26-8-2015

David Letrán González

Miguel Ángel Núñez-Romero Olmo

Documento de Diseño

Proyecto Predicción de Resultados Deportivos

[1. Introducción. 2](#_Toc428712520)

[2. Tecnología 2](#_Toc428712521)

[3. Arquitectura 2](#_Toc428712522)

[4. Implementación 3](#_Toc428712523)

[4.1. Modelo-Vista-Controlador 3](#_Toc428712524)

[4.2. Data-Binding 3](#_Toc428712525)

[4.3. Módulos 4](#_Toc428712526)

[4.3.1. Angular-Translate 4](#_Toc428712527)

[4.3.2. Tc-Angular-ChartJs 4](#_Toc428712528)

[4.3.3. FlowJS 5](#_Toc428712529)

[4.4. Bootstrap 5](#_Toc428712530)

# Introducción.

Éste documento estará dirigido a los desarrolladores del sistema, no a usuarios finales ni a clientes. Contiene información del diseño de la parte correspondiente al front-end del sistema.

# Tecnología

La tecnología escogida para el desarrollo de la interfaz del sistema es AngularJS, un framework de JavaScript que implementa el patrón Modelo, Vista, Controlador (MVC). Está mantenido por Google y dispone de una amplia comunidad que ayuda a su mantenimiento y con una amplia variedad de contribuciones de usuarios. Además se utilizará Bootstrap, un framework HTML y CSS que permitirá adaptar la aplicación para usuarios que accedan desde dispositivos móviles de manera fácil y rápida, añadiendo además sencillez a la hora de crear un diseño atractivo.

# Arquitectura

La interfaz estará basada en el concepto de SPA (Single Page Application), es decir, estará concentrada en una misma página que cambiará de manera dinámica dependiendo de las interacciones del usuario sin necesidad de comunicarse en todo momento con el servidor.

Se dispondrá de una página principal que contendrá elementos comunes para toda la aplicación, contenida en ella se encontrará una zona dedicada a mostrar el contenido dinámico, es decir aquel correspondiente a cada sección de la aplicación. Los elementos comunes serán el menú superior y el pie de página, quedando todo el espacio entre ellos destinado a el contenido dinámico.

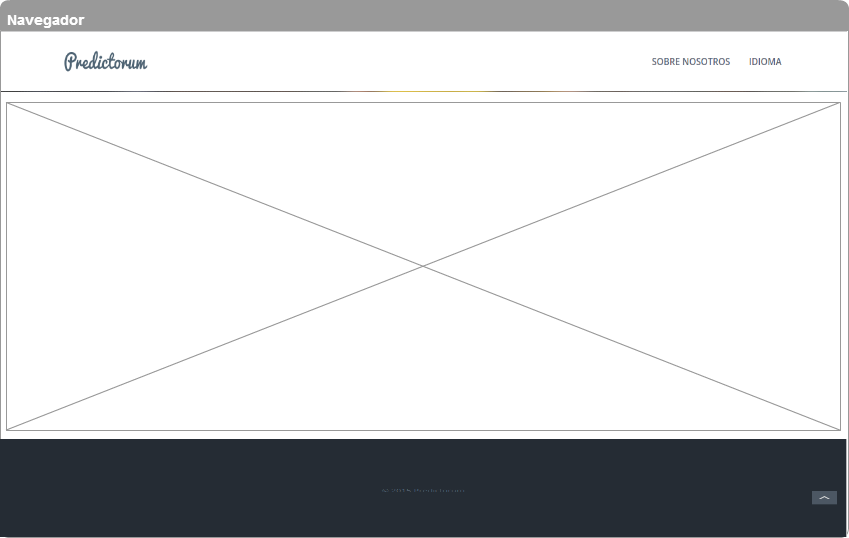


Ilustración : Arquitectura de la interfaz

# Implementación

## Modelo-Vista-Controlador

Para lograr el objetivo marcado se utiliza el patrón Modelo-Vista-Controlador, cada sección de la aplicación dispondrá de su propio controlador y servicio, así como de vistas parciales, las cuales se corresponderán con las distintas secciones. De esta manera cuando un usuario interactúa con la aplicación, sus acciones son recogidas por el controlador que tras tratarlas debidamente se comunica con el servicio para que este se encargue de comunicarse con el servidor en caso de que sea necesario.

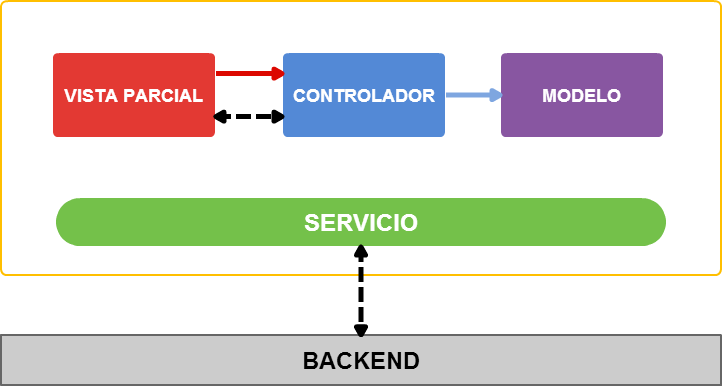


Ilustración : Modelo-Vista-Controlador

## Data-Binding

Esto es así gracias al llamado Data-Binding de AngularJS, el cual actualiza la vista siempre que el modelo cambia o actualiza el modelo cuando se cambia este en la vista. Así cada interacción del usuario será capturada por el controlador que podrá mostrar, ocultar o editar la información presente en el modelo sin necesidad de llamar al servidor para ello. Para implementar esto se utilizan las llamadas “directivas”, marcadores localizados en elementos del DOM que se comportan como atributos HTML y se encargan de relacionar dichos elementos con determinados comportamientos o incluso transformarlo tanto a él como a sus subelementos. Cuando la vista parcial se compila, se analizan todas las directivas y se transforman en los elementos que conforman la vista final a partir de la información contenida en el modelo.

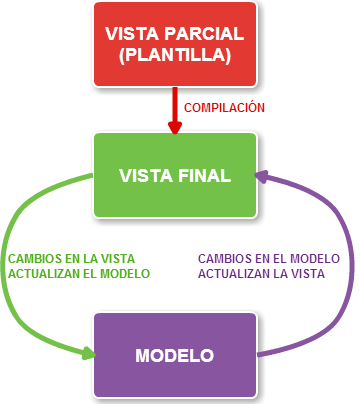


Ilustración : Data-Binding

La combinación de estos elementos provoca una experiencia de uso más rápida al usuario gracias a los cambios asíncronos que se producen en la vista que ahorran tiempos de carga que se añadirían si fuera necesario conectarse continuamente con el servidor.

## Módulos

AngularJS permite la agrupación de funcionalidades en módulos que pueden ser más tarde reutilizados por otras aplicaciones como si de librerías se tratasen. Esta utilidad permite que se hagan muchos aportes a la comunidad que facilitan mucho el desarrollo al utilizar los avances de otros desarrolladores. En nuestro caso nos hemos ayudados de ciertos módulos para implementar varias funcionalidades sin tener que desarrollarlas desde cero.

### Angular-Translate

Se trata de un módulo que permite implementar el cambio de idioma para la aplicación, utiliza archivos en formato JSON que contienen las claves que representan a los mensajes que queremos traducir junto con el valor asociado al idioma en concreto. Utiliza cookies para almacenar el idioma elegido por el usuario y que así no tenga que elegirlo cada vez que visita la aplicación. Permite el cambio de idioma sin necesidad de recargar la página.

### Tc-Angular-ChartJs

Consiste en un módulo que integra ChartJs que permite crear gráficos de todo tipo de manera sencilla, una funcionalidad de mucha utilidad cuando se trata de mostrar estadísticas y datos, algo muy presente en Predictorum.



Ilustración : tc-angular-chartjs

### FlowJS

Consiste en un módulo que permite la subida de archivos con tolerancia a fallos, se divide el archivo en varias partes y si la subida de alguna de ellas fallase se intentaría de nuevo hasta que se consiguiera. En Predictorum se utiliza para la subida de imágenes en la personalización del perfil de los usuarios, permitiendo previsualizarla antes de guardarla.

## Bootstrap

Para conseguir que la interfaz esté disponible en un formato adecuado para cualquier dispositivo independientemente del tamaño de su pantalla se ha utilizado el framework HTML Y CSS, Bootstrap. Esto se consigue mediante las clases predefinidas en Bootstrap que ayudan a definir el espacio que ocuparan los elementos dependiendo del tamaño de la pantalla del dispositivo en el que se está mostrando. Para ello se parte de que el espacio está dividido en 12 columnas, de manera que es posible definir que dos elementos ocupen una fila, teniendo 6 columnas cada uno de ellos, para pantallas grandes pero que ocupe cada uno 12 columnas, apareciendo así uno encima del otro, para una pantalla pequeña.

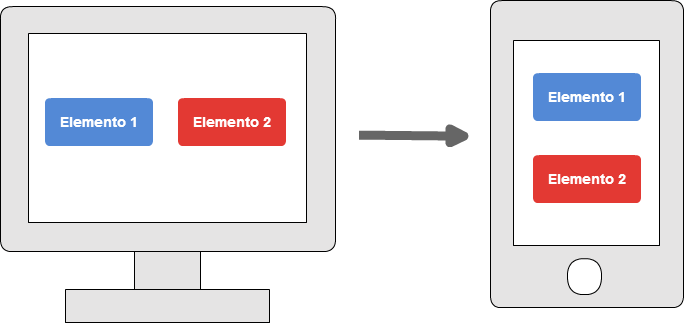


Ilustración 5: Bootstrap