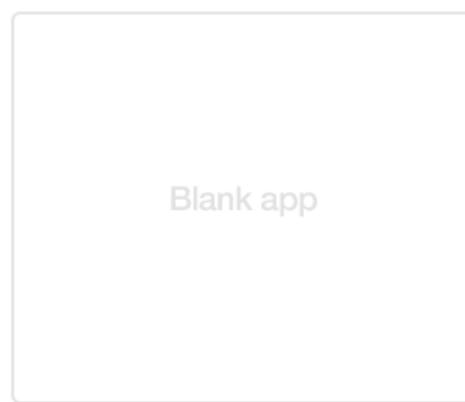
Recomendamos guardar todos los proyectos de ionic en un mismo directorio: C:\ionic

(a) Creación nuevo proyecto y comandos útiles:

Una vez que está todo configurado, podemos comenzar a usar ionic de verdad.

Para crear una aplicación tenemos tres plantillas:

- **Blank:** aplicación vacía
 - `ionic start appName blank`



```
$ ionic start myApp blank
```

Ilustración 97 ionic puesta en marcha 1

- **Tabs:** ejemplo de app con tres pestañas y navegación hecha.
 - `ionic start appName tabs`

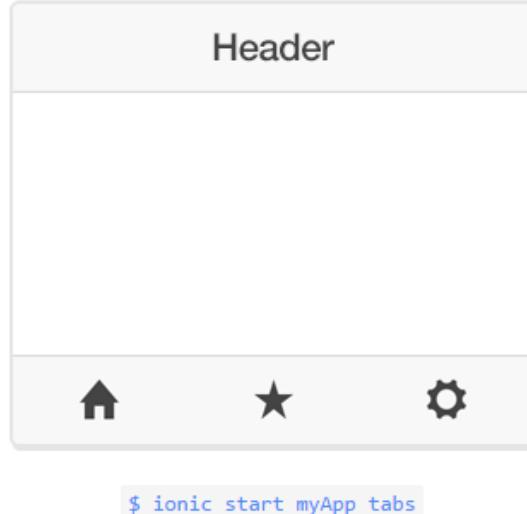


Ilustración 98 ionic puesta en marcha 2

- **Sidemenu:** app con menú lateral
 - `ionic start appName sidemenu`

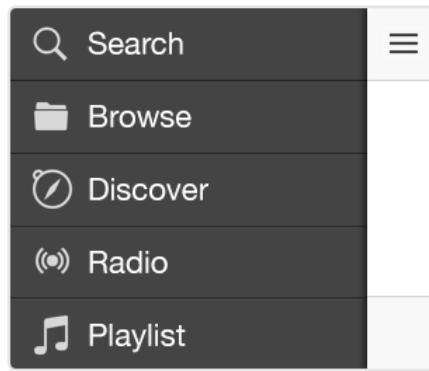


Ilustración 99 ionic puesta en marcha 3

Añadir plataformas:

Con estos commandos, habilitamos nuestro proyecto para las plataformas que deseemos:

```
$ ionic platform add ios
$ ionic platform add android
```

* NOTA: si no estás trabajando en MacOs, no se podrá usar la plataforma ios.

Test:

Como hemos mencionado anteriormente, podemos probar nuestra aplicación directamente desde el navegador:

- `ionic serve`

```
C:\ionic\top_games_ionic>ionic serve
Running live reload server: http://localhost:35729
Watching : [ 'www/**/*', '!www/lib/**/*' ]
Running dev server: http://localhost:8100
Ionic server commands, enter:
  restart or r to restart the client app from the root
  goto or g and a url to have the app navigate to the given url
  consolelogs or c to enable/disable console log output
  serverlogs or s to enable/disable server log output
  quit or q to shutdown the server and exit
```

Ilustración 100 ionic puesta en marcha 4

Este comando nos arranca el proyecto en: <http://localhost:8100/>, si así lo hemos configurado.

- `ionic serve -lab`

Este comando nos muestra desde nuestro navegador una vista de cómo quedaría la aplicación en ios y android, realmente útil.

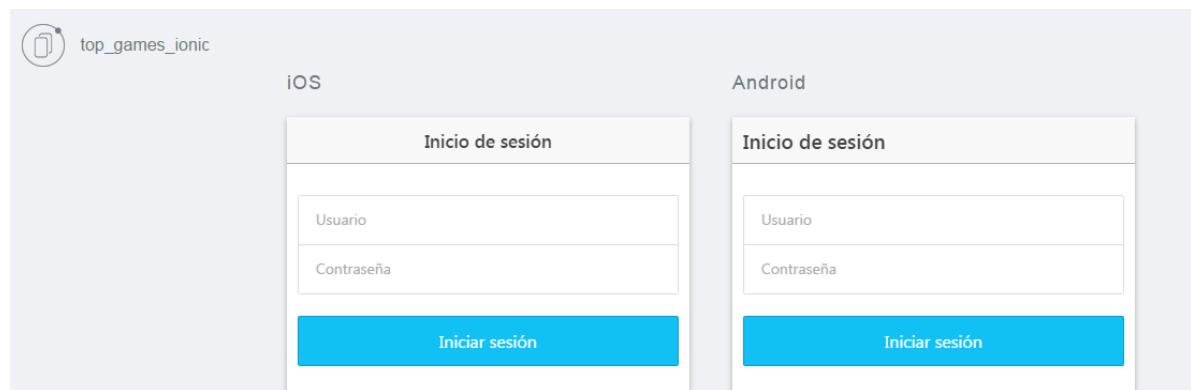


Ilustración 101 ionic puesta en marcha 5

Construir aplicaciones:

Una vez terminado el desarrollo, si queremos probarlo en un dispositivo, debemos generar la aplicación:

```
$ ionic build android
$ ionic emulate android
```

```
$ ionic build ios  
$ ionic emulate ios
```

Extra: generación de iconos para la app:

Documentación: <http://blog.ionic.io/automating-icons-and-splash-screens/>

Ionic tiene una interfaz de línea de comandos muy potente. Gracias a ella, nos permite generar iconos y pantallas de inicio (splash screen). Para ello, tan solo basta con colocar nuestra imagen en la carpeta resources del proyecto:

- icon.png para el ícono
- splash.png para la pantalla de inicio

Y ejecutar el comando:

```
$ ionic resources  
  
//generar sólo ícono  
$ ionic resources --icon  
//generar sólo splash  
$ ionic resources --splash
```

Esto nos genera las carpetas:

- android
- ios

Con sus respectivas carpetas, icon y splash, con las imágenes preparadas para los distintos tamaños de pantalla y densidad de pixeles.

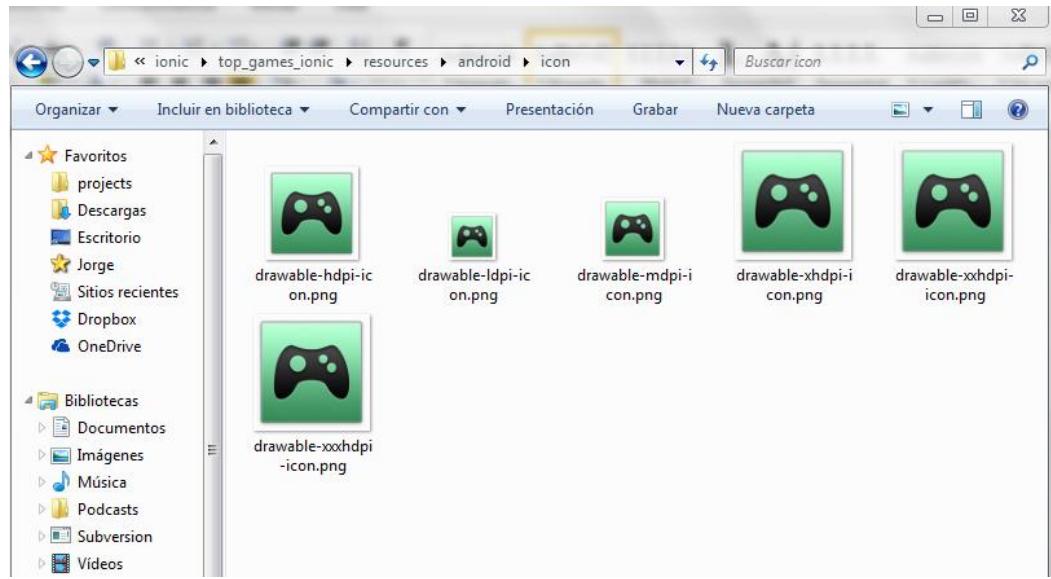


Ilustración 102 ionic puesta en marcha 6

Podemos comprobar que los iconos funcionan tras compilar la aplicación de nuevo (ionic build android por ejemplo) e intentar instalar la apk generada: (platforms\android\build\outputs\apk\android-debug.apk):

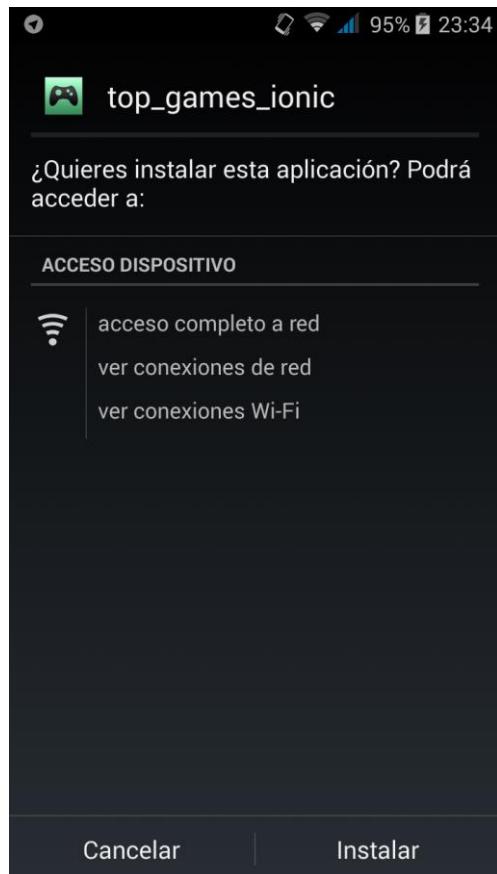


Ilustración 103 ionic puesta en marcha 7

(b) Importar Top Games ionic:

Para importar el proyecto **Top Games** debemos descargar el proyecto de:

https://github.com/jorgelillo7/top_games_ionic o copiarlo desde el cd que incluye esta memoria.

Y guardarla en la carpeta dónde tengamos todos los proyectos de ionic; por seguir buenas prácticas, en nuestro caso C:\ionic. Una vez tengamos el proyecto, debemos arrancarlo con ionic serve o ionic serve -lab

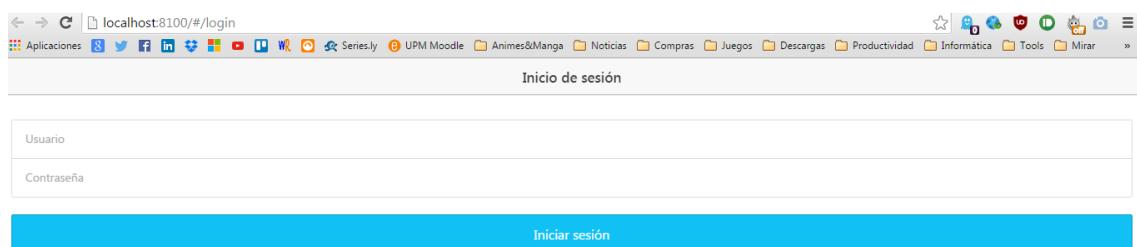


Ilustración 104 ionic TopGames serve

Ilustración 105 ionic TopGames serve --lab

Con el comando serve, conseguimos lanzar la aplicación, pero, para poder interactuar con los datos, debemos tener la aplicación web levantada para poder acceder a la api y obtener los datos. Si no tenemos la aplicación web levantada, no conseguiremos avanzar del login.

Para ello, [ver anexo 1: instalación aplicación web.](#)