Universidad de Valladolid

E.T.S Ingeniería Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Estudio comparativo entre *frameworks* MVC JavaScript: BackboneJS

Autor:

David Tejedor Zurdo

Tutor (Uva):

Valentín Cardeñoso Payo

Tutor (Everis):

Jose Carlos Jiménez Sánchez

Contenidos:

1 INTRODUCCIÓN

El proyecto se ha realizado por medio de un convenio entre la UVa y la empresa Everis Spain SLU. Esto ha posibilitado tener acceso a recursos, tanto humanos como de documentación, muy provechosos para la realización del mismo.

Este estudio ha sido elaborado en su gran mayoría conjuntamente con otro de los miembros que va a presentar un proyecto mediante dicho convenio, Israel González Aguayo. Además, se ha compartido también una pequeña parte del trabajo con Fernando Martín Sánchez. En la presentación de la estructura que contiene el documento se detallarán las partes en las que ha tenido más peso cada uno.

El objetivo del **TFG** es realizar un análisis de los mejores frameworks[1] y herramientas orientadas a cubrir las necesidades de desarrollo de interfaces de usuario, dentro del amplio conjunto de opciones que ofrece el mercado, de cara a su posible integración en la arquitectura **J-everis**, desarrollado y mantenido por la empresa **Everis**.

El estudio está dirigido esencialmente a la comparativa entre *frameworks* **Javascript** de capa visual basados en el patrón **MVC**, como se explicará más adelante. No obstante, es imprescindible hacer referencia a otros *frameworks* y tecnologías que serán necesarias a la hora de realizar un desarrollo de capa visual en JavaScript.

Para ello se han analizado los siguientes tipos de *frameworks*:

- ***Frameworks* de ejecución:** conjunto de *framework*s para la construcción de interfaces de usuario.
- ***Framework*s de presentación de gráficos:** conjunto de *framework*s utilizados para la generación de gráficos.
- **Herramientas de desarrollo:** herramientas que facilitan el desarrollo de la capa de presentación.

Tras el estudio de estas áreas se procederá a la elección de un *framework* de capa visual con el cual se realizará una pequeña prueba de concepto basada en una aplicación web de funcionalidad básica.

Puesto que este estudio será realizado por dos integrantes, cada uno elegirá un *framework* y realizará la una aplicación piloto desarrollando la parte *front-end*[2] con la tecnología seleccionada, de forma que a posteriori se pueda hacer una comparación entre ambas encontrando ventajas y desventajas a través de un caso práctico.

La parte *back-end*[3] de la aplicación demo será desarrollada con la arquitectura J-everis, que introducimos en el siguiente punto.

El interés de este estudio está en la posibilidad de acoplar una estructura base de estos *frameworks* a las posibilidades que ofrece J-everis.

A continuación se muestra la estructura del documento, indicando brevemente el contenido de cada una de las secciones y la implicación de cada alumno.

[1] Estructura conceptual y tecnológica de soporte definido, compuesta de componentes personalizables e intercambiables, que puede servir de base para la organización y desarrollo de aplicaciones software.

[2] Parte del software que interactúa con los usuarios.

[3] Parte del software que procesa la entrada desde el *front-end* para resolver las peticiones de los usuarios.

- **Capítulo 1**: Se introduce el proyecto, incluyendo los objetivos fijados y su motivación.

  Los objetivos finales y requisitos del proyecto han sido detallados por Israel González, mientras que el contexto y la primera introducción de la memoria ha sido realizada por David Tejedor.

- **Capítulo 2**: Se presenta la arquitectura J-everis con el que se va a desarrollar la aplicación piloto y la cual es el objeto final de este estudio.

  Este apartado ha sido realizado conjuntamente entre los dos miembros, junto a Fernando Martín Sánchez. Es la única parte compartida por los tres alumnos.

- **Capítulo 3:** Se recopila el estado del arte de los diferentes *frameworks* JavaScript MVC existentes, realizando un análisis de los principales y concluyendo en la elección de los dos más relevantes, con los que se realizarán las posteriores aplicaciones *demo*.

  Este apartado contiene toda la información relativa al estudio, el cual se ha realizado de forma conjunta, participando ambos en la búsqueda de información para después unificar ciertos criterios que definan cada tipo de frameworks, así como para asignar las notas a cada tecnología a partir de la información encontrada. Israel González Aguayo ha tenido mayor peso en la elaboración de las tablas comparativas de los *frameworks* gráficos y herramientas de desarrollo, mientras que David Tejedor ha profundizado más en l*os frameworks* visuales y de servidor. En cuanto al estudio e investigación sobre los *frameworks* Client MVC, se ha realizado en conjunto entre ambos miembros.

  La descripción de la metodología empleada ha sido elaborada por Israel González Aguayo, así como el grueso de las descripciones de los *frameworks* Client MVC. Las conclusiones de las diferentes comparaciones han sido elaboradas por David Tejedor Zurdo, y las conclusiones generales se han realizado en conjunto por los dos integrantes.

- **Capítulo 4**: Se describe la aplicación piloto que se ha desarrollado utilizando la arquitectura de J-everis mediante los *frameworks* JavaScript MVC seleccionados.

  David Tejedor Zurdo ha detallado el objetivo y la descripción general de la aplicación, y ha tenido más peso en la elaboración de requisitos y descripción de las entidades de información, mientras que Israel Aguayo ha elaborado la descripción de las interfaces tanto de usuario como de sistema.

  En cuanto al desarrollo de la aplicación demo, cada integrante ha realizado la suya, aunque toda la parte del servidor es la misma para ambos desarrollos, cambiando solo la capa de presentación. En cuanto a la parte común, ambos integrantes han participado activamente en el desarrollo, aunque Israel González ha tenido más peso en la lógica de negocio y definición de servicios web, mientras que David Tejedor ha tenido mayor peso en la capa de persistencia y creación de la base de datos.

  El apartado de detalle de la tecnología Backbone.JS ha sido elaborada en su totalidad por David Tejedor Zurdo, ya que al igual que el desarrollo de la aplicación, es una parte totalmente individual del proyecto.

- **Capítulo 5**: Se presentan las conclusiones del trabajo realizado y se muestra una comparativa de los dos *frameworks* elegidos, según los criterios más relevantes encontrados en la etapa de desarrollo de la aplicación piloto.

  En esta sección cada uno ha detallado la tecnología utilizada para el desarrollo de cada funcionalidad del sistema. Las conclusiones generales han sido redactadas por Israel González Aguayo.

- **Capítulo 6**: Referencias bibliográficas del proyecto.

  Ambos participantes han recopilado las referencias utilizadas durante el estudio del proyecto. David Tejedor se ha encargado de la inserción de estas referencias en el documento.

- **Capítulo 7**: Anexos. Se presentan las tablas originales utilizadas para realizar el estudio.

1.1 CONTEXTO

Las aplicaciones web son cada vez más ricas en funcionalidad, lo que conlleva una mayor carga de trabajo y hace que delegar estas funcionalidades al cliente sea la mejor forma de conseguir que esta carga se vea distribuida entre los usuarios de la web, aligerando el servidor de peticiones y procedimientos que dependen del usuario.

Por este motivo la integración de patrones MVC en JavaScript está siendo cada vez más demandada en la actualidad. Una de las principales razones es que a medida que la funcionalidad de una página web se ve incrementada, el mantenimiento se puede volver muy complicado sin estas herramientas.

Dada la gran tendencia que existe en la actualidad por el uso de este tipo de tecnologías, resultaría muy interesante la integración de alguna de ellas en un *framework* como J-everis, ofreciendo la posibilidad de desarrollar la parte *front-end* de la aplicación utilizando un *framework* JavaScript MVC y dando las mayores facilidades posibles a la hora de comenzar el desarrollo de una aplicación.

1.2 OBJETIVOS DEL PROYECTO

- Realizar un estudio sobre los diferentes *frameworks* de desarrollo de JavaScript existentes y clasificarlos según sus principales características, para posteriormente elegir los dos que más se ajusten a nuestras necesidades.
- Desarrollar una aplicación piloto con la arquitectura de ejecución J-everis y los *frameworks* MVC JavaScript seleccionados.
- Comparar los dos *frameworks* elegidos y sacar conclusiones en base a los resultados obtenidos.

1.3 REQUISITOS GENERALES

Previamente a la realización del proyecto se han fijado una serie de requisitos que deberían cumplirse con el fin de completar los objetivos de forma satisfactoria:

- **Uso de J-everis:** el proyecto ha sido pensado para utilizar la arquitectura J-everis en el desarrollo de los prototipos. Además se ha adoptado la metodología de desarrollo COM-Everis, basada en el Proceso Unificado, aunque no se ha seguido de una forma estricta debido a que el desarrollo ha consistido en una pequeña prueba de concepto con una funcionalidad limitada.
  - **Mantenimiento:** una de las propiedades más importantes de la arquitectura es que ofrece una capa de abstracción con el fin de que programadores con poca experiencia puedan utilizarlo sin necesidad de conocer en profundidad la totalidad de los componentes que utiliza.
  - **Portabilidad:** es importante que la adaptación que se haga de la arquitectura no se acople demasiado a la plataforma de manera que la migración a otra plataforma se pueda realizar con facilidad.
- **Desarrollo de aplicación:** se estudiarán los *frameworks* más relevantes según las características que se consideren más destacables y se seleccionarán aquellos cuyas propiedades sean más importantes para nuestro proyecto.

  Con los *frameworks* seleccionados se desarrollará la capa de presentación de la aplicación demo.

2 J-EVERIS

En esta sección se va a ofrecer una pequeña idea de lo que ofrece **J-everis**, arquitectura empresarial en torno a la cual se basa la realización de este proyecto y sobre la que se desarrollará la aplicación piloto.

Tanto la descripción realizada de la arquitectura como las figuras utilizadas en esta sección han sido extraídas de la documentación oficial de J-everis, almacenada en el portal interno Confluence de Everis y accesible solo para los trabajadores de la empresa.

[http://quark.everis.int/confluence/display/ARCHEX/Framework+j-everis]

2.1 INTRODUCCIÓN

J-everis es una arquitectura Java Empresarial creada, gestionada y mantenida por **Everis**, que proporciona cobertura a las actividades relacionadas con el diseño, construcción, implantación y mantenimiento de sistemas y aplicaciones.

Los principales objetivos que persigue su uso:

- Ofrecer una arquitectura común de construcción de aplicaciones JavaEE.
- Proporcionar un entorno de trabajo, documentación, soporte y mantenimiento de sus componentes.
- Simplificar la complejidad inherente a JavaEE, ofreciendo un marco de referencia de trabajo.
- Ofrecer una solución alineada con los estándares y soluciones más utilizadas para la comunidad *OpenSource*[4].
- Ofrecer una solución abierta que permita agregar e intercambiar cualquier pieza con un coste reducido.

La arquitectura de J-everis se basa en la arquitectura MVC (Modelo Vista Controlador), donde existe un proceso de abstracción que permite dividir una aplicación en componentes lógicos con responsabilidades diferenciadas y que pueden ser desarrolladas incluso por diferentes roles de equipo.

2.2 ARQUITECTURA DE EJECUCIÓN

La arquitectura de ejecución de J-everis es una arquitectura multicapa dividida de la siguiente forma:

- **Capa de presentación:** esta capa es la responsable de la interacción con el usuario a través de una interfaz visual. Ofrece también acceso a servicios de negocio que se encuentran en la capa de servicios. Sigue el patrón MVC, con el fin de desacoplar el diseño de la interfaz (vistas, flujos, etc.), del resto de funcionalidades. Esta capa es opcional, pudiéndose integrar una capa de presentación propia mediante la capa de integración. Estos es precisamente lo que se realiza en el desarrollo del prototipo en este proyecto.
- **Capa de servicios:** esta capa contiene el modelo de la aplicación y todos sus servicios de negocio.
- **Capa de integración:** contiene el conjunto de funcionalidades que permiten a la aplicación integrarse con otras aplicaciones.

[4] Software distribuido y desarrollado libremente.

- **Capa de datos:** contiene la interfaz para conectar la capa de negocios con la base de datos.
- **Funcionalidades transversales:** esta capa ofrece un conjunto de funcionalidades al resto de capas, así como la gestión de excepciones, trazas y *logs*, seguridad, etc.

Una de las características más importantes de J-everis es su modularidad, es decir, cuenta con un conjunto de módulos que se pueden agregar o eliminar a un proyecto J-everis, y cada uno de estos cuenta con un conjunto de funcionalidades que se adaptan al proyecto ya existente pudiendo ampliarlo fácilmente sin afectar al resto de funcionalidades.

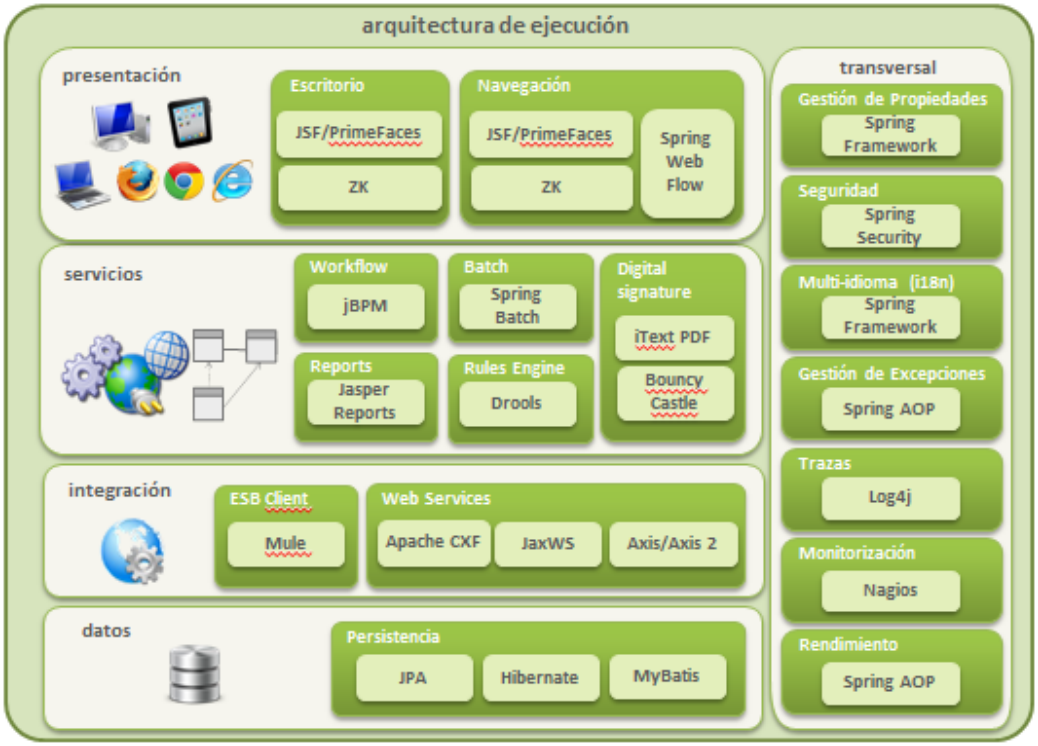


Figura 1: Esquema de los módulos de J-everis

En la figura 1 se muestran los módulos de la versión 3.2 de J-everis, aunque este está en continua evolución para atender las necesidades que puedan demandarse desde los proyectos que utilizan o van a utilizar esta arquitectura.

Para poder empezar a describir los módulos, es necesario dar a conocer que hay dos tipos de proyectos J-everis: **J-everis Web** y **J-everis Business**. El primero contiene la capa web, mientras que el segundo contiene la lógica de negocio y el acceso a base de datos.

También puede crearse únicamente una capa *business* que despliega servicios web para que sean posteriormente consumidos por clientes externos.

2.2.1 Capa de presentación

La capa de presentación, tal y como se ha comentado anteriormente, es la responsable de la interacción con el usuario. Ésta se constituye por dos partes, el escritorio y la navegación:

Figura 2: Esquema de los módulos de la capa de presentación de J-everis

- **Escritorio:** El término abstracto "Escritorio" es dado a la parte visual de la aplicación, es decir, aquello que va a ser mostrador al usuario. Las funcionalidades que maneja son: gestión de vistas, inicio de los flujos de navegación y hace uso de servicios como seguridad y i18n (internacionalización).
- **Navegación:** La parte de navegación es aquella que gestiona qué va a mostrarse al usuario después de que este realice una acción. Para ello, la tecnología que se utiliza en el *framework* es *Spring Web Flow*. Esta tecnología permite definir flujos de navegación, los cuales definen el conjunto de acciones que se realizarán y en qué orden.

La finalidad de este proyecto es que la arquitectura llegue a ofrecer en esta capa la posibilidad de elegir un *framework* MVC, generando las configuraciones necesarias y una base de código genérico que agilice la puesta en marcha de la aplicación y facilite su posterior desarrollo.

2.2.2 Capa de negocio

La capa de negocio es la parte que contiene los servicios de la aplicación y el modelo. En este caso, se definen cinco módulos diferentes e independientes, que podrán ser agregados a cualquier aplicación desarrollada con J-everis:

Figura 3: Esquema de los módulos de la capa de servicios de J-everis

2.2.2.1 Módulo batch

El módulo de *batch* permite a la arquitectura disponer de una herramienta para crear procesos *batch*, es decir, procesos que se ejecutan según programación de forma repetitiva o puntual en el momento de tiempo que se le defina. Este módulo está basado en el *framework Spring Batch*, que ofrece una manera más sencilla de lidiar con los procesos periódicos, proporcionando una interfaz que ofrece una mejor gestión del mantenimiento.

Además de ejecutar trabajos y tareas, es capaz de guardar la contabilidad de todo lo que ha pasado con estos trabajos y tareas en la base de datos, por lo que se tiene *logging/tracing* de todo lo que ha ido sucediendo con los trabajos y tareas en la base de datos.

2.2.2.2 Módulo de reportes

Mediante el módulo de reportes se añade al proyecto la capacidad de generar informes, con la característica de ser independiente de la tecnología de informes que se utilice. El desarrollador solo debe encargarse del diseño de los informes con la tecnología que se haya seleccionado.

J-everis ofrece una implementación con *Jasper Reports*, que nos da la posibilidad de una fácil creación de informes con un amplio abanico de formatos.

2.2.2.3 Módulo de firma digital

El módulo de firma digital añade a una aplicación J-everis las funcionalidades necesarias para trabajar con la propia firma digital. Las funcionalidades que aporta el módulo son las siguientes:

- Firma en cliente: permite firmar cualquier texto o formulario web completo.
- Acuse de recibo de la firma en cliente.
- Validación de *TimeStamp*: mecanismo que permite demostrar que una serie de datos han existido y no han sido alterados desde un instante específico de tiempo.
- Validación de firma y extracción de información del certificado firmante.
- Validación de certificados.
- Firma de documentos en el servidor: mediante esta funcionalidad se pueden sellar archivos PDF en el servidor.
- Sellado de tiempo.

2.2.2.4 Módulo de workflow

El propósito del módulo de workflow es ofrecer una API genérica desacoplada de cualquier motor de workflow. Esta API ofrece los servicios básicos y comunes para los principales motores de workflow del mercado. Su utilización da total libertad para cambiar el proveedor de workflow sin apenas tener que hacer cambios en el código.

El propósito del módulo de workflow contempla la integración de cualquier motor BPM dentro de la arquitectura J-everis para la interactuación de los procesos de negocio dentro de las aplicaciones desarrolladas con esta arquitectura.

Este módulo tiene las siguientes propiedades:

- **Genérico:** engloba acciones comunes a todos los motores de BPM.
- **Independiente:** no está sujeto a ninguna especificación propia de ningún motor de procesos BPM.
- **Alcance del ciclo de vida de las tareas humanas:** los procesos BPM se caracterizan por tener tareas humanas o automáticas. Se abarca el proceso de ejecución de las tareas humanas.
- **Arranque de las instancias de procesos:** puede iniciar instancias de procesos BPM de forma que un actor pueda a través de la capa de presentación realizar el arranque de un proceso BPM.

En el momento en el que se realiza este proyecto, la única implementación que se da del BPM es con jBPM, un motor open source que está adaptado a la arquitectura de j- Everis.

2.2.2.5 Módulo de motor de reglas

El propósito del módulo de motor de reglas es ofrecer una API genérica desacoplada de cualquier motor de reglas. Esta API ofrece los servicios básicos y comunes para los principales motores de reglas del mercado. Su utilización da total libertad para cambiar el proveedor sin apenas tener que hacer cambios en el código. Además se contempla la integración de cualquier motor dentro de la arquitectura J-everis para la ejecución de reglas de negocio.

Las siguientes propiedades son propias del módulo:

- **Genérico:** engloba acciones comunes a todos los motores de reglas.
- **Independiente:** no está sujeto a ninguna especificación propia de ningún motor de reglas.

En el momento en el que se realiza este proyecto, la única implementación que se da del motor de reglas es con *Drools*, totalmente compatible con la implementación del BPM. Hay que tener en cuenta que el uso de ambos módulos viene muchas veces ligado, pues dentro de un proceso de workflow pueden incluirse reglas definidas en el motor de reglas.

2.2.3 Capa de integración

El objetivo de las funcionalidades de la capa de integración es ofrecer servicios para conectar la aplicación que se desarrolle con otras aplicaciones ya existentes, bien sea ofreciendo algún tipo de servicio o bien consumiendo servicios ofrecidos desde otras aplicaciones

Figura 4: Esquema de los módulos de la capa de integración de J-everis

2.2.3.1 Módulo de servicios web

Los módulos de *Web Services* permiten a la arquitectura publicar o consumir servicios web. En la arquitectura J-everis, existen dos módulos de *Web Services*: publicación y consumo.

- **Publicación de Web Services:** este módulo permite publicar servicios web a partir de clases anotadas o mediante la configuración de los ficheros de Spring. Con este fin, se utiliza *Apache CXF*, un *framework Open Source* de servicios web que en este caso se utiliza para publicarlos. Se ha escogido este *framework* por su soporte a muchos estándares de *Web Services* como *SOAP, WS-Addressing, WS-Policy, WS-Security*, entre otros.
- **Consumo de Web Services:** mediante el módulo de consumo de *Web Services* se añade a la arquitectura la posibilidad de utilizar (consumir) servicios web ya existentes y publicados por otras aplicaciones. Gracias a las funcionalidades ya ofrecidas por el módulo, el desarrollador no deberá trabajar en cuestiones complejas como pueden ser el establecimiento de conexiones seguras (SSL) o la conexión a través de un servidor Proxy.

2.2.3.2 Módulo de integración ESB

Un ESB es un "Bus de servicios de empresa" que consiste en un combinado de arquitectura software que proporciona servicios fundamentales para arquitecturas complejas a través de un sistema de mensajes, que forman el bus, basado en las normas y que responde a eventos. Así pues se puede entender que un ESB

proporciona una capa de abstracción construida sobre una implementación de un sistema de mensajes de empresa que permita a los expertos en integración explotar el valor del envío de mensajes sin tener que escribir código.

El propósito del módulo de ESB de J-everis es ofrecer una API genérica desacoplada de cualquier *Bus* de datos. Esta API ofrece los servicios básicos y comunes de comunicación con buses. Su utilización da libertad para cambiar de tecnología sin apenas tener que hacer cambios en el código. De esta forma, el módulo permite:

- Enviar mensajes desde la aplicación a través del bus de manera síncrona y asíncrona.
- Solicitar mensajes a través del bus.

2.2.4 Capa de datos

La capa de datos añade las funcionalidades necesarias para que haya interacción entre la aplicación desarrollada y la base de datos que se vaya a utilizar. Para ello disponemos del módulo de persistencia.

Figura 5: Esquema de los módulos de la capa de datos de J-everis

2.2.4.1 Módulo de persistencia

El módulo de persistencia hace que la aplicación disponga de un conjunto de funcionalidades para que se puedan hacer consultas a la base de datos. Este módulo consta de distintas capas para facilitar la modularización, flexibilidad e incorporación de cualquier tipo de tecnología dependiendo de las necesidades de la aplicación.

Contiene una interfaz base con las operaciones básicas de cualquier servicio de persistencia, además de servicios transversales comunes asociados al servicio de persistencia. Este módulo posee las siguientes características:

- **Interfaz base de persistencia:** esta interfaz proporciona las operaciones básicas comunes asociadas al servicio de persistencia.
- **Configuración de transaccionalidad:** configuración de transaccionalidad mediante *Spring AOP* para las operaciones definidas en el repositorio genérico. Se puede definir el tipo de transaccionalidad a aplicar para cada tipo de operación.
- **Traducción de excepciones:** mediante *Spring AOP* se capturan las excepciones que se puedan producir al ejecutar los métodos del repositorio genérico y se traducen a excepciones J-everis internacionalizadas para su mejor comprensión.
- **Auditoria SQL:** como componente añadido se ha desarrollado una funcionalidad para poder auditar y sacar estadísticas de las consultas a la base de datos.

J-everis consta de implementaciones de persistencia con las tecnologías *JPA, Hibernate* y *MyBATIS*, pero gracias a la arquitectura, añadir una nueva tecnología de persistencia es un proceso sencillo.

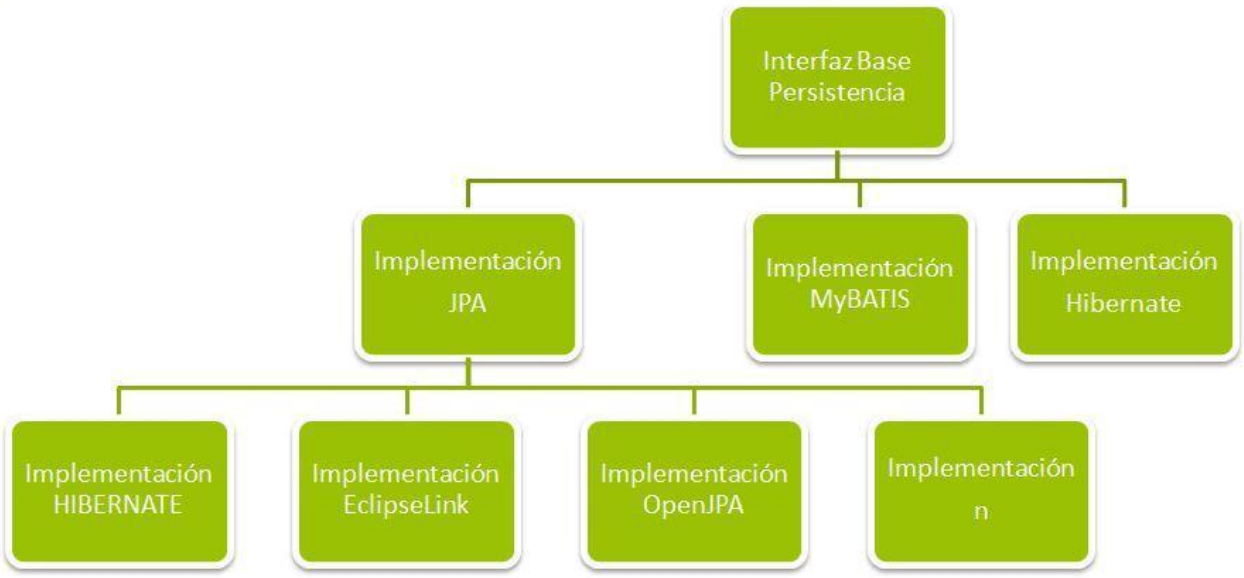


Figura 6: Esquema de la estructura del módulo de persistencia de J-everis

Para la realización del prototipo funcional se utilizará la **implementación Hibernate de JPA**.

2.2.5 Funcionalidades transversales

Las funcionalidades transversales de J-everis son aquellas que están presentes en cualquier proyecto creado con esta arquitectura, de forma que se pueden utilizar siempre y en todos los módulos que se vayan agregando a la aplicación.

Figura 7: Módulos de la capa transversal de J-everis

2.2.5.1 Módulo de seguridad

El módulo de seguridad tiene como propósito principal gestionar la autenticación y autorización de los usuarios a la aplicación desarrollada con J-everis. La autenticación permite comprobar que el usuario es quien dice ser, mientras que con la autorización se comprueba que el usuario tenga permiso para acceder al recurso que solicita.

Para el desarrollo de este módulo se ha utilizado Spring Security, el cual nos permite definir de una forma sencilla roles de usuario, a los que se les puede asignar una serie de permisos de acceso a los diferentes recursos de la aplicación. Además, puesto que pertenece al *framework* de Spring, es compatible con el resto de componentes.

2.2.5.2 Módulo de monitorización

El módulo de monitorización dota a las aplicaciones de la capacidad de ser monitorizadas a través de un sistema externo mediante JMX (*Java Management Extensions*[5]). El módulo desarrollado para J-everis es independiente del sistema de monitorización que se quiera utilizar, aunque será necesario agregar la implementación para el sistema escogido por parte de las aplicaciones.

J-everis consta de una implementación para un sistema de monitorización llamado *Nagios*. Esta implementación cubre los siguientes aspectos:

- **Monitorización de arquitectura**
  - Estado y tiempo de respuesta de los diferentes *backends* a los que se tiene acceso desde la aplicación. Es la propia aplicación quien realiza la monitorización de su conectividad.
  - Listado de EJBs visibles desde la aplicación.
- **Monitorización de la aplicación:** hace referencia a las conexiones externas a la aplicación, así como la conexión a base de datos.
  - Número de sesiones en la aplicación.
  - Número de usuarios autenticados conectados a la aplicación.
  - Número de excepciones producidas en la aplicación.

2.2.5.3 Módulo de rendimiento

El objetivo principal del módulo de rendimiento es servir de ayuda en el desarrollo de aplicaciones, permitiendo identificar los métodos más costosos en cuanto a tiempo de respuesta. De esta forma, la aplicación va guardando los tiempos de retardo que cuesta la realización de una función y el tiempo de respuesta de la base de datos, de forma que estos tiempos pueden ser listados.

2.3 ARQUITECTURA DE DESARROLLO

La arquitectura de desarrollo de J-everis pretende ayudar al desarrollador a crear y desarrollar aplicaciones basadas en J-everis. Para ello, el entorno de trabajo que se sugiere consta del uso de las siguientes herramientas:

· Eclipse Indigo **· Maven** **· Tomcat**

[5] Tecnología que aporta herramientas para manejar y monitorizar aplicaciones

En la web de J-everis se ofrece un entorno ya configurado con las herramientas mencionadas, de forma que el desarrollador solo tiene que descargárselo, instalarlo y empezar a programar.

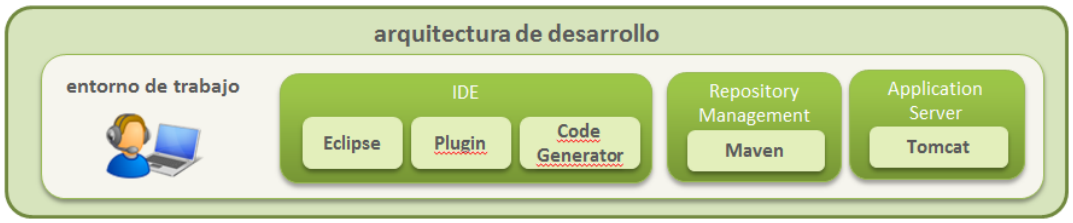


Figura 8: Esquema de la arquitectura de desarrollo de J-everis

Aunque se aconseja el uso del servidor de aplicaciones Tomcat, el *framework* J-everis es compatible con JBoss, Weblogic, WebSphere y Glassfish.

2.3.1 *Plugin* para eclipse

El uso del IDE Eclipse, viene ligado al uso del *plugin*[6] de J-everis desarrollado para eclipse. Mediante este *plugin*, se pueden crear proyectos J-everis, agregar o eliminar módulos y filtros a un proyecto J-everis, y utilizar el generador de código que se verá más adelante. La ventaja del uso de este módulo es que para cada una de sus funcionalidades da una configuración válida, de forma que si se agrega un módulo, el proyecto quedará configurado y seguirá siendo usable aunque no se dé una configuración adaptada a la aplicación. De forma similar, al eliminar un módulo el proyecto, este seguirá siendo consistente y se eliminarán tanto las dependencias como los archivos de configuración que estaban únicamente relacionados con el módulo eliminado.

[6] Complemento que puede incorporarse a una aplicación para aportarle nueva funcionalidad.

3 COMPARATIVA DE *FRAMEWORKS*

En esta sección se realiza un estudio de los principales *frameworks* JavaScript y otras herramientas que son de utilidad a la hora de realizar desarrollos utilizando estas tecnologías.

Este análisis se ha centrado principalmente en los *frameworks* que se ajustan al modelo *Client MVC*[7], aunque nos ha parecido conveniente analizar también *frameworks* asociados a otros modelos, tanto para ampliar la visión ante diferentes requerimientos y situaciones. También se ha realizado un estudio de *frameworks* orientados a funcionalidades muy concretas que son de mucha utilidad como complemento a los mencionados anteriormente. Además, se han analizado también diferentes herramientas de desarrollo que resultan muy beneficiosas a la hora de realizar desarrollos en lenguaje *JavaScript.*

Los tres tipos de *frameworks* comparados han sido:

- ***Frameworks* de ejecución:** conjunto de *frameworks* de mercado para la construcción de interfaces de usuario.
- ***Frameworks* de presentación de gráficos:** conjunto de *frameworks* utilizados para la generación de gráficos.
- **Herramientas de desarrollo:** herramientas que facilitan el desarrollo de la capa de presentación.

3.1 METODOLOGÍA

A continuación se detalla la metodología que se ha utilizado para llevar a cabo la comparativa de los diferentes tipos de *frameworks*, así como una explicación del funcionamiento de las tablas que contienen los datos de las comparaciones.

Durante la comparativa se harán referencias a estas tablas, que serán adjuntadas en el apartado Anexos, al final del documento.

En primer lugar se han seleccionado los *frameworks* más relevantes de las distintas categorías: ***frameworks* de ejecución, *frameworks* gráficos y herramientas de desarrollo.**

Una vez seleccionados los *frameworks* sobre los que realizar la comparativa, se han definido los criterios a analizar y se han seleccionado las características más relevantes para cada tipología. Estas características han sido agrupadas en cuatro áreas:

- **Ejecución:** criterios fundamentados en las capacidades del *framework*.
- **Desarrollo:** criterios que facilitan el trabajo del desarrollador de la aplicación, desde la escritura del código hasta las pruebas, automatización de tareas o tecnologías a emplear.
- **Operación:** atiende a criterios para el mantenimiento de la aplicación.
- **Soporte:** criterios que hacen referencia a la calidad de la documentación, soporte del proveedor, soporte de la comunidad de usuarios, existencia de certificados, licencia del producto y evolución histórica del *framework*.

[7] Frameworks de la capa del cliente, es decir, de la capa de presentación, que siguen el patrón MVC o similar.

Para cada tipo de *framework* a analizar se han definido las siguientes tablas:

- **Análisis:** existe una tabla de este tipo por cada *framework*, en la que se detalla la puntuación obtenida en cada criterio junto a una explicación aclaratoria de dicha nota.
- **Pesos:** en esta tabla se pondera la puntuación de cada criterio con el peso asociado a este.
- **Resultados:** tabla resumen con la puntuación final de cada *framework* en cada criterio, junto a la nota final del *framework*.

La nota asignada a cada *framework* para cada una de las características analizadas está definida por los siguientes niveles:

Cobertura		Nivel
N	0%	**Ninguna**
B	25%	**Bajo**
M	50%	**Medio**
A	75%	**Alto**
T	100%	**Total**

Una vez asignado el nivel, este será multiplicado por el peso asociado al criterio de comparación en cuestión, obteniendo como resultado una nota.

Los pesos están comprendidos en un rango del 1 al 3:

Coeficiente de Peso	
Imprescindible	**3**
Importante	**2**
Aconsejable	**1**

Una vez realizado este proceso para cada criterio, se calcula la media ponderada de cada *framework*, sumando las notas obtenidas y dividiendo el resultado entre la suma total de los pesos. Ésta será la nota final del *framework* en la comparativa.

3.2 *FRAMEWORKS* EJECUCIÓN

Se han seleccionado un conjunto de *frameworks* que cubran los principales representantes de los diferentes enfoques y capacidades. Dependiendo de las características de los *frameworks* analizados se han dividido en:

- **Estructurales:** permiten crear la estructura de una aplicación en la capa de presentación. En un enfoque **MVC** nos proporcionan al menos el modelo y el controlador. Cubren aspectos como navegación, invocación a servicios y modelo.
- **Visuales:** proporcionan componentes visuales y capacidades como validación de formularios, drag and drop. En un enfoque **MVC** nos proporcionan la vista.

Aunque algunos de ellos pueden proporcionarnos todo, nos ofrecen mayor valor en uno de los dos apartados.

En el apartado de *frameworks* estructurales se han seleccionado *frameworks* que cubren los siguientes enfoques:

- **Client MVC:** *frameworks JavaScript* que cumplen con el patrón modelo-vista-controlador. Se ejecutan en el navegador y se comunican con una capa de servicios para obtener los datos. Ejemplos: Angular, Backnone, Ember, ExtJS.
- **Server MVC:** *frameworks* java que cumplen con el patrón modelo-vista-controlador. Se ejecutan en el servidor que genera html. Ejemplos: zk, vaadin.

3.2.1 *Frameworks* Estructurales

Los *frameworks* modernos basados en el patrón Modelo-Vista-Controlador ofrecen al desarrollador una manera sencilla de organizar el código.

El patrón está basado en las siguientes capas:

- **Modelo:** representa el conocimiento del dominio específico y los datos utilizados en la aplicación. Se pueden ver como los 'tipos' de datos que pueden ser modelados, como un Usuario, Foto, Nota, etc. El modelo debería notificar su estado actual a las vistas que estén observándolo.
- **Vista:** se encarga de mostrar al usuario la información del modelo en un formato adecuado para que éste pueda interactuar con la aplicación. Por tanto las vistas tienen que observar a los modelos, aunque no se comunicarán directamente con ellos.
- **Controlador:** es el encargado de manejar las entradas a la aplicación, realizadas por el usuario e invoca peticiones al 'modelo' cuando se hace alguna solicitud sobre la información (por ejemplo, editar un documento o un registro en una base de datos). En este sentido las vistas se pueden considerar las encargadas de manejar las salidas.

Los *frameworks* MVC **no** siempre siguen literalmente este patrón, pero son muy útiles para ayudar a estructurar el código de la aplicación. Por ejemplo el *framework* Backbone.js, el cual analizaremos posteriormente, atribuye la responsabilidad del Controlador a la propia Vista, mientras que otros directamente añaden componentes propios intentando hacer el patrón más efectivo.

Por esta razón, se considera que estos *frameworks* siguen patrones **MV***, ya que siempre tendremos Modelo, Vista y algo más, que se presenta en diferentes formas dependiendo del *framework* utilizado.

3.2.1.1 Evolución de los *frameworks* de presentación

En los últimos años, los *frameworks* creados para la capa de presentación únicamente se centran en modelar la vista, lo que hace que su customización requiera un alto coste y que sean dependientes de un servidor, aunque cuentan con una fácil integración con los *frameworks* MVC y tienen un alto grado de madurez. Algunos ejemplos son Jquery y PrimeFaces.

Más adelante han surgido los llamados Servidores MVC, que requieren de un servidor de aplicaciones y ofrecen gran flexibilidad en la integración con otros *frameworks* como Jquery o JSF. Estos disponen también de un alto grado de madurez.

Por último nos encontramos con la tendencia actual, en dirección a los llamados **clientes MVC**. Con un total desacoplamiento entre la presentación y la capa de negocio, requieren de un servidor menos pesado ya que parte de la lógica de negocio se realiza directamente en la capa de presentación, además facilitan el despliegue en *cloud*[8] de las aplicaciones.

[8] Término asociado a la computación en la nube, que permite ofrecer servicios de computación a través de Internet

Algunos ejemplos de estos *frameworks* son **Angular**, **Ember**, **BackBone**, **Knockout** o **JavaScriptMVC**.

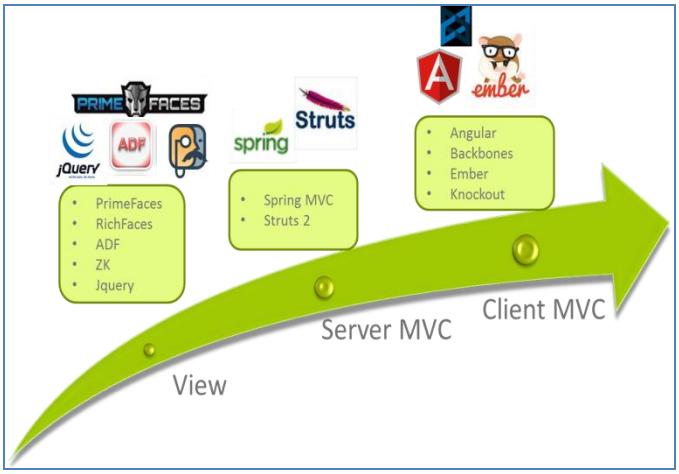


Figura 9: Evolución frameworks de presentación

Los más relevantes son los tres primeros y en especial Angular, que mantenido por Google, no para de ganar funcionalidades y seguidores. Este *framework* ofrece multitud de funcionalidades muy potentes y permite una estructuración del código muy clara. Debido a ese gran apoyo por parte de la comunidad dispone de mucha documentación a pesar de su juventud.

3.2.1.2 Por qué usar un *framework* MVC JavaScript

Si tenemos que construir una aplicación de una única página dinámica mediante JavaScript es muy probable que nos encontremos creando componentes como los que existen en *frameworks* MV* como Backbone o Angular.

Estos *frameworks* nos ahorran gran parte del trabajo estructurando la aplicación. Son de mucha ayuda para realizar una página web con gran funcionalidad escribiendo mucho menos código del que tendríamos que escribir sin disponer de estos *frameworks*, haciendo posible que sólo tengamos que centrarnos en la lógica de negocio de nuestra aplicación, y no prestar tanta atención a partes del código repetitivas y de configuración.

En este sentido, algunos *frameworks* ofrecen más funcionalidad que otros, como Angular, pero es importante también tener en cuenta que en este caso tendremos menos "libertad", ya que será mucho más fácil hacer las cosas para las que el *framework* está preparado, pero mucho más difícil las que no están contempladas por el mismo. Lo contrario nos ocurre con *frameworks* más ligeros como Backbone.

Este tipo de *frameworks* son también muy útiles para aplicaciones que se comunican con una API o un servicio *back-end*, donde el trabajo más pesado es la manipulación de los datos que se muestran en el navegador. En este sentido es muy importante que el *framework* utilizado disponga de "Data-Binding".

Por el contrario, si la aplicación basa en el servidor la mayor parte del trabajo pesado de vistas o páginas y sólo se utiliza una pequeña parte de JavaScript para hacer la página algo más interactiva y dinámica, utilizar un *framework* MV* sería excesivo e innecesario.

En resumen, elegir un buen *framework* que se adecúe a las necesidades de la aplicación hará posible que ésta sea desarrollada en mucho menos tiempo, aportando código, una base de la estructura a utilizar y ofreciendo soluciones a problemas a los que habría que dedicar tiempo para poder solucionar sin esta ayuda.

3.2.1.3 Cómo seleccionar un *framework* de este tipo

Existen diferentes cuestiones a tener en cuenta a la hora de seleccionar un *framework* para nuestra aplicación. Ya nos hemos planteado algunas de ellas anteriormente, pero ahora veremos diferentes preguntas que son imprescindibles solventar antes de decantarse por uno u otro.

¿De qué es el *framework* capaz?

Es importante dedicar tiempo a revisar las características oficiales que ofrece, así como su código fuente para asegurarnos de que encaja con las necesidades de la aplicación.

Sería muy recomendable que, antes de adentrarse en el desarrollo de una gran aplicación con uno de estos *frameworks*, se investigue con ellos y se realicen algunas pruebas en modo de pequeñas aplicaciones para confirmar si cumple con nuestros requisitos.

¿Ha sido probado en un entorno de producción?

Es muy interesante también saber si los *frameworks* que estamos pensando elegir han sido utilizados por los desarrolladores para construir y desplegar aplicaciones de gran tamaño que sean accesibles al público. Por ejemplo, podemos encontrar una gran lista de ejemplos de este tipo con Backbone, como los conocidos SoundCloud o LinkedIn. JavaScriptMVC se ha utilizado para aplicaciones de gran potencia en IBM entre otros, y Angular se ha utilizado para realizar productos de Google como Youtube mobile.

¿Es el *framework* suficientemente maduro?

Si se elige un *framework* en desarrollo y poco probado, existe el riesgo de tener que hacer correcciones y refactorizaciones en el código por no tener la suficiente estabilidad. Si va a ser usado para un entorno de producción habrá que tener especial cuidado en este aspecto.

Además, algo muy importante a tener en cuenta es que un *framework* más maduro siempre dispondrá de mayor cantidad de documentación y apoyo de su comunidad que facilitará el desarrollo de la aplicación.

¿Flexible o rígido?

En este caso no es posible decir si un tipo y otro es mejor o peor, simplemente es necesario saber qué se está buscando. Por un lado, un *framework* más rígido siempre sugerirá realizar las cosas de una manera específica y ofrecerán facilidades para hacerlo así.

3.2.1.4 Criterios de comparación de los *frameworks* de ejecución estructurales

A continuación se expone un análisis más exhaustivo de los *frameworks* atendiendo a los siguientes criterios:

- Dentro del área de **ejecución** los criterios son:
  - **Componentización:** riqueza de componentes ofrecidos por el *framework*, funcionalidad de los componentes y capacidad de personalización.
  - **Navegación:** capacidad del *framework* para enrutar servicios. otros frontales.
  - **Integración otros frontales:** capacidad para integrar *frameworks* de terceros.
  - **Manejo de Eventos:** eventos ofrecidos por el *framework* al interaccionar con la aplicación.

- **Seguridad:** gestión de la autenticación y autorización de los usuarios de la aplicación. La autenticación permite comprobar que el usuario es quien dice ser, mientras que con la autorización se comprueba que el usuario tenga permiso para acceder al recurso que solicita.
- **Gestores comunes:** gestión que ofrece el *framework* relativo a captura de errores, gestión de sesiones y gestión de la configuración.
- **Multidispositivo/Multinavegador:** soporte de los diferentes navegadores y dispositivos.

- Dentro del área de **desarrollo** los criterios son:
  - **Herramientas de desarrollo:** herramientas que facilitan el desarrollo como *plugins*, *toolkits* y compatibilidad con los diferentes IDE's.
  - **Herramientas de test:** capacidades que permiten acciones de *debug*, trazado, medición de rendimiento.
  - **Herramientas de despliegue:** facilidad para el despliegue de la aplicación que utiliza el *framework*. Analizando el tamaño de la librería, las dependencias de terceros que tenga.
  - **Facilitadores:** plantillas, proyectos demo, para acelerar la fase inicial.
  - **Análisis de la calidad:** análisis automático de la calidad del sistema.
  - **Independencia del servidor:** capacidad de la capa del cliente para poder trabajar con un servidor de forma transparente a la tecnología de este.
- Dentro del área de **operación** los criterios son:
  - **Monitorización de la aplicación:** herramientas ofrecidas por el *framework* para la monitorización de aplicación.
  - **Tareas de mantenimiento:** herramientas que porporciona el *framework* para realizar *backups*, ejecución de *jobs* y explotación de *logs*.
- Dentro del área de **soporte** los criterios son:
  - **Soporte Proveedor:** documentación disponible del *framework*, certificaciones disponibles, soporte de la comunidad y foros.
  - **Licencia del producto:** condiciones económicas del uso de las herramientas.
  - **Evolución:** grado de actualización, incorporación de nuevas funcionalidades y continuidad.

3.2.1.5 Tabla comparativa *frameworks* Estructurales Client MVC

Área	Angularjs	Backbone	Capuccino	Ember	Ext JS	JQuery	Knockout	SproutCore
Ejecución	**68%**	**61%**	**61%**	**69%**	**69%**	**47%**	**50%**	**69%**
Componentización	50%	50%	50%	75%	75%	75%	0%	75%
Navegación	75%	75%	75%	75%	75%	25%	50%	75%
Integración otros frontales	50%	50%	50%	50%	50%	50%	50%	50%
Manejo de Eventos	75%	75%	75%	75%	75%	25%	50%	75%
Seguridad	75%	50%	50%	50%	50%	50%	50%	50%
Gestores comunes (errores, sesiones,configuración)	75%	50%	50%	75%	75%	25%	75%	75%
Multidispositivo/Multinavegador	75%	75%	75%	75%	75%	75%	75%	75%
Desarrollo	**73%**	**77%**	**50%**	**67%**	**83%**	**77%**	**71%**	**71%**
Herramientas de desarrollo	75%	75%	25%	50%	100%	75%	50%	50%
Herramientas de test	75%	75%	50%	75%	75%	75%	75%	75%
Herramientas de despliegue	75%	75%	25%	75%	75%	75%	75%	75%
Facilitadores (Plantillas, proyectos demo)	75%	75%	50%	50%	75%	75%	75%	75%
Analisis de la calidad (automática)	25%	50%	25%	50%	50%	50%	50%	50%
Independencia del servidor	100%	100%	100%	100%	100%	100%	100%	100%
Operación	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**
Monitorización de aplicación	0%	0%	0%	0%	0%	0%	0%	0%
Tareas de mantenimiento (Jobs, backups y explotación de logs)	0%	0%	0%	0%	0%	0%	0%	0%
Soporte	**75%**	**75%**	**34%**	**59%**	**44%**	**75%**	**69%**	**50%**
Soporte proveedor	75%	75%	25%	50%	25%	75%	50%	50%
Licencia producto	75%	75%	50%	75%	75%	75%	75%	75%
Evolución	75%	75%	25%	50%	25%	75%	75%	25%
Media	**54,03%**	**53,26%**	**36,37%**	**49,03%**	**48,97%**	**49,79%**	**47,48%**	**47,65%**

Los resultados de esta tabla han sido generados a partir de las tablas adjuntadas en el Anexo 7.1.1

3.2.1.6 Descripción de los *frameworks* Client MVC

En esta sección mostraremos una breve descripción de cada *framework* y las principales ventajas y desventajas de cada uno.

3.2.1.6.1 Angularjs

AngularJS es un *framework* de JavaScript de código abierto, mantenido por Google, que ayuda con la gestión de lo que se conoce como aplicaciones de una sola página, que extiende el tradicional HTML con etiquetas propias.

[https://angularjs.org/]

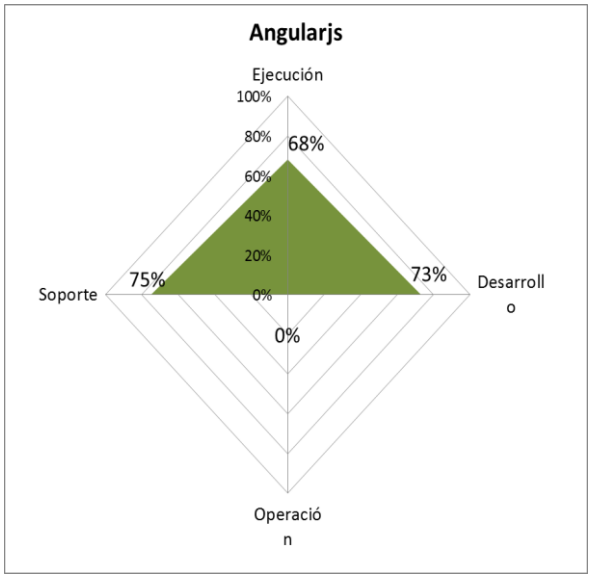
Ventajas:

- ✓ Ligero y con buena gestión de dependencias.
- ✓ Potente sistema de plantillas, extendiendo vocabulario de html básico.
- ✓ El concepto de directivas, que permite crear nuevos *tags* customizados que incorporan tanto funcionalidad como capa visual.
- ✓ Posee un potente enlace de interfaces (UI-Binding).
- ✓ Posee buenas herramientas para hacer *debug*.
- ✓ Inyección automática de dependencias.
- ✓ Desacoplamiento del DOM de javascript
- ✓ Internacionalización i18n y l10n.

Desventajas:

- Curva de aprendizaje muy elevada.
- La escritura de directivas es compleja, es la parte más difícil de escribir código en Angular.

En la siguiente gráfica se muestra una foto de las áreas en las que destaca frente a las que es más débil. Hay que tener en cuenta que no todas las áreas tienen la misma importancia, como se puede ver en los pesos asignados a los criterios de la comparación.



3.2.1.6.2 BackboneJS

Backbone es una herramienta de desarrollo/API para el lenguaje de programación Javascript con un interfaz RESTful por JSON , basada en el paradigma de diseño de aplicaciones Modelo Vista Controlador. Está diseñada para desarrollar aplicaciones de una única página y para mantener las diferentes partes de las aplicaciones web (p.e. múltiples clientes y un servidor) sincronizadas.

Es de gran utilidad para tener la capa de presentación correctamente modulada y con una estructura claramente definida que hace que la aplicación sea mucho más fácil de mantener, mejorar y reutilizar.

Estas características son tremendamente ventajosas en el contexto de este proyecto, ya que se persigue poder generar automáticamente una base que haga más fácil el desarrollo de aplicaciones con esta tecnología. Aunque si bien ofrece facilidades para poder crear esta estructura y separar correctamente la aplicación en distintos módulos, ésta no ofrece una estructura claramente definida y deja que sea el desarrollador el que tome las decisiones oportunas.

Además de la fuerte comunidad que respalda esta tecnología, BackboneJS puede presumir de haber sido utilizado en muchos sitios web de renombre, como son *LinkedIn*, *WordPress.com* o *Foursquare*.

[http://backbonejs.org/]

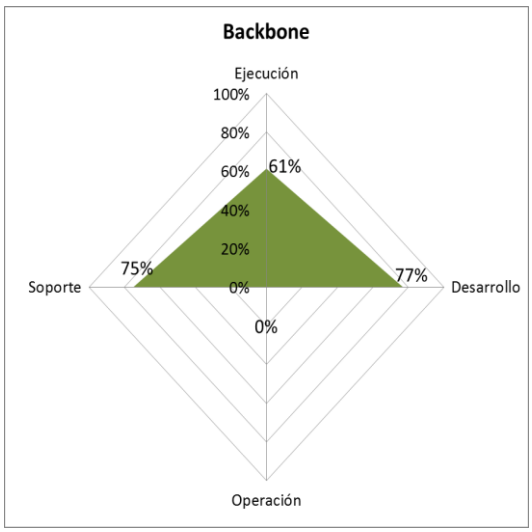
Ventajas:

- ✓ Posee una comunidad muy fuerte.
- ✓ Existe mucha documentación de calidad.
- ✓ Clara estructura
- ✓ Permite modularizar la capa de presentación
- ✓ Simplicidad, sólo posee 4 componentes básicos *(Collection, Model, View, Router).*
- ✓ Es muy fácil inicializarse en él.
- ✓ Es muy customizable.

Desventajas:

- Debido a la ligereza del *framework*, hay que repetir tareas similares en multitud de ocasiones.
- No posee una estructura rígida y definida.

Gráfica de las áreas en las que destaca frente a las más débiles, siempre teniendo en cuenta los pesos asociados:



3.2.1.6.3 Capuccino

Capuccino es un *framework* open source que utiliza tecnologías estándares como Javascript y nos da la posibilidad de desarrollar aplicaciones web que serán prácticamente iguales a una aplicación de escritorio.

Este *framework* fue implementado usando el lenguaje de programación Objective-J, que fue modelado en base al Objective-C utilizado por Cocoa. Todo el conocimiento de Objective C se traslada al navegador sin tener que aprender toda una nueva tecnología, es decir, Cappuccino es Cocoa para la web. Capuccino es una utilidad interesante para realizar aplicaciones complejas de una forma verdaderamente simple.

[http://www.cappuccino-project.org/]

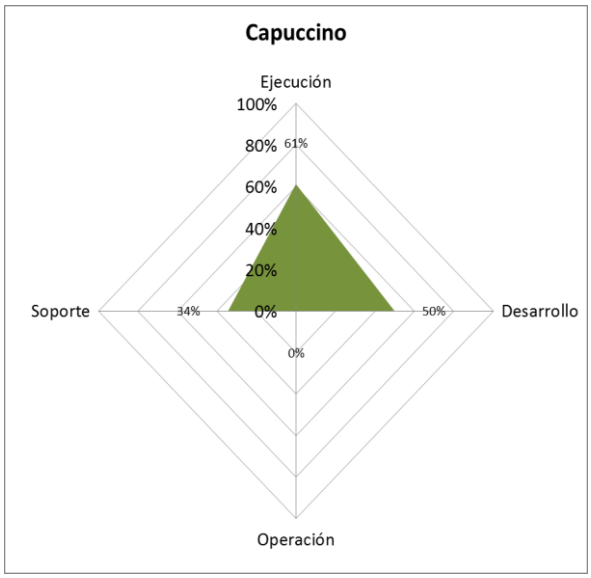
Ventajas:

- ✓ Los programas escritos con Objective-J son interpretados en el cliente por lo que no se necesitan ni compilaciones ni *plugin*s.
- ✓ Sintaxis que resultará familiar a aquellos acostumbrados al Objective-C que se usa con Cocoa.

Desventajas:

- Poco extendido y con comunidad débil.
- Documentación de calidad difícil de conseguir.

Gráfica de las áreas en las que destaca frente a las más débiles, siempre teniendo en cuenta los pesos asociados:



3.2.1.6.4 Ember

Ember.js está catalogado como unos de los principales *framework* a en el mundo de JavaScript ya que permite a los desarrolladores crear aplicaciones de una sola página (*single-page*) escalables.

[http://emberjs.com/]

Ventajas:

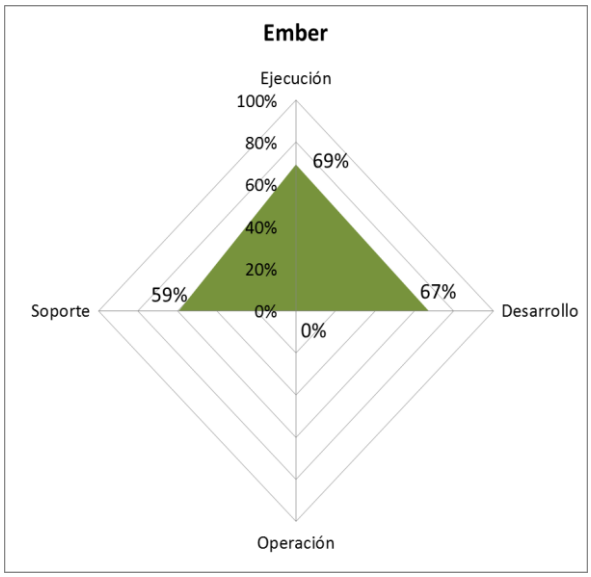
- ✓ Intuitiva separación entre interfaz y controlador.
- ✓ Complejas funciones con código simple.
- ✓ Estructura determinada y consistente.
- ✓ Sistema de plantillas extremadamente rico, con vistas compuestas y enlace de interfaces (UI-Binding).

Desventajas:

Su versión 1.0 estable es relativamente nueva (31/08/2013).

Dificultad para encontrar buena documentación, debido a que los radicales cambios de la versión 1.0.

Gráfica de las áreas en las que destaca frente a las más débiles, siempre teniendo en cuenta los pesos asociados:



3.2.1.6.5 Ext JS

Framework JavaScript para el desarrollo de aplicaciones web interactivas usando tecnologías como AJAX, DHTML y DOM. Fue desarrollada por Sencha.

Originalmente construida como una extensión de la biblioteca YUI, en la actualidad puede usarse como extensión para las bibliotecas jQuery y Prototype. Desde la versión 1.1 puede ejecutarse como una aplicación independiente.

[http://www.sencha.com/products/extjs/]

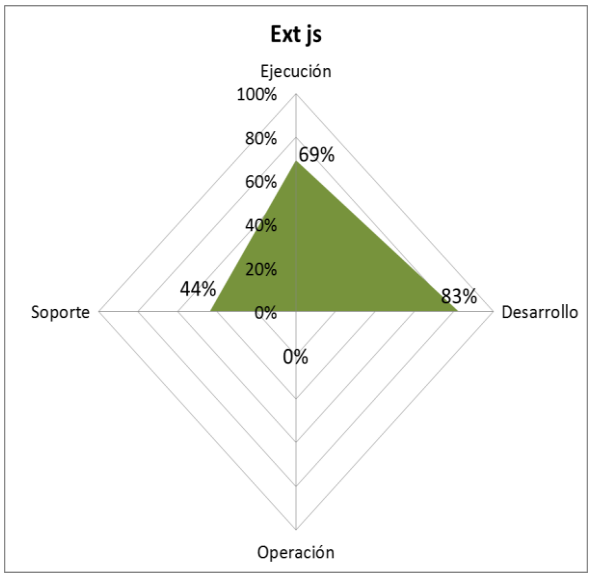
Ventajas:

- ✓ Independiente y adaptable a diferentes *frameworks* (prototyoe, jquery, YUI).
- ✓ Ofrece una gran cantidad de widgets para crear interfaces de usuario complejas e interactivas.
- ✓ El API es homogeneizado independientemente del adaptador usado, los controles siempre se verán igual.
- ✓ Extensa comunidad de usuarios.

Desventajas:

- Librería muy pesada que afecta al rendimiento.
- Licencia propietaria (GPL v3).

Gráfica de las áreas en las que destaca frente a las más débiles, siempre teniendo en cuenta los pesos asociados:



3.2.1.6.6 JQuery

Jquery es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.

[http://jquery.com/]

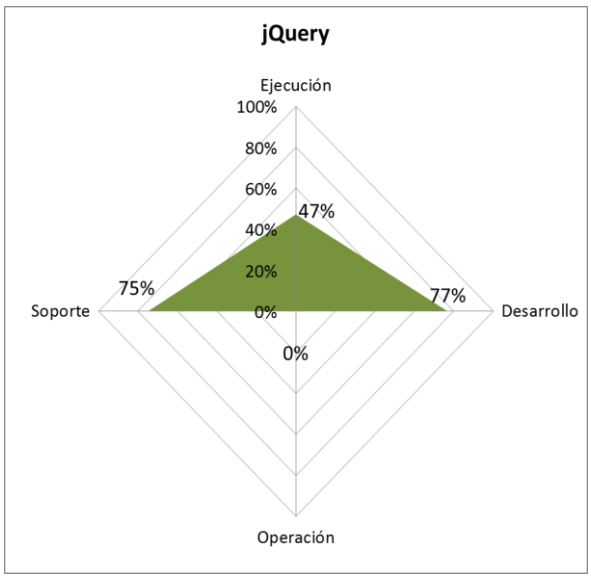
Ventajas:

- ✓ Es Open Source.
- ✓ Se integra con AJAX.
- ✓ Se pueden agregar otros *plugin*s.
- ✓ Extensa documentación de gran calidad.
- ✓ Compatibilidad crossbrowser (multi-navegador).
- ✓ Reutilización de código y la capacidad de abstraerlo en capas (por ejemplo definir todos los eventos en un bloque y no embebido en los tag html).

Desventajas:

- Es difícil de mantener a largo plazo.
- Mucha libertad de codificación.
- Su código principal es difícil de entender.
- Su CSS es complejo a la hora de manipular.

Gráfica de las áreas en las que destaca frente a las más débiles, siempre teniendo en cuenta los pesos asociados:



3.2.1.6.7 Knockout

Framework liviano, enfocado en implementar el modelo MVVM, es totalmente libre, y construido sobre JavaScript, lo que le permite funcionar con cualquier *framework* adicional.

[http://knockoutjs.com/]

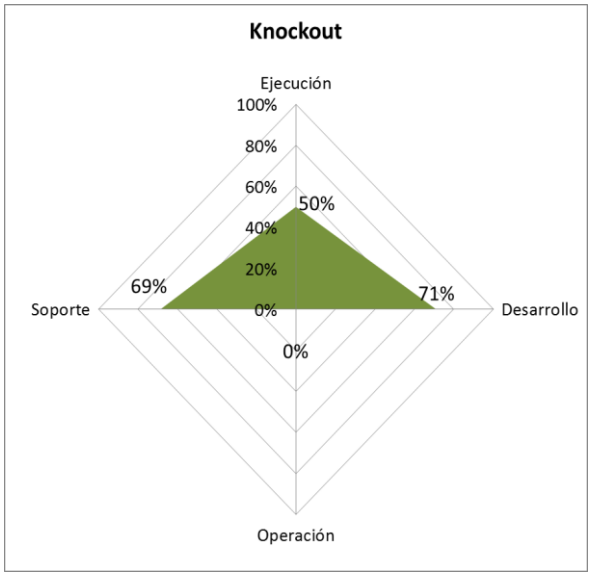
Ventajas:

- ✓ Posee buena documentación.
- ✓ Recibe mucho soporte por parte de su comunidad.
- ✓ Permite realizar Bindings declarativos
- ✓ Posee un tracking de dependencias, detecta los cambios realizados en la vista o en el modelo y es capaz de propagarlos.
- ✓ Permite generar rápidamente plantillas de los datos del modelo.

Desventajas:

- Dificultad en la reutilización de componentes.
- Carece de una jerarquía sólida de componentes.

Gráfica de las áreas en las que destaca frente a las más débiles, siempre teniendo en cuenta los pesos asociados:



3.2.1.6.8 SproutCore

SproutCore es un framwework de Javascript inspirado en Cocoa, de código abierto y multiplataforma, disponible para crear aplicaciones web que tengan un aspecto idéntico al de una aplicación nativa del ordenador.

[http://sproutcore.com/]

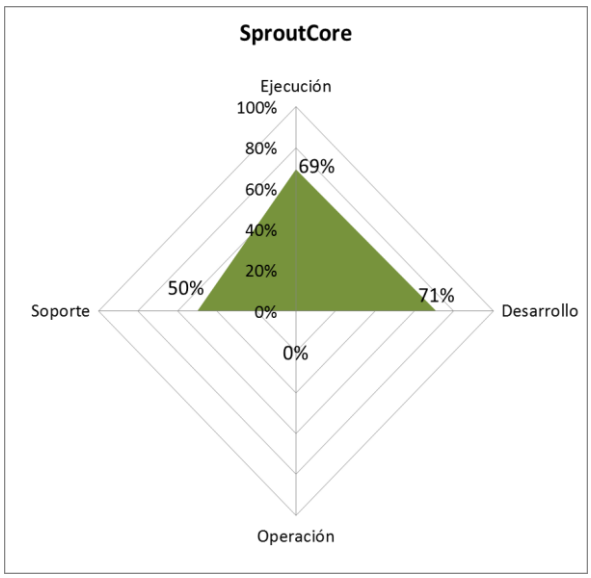
Ventajas:

- ✓ Posee gran cantidad de Widgets personalizables.
- ✓ SproutCore se combina a la perfección con el sistema de almacenamiento de datos de HTML5, dando como resultado un alto rendimiento y una buena experiencia de usuario.

Desventajas:

- Documentación difícil de conseguir y de poca calidad.

Gráfica de las áreas en las que destaca frente a las más débiles, siempre teniendo en cuenta los pesos asociados:



3.2.1.7 Tabla comparativa *frameworks* Estructurales Server MVC

Área	Vaadin	ZK	JavaFX
Ejecución	**69%**	**74%**	**58%**
Componentización	75%	75%	75%
Navegación	75%	100%	100%
Integración otros frontales	50%	50%	25%
Manejo de Eventos	75%	75%	75%
Seguridad	50%	50%	50%
Gestores comunes (errores, sesiones,configuración)	75%	75%	75%
Multidispositivo/Multinavegador	75%	75%	0%
Desarrollo	**46%**	**50%**	**46%**
Herramientas de desarrollo	75%	75%	75%
Herramientas de test	50%	50%	50%
Herramientas de despliegue	75%	75%	25%
Facilitadores (Plantillas, proyectos demo)	50%	75%	75%
Analisis de la calidad (automática)	50%	50%	50%
Independencia del servidor	0%	0%	0%
Operación	**25%**	**25%**	**25%**
Monitorización de aplicación	25%	25%	25%
Tareas de mantenimiento (Jobs, backups y explotación de logs)	25%	25%	25%
Soporte	**69%**	**50%**	**66%**
Soporte proveedor	50%	50%	75%
Licencia producto	75%	50%	50%
Evolución	75%	50%	75%
Media	**52,34%**	**49,65%**	**48,78%**

Los resultados de esta tabla han sido generados a partir de las tablas adjuntadas en el Anexo 7.1.2

3.2.1.8 Conclusiones de la comparación entre *frameworks* de ejecución estructurales

Dentro de los *frameworks* javascript **MV*** los mejor valorados han sido **angularjs** **y** **backbone** donde sus puntos diferenciales han sido sus capacidades en el área de ejecución y soporte.

La diferencia con otros frameworks identificadas son:

- Frente a **ExtJS** y **Capuccino** la diferencia es el nivel de aprendizaje ya que son frameworks en los que la capa de vista y controladora no están tan claramente diferenciadas, ya que están embebidas en sus componentes. Esto hace también que necesite herramientas específicas para su desarrollo y debug lo que conlleva una especialización de conocimiento frente al resto de frameworks javaScript. Además dispone de un bajo soporte respecto a sus competidores, ya que las últimas versiones estables de ExtJS y de Capuccino son de 2011.
- Frente a **Knockout** y **jQuery** ha sido el hecho de que proporcionan un mayor número de capacidades ya existentes de forma nativa mediante servicios, anotaciones, directivas, modelado de objetos, filtros y gestión de alcance de variables.
- Frente a **Ember** el mayor posicionamiento en el mercado de angularjs y backbone, hace que estos dispongan de mayor documentación y apoyo por parte de la comunidad de desarrolladores, lo que acelerará las fases iniciales del desarrollo. Si bien este cuenta con capacidades en el área de ejecución ligeramente superiores al proporcionar componentes visuales propios, con gestión de eventos de

estos. La madurez del framework también es ligeramente inferior a sus competidores y cuenta con mayor número de herramientas y api de terceros.

- **SproutCore** aunque presenta grandes características en el área de ejecución, se ha descartado debido a la falta de soporte y de documentación sumado a la escasa evolución frente a sus competidores.

La diferencia entre los frameworks javaScript y java (**ZK**, **Vaadin** y **JavaFX**) es que los segundos tienen mayores capacidades de ejecución y operación debido a la utilización de un lenguaje más robusto y maduro, java frente a JavaScript. Es por esto que se pueden utilizar todas las herramientas compatibles con Java, herramientas maduras y altamente probadas. Si bien estos tienen la limitación de su ejecución en servidor y la necesidad de una máquina virtual Java y el aprendizaje de sus *APIs* y *tags* en el caso de ZK.

3.2.2 *Frameworks* Visuales

El objetivo es identificar los *frameworks*, que nos faciliten el desarrollo de interfaces gráficas, mediante componentes visuales y funcionalidades adicionales como la validación de formularios y *drag and drop*.

3.2.2.1 Criterios de comparación de los *frameworks* visuales

Para estos *frameworks* hay apartados de los criterios que no aplican y otros en los que los puntos analizados difieren de los estructurales, por esta razón se han establecido unos criterios específicos. La razón es que su enfoque es diferente a los *frameworks* estructurales, ya que su principal función es la de complementar a los anteriores proporcionando un conjunto de componentes visuales que agilicen el desarrollo y hagan más atractiva la interfaz de usuario. Por esta razón se explica a continuación lo criterios analizados:

- Dentro del área de **ejecución** los criterios son:
  - **Componentización:** riqueza de componentes ofrecidos por el *framework*, funcionalidad de los componentes y capacidad de personalización.
  - **Integración otros *frameworks*:** facilidad para integrar el *framework* con otros *frameworks* visuales.
  - **Manejo de Eventos:** eventos ofrecidos por el *framework* al interaccionar con los componentes gráficos.
  - **Multidispositivo/multinavegador:** soporte de los diferentes navegadores y dispositivos.
- Dentro del área de **desarrollo** los criterios son:
  - **Herramientas de desarrollo:** herramientas que facilitan el desarrollo como *plugin*s, toolkits y compatibilidad con los diferentes IDE's.
  - **Facilitadores:** plantillas, proyectos demo, para acelerar la fase inicial.
- Dentro del área de **soporte** los criterios son:
  - **Soporte Proveedor:** documentación disponible del *framework*, certificaciones disponibles, soporte de la comunidad y foros.
  - **Licencia del producto:** condiciones económicas del uso de las herramientas.
  - **Evolución:** grado de actualización, incorporación de nuevas funcionalidades y continuidad.

3.2.2.2 Tabla comparativa *frameworks* visuales

Área	Bootstrap	JQueryUI	Dojo
Ejecución	**75%**	**88%**	**80%**
Componentización	**75%**	**100%**	**75%**
Integración otros *frameworks*	**100%**	**100%**	**100%**
Manejo de Eventos	**50%**	**75%**	**75%**
Multidispositivo/Multinavegador	**75%**	**75%**	**75%**
Desarrollo	**75%**	**60%**	**65%**
Herramientas de desarrollo	**75%**	**50%**	**75%**
Facilitadores (Plantillas, proyectos demo)	**75%**	**75%**	**50%**
Soporte	**75%**	**66%**	**69%**
Soporte proveedor	**75%**	**75%**	**50%**
Licencia producto	**75%**	**75%**	**100%**
Evolución	**75%**	**50%**	**50%**
Media	**75,00%**	**71,04%**	**71,25%**

Los resultados de esta tabla han sido generados a partir de las tablas adjuntadas en el Anexo 7.1.3

3.2.2.3 Conclusiones *frameworks* Visuales

Dentro de los *frameworks* visuales, el mejor valorado en la comparativa ha sido Bootstrap, donde su punto diferencial ha sido su capacidad en el área de soporte, aunque la puntuación de los tres frameworks es muy similar.

Del análisis realizado destacamos los siguientes puntos:

- La gran similitud de capacidades entre JQueryUI y Bootstrap, tanto en capacidades de componentes como de integración con *frameworks* estructurales ya que ambas se integran con los *frameworks* js analizados. Las principales diferencias identificadas son:
  - Bootstrap es más ligero y no tiene dependencias, presenta una mayor evolución y tiene una curva de aprendizaje menor.
  - JqueryUI es más estable y maduro, presentando una mayor compatibilidad con los navegadores.
- Dojo se diferencia de los anteriores en su mayor complejidad y su menor utilización en el mercado. La evolución del *framework* es menor que la de Bootstrap y jQuery y es un *framework* menos maduro que estos.

3.3 HERRAMIENTAS DE DESARROLLO

El objetivo es identificar las herramientas que nos faciliten el desarrollo del interfaz de usuario de las aplicaciones. Se ha seleccionado un conjunto de herramientas de desarrollo web que cubran las necesidades de manejo de elementos html+js+css dejando al margen soluciones específicas para *frameworks*.

3.3.1 Criterios de comparación herramientas de desarrollo

En este apartado se explican los puntos analizados para el análisis de las herramientas de desarrollo:

- Dentro del área de **ejecución** los criterios son:
  - **Previsualización:** capacidad de visualizar los elementos visuales en nuestro entorno de desarrollo mientras son codificados.
  - **Autocompletado y corrector sintáctico:** funcionalidad que facilita la codificación de nuestros componentes mediante la sugerencia de elementos y la identificación de errores antes de su compilación.
  - **Productividad:** funcionalidad que proporciona el IDE para búsquedas, refactorizaciones, comprobación de calidad, construcción y despliegue de artefactos. Facilidad para la integración con herramientas CI (Integración Continua) que integran con una alta frecuencia y automáticamente los cambios de un proyecto, para así poder detectar fallos lo más rápido posible.
  - **Test:** capacidades que permiten acciones de debug, trazado, medición de rendimiento.
  - **Cobertura de *frameworks*:** grado de compatibilidad con diferentes *frameworks*.
  - **Licencias:** condiciones económicas del uso de las herramientas.

3.3.2 Tabla comparativa herramientas de desarrollo

En la siguiente tabla se muestran los resultados por los criterios de los IDE's.

Área	Webstorm	SublimeText	Aptana
Previsualización	**100%**	**100%**	**75%**
Autocompletado y corrector sintáctico	**100%**	**100%**	**100%**
Productividad	**100%**	**75%**	**75%**
Test (debug, performance)	**100%**	**50%**	**75%**
Cobertura de *frameworks*	**75%**	**75%**	**75%**
Licencias	**0%**	**50%**	**100%**
Media	**78%**	**73%**	**84%**

Los resultados de esta tabla han sido generados a partir de las tablas adjuntadas en el Anexo 7.1.5

3.3.3 Conclusiones herramientas de desarrollo

El análisis nos muestra como la mejor herramienta es **WebStorm** debido a la mayor funcionalidad de la que dispone y el soporte que proporciona, aunque su coste hacen que la opción mayor valorada sea la utilización de **Aptana** como *plugin* IDE. Otra de las ventajas de Aptana es la capacidad de integrarlo en J-everis, por lo se tendría la misma herramienta para desarrollar todas las capas de la aplicación.

SublimeText sin llegar a las capacidades de WebStorm tiene las mismas desventajas con un rango de precios ligeramente menor por licencia.

3.4 *FRAMEWORKS* GRÁFICOS

El objetivo es identificar los *frameworks* más adecuados para la creación de gráficos dependiendo de los requerimientos de las aplicaciones.

3.4.1 Criterios de comparación *frameworks* gráficos

En este apartado se explican los puntos analizados para el análisis de los *frameworks* de representación de gráficos:

- Dentro del área de **ejecución** los criterios son:
  - **Riqueza de gráficos:** variedad de los tipos de gráficos soportados y opciones de personalización. (3D, Etiquetación de ejes, áreas, puntos)
  - **Resizing:** redimensionamiento ante los cambios del tamaño de la ventana o el espacio de esta en el que están alojados.
  - **Zoom/scrolling:** capacidades de acceder a la información visualmente como la utilización del zoo y el scroll para consultar información detallada.
  - **Exportación:** utilidades de conversión del gráfico a diferentes formatos.
  - **Mecanismo de randerización:** especificación utilizada para la generación de gráficos.
  - **Eventos:** capacidades de manejo de los eventos generados en la interacción del usuario con los gráficos.
  - **Actualización dinámica:** modos de actualización de los gráficos ante cambios de los datos.
  - **Personalización:** capacidad para personalizar los componentes.
  - **Compatibilidad navegadores:** conjunto de navegadores soportados y detección de condicionantes en algún navegador.
- Dentro del área de **desarrollo** los criterios son:
  - **Curva de aprendizaje:** facilidad de aprendizaje para la utilización de los *frameworks*, mayor puntuación indica mayor facilidad de aprendizaje.
- Dentro del área de **operación** los criterios son:
  - **Gestión de librerías:** mecanismos en la que se permite el manejo de las librerías necesarias y dependencias de librerías de terceros que tienen los *frameworks*.
- Dentro del área de **soporte** los criterios son:
  - **Licencia producto:** modo de licenciamiento del *framework*.

3.4.2 Tabla comparativa de *frameworks* gráficos

En la siguiente tabla se muestran los resultados por los criterios de los *frameworks* gráficos.

Area	Jqplot	Highcharts	googlecharts	d3js
Ejecución	**71%**	**84%**	**74%**	**60%**
Riqueza de gráficos	75%	100%	75%	50%
Resizing	50%	75%	50%	50%
Zoom/scrolling	50%	75%	50%	50%
Exportación	50%	75%	50%	50%
Mecanismo de randerización	100%	100%	100%	75%
Eventos	100%	100%	100%	100%
Actualización dinámica	50%	75%	50%	75%
Personalización	100%	100%	100%	75%
Compatibilidad navegadores	50%	50%	75%	25%
Desarrollo	**50%**	**50%**	**75%**	**25%**
Curva de aprendizaje	50%	50%	75%	25%
Operación	**75%**	**75%**	**50%**	**75%**
Gestión de librerias	75%	75%	50%	75%
Soporte	**100%**	**25%**	**100%**	**100%**
Licencia producto	100%	25%	100%	100%
Media	**69,71%**	**73,91%**	**71,59%**	**60,65%**

Los resultados de esta tabla han sido generados a partir de las tablas adjuntadas en el Anexo 7.1.4

3.4.3 Tabla comparativa de recursos

En la siguiente tabla se muestran los datos recogidos de los recursos necesarios para la utilización de los diferentes *frameworks* gráficos comparados. Debido a que algunos de estos frameworks son bastante *pesados* esta información se erige como un factor muy importante, tanto si las fuentes son importadas físicamente como si se importan de forma dinámica a través de una URL.

La información ha sido recogida mediante la monitorización de los recursos descargados de la aplicación al navegador.

Recurso	Tipo	Tamaño
Highcharts	**347kb**	
jquery.min.js	Lib Externa	138,96
highstock.js	Lib	187,62
highcharts-3d.js	Lib	17,04
highcharts.js	Ctrl	3,38
Google Charts	**680,72 kb**	
Jspapi	Lib	23,97
/?file=visualization&v=1&packages=corechart	Lib	0,65

/format+en,default+en,ui+en,corechart+en.I.js	Lib	646,58
Google Charts	Ctrl	1,71
ui+en.css	Css	6,35
tooltip.css	Css	1,46
JqPlot	**318,14 kb**	
jquery.min.js	Lib	90,45
jquery.jplot.min.js	Lib	168,09
jquery.pieRenderer.min.js	Lib	13,32
excanvas.js	Lib	41,1
jplotcharts.js	Ctrl	1,75
jquery.jplot.min.css	Css	3,43

3.4.4 Conclusiones *frameworks* gráficos

Todos los *frameworks* analizados son compatibles con los estructurales tanto directamente como pudiendo ser encapsulados para simplificar su utilización.

El análisis y los datos recogidos nos muestran cómo el *framework* mejor valorado es **Highcharts,** destacando por sus capacidades de ejecución:

- Amplio conjunto de gráficos (elegantes gráficos de históricos)
- Api que permite la actualización por puntos.
- Altas capacidades gráficas: 3D y zoom, en la gran mayoría de sus gráficos.

Sin embargo el *framework* **Googlecharts,** es gratuito y proporcionar gráficos elegantes con un código muy simple. Al tener que utilizarse como una api remota tiene sus ventajas y sus puntos débiles. Por un lado es necesario que las aplicaciones tengan salida a la red externa donde está la api, pero resolver errores puede resultar más complicado.

Jqplot se presenta como un *framework* con gran funcionalidad y ricos componentes visuales, pero tiene una curva de aprendizaje mayor a los dos anteriores, sin ofrecer mejor funcionalidad.

D3js no es un *framework* de creación de gráficas sino una librería base para crear gráficos, por lo cual se recomienda su utilización para dibujar componentes gráficos complejos no cubiertos por las librerías de gráficos.

3.5 CONCLUSIONES DEL ESTUDIO

En esta sección vamos a exponer los *frameworks* que hemos elegido para las distintas funciones.

Respecto a los *frameworks* de ejecución estructurales la conclusión nos lleva a la recomendación de los ***frameworks* Angularjs y Backbone** utilizando un enfoque **Client MVC**. Esta propuesta se basa en:

- Los beneficios identificados en la prueba de concepto.
- La gran funcionalidad que incorporan o la gran cantidad de herramientas desarrolladas para estos *frameworks*.
- La popularidad de los *frameworks*, lo que se traduce en abundante documentación de calidad y soporte por parte de la comunidad.

- Estabilidad y madurez de los *frameworks*.
- Al tratarse de los *frameworks* más relevantes del mercado, existen multitud de soluciones en el ámbito del desarrollo como puede ser el *debug* de la aplicación, herramientas de desarrollo, análisis de calidad de código.
- Total independencia de la tecnología del servidor, a diferencia de los *frameworks* Server MVC.

Como *framework* visual se ha seleccionado **Bootstrap** debido a las siguientes características:

- Mínima curva de aprendizaje.
- Facilidad de inyección de dependencias.
- Buena integración con Angularjs y BackBone.
- Elementos muy configurables.
- Diseños atractivos.
- Código no intrusivo.

Respecto a los *frameworks* de gráficos a pesar de que los mejores valorados en la comparativa han sido Highcharts y googlecharts, en nuestro caso al únicamente necesitar un tipo de gráfico, un gráfico de tipo tarta (pie charts), se ha utilizado el *framework* JqPlot, debido a que presentaba el mayor tipo de gráficos de tarta distintos, aunque la dificultad ha resultado ser mayor.

Como herramienta de desarrollo, a pesar de que las herramientas mejor valoradas en la comparativa han sido Webstrom y SublimeText se ha elegido **Aptana**, por los siguientes motivos:

- Licencia libre.
- Al integrarse mediante un *plugin* para Eclipse, puede integrarse en J-everis, ya que éste se apoya en Eclipse.
- Permite desarrollar todas las capas de la aplicación en Eclipse, una herramienta robusta y con la que tenemos experiencia.
- Gran funcionalidad: refactorización, búsquedas avanzadas, autocompletado, corrección sintáctica.
- Soporta Angularjs y Backbone.
- Permite customización.

3.6 BACKBONE.JS AL DETALLE

Como se ha explicado anteriormente, uno de los *frameworks* elegidos para la realización de una demo utilizando la arquitectura J-everis es **BackboneJS**. En este apartado se pretende dar a conocer más en profundidad las diferentes características y funcionalidades que ofrece el *framework*, entrando en aspectos más técnicos que los vistos hasta el momento.

A continuación describiremos los componentes esenciales que se utilizan en BackboneJS y las funcionalidades que estos ofrecen:

- **Model**: el modelo define una entidad u objeto del sistema. Son la parte fundamental de una aplicación JavaScript ya que contiene la información con la que se trabaja, así como una gran parte de la lógica de negocio que le rodea. Algunas funciones destacadas de estos objetos son:
  - Initialize: se ejecuta cada vez que instanciamos un objeto de este tipo, por lo que contiene todo lo que deba ser declarado, instanciado o realizado en ese momento. Esta función se encuentra también en los componentes *Collection* y *View*, que se explican a continuación.
  - Definición de la información: se puede definir los atributos que contiene un modelo tanto al ser definido como al ser instanciado, aunque lo normal es que estos datos sean recogidos automáticamente del servidor, conteniendo la información declarada en la lógica de negocio de forma dinámica sin necesidad de ser declarado de nuevo en esta capa. Para recoger la información de un objeto de tipo modelo es tan sencillo como hacer un *get* del atributo que corresponda.

    Ejemplos:

```
Persona = Backbone.Model.extend({
        defaults: {
            nombre: 'Daniel',
            edad: 12,
        },
        initialize: function(){
            alert("Welcome to this world");
        }
    });
```

```
var person = new Person({ nombre: 'Eduardo', altura: 1,78});
```

  - Escuchar cambios: si se produce algún cambio en la información, por defecto estos objetos no son actualizados automáticamente, pero existe una forma sencilla de implementar esta funcionalidad para los objetos que se desee, definiendo una función que se ejecuta en el momento que el atributo que se especifique del objeto sea modificado.
  - Peticiones al servidor: esta es una de las mayores facilidades que ofrece el *framework*. Cuando se define un modelo se indica una *url* a la que se enviarán por defecto las peticiones que se hagan al servidor, de forma que al ejecutar ciertas funciones que ofrece el componente, Backbone envía automáticamente estas peticiones, sin necesidad de que el desarrollador tenga que definirlas manualmente.

```
var Oportunidad = Backbone.Model.extend({
      urlRoot: '/om-web-il/rest/oportunidad/oportunidad'
});
```

    Una vez hemos declarado esta propiedad, las peticiones básicas al servidor se hacen de una manera muy fácil. Por ejemplo si se tiene el ID de un objeto y se quiere recuperar la información relativa a este tan solo tenemos que hacer un *fetch*, y el servidor

automáticamente envía la petición oportuna al servidor, en este caso se enviaría un GET pasando el ID como parámetro.

```
var user = new Usermodel({id: 1});

oportunidad.fetch({

        success: function (user) {

                //----------//

        }

})
```

Para crear un nuevo objeto o modificarlo solo habría que hacer un *save*, la diferencia está en que con el primero no sabemos el ID, y en el segundo caso sí. Del mismo modo se puede mandar una petición DELETE para eliminar un objeto mediante un *destroy*.

- Validaciones: este componente ofrece también una función por defecto denominada *validate* en la que se pueden añadir las validaciones oportunas para los atributos del objeto. Se realizan las comprobaciones especificadas y en caso de no cumplir la validación, basta con retornar un string, que será el mensaje de error que se muestre por pantalla.

- **Collection:** este componente define colecciones de modelos y tiene las mismas características que las mencionadas anteriormente, a excepción de las peticiones al servidor que en este caso implmenta solo el *fetch*, ya que el guardado, modificado y borrado de los objetos no se hace de forma masiva si no de uno en uno.

- **View:** este componente define las vistas de la aplicación, reflejando la información que contienen los modelos de la aplicación y reaccionando a los eventos definidos. Al igual que los otros componentes define una función *initialize*, que se ejecutará cuando instanciemos un objeto de este tipo. También contiene una función muy important denominada *render*, que será la función principal de la vista, que se ejecutará tras instanciarla. Esta función normalmente realiza alguna acción necesaria antes de mostrar por la interfaz de usuario la información que corresponda.

  - Templates: este *framework* está diseñado para realizar aplicaciones web de una sola página (*single-page*), por lo que se define un código html básico que servirá de base, y en un punto se añade la asignación *.page*, que será el lugar donde la Backbone inserte el código luego definido desde estas vistas. La manera de mostrar la información se define a partir de unas plantillas, los llamados *templates*, que normalmente se realizan por medio del *framework* **Underscore.js**. En estas plantilla se define el código html y la lógica que debe ser implementada, pudiendo recibir objetos por parámetro a la hora de ser instanciadas, para así poder mostrar información realtiva a los modelos y colecciones definidos en la aplicación.
  - Eventos: define las funciones que serán ejecutadas cuando sucedan los eventos definidos. Por ejemplo que cuando se haga un *submit* desde el template instanciado se ejecute cierta función que almacene los datos en el servidor.

  Como se puede apreciar, es este componente, la vista, el encargado de actuar también como **controlador**. Como se explicó anteriormente, no todos estos *frameworks* siguen literalmente un patrón MVC, sino que se puede considerar como **MV***.

- **Router:** este componente hace de enrutador, y por tanto define lo que ocurrirá cuando entremos a las rutas definidas desde el navegador. Este instanciará los objetos y ejecutará las funciones que correspondan. Una propiedad muy ventajosa es la enrutación dinámica, es decir, existen algunas

especie de *expresiones regulares*, para que no sea necesario definir la ruta exacta, y se puedan escoger elementos de la ruta, como por ejemplo el ID.

Ejemplo:

```
var Router = Backbone.Router.extend({
      routes: {
        '': 'login',
        'home': 'home',
        'nuevaOportunidad': 'editOportunidad',
        'editar/:id' : 'editarOportunidad'
      }
 });


 router.on('route:editarOportunidad', function(id) {
      editOp.render({id: id});  // editOp será una vista como las explicadas
                                   anteriormente
 });
```

Un inconveniente que se ha visto es la necesidad de repetir muchas partes del código que son muy similares al tener varias entidades de información. Por otro lado, tampoco es trivial la correcta estructuración del código, que aunque para la demo realizada en este proyecto no es de vital necesidad, sí lo sería en proyectos de mayor embergadura, ya que si no se perdería una de las principales razones para utilizar este tipo de tecnologías.

4 DESARROLLO DE LA PRUEBA DE CONCEPTO

4.1 OBJETIVO

El objetivo de la realización de este prototipo es el uso de uno de los *frameworks* estudiados, de forma que se pueda comprobar de forma práctica las características de los mismos y añadir conclusiones extraídas a partir de la experiencia personal.

Se ha propuesto realizar una aplicación a la que se pueda dar uso desde la Oficina Técnica de J-everis. El objetivo es facilitar la gestión de proyectos para clientes que serán realizados con el *framework* J-everis.

Para ello se ha realizado un análisis de requisitos de la aplicación. Es importante tener en cuenta que la aplicación será un **pequeño prototipo** cuya finalidad no es la aplicación en sí misma si no el uso de un *framework* MVC para la capa visual. Es por ello que se realizarán **algunas funciones básicas** de la aplicación, aunque para darle un uso real habría que llevar a cabo mejoras expuestas en el apartado de *posibles mejoras*.

Puesto que no puede considerarse una aplicación completa, y el objetivo que persigue es el de utilizar una tecnología de las estudiadas y su posible integración con la arquitectura, no se ha realizado una documentación completa de la aplicación, ya que las fases de toma de requisitos, análisis y diseño no han sido realizadas de la misma manera que se haría para una aplicación completamente funcional y de mayor complejidad. A continuación se detalla un breve diseño funcional, suficiente para entender las funcionalidades que son cubiertas por el prototipo, las entidades de información utilizadas y sus relaciones, así como una descripción de las diferentes interfaces de usuario y del sistema que contiene la aplicación.

4.2 ESPECIFICACIÓN DE REQUISITOS

4.2.1 Objetivos del Sistema

Aplicación Web que permita gestionar los proyectos y las oportunidades de emprendimiento de proyectos para clientes, realizados mediante la arquitectura J-everis.

4.2.2 Requisitos funcionales

Código	Nombre	Descripción
RF-01	**Login**	El sistema debe permitir logearse a los usuarios que estén registrados en el sistema
RF-02	**Posición global**	El sistema debe proporcionar el porcentaje de oportunidades de proyectos J-everis que hay en cada estado.
RF-03	**Importación de información**	El sistema debe permitir agregar información al histórico actual mediante la importación de un fichero Excel con información de nuevas oportunidades siguiendo el formato de una plantilla
RF-04	**Alta oportunidad manual**	El sistema debe permitir agregar información de nuevas oportunidades mediante un formulario proporcionado por la aplicación
RF-05	**Edición de oportunidad**	El sistema debe permitir modificar la información de las oportunidades registradas
RF-06	**Listado de oportunidades**	El sistema debe permitir ver un listado de las oportunidades y filtrar por atributos
RF-07	**Detalle de oportunidad**	El sistema debe permitir ver la información detallada de una oportunidad
RF-08	**Borrado de una oportunidad**	El sistema debe permitir realizar borrados lógicos de las oportunidades
RF-09	**Administración de la información asociada a las oportunidades**	El sistema debe permitir administrar los posibles usuarios, clientes, oficinas, estados, etc.

4.2.3 Requisitos no funcionales

Código	Nombre	Descripción
RNF-01	**Aplicación Web**	El sistema deberá ser una aplicación web
RNF-02	**Uso J-everis**	El sistema deberá ser realizado mediante la arquitectura J-everis
RNF-03	**Capa de persistencia Hibernate: JPA**	La capa de persistencia deberá estar desarrollada por la implementación Hibernate de JPA
RNF-04	**Publicación de servicios REST**	El sistema deberá implementar servicios REST
RNF-05	**Capa visual Javascript**	La interfaz de usuario debe estar desarrollada en un *framework* cliente javascript MVC
RNF-06	**BBDD MySQL**	La base de datos deberá estar realizada en MySQL

4.3 DISEÑO FUNCIONAL

4.3.1 Objetivos del sistema

El sistema permitirá gestionar las oportunidades de proyectos J-everis, aportando una imagen de la situación global y haciendo más fácil la manipulación de los datos.

4.3.2 Contexto del sistema

Actualmente, esta gestión se realiza por medio de Excels, en las que se engloba toda la información relativa a las oportunidades de proyecto en J-everis, así como los actuales proyectos que se están realizando, o han finalizado.

A medida que crece la cantidad de proyectos, esto se hace más pesado de manejar, además de requerir más tiempo por parte de los trabajadores de la Oficina Técnica, por lo que la aplicación debe dar solución a estos problemas, además de aportar información valiosa a nivel de estadísticas para tener una mejor visión de la situación actual.

La aplicación permitirá la realización de altas, bajas y modificaciones, mostrando estadísticas sobre la información de los datos almacenados. Los datos podrán importarse también de forma masiva, haciendo más fácil el paso del actual sistema de gestión (Excels) al nuevo.

4.3.3 Estructura lógica de la información

A continuación se detallan las diferentes entidades del sistema y las relaciones existentes entre sí, junto con la información que almacenará cada una de ellas en la base de datos.

Código Entidad/Relación (Especificación Funcional)	Nombre de la Entidad	Descripción de la Entidad
EF-01	**Oportunidad**	Oportunidad de proyecto com J-everis
EF-02	**Cliente**	Empresa cliente
EF-03	**Actividad**	Sector de la empresa
EF-04	**Estado**	Situación de la oportunidad
EF-05	**Contacto**	Responsable en Everis
EF-06	**Oficina**	Oficina de Everis
EF-07	**Lugar**	Ubicación empresa
EF-08	**Usuario**	Usuario de la aplicación
EF-09	**Rol**	Rol del usuário

*Se ha incluido la entidad Rol de cara a futuras mejoras, aunque no será utilizada en esta prueba de concepto.

4.3.3.1 Descripción de las entidades de información

Nombre de la Entidad	OPORTUNIDAD
Descripción	Proyecto que puede ser desarrollado con la arquitectura J-everis
Volumen Actual	50
Atributos	IDOPORTUNIDAD
	NOMBRE_PROYECTO
	IDCLIENTE
	IDACTIVIDAD
	IDESTADO
	PROBABILIDAD
	PRECIO_VENTA
	SOPORTE_COMERCIAL
	USO_CENTROS
	IDCONTACTO
	MES_REFERENCIA
	PREV_INICIO
	DURACION
	DESCRIPCIÓN
	IDPROPIETARIO
	IDCREADOR
	FECHA_CREACÍON
	IDMODIFICADOR
	FECHA_MODIFICACIÓN
Restricciones	PRIMARY KEY (IDOPORTUNIDAD)
	FOREIGN KEY (IDCLIENTE) REFERENCES CLIENTE(IDCLIENTE)
	FOREIGN KEY (IDACTIVIDAD) REFERENCES ACTIVIDAD(IDACTIVIDAD)
	FOREIGN KEY (IDESTADO) REFERENCES ESTADO(ESTADOACTUAL)
	FOREIGN KEY (IDCONTACTO) REFERENCES CONTACTO(IDCONTACTO)
	FOREIGN KEY (IDPROPIETARIO) REFERENCES USUARIO(IDUSUARIO)
	FOREIGN KEY (IDCREADOR) REFERENCES USUARIO(IDUSUARIO)
	FOREIGN KEY (IDMODIFICADOR) REFERENECES USUARIO(IDUSUARIO)

Nombre de la Entidad	USUARIO
Descripción	Usuario registrado en la aplicación
Volumen Actual	3
Atributos	IDUSUARIO
	LOGIN
	PASSWORD
	IDROL
	CORREO
	FECHA_ALTA
	ULTIMA_CONEXION
Restricciones	PRIMARY KEY (IDCLIENTE)

Nombre de la Entidad	ROL
Descripción	Define los permisos de los diferentes usuarios en la aplicación
Volumen Actual	3
Atributos	IDROL
	NOMBRE_ROL
Restricciones	PRIMARY KEY (IDROL)

Nombre de la Entidad	CLIENTE
Descripción	Empresa para la que se desarrolla el proyecto
Volumen Actual	45
Atributos	IDCLIENTE
	EMPRESA
Restricciones	PRIMARY KEY (IDCLIENTE)

Nombre de la Entidad	ACTIVIDAD
Descripción	Sector para el cual trabaja la empresa para la que se desarrolla el proyecto
Volumen Actual	7
Atributos	IDACTIVIDAD
	SECTOR
Restricciones	PRIMARY KEY (SECTOR)

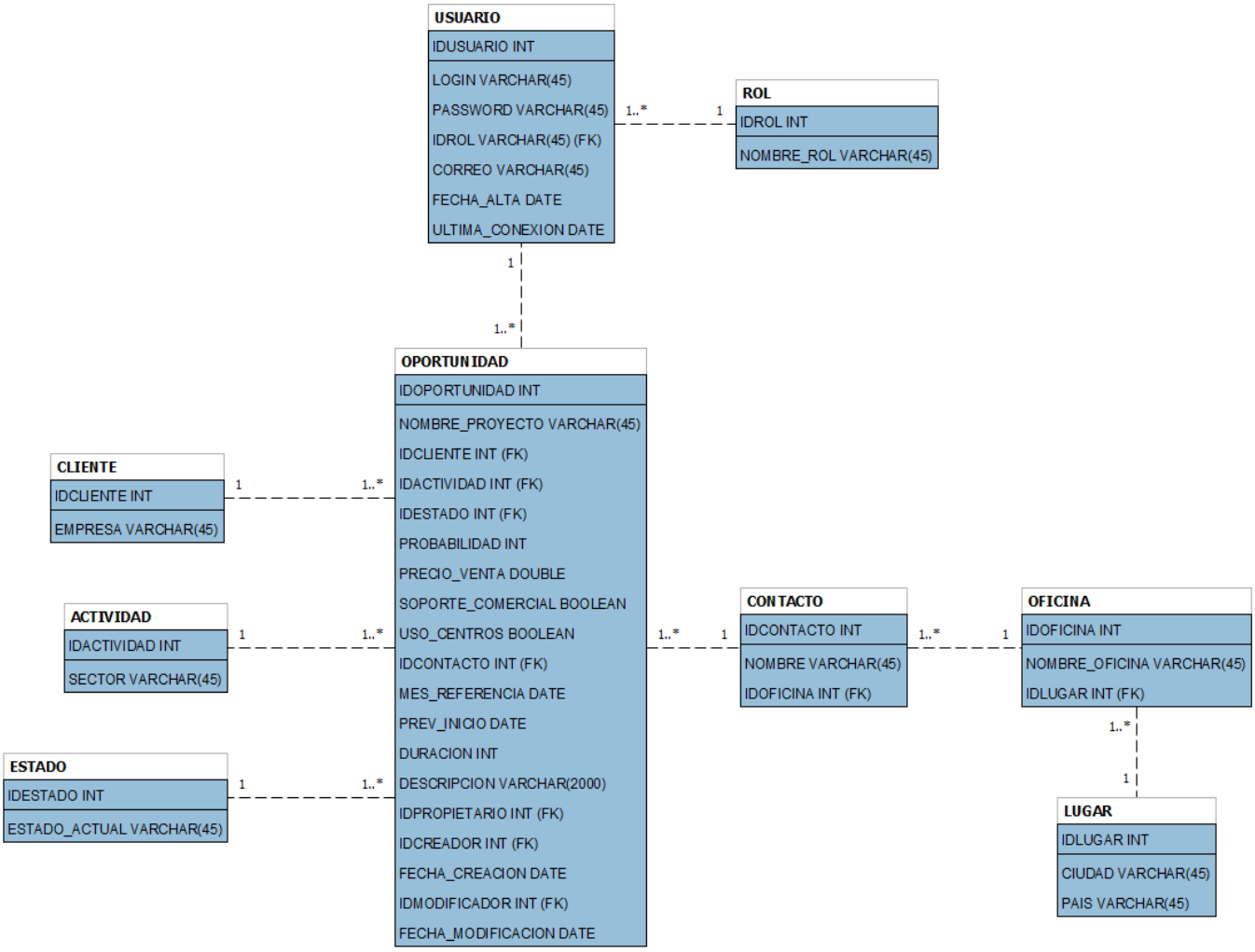
Nombre de la Entidad	ESTADO
Descripción	Define la situación en la que se encuentra la oportunidad
Volumen Actual	4
Atributos	IDESTADO
	ESTADO_ACTUAL
Restricciones	PRIMARY KEY (IDESTADO)

Nombre de la Entidad	CONTACTO
Descripción	Responsable del proyecto dentro de Everis
Volumen Actual	25
Atributos	IDCONTACTO
	NOMBRE
	IDOFICINA
Restricciones	PRIMARY KEY (IDCONTACTO)
	FOREIGN KEY (IDOFICINA) REFERENCES CLIENTE(IDOFICINA)

Nombre de la Entidad	OFICINA
Descripción	Oficina de Everis en la que se desarrollará el proyecto
Volumen Actual	5
Atributos	IDOFICINA
	NOMBRE_OFICINA
	IDOFICINA
Restricciones	PRIMARY KEY (IDOFICINA)
	FOREIGN KEY (IDLUGAR) REFERENCES CLIENTE(IDLUGAR)

Nombre de la Entidad	LUGAR
Descripción	Ciudad y país
Volumen Actual	18
Atributos	IDLUGAR
	CIUDAD
	PAÍS
Restricciones	PRIMARY KEY (IDLUGAR)

4.3.3.2 Diagrama de entidad-relación



4.3.4 Definición de casos de uso

A continuación se detallan los casos de uso de la aplicación:

Logueo de usuarios	
Propósito	Este caso de uso permite el acceso de los usuarios a la plataforma
Actores	Usuario
Precondiciones	El usuario debe estar dado de alta en el sistema
Post-condiciones	El usuario está logueado en la aplicación
Flujo de eventos	El usuario introduce su login y contraseña, el sistema valida los datos y, de ser correctos, permite al usuario acceder a la aplicación

Acceso a posición global	
Propósito	Este caso de uso permite ver diferentes datos y estadísticas de las oportunidades
Actores	Usuario
Precondiciones	Estar logueado en el sistema
Post-condiciones	
Flujo de eventos	El usuario accede a la aplicación o al apantalla de inicio, el sistema recoge diferentes datos de las oportunidades existentes y los muestra por pantalla en diferentes formatos

Alta de Oportunidad	
Propósito	El caso de uso permite dar de alta una nueva oportunidad en el sistema de forma manual
Actores	Usuario
Precondiciones	El usuario está logueado en el sistema
Post-condiciones	La nueva oportunidad es registrada en el sistema
Flujo de eventos	El usuario rellena un formulario con los metadatos de la oportunidad para que el sistema recoja esta información y cree la nueva oportunidad

Edición de Oportunidad	
Propósito	El caso de uso permite mofificar una oportunidad existente en el sistema
Actores	Usuario
Precondiciones	El usuario está logueado en el sistema La oportunidad a editar está registrada en el sistema
Post-condiciones	La nueva oportunidad es modificada
Flujo de eventos	El sistema muestra un formulario con los campos rellenos según los datos de la oportunidad que se esté editando. El usuario modifica los campos oportunos y la oportunidad guarda en el sistema los cambios realizados.

Borrado de Oportunidad	
Propósito	El caso de uso permite eliminar del sistema una oportunidad existente
Actores	Usuario
Precondiciones	El usuario está logueado en el sistema La oportunidad a eliminar está registrada en el sistema
Post-condiciones	La oportunidad es eliminada del sistema
Flujo de eventos	El sistema muestra los datos de la oportunidad seleccionada, el usuario elige la opción de eliminar y el sistema borra el registro de esta oportunidad

Importación masiva de Oportunidades	
Propósito	El caso de uso permite dar de alta oportunidades en el sistema de forma masiva a través de un archivo Excel
Actores	Usuario
Precondiciones	El usuario está logueado en el sistema La Excel contiene las columnas que corresponden
Post-condiciones	Las oportunidades son registradas en el sistema
Flujo de eventos	El sistema permite al usuario elegir un archivo Excel, recoge los datos y registra las oportunidades

Administración de datos	
Propósito	El caso de uso permite administrar la información relativa a las oportunidades, como los posibles estados, las oficinas existentes, etc.
Actores	Usuario
Precondiciones	El usuario está logueado en el sistema
Post-condiciones	Los datos son modificados en el sistema
Flujo de eventos	El sistema permite al usuario crear, eliminar o mantener la posible información que se puede asignar a las oportunidades, como los posibles estados, las oficinas existentes, etc

4.3.5 Interfaces de sistema

Se detallan los servicios REST publicados por el sistema para interactuar con las peticiones de la capa de presentación.

Servicio	Descripción
autenticarUsusario	Identifica a un usuario
getPosiciónGlobal	Proporciona la proporción de oportunidades que hay en cada estado
setOportunidad	Registra una nueva oportunidad
getOportunidades	Obtiene todas las oportunidades registradas en el sistema
getOportunidad	Obtiene el detalle de una oportunidad.
updateOportunidad	Modifica los datos de una oportunidad
deleteOportunidad	Elimina una oportunidad del sistema

importOportunidades	Registra oportunidades mediante un Excel en formato Base64.	
set*	Registro de cada tipo de entidad del sistema	Entidades afectadas: - Usuario - Cliente - Actividad - Contacto - Oficina - Estado
get*	Obtiene el detalle de cada tipo de entidad	
get*s	Obtiene todos los registros de una determinada entidad existentes en el sistema	
update*	Modifica los datos de una determinada entidad del sistema	
delete*	Elimina una entidad del sistema	

4.3.6 Interfaces de usuario

Se detallan las diferentes interfaces de usuario que tendrá la aplicación:

- **IU - Login:** pantalla con un pequeño formulario donde introducir el login del usuario y la contraseña para loguearse en el sistema. Será la primera pantalla que se encuentre al acceder a la aplicación.
- **IU - PosiciónGlobal:** pantalla con diferentes gráficas que permiten ver el porcentaje de oportunidades registradas en el sistema en cada estado. Será la página principal de la aplicación, a la que se accede tras loguarse en el sistema, y a la que se tendrá acceso desde el menú principal.
- **IU - ListadoOportunidades:** listado de un breve detalle todas las oportunidades existentes en el sistema. Se podrá acceder desde el menú principal.
- **IU - NuevaOportunidad**: formulario donde se rellena la información relativa a una oportunidad para que sea registrada en el sistema. Se accede desde el listado de oportunidades, así como desde el menú principal.
- **IU - EdiciónOportunidad:** formulario con el detalle de la oportunidad seleccionada. Desde esta pantalla se podrá editar la información o eliminar la oportunidad. Se accede desde el listado de oportunidades, seleccionando la oportunidad que se queira ver/editar/eliminar.
- **IU - AdministraciónAplicación:** desde esta pantalla se tendrá acceso a la edición del resto de entidades del sistema. Se mostrará un menú con cada una de ellas desde el que se accederá a ver su detalle y gestionar las altas, modificaciones y bajas del sistema. A esta interfaz se accederá desde el menú principal de la aplicación

Existirán además las siguientes interfaces con las mismas características para cada entidad del sistema:

- **IU – Listado***
- **IU – Nuevo***
- **IU – Edición***

Estas interfaces son homologas de las correspondientes a las oportunidades, pero a estas se tendrá acceso a dese la interfaz de administración de la aplicación, eligiendo la entidad de la cual queramos gestionar la información.

5 CONCLUSIONES

5.1 COMPARACIÓN DE LAS TECNOLOGÍAS SELECCIONADAS

En esta sección se va a comparar cómo se han implementado las distintas funcionalidades de la aplicación utilizando los *frameworks* **AngularJS** y **BackboneJS**.

5.1.1 RF-02 Posición global

Para la implementación de esta funcionalidad se ha creado un servicio REST en la capa del servidor que devuelve el número de proyectos existentes en cada uno de los diferentes estados almacenados en el sistema.

En la capa de implementación se implementa la funcionalidad que calcula el porcentaje y muestra diferentes *tartas* con los porcentajes de proyectos en cada estado.

En ambos casos existen dos partes diferenciadas, la vista donde se cargará el gráfico (*pie chart*), común para ambas, y la función que llama al servicio REST y se encarga de informar los datos correctamente.

- En el controlador se define la función que recoge los datos del servidor y actualiza la variable que utiliza el *framework* JqPlot para dibujar el gráfico con los datos actualizados.
- En la vista indicamos en que sección se dibujará el gráfico e indicamos que al acceder a la página se llame a la función del controlador, para actualizar los datos.

En este caso la única diferencia reside a la hora de llamar a la función. En AngularJS se utiliza una etiqueta que hace que al entrar en la página se llame a una función JavaScript del controlador, mientras que en Backbone existe un objeto *Router*, que hace de "enrutador" y se encarga de gestionar las funciones que se lanzan al entrar en una determinada url. Éste llamará a la función *render* de la vista asociada a la interfaz encargada de mostrar las gráficas. Esta función la contienen todas las vistas y definen el código que se ejecutará al instanciar la vista. En ella se define el *template* o plantilla a la que se llama, que no es otra cosa que el código html y los scripts que será mostrado en la página.

5.1.2 RF-03 Importación de información

El sistema debe permitir agregar información al histórico actual mediante la importación de un fichero Excel con información de nuevas oportunidades siguiendo el formato de una plantilla.

Ninguno de los frameworks utilizados incorpora herramientas que faciliten esta utilidad, por lo que ha sido necesario desarrollar esta funionalidad con JavaScript y JQuery, siendo la misma en ambos casos.

El fichero se importa en formato multi-part para su correcta interpretación por parte del servicio Rest.

La única diferencia la encontramos en que en Angular se ha implementado una *directiva*, que permite tener acceso a la función desde cualquier parte de la aplicación de una manera muy sencilla, aunque esta es más compleja de implementar.

5.1.3 RF-04 Alta oportunidad manual

Esta funcionalidad permite agregar información de nuevas oportunidades mediante un formulario proporcionado por la aplicación.

Se ha creado un servicio REST en la capa del servidor, que será utilizado por ambos frameworks, el cual recibe un objeto Oportunidad y lo registra en la base de datos.

En el caso de AngularJS están involucradas dos partes, la vista del formulario donde se introducen los datos de la oportunidad a registrar y el controlador de esta vista.

- En el controlador se define el objeto oportunidad y el alcance del mismo, aunque no es necesario definir los atributos de este, se añaden dinámicamente. También se define la función que enviará la oportunidad al servicio REST indicado anteriormente. AngularJs automáticamente transforma el objeto a JSON para poder ser enviado a través del servicio REST.
- En la vista se indica a qué controlador está asociada. Para registrar los atributos de la oportunidad se indican los atributos de la oportunidad mediante una etiqueta <input> de html, en esos input se le indica a que objeto está asociado y a que atributo de ese objeto, gracias al data-binding una vez que el usuario introduzca un valor en los input, el valor se registra en el objeto automáticamente.
- Una vez pulsado el botón de guardar de la vista, se lanza la función del controlador que envía la oportunidad al servicio Rest.

En Backbone se ha creado una vista encargada de manejar la interfaz de alta y edición de oportunidades, que define al igual que antes la funcionalidad que se ejecutará y la plantilla a la que se llamará para mostrar los datos por pantalla. Antes, se ha creado una variable de tipo Model, que define un modelo, es decir una entidad del sistema. Al igual que con las colecciones, se define la ruta a la que tiene que invocar los servicios y Backbone se encargará de realizar las peticiones PUT, GET, POST o DELETE automáticamente cuando se llamen a algunas de las funciones que definen los modelos.

La plantilla mostrará el formulario y recogerá los datos. Cuando se haga el *submit*, el manejador de eventos se encargará de llamar a otra función de la vista, que se encarga de convertir los datos del formulario en un objeto JSON, y posteriormente hacer un *save* del objeto con estos datos como argumento. Backbone se encargará de hacer mandar un POST al servidor, de forma que el servidor automáticamente creará una nueva oportunidad en la base de datos.

Un punto muy interesante es el hecho de que esta vista y esta plantilla sean las mismas para realizar el alta y la edición del modelo Oportunidad. La diferencia está en que cuando el enrutador llame a la vista lo puede hacer pasando el ID de una objeto Oportunidad (en el caso de la edición) o sin pasarle ningún argumento, esto hará que la vista sepa si debe recuperar el objeto con este id y mandárselo a la plantilla que mostrará el formulario, o simplemente instanciar la plantilla sin pasar ningún argumento como parámetro.

Por otro lado, en la plantilla se define código JavaScript que muestre o no datos en el formulario en función de si le han pasado algún objeto o no.

5.1.4 RF-06 Edición de oportunidad

El sistema debe permitir modificar la información de las oportunidades registradas.

Se han creado dos servicios REST en la capa del servidor, uno de ellos recibe un objeto Oportunidad y mediante su ID modifica la oportunidad, mientras que el otro devuelve la oportunidad a modificar a la capa de presentación.

En el caso de AngularJs, están involucradas 2 partes, la vista del formulario donde se carga la oportunidad y donde se introducen los datos de la oportunidad a modificar, y el controlador de esta vista.

- En el controlador se recoge el objeto oportunidad recibido por el servicio Rest. También se define la función que enviará la oportunidad modificada al servicio Rest indicado anteriormente, AngularJs

automáticamente transforma el objeto de Json a un objeto AngularJs con todos los atributos al igual que los que tenía en Java.

- En la vista se indica a que controlador está asociada esta. Para modificar los atributos de la oportunidad se indican los atributos de la oportunidad mediante una etiqueta <input> de html, en esos input se le indica a que objeto está asociado, es este caso al objeto oportunidad y a que atributo de ese objeto, gracias al data-binding los atributos del objeto se cargarán automáticamente del objeto oportunidad recibido por el servicio Rest y una vez que el usuario introduzca un valor en los input, el valor se registra en el objeto automáticamente.
- Una vez pulsado el botón de editar de la vista, se lanza la función del controlador que envía la oportunidad al servicio Rest para su modificación.

En backbone le funcionamiento es el mismo que el definido en la anterior sección, con la diferencia de que en este caso se habrá seleccionado una Oportunidad antes de elegir la opción de editar, y por tanto el enrutador le pasará a la vista el ID del objeto seleccionado para que esta sepa que debe recuperar la información de este objeto por medio de la función *fetch*, que definen todos los modelos de Backbone, y mostrar sus datos en el formulario. Además tendrá en cuenta que cuando se haga el *save* del objeto no estaremos creando uno nuevo si no modificándolo, ya que tenemos su ID, y la petición al servidor será de distinto tipo, en este caso PUT.

5.1.5 RF-07 Listado de oportunidades

Permite ver un listado de las oportunidades existentes y sus propiedades más relevantes.

Para ambas tecnologías se ha creado un servicio REST en la capa del servidor, el mismo servicio para los dos *frameworks*, que devuelve las oportunidades a mostrar en la capa de presentación.

En el caso de AngularJs, están involucradas dos partes, la vista del formulario donde se carga la oportunidad y donde se introducen los datos de la oportunidad a modificar, y el controlador de esta vista.

- En el controlador se recoge el objeto lista de oportunidades recibido por el servicio Rest. También se define la función que recoge las oportunidades a través del servicio Rest indicado anteriormente, AngularJs automáticamente transforma el objeto de Json a un objeto AngularJs con todos los atributos al igual que los que tenía en Java.
- En la vista se indica a que controlador está asociada esta. Para mostrar las oportunidades únicamente llamamos al método indicado anteriormente, el cual recupera la lista de oportunidades a mostrar y hacemos referencia al objeto que contiene la lista. En esta página se define que atributos se van a mostrar de cada oportunidad y se define que se van a mostrar a través de la lista de oportunidades mencionada anteriormente y angular automáticamente creará una fila por cada elemento en la lista de oportunidades. AngularJs proporciona un filtro de los elementos de la lista mediante un tag, de manera muy sencilla.

En el caso de Backbone, primero definimos un objeto de tipo *Collection*, es decir, una colección de objetos o modelos, al cual le indicamos la ruta por la que se accede al servicio REST que ofrece los servicios referentes a estos objetos (no es necesario especificar nada al respecto del objeto o modelo). Tras instanciar una variable de este tipo collection, Backbone ofrece una función denominada *fetch*, que automáticamente envía una petición al servidor de tipo GET y recupera los objetos de la base de datos.

5.1.6 RF-08 Detalle de oportunidad

Permite ver la información detallada de una oportunidad concreta.

En esta funcionalidad se hace uso de uno de los servicios REST mencionados en el punto anterior, el cual devuelve toda la información respectiva a una oportunidad.

En el caso de AngularJs, están involucradas dos partes, la vista del formulario donde se carga la oportunidad y donde se introducen los datos de la oportunidad a modificar, y el controlador de esta vista.

- En el controlador se recoge el objeto oportunidad recibido por el servicio Rest. También se define la función que recoge la oportunidad a través del servicio Rest indicado anteriormente, AngularJs automáticamente transforma el objeto de Json a un objeto AngularJs con todos los atributos al igual que los que tenía en Java.
- En la vista se indica a que controlador está asociada esta. Para mostrar los atributos únicamente llamamos al método indicado anteriormente, el cual recupera la oportunidad a mostrar y hacemos referencia al atributo del objeto oportunidad.

En Backbone se reutiliza la funcionalidad de edición de una oportunidad, ya que ésta muestra todos los detalles del objeto seleccionado para modificar.

5.1.7 RF-09 Borrado de una oportunidad

Para ambas tecnologías se han creado dos servicios Rest en la capa del servidor, los mismos servicios para los dos frameworks, uno de ellos recibe un objeto oportunidad y mediante su id elimina la oportunidad, el otro devuelve la oportunidad a eliminar a la capa de presentación.

En el caso de AngularJs, están involucradas 2 partes, la vista del formulario donde se carga la oportunidad y el controlador de esta vista.

- En el controlador se recoge el objeto oportunidad recibido por el servicio Rest. También se define la función que enviará la oportunidad a eliminar al servicio Rest indicado anteriormente.
- En la vista se indica a que controlador está asociada esta.
- Una vez pulsado el botón de eliminar de la vista, se lanza la función del controlador que envía la oportunidad al servicio Rest para su eliminación.

En backbone se vuelve a utilizar la vista definida para las funcionalidades de edición y alta de los objetos Oportunidad, añadiendo una función a la que el manejador de eventos llama cuando se selecciona borrar una oportunidad. Esta función recibe el ID de la oportunidad a borrar y con tan solo hacer un *destroy*, que implementan todos los modelos, Backbone se encarga de hacer una petición al servidor de tipo DELETE, indicando el id del objeto Oportunidad a eliminar del sistema.

5.1.8 RF-10 Administración de la información asociada a las oportunidades

Para ambas tecnologías se ha creado un servicio REST en la capa del servidor por cada objeto asociado a las oportunidades (Usuario, cliente, Creador, Oficina, etc), el cual permite administrar la información asociada a las oportunidades, en nuestro caso únicamente está implementada la función de añadir.

En el caso de AngularJs, están involucradas dos partes, la vista del formulario donde se introducen los datos de la información asociada a las oportunidades, y el controlador de esta vista.

- En el controlador se define el objeto asociado a la oportunidad y el alcance del mismo, aunque no es necesario definir los atributos de este, se añaden dinámicamente. También se define la función que enviará la oportunidad al servicio Rest indicado anteriormente, AngularJs automáticamente transforma el objeto a Json para poder ser enviado a través del servicio Rest.
- En la vista se indica a que controlador está asociada esta. Para registrar los atributos de la oportunidad se indican los atributos del objeto asociado a la oportunidad mediante una etiqueta <input> de html, en esos input se le indica a que objeto está asociado y a que atributo de ese objeto, gracias al data-binding una vez que el usuario introduzca un valor en los input, el valor se registra en el objeto automáticamente.
- Una vez pulsado el botón de guardar de la vista, se lanza la función del controlador que envía la objeto asociado a la oportunidad al servicio Rest, para su guardado.

En el caso de Backbone será necesario crear, igual que se hizo para el objeto Oportunidad, una colección de cada tipo de objeto, así como un modelo que representa esta entidad, además de una vista para mostrar el listado de objetos y otra para la edición, alta y eliminación de los mismos.

Cada modelo define la url a la que se deben mandar las peticiones relativas al objeto o entidad en cuestión. El enrutador y el manejador de eventos tendrán las mismas funcionalidades comentadas en los puntos anteriores y las vistas definirán las mismas funciones antes mencionadas para realizar las funcionalidades necesarias para cada objeto. Será necesario también definir una plantilla para la lista de cada objeto y otra para la edición y alta de los mismos, de nuevo igual que con el objeto Oportunidad.

5.2 CONCLUSIONES GENERALES DEL PROYECTO

Este proyecto ha sido un punto de partida para poder integrar los dos *frameworks* javaScript MVC más relevantes del mercado en la arquitectura j-everis, para lo cual se han analizado los *frameworks* JavaScript MVC más relevantes.

El análisis se ha extendido a *frameworks* visuales, de creación de gráficos y herramientas de desarrollo. Dentro de los primeros, el estudio no se ha centro únicamente en los que son ejecutados en el cliente, si no que también se han analizado otros que son ejecutados en el servidor.

Para cada tipo analizado se han establecido unos criterios adecuados y un peso asociado a cada criterio. A continuación se investigó por dichas características de cada *framework* y se les estableció una nota, que tras ser ponderada en función de los pesos asociados a los criterios correspondientes ha generado finalmente una nota por cada framework.

Una vez elegidos los frameworks, se han desarrollado dos aplicaciones con las mismas funcionalidades, una aplicación con AngularJs y otra con Backbone, para posteriormente comparar el desarrollo con ambas tecnologías y analizar las ventajas y desventajas de ambos frameworks, este desarrollo también ha servido para establecer una estructura a seguir por los proyectos realizados con j-everis mediante estas tecnologías. Dicha estructura se integrará en j-everis para que se cree automáticamente al crear un proyecto con estas tecnologías y acelerar el proceso inicial del desarrollo de una aplicación en AngularJs y BackboneJS. También

ha servido para detectar patrones o funcionalidades repetitivas que se podrían integrar automáticamente mediante la arquitectura j-everis y evitar la escritura de código repetitivo, como por ejemplo al crear una vista automáticamente crear su controlador e inyectar las dependencias.

Estos desarrollos servirán de base para poder añadir dichas demos en la arquitectura J-everis y que sirvan de ejemplo funcional para los nuevos desarrolladores en dichas tecnologías.

5.3 POSIBLES MEJORAS

5.3.1 Mejoras de la aplicación

En esta sección explicamos una serie de mejoras para la aplicación:

- Uso beans con información deseada (en la capa del servidor) y no llevar todo el contenido de las entidades y no hacer tantas consultas a la base de datos, lo que penaliza el rendimiento.
- Búsqueda filtrada con máximo de resultados para no llevar tanta cantidad de contenido y agilizar acceso a datos
- Uso de roles, teniendo cada usuario acceso a la información que corresponda dependiendo del rol al que pertenezca.
- Exportar información restringiendo oportunidades seleccionadas o que cumplan ciertas condiciones

5.3.1.1 Ampliación de requisitos

Código	Nombre	Descripción
RF-10	Login	El sistema debe permitir loguearse a los usuarios y mostrar la información que corresponda en función del usuario y rol al que pertenezca.
RF-11	Notificaciones en pantalla	El sistema debe notificar en la pantalla principal las oportunidades por las que no ha habido contacto con el cliente en un tiempo determinado
RF-12	Notificaciones vía email	El sistema debe notificar semanalmente mediante correo electrónico las oportunidades por las que no ha habido contacto con el cliente en un tiempo determinado
RF-13	Administración de usuarios	El sistema debe permitir el registro, modificación y borrado de usuarios
RF-14	Roles de usuarios	El sistema debe permitir realizar distintas funciones dependiendo del rol al que esté asociado el usuario. Los roles serán Administrador, Contribuidor Restringido y Lector
RF-15	Permisos Administrador	El sistema debe permitir realizar todas las acciones al administrador. Podrá por tanto realizar tanto la gestión de las oportunidades como de los usuarios.
RF-17	Permisos Contribuidor Restringido	El sistema debe permitir al Contribuidor Restringido dar de alta nuevas oportunidades vía formulario, así como acceder a las mismas y modificalas. Sólo verá las oportunidades dadas de alta por él, así como exportarlas.
RF-18	Permisos Lector	El sistema debe permitir al Lector ver y exportar todas las oportunidades existentes, pero no podrá realizar ningún alta ni modificación.

6 BIBLIOGRAFÍA

AngularJS. [En línea] https://angularjs.org/.

Aptana. [En línea] http://www.aptana.com/.

Arrigoni, Ricardo. MrBool. [En línea] http://mrbool.com/top-11-javascript-frameworks/27382.

BackboneJS. [En línea] http://backbonejs.org/.

Bégaudeau, Stéphane. Generating blog posts. [En línea] http://stephanebegaudeau.tumblr.com/post/48776908163/everything-you-need-to-understand-to-start-with.

Bodnarchuk, Michael. 2013. JSter JavaScript Catalog. [En línea] 6 de Febrero de 2013. http://jster.net/blog/why-should-you-use-client-side-mvc-framework#.VAhVdfl_snU.

Bootstrap. [En línea] http://getbootstrap.com/.

Capuccino. [En línea] http://www.cappuccino-project.org/.

D3.js. [En línea] http://d3js.org/.

Davis, Thomas. 2012. *Backbone Tutorials.* s.l. : Leanpub, 2012.

—. 2011. Github. [En línea] 1 de Febrero de 2011. http://thomasdavis.github.io/2011/02/01/backbone-introduction.html.

Daws, Sharon. 2012. Pikemere Web Services. [En línea] 16 de Marzo de 2012. http://www.pikemere.co.uk/blog/flot-how-to-create-pie-charts/.

Dojo. [En línea] http://dojotoolkit.org/.

Ember. [En línea] http://emberjs.com/.

Google Charts. [En línea] https://developers.google.com/chart/?hl=es.

Gube, Jacob. 2010. Smashing Magazine. [En línea] 5 de Octubre de 2010. http://www.smashingmagazine.com/2009/02/08/50-extremely-useful-javascript-tools/.

Hempton, Gordon. CodeBrief. [En línea] http://codebrief.com/2012/01/the-top-10-javascript-mvc-frameworks-reviewed/.

Hethey, Jonathan M. 2013. JonathanMH Digital Media. [En línea] 24 de Septiembre de 2013. http://jonathanmh.com/best-javascript-mvc-frameworks-2013-2014/.

HighCharts. [En línea] http://www.highcharts.com/.

JavaFX. [En línea] http://www.oracle.com/technetwork/es/java/javafx/overview/index.html.

JqPlot. [En línea] http://www.jqplot.com/.

JQueryUI. [En línea] http://jqueryui.com/.

Kerr, Dean. 2013. Bespoke Software Development Company. [En línea] 6 de Diciembre de 2013. http://www.scottlogic.com/blog/dkerr/.

Louthan, Jenny. 2013. Uncorked Studios. [En línea] 4 de Diciembre de 2013. http://uncorkedstudios.com/blog/multipartformdata-file-upload-with-angularjs.

McDearmon, Mike. Mike McDearmon Posts. [En línea] http://mikemcdearmon.com/portfolio/techposts/charting-libraries-using-d3.

McKeachie, Craig. 2013. Funny Ant. [En línea] 19 de Septiembre de 2013. http://www.funnyant.com/choosing-javascript-mvc-framework/.

Oliveira, Hébert. 2013. uaiHebert. [En línea] 3 de Noviembre de 2013. http://uaihebert.com/complete-web-application-angular-twitter-bootstrap-spring-mvc-data-and-security/.

Osmani, Addy. 2012. Smashing Magazine. [En línea] 27 de Julio de 2012. http://www.smashingmagazine.com/2012/07/27/journey-through-the-javascript-mvc-jungle/.

Picca, Carlos. Codehero. [En línea] http://codehero.co/ember-js-desde-cero-introduccion-e-instalacion/.

QUARK - ARCHEX - Framework j-everis. [En línea] http://quark.everis.int/confluence/display/ARCHEX/Framework+j-everis.

Roldán, Carlos Santana. 2013. CodeJobs. [En línea] 11 de Abril de 2013. http://www.codejobs.biz/es/blog/2013/04/11/los-5-mejores-frameworks-mvc-de-javascript#sthash.eSYlwyZb.xjoDHUkM.dpbs.

Sanderson's, Steven. 2012. Steven Sanderson's Blog. [En línea] 1 de Agosto de 2012. http://blog.stevensanderson.com/2012/08/01/rich-javascript-applications-the-seven-frameworks-throne-of-js-2012/.

Sencha Ext JS. [En línea] http://www.sencha.com/products/extjs/.

Sorst, Philip. 2013. GitHub. [En línea] 2 de Abril de 2013. https://github.com/philipsorst/angular-rest-springsecurity/blob/master/README.md.

Sproutcore. [En línea] http://sproutcore.com/.

Sublime Text. [En línea] http://www.sublimetext.com/.

Synodinos, Dio. 2013. InfoQ. [En línea] 5 de Febrero de 2013. http://www.infoq.com/author/Dio-Synodinos.

TodoMVC. [En línea] http://todomvc.com/.

Vaadin. [En línea] https://vaadin.com/home.

Webstorm. [En línea] http://www.jetbrains.com/webstorm/.

Wiederkehr, Benjamin. 2009. Data Visualization. [En línea] 29 de Julio de 2009. http://datavisualization.ch/tools/13-javascript-libraries-for-visualizations/.

Zarate, Henry Alvaro. 2013. enbolivia Blog - Soluciones en Internet. [En línea] 1 de Marzo de 2013. http://enboliviacom.wordpress.com/2013/03/01/highcharts-libreria-para-creacion-de-graficos/.

—. 2014. enbolivia Blog - Soluciones en Internet. [En línea] 21 de Febrero de 2014. http://enboliviacom.wordpress.com/2014/02/21/desarrollo-web-con-sublime-text/.

ZK. [En línea] http://www.zkoss.org/.

7 ANEXOS

7.1 DETALLE DE LAS TABLAS COMPARATIVAS

En este apartado se mostrarán todas las tablas con los resultados de cada comparación detallados, con la explicación de la nota asignada para cada característica de cada *framework*, así como la relación de pesos impuesta en cada comparación.

A continuación se detalla de nuevo la estructura de las tablas comparativas y la metodología que se ha utilizado para la creación de las mismas.

Para cada tipo de *framework* a analizar se han definido las siguientes tablas:

- **Análisis:** existe una tabla de este tipo por cada *framework*, en la que se detalla la puntuación obtenida en cada criterio junto a una explicación aclaratoria de dicha nota.
- **Pesos:** en esta tabla se pondera la puntuación de cada criterio con el peso asociado a este.
- **Resultados:** tabla resumen con la puntuación final de cada *framework* en cada criterio, junto a la nota final del *framework*.

La nota asignada a cada *framework* para cada una de las características analizadas está definida por los siguientes niveles:

Cobertura		Nivel
N	0%	**Ninguna**
B	25%	**Bajo**
M	50%	**Medio**
A	75%	**Alto**
T	100%	**Total**

Una vez asignado el nivel, este será multiplicado por el peso asociado al criterio de comparación en cuestión, obteniendo como resultado una nota.

Los pesos están comprendidos en un rango del 1 al 3:

Coeficiente de Peso	
Imprescindible	**3**
Importante	**2**
Aconsejable	**1**

Una vez realizado este proceso para cada criterio, se calcula la media ponderada de cada *framework*, sumando las notas obtenidas y dividiendo el resultado entre la suma total de los pesos. Ésta será la nota final del *framework* en la comparativa.

7.1.1 Tablas detalladas de los *frameworks* Client MVC analizados

7.1.1.1 Tablas de análisis

7.1.1.1.1 AngularJS

		Angularjs		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	M	**50%**	Proporciona componentes propios mediante extension de etiquetas html y permite crear sus propios componentes mediante directivas.
	Navegación	A	**75%**	Dispone de un potente enrutador.
	Integración otros frontales	M	**50%**	Proporciona gran flexibilidad para enriquecerse de otros *frameworks* o librerias como jquery, ui -angular o bootstrap.
	Manejo de Eventos	A	**75%**	Dispone de un escuchador de eventos.
	Seguridad	A	**75%**	Existen multitud de librerias que se pueden integrar como: lOauth/2 y google account. Se puede extender su comportamiento usando directivas de angular.
	Gestores comunes (errores, sesiones,configuración)	A	**75%**	Dispone de un manejador de errores. También dispone de un rico servicio de llamadas http para consumir servicios, internacionalización i18n.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por todos los navegadores modernos de escritorio y móvil. La versión 8 de internet explorer lo soporta, pero con algunas restricciones, no soporta la version 1.3 de Angularjs.
Desarrollo	Herramientas de desarrollo	A	**75%**	Existen multitud de *plugin*s para eclipse y otros IDE's como WebStorm y SublimeText. Para debugger existen *plugin*s como Batarang.
	Herramientas de test	A	**75%**	Buena integración con herramientas de Test como: Karma, jasmine y phantom js.
	Herramientas de despliegue	A	**75%**	Fácil integración con la herramienta de despliegue Grunt.
	Facilitadores (Plantillas, proyectos demo)	A	**75%**	Fácil integración con Yeoman y con el *plugin* generator angularjs, una de las mejores herramientas para acelerar las primeras fases de desarrollo de la aplicación.
	Analisis de la calidad (automática)	B	**25%**	No proporciona una funcionalidad específica, se debe desarrollar reglas específicas.
	Independencia del servidor	T	**100%**	Al comunicarse a traves de servicios Rest y a traves de Json, la tecnología del servidor no influye.
Operación	Monitorización de aplicación	N	**0%**	Ninguna.
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	N	**0%**	Ninguna.
Soporte	Soporte proveedor	A	**75%**	Tiene el respaldo de un gigante como Google, la cual usa en algunos de sus productos.
	Licencia producto	A	**75%**	Licencia MIT, permite usar, copiar, modificar, publicar, sublicenciar o vender el código, incluyendo el copyright.
	Evolución	A	**75%**	Es uno de los *frameworks* js mas populares y su comunidad es la que mayor crecimiento está experimentando.

7.1.1.1.2 Backbone

		Backbone		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	M	**50%**	No provee componentes, necesita apoyarse en *frameworks* complementarios como jquery, bootstrap. Aunque ermite la creación y reutilización de componentes.
	Navegación	A	**75%**	Dispone de un enrutador.
	Integración otros frontales	M	**50%**	Proporciona gran flexibilidad para enriquecerse de otros *frameworks* o librerias como jquery, ui -angular o bootstrap.
	Manejo de Eventos	A	**75%**	Dispone de un potente manejador de eventos.
	Seguridad	M	**50%**	No proporciona una funcionalidad específica, aunque permite la integración con los principales servicios de loggin.
	Gestores comunes (errores, sesiones,configuración)	M	**50%**	Proporciona una gran variedad de servicios, con los cuales puedes contruir tus propios servicios.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por todos los navegadores modernos. La unica restricción es la dependencia con jquery.
Desarrollo	Herramientas de desarrollo	A	**75%**	Existen multitud de *plugin*s para eclipse y otros other IDE's como WebStorm y SublimeText. Para debugger existe la herramienta BackBone Eye and debugger.
	Herramientas de test	A	**75%**	Permite la integración con los *Frameworks* de test: Jasmine, karma.
	Herramientas de despliegue	A	**75%**	Fácil integración con la herramienta de despliegue Grunt.
	Facilitadores (Plantillas, proyectos demo)	A	**75%**	Fácil integración con Yeoman, una de las mejores herramientas para acelerar las primeras fases de desarrollo de la aplicación.
	Analisis de la calidad (automática)	M	**50%**	No proporciona una funcionalidad específica, se debe desarrollar reglas específicas.
	Independencia del servidor	T	**100%**	Al comunicarse a traves de servicios Rest y a traves de Json, la tecnología del servidor no influye.
Operación	Monitorización de aplicación	N	**0%**	Ninguna.
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	N	**0%**	Ninguna.
Soporte	Soporte proveedor	A	**75%**	DocumentCloud la empresa detrás de Backbone y su creador Jeremy Ashkenas, participan activamente en el proyecto.
	Licencia producto	A	**75%**	Licencia MIT, permite usar, copiar, modificar, publicar, sublicenciar o vender el código, incluyendo el copyright.
	Evolución	A	**75%**	Se trata de uno de los *Frameworks* MVC javaScript mas populares, lo utilizan grandes compañias y tiene una gran comunidad de usuarios.

7.1.1.1.3 Capuccino

		Capuccino		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	M	**50%**	Proporciona potentes componentes visuales, pero es dificil integrarlo con otros *Frameworks* complemetarios.
	Navegación	A	**75%**	Dispone de un enrutador.
	Integración otros frontales	M	**50%**	Se puede integrar con otros *Frameworks* mediante div's e iFrame's
	Manejo de Eventos	A	**75%**	Proporciona un potente gestor de eventos.
	Seguridad	M	**50%**	Necesita ser desarrollada.
	Gestores comunes (errores, sesiones,configuración)	M	**50%**	Necesitan serdesarrollados, aunque existe herramientas de terceros.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por la mayoria de navegadores. Explorer 7+, Firefox 2+, Safari 3+, Opera 9+, y Google Chrome.
Desarrollo	Herramientas de desarrollo	B	**25%**	No existen *plugin*s para eclipse. Webstorm, Sublimetext y el SDK de desarrollo están soportados por Xcode para MacOS.
	Herramientas de test	M	**50%**	Posee una herramienta de Test de Objetos.
	Herramientas de despliegue	B	**25%**	Muestra dificultades para integrarse con herramientas de despliege, por el ecositema cerrado de sus herramientas.
	Facilitadores (Plantillas, proyectos demo)	M	**50%**	El SDK porporciona facilitadores.
	Analisis de la calidad (automática)	B	**25%**	No existe una herramienta específica.
	Independencia del servidor	T	**100%**	Al comunicarse a traves de servicios Rest y a traves de Json, la tecnología del servidor no influye.
Operación	Monitorización de aplicación	N	**0%**	Ninguna.
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	N	**0%**	Ninguna.
Soporte	Soporte proveedor	B	**25%**	Propiedad de Motorola, aunque no participa activamente.
	Licencia producto	M	**50%**	GNU (Licencia Pública General, ofrece la libertad de usar, estudiar, compartir y modificar el software).
	Evolución	B	**25%**	Tras la compra por parte de Motorola en 2010, la evolición del *framework* se ha visto disminuida.

7.1.1.1.4 Ember

		Ember		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	A	**75%**	Proporciona componentes visuales y permite crear tus propios componentes.
	Navegación	A	**75%**	Dispone de un servicio de enrutado.
	Integración otros frontales	M	**50%**	Se puede integrar con otros *Frameworks* mediante div's e iFrame's
	Manejo de Eventos	A	**75%**	Proporciona un manejador de eventos.
	Seguridad	M	**50%**	Compatible con otras medidas de seguridad como Content Security Policy (CSP), HTTPS (SSL/TLS).
	Gestores comunes (errores, sesiones,configuración)	A	**75%**	Dispone de un gestor de errores , invocador de servicios rest, i18n y proporciona servicios customizables.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por la mayoría de navegadores. Soportado por IE8 con algunas restricciones.
Desarrollo	Herramientas de desarrollo	M	**50%**	No existen *plugin*s para eclipse, se pueden utilizar otros editores como sublimetext y webstorm.
	Herramientas de test	A	**75%**	Fácil integración con herramientas de Test como: Karma, jasmine y phantom js.
	Herramientas de despliegue	A	**75%**	Fácil integración con la herramienta de despliegue Grunt.
	Facilitadores (Plantillas, proyectos demo)	M	**50%**	Fácil integración con Yeoman, una de las mejores herramientas para acelerar las primeras fases de desarrollo de la aplicación.
	Analisis de la calidad (automática)	M	**50%**	No proporciona una funcionalidad específica, se debe desarrollar reglas específicas.
	Independencia del servidor	T	**100%**	Al comunicarse a traves de servicios Rest y a traves de Json, la tecnología del servidor no influye.
Operación	Monitorización de aplicación	N	**0%**	Ninguna.
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	N	**0%**	Ninguna.
Soporte	Soporte proveedor	M	**50%**	El desarrollo de este *framework* es muy activo debido a la gran cantidad de contribuidores que posee, se desarrollan versiones estables con una alta frecuencia.
	Licencia producto	A	**75%**	Licencia MIT, permite usar, copiar, modificar, publicar, sublicenciar o vender el código, incluyendo el copyright.
	Evolución	M	**50%**	Se trata de uno de los *frameworks* que más evolución ha experimentado en los últimos años junto con angularjs y backbone.

7.1.1.1.5 Ext JS

		Ext JS		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	A	**75%**	Ofrece gan cantidad de componentes visuales.
	Navegación	A	**75%**	Proporciona un enrutador de servicios.
	Integración otros frontales	M	**50%**	Se pueden integrar otros *Frameworks* mediante div's e iFrame's
	Manejo de Eventos	A	**75%**	Proporciona un manejador de eventos.
	Seguridad	M	**50%**	No proporciona una funcionalidad específica, aunque permite la integración con los principales servicios de loggin.
	Gestores comunes (errores, sesiones,configuración)	A	**75%**	Proporciona una serie de servicios que permiten customizar el comportamiento de la aplicación.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por la mayoría de navegadores. Soportado por IE8 con algunas restricciones.
Desarrollo	Herramientas de desarrollo	T	**100%**	Existe un *plugin* para eclipse, tambien se pueden utilizar otros editores como sublimetext, webstorm y las herramientas del SDK.
	Herramientas de test	A	**75%**	Fácil integración con herramientas de Test como: Karma, jasmine y phantom js.
	Herramientas de despliegue	A	**75%**	Fácil integración con la herramienta de despliegue Grunt.
	Facilitadores (Plantillas, proyectos demo)	A	**75%**	Fácil integración con Yeoman, una de las mejores herramientas para acelerar las primeras fases de desarrollo de la aplicación.
	Analisis de la calidad (automática)	M	**50%**	No proporciona una funcionalidad específica, se debe desarrollar reglas específicas.
	Independencia del servidor	T	**100%**	Al comunicarse a traves de servicios Rest y a traves de Json, la tecnología del servidor no influye.
Operación	Monitorización de aplicación	N	**0%**	Ninguna.
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	N	**0%**	Ninguna.
Soporte	Soporte proveedor	B	**25%**	Tras la fusión en 2010 con JQTouch y Raphaël para crear Sencha, centrado en la creación de interfaces móviles, la evolución ha sido muy escasa.
	Licencia producto	A	**75%**	GNU (Licencia Pública General, ofrece la libertad de usar, estudiar, compartir y modificar el software).
	Evolución	B	**25%**	Desde 2011 no aparece ninguna versión estable.

7.1.1.1.6 JQuery

		JQuery		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	A	**75%**	Existe gran cantidad de componentes, librerías y *plugin*s basados en jquery, lo que implica la existencia de gran cantidad de componentes robustos.
	Navegación	B	**25%**	No dispone de un enrutador nativo.
	Integración otros frontales	M	**50%**	Se pueden integrar otros *Frameworks* mediante div's e iFrame's
	Manejo de Eventos	B	**25%**	No dispone de un despachador de eventos, aunque soporta algunos eventos.
	Seguridad	M	**50%**	Necesita ser desarrollada, aunque existen multitud de herramientas de terceros basdas en jquery.
	Gestores comunes (errores, sesiones,configuración)	B	**25%**	Proporciona llamadas AJAX, aunque tenemos que contruir los servicios rest. No dispone de data-binding.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por los navegadores mas modernos. Soportado por IE8 con algunas restricciones.
Desarrollo	Herramientas de desarrollo	A	**75%**	Existen *plugin*s para Eclipse y existen editores que lo soportan como Sublimetext y Webstorm.
	Herramientas de test	A	**75%**	Fácil integración con herramientas de Test como: Karma, Jasmine y phantom js.
	Herramientas de despliegue	A	**75%**	Fácil integración con la herramienta de despliegue Grunt.
	Facilitadores (Plantillas, proyectos demo)	A	**75%**	Fácil integración con Yeoman, una de las mejores herramientas para acelerar las primeras fases de desarrollo de la aplicación.
	Analisis de la calidad (automática)	M	**50%**	No proporciona una funcionalidad específica, se debe desarrollar reglas específicas.
	Independencia del servidor	T	**100%**	Al comunicarse a traves de servicios Rest y a traves de Json, la tecnología del servidor no influye.
Operación	Monitorización de aplicación	N	**0%**	Ninguna.
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	N	**0%**	Ninguna.
Soporte	Soporte proveedor	A	**75%**	La empresa desarrolladora jQuery Team, posee una gran apoyo por parte de la comuidad, impulsado por grandes empresas.
	Licencia producto	A	**75%**	Licencia MIT, permite usar, copiar, modificar, publicar, sublicenciar o vender el código, incluyendo el copyright.
	Evolución	A	**75%**	Desde 2006, ha experimentado un gran crecimiento, convirtiendose practicamente en un estándar. Tiene una versión estable cada año.

7.1.1.1.7 Knockout

		Knockout		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	N	**0%**	No porporciona ningún compenente visuales ,se necesitan *Frameworks* de terceros.
	Navegación	M	**50%**	Utiliza usingSammy.js para enrutar.
	Integración otros frontales	M	**50%**	Se pueden integrar otros *Frameworks* mediante div's e iFrame's
	Manejo de Eventos	M	**50%**	Dispone de un manejador de eventos.
	Seguridad	M	**50%**	Necesita ser desarrollado. Existen módulos para manejar la seguridad como knockout-secure-binding.
	Gestores comunes (errores, sesiones,configuración)	A	**75%**	Existen módulos para manejar la internacionalización como KnockoutJS-i18n). Existen también módulos para manejar llamas a servicios y Ajax como Knockout-Localization-Binding.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por los navegadores mas modernos. Las dependencias vienen de los *Frameworks* complementarios como jQuery.
Desarrollo	Herramientas de desarrollo	M	**50%**	Existen *plugin*s para Eclipse. También existen otros IDE's como WebStorm y SublimeText. Existen herramientas para realizar debugg como: Knockout Context Debugger y Knockout-debug.
	Herramientas de test	A	**75%**	Fácil integración con herramientas de Test como: Karma, Jasmine y phantom js.
	Herramientas de despliegue	A	**75%**	Fácil integración con la herramienta de despliegue Grunt.
	Facilitadores (Plantillas, proyectos demo)	A	**75%**	Fácil integración con Yeoman, una de las mejores herramientas para acelerar las primeras fases de desarrollo de la aplicación.
	Analisis de la calidad (automática)	M	**50%**	No proporciona una funcionalidad específica, se debe desarrollar reglas específicas.
	Independencia del servidor	T	**100%**	Al comunicarse a traves de servicios Rest y a traves de Json, la tecnología del servidor no influye.
Operación	Monitorización de aplicación	N	**0%**	Ninguna.
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	N	**0%**	Ninguna.
Soporte	Soporte proveedor	M	**50%**	Existen versiones estables con frecuenía, posee una comunidad muy activa en su desarrollo.
	Licencia producto	A	**75%**	Licencia MIT, permite usar, copiar, modificar, publicar, sublicenciar o vender el código, incluyendo el copyright.
	Evolución	A	**75%**	Desde 2010 ha experimentado un gran crecimiento.

7.1.1.1.8 SproutCore

		SproutCore		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	A	**75%**	Proporciona un rico conjunto de componentes visuales mendiante la librería SproutCore UI.
	Navegación	A	**75%**	Dispone de un enrutador que puede usar el histórico del navegador.
	Integración otros frontales	M	**50%**	Se pueden integrar otros *Frameworks* mediante div's e iFrame's
	Manejo de Eventos	A	**75%**	Dispone de la api SC.Event para manejar complejos eventos.
	Seguridad	M	**50%**	Necesita ser desarrolado.
	Gestores comunes (errores, sesiones,configuración)	A	**75%**	Proporciona una gran variedad de servicios como SC.Log, SC.ExceptionHandler, SC.Controller.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por todos los navegadores modernos.
Desarrollo	Herramientas de desarrollo	M	**50%**	Existen *plugin*s para Eclipse. También existen otros IDE's como WebStorm y SublimeText. Existen herramientas para realizar debugg como: SC.StatechartManager.
	Herramientas de test	A	**75%**	Fácil integración con herramientas de Test como: Karma, Jasmine y phantom js.
	Herramientas de despliegue	A	**75%**	Fácil integración con la herramienta de despliegue Grunt.
	Facilitadores (Plantillas, proyectos demo)	A	**75%**	Fácil integración con Yeoman, una de las mejores herramientas para acelerar las primeras fases de desarrollo de la aplicación.
	Analisis de la calidad (automática)	M	**50%**	No proporciona una funcionalidad específica, se debe desarrollar reglas específicas.
	Independencia del servidor	T	**100%**	Al comunicarse a traves de servicios Rest y a traves de Json, la tecnología del servidor no influye.
Operación	Monitorización de aplicación	N	**0%**	Ninguna.
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	N	**0%**	Ninguna.
Soporte	Soporte proveedor	M	**50%**	Apple contribuyó activamente en el desarrollo de este *framework* y lo utiliza en algunos de sus productos como iWork.
	Licencia producto	A	**75%**	Licencia MIT, permite usar, copiar, modificar, publicar, sublicenciar o vender el código, incluyendo el copyright.
	Evolución	B	**25%**	Experimentó una gran evolución desde 2007, anque desde que Facebook compró la empresa en 2011 no han aparecido mas versiones.

7.1.1.2 Tablas de pesos

Área	Caracterítica	Coeficiente de Peso	Angularjs		Backbone		Capuccino		Ember		Ext JS		JQuery		Knockout		SproutCore	
			Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación	
Ejecución	Componentización	3	50%	1,5	50%	1,5	50%	1,5	75%	2,3	75%	2,3	75%	2,3	0%	0,0	75%	2,3
	Navegación	3	75%	2,3	75%	2,3	75%	2,3	75%	2,3	75%	2,3	25%	0,8	50%	1,5	75%	2,3
	Integración otros frontales	2	50%	1,0	50%	1,0	50%	1,0	50%	1,0	50%	1,0	50%	1,0	50%	1,0	50%	1,0
	Manejo de Eventos	2	75%	1,5	75%	1,5	75%	1,5	75%	1,5	75%	1,5	25%	0,5	50%	1,0	75%	1,5
	Seguridad	2	75%	1,5	50%	1,0	50%	1,0	50%	1,0	50%	1,0	50%	1,0	50%	1,0	50%	1,0
	Gestores comunes (errores, sesiones,configuración)	3	75%	2,3	50%	1,5	50%	1,5	75%	2,3	75%	2,3	25%	0,8	75%	2,3	75%	2,3
	Multidispositivo/Multinavegador	3	75%	2,3	75%	2,3	75%	2,3	75%	2,3	75%	2,3	75%	2,3	75%	2,3	75%	2,3
				68%		61%		61%		69%		69%		47%		50%		69%

Área	Caracterítica	Coeficiente de Peso	Angularjs		Backbone		Capuccino		Ember		Ext JS		JQuery		Knockout		SproutCore	
			Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación	
Desarrollo	Herramientas de desarrollo	3	75%	2,3	75%	2,3	25%	0,8	50%	1,5	100%	3,0	75%	2,3	50%	1,5	50%	1,5
	Herramientas de test	2	75%	1,5	75%	1,5	50%	1,0	75%	1,5	75%	1,5	75%	1,5	75%	1,5	75%	1,5
	Herramientas de despliegue	1	75%	0,8	75%	0,8	25%	0,3	75%	0,8	75%	0,8	75%	0,8	75%	0,8	75%	0,8
	Facilitadores (Plantillas, proyectos demo)	2	75%	1,5	75%	1,5	50%	1,0	50%	1,0	75%	1,5	75%	1,5	75%	1,5	75%	1,5
	Analisis de la calidad (automática)	2	25%	0,5	50%	1,0	25%	0,5	50%	1,0	50%	1,0	50%	1,0	50%	1,0	50%	1,0
	Independencia del servidor	3	100%	3,0	100%	3,0	100%	3,0	100%	3,0	100%	3,0	100%	3,0	100%	3,0	100%	3,0
				73%		77%		50%		67%		83%		77%		71%		71%

Área	Caracterítica	Coeficiente de Peso	Angularjs		Backbone		Capuccino		Ember		Ext JS		JQuery		Knockout		SproutCore	
			Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación	
Operación	Monitorización de aplicación	1	0%	0,0	0%	0,0	0%	0,0	0%	0,0	0%	0,0	0%	0,0	0%	0,0	0%	0,0
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	1	0%	0,0	0%	0,0	0%	0,0	0%	0,0	0%	0,0	0%	0,0	0%	0,0	0%	0,0
				0%		0%		0%		0%		0%		0%		0%		0%

Área	Caracterítica	Coeficiente de Peso	Angularjs		Backbone		Capuccino		Ember		Ext JS		JQuery		Knockout		SproutCore	
			Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación		Puntuación	
Soporte	Soporte proveedor	2	75%	1,5	75%	1,5	25%	0,5	50%	1,0	25%	0,5	75%	1,5	50%	1,0	50%	1,0
	Licencia producto	3	75%	2,3	75%	2,3	50%	1,5	75%	2,3	75%	2,3	75%	2,3	75%	2,3	75%	2,3
	Evolución	3	75%	2,3	75%	2,3	25%	0,8	50%	1,5	25%	0,8	75%	2,3	75%	2,3	25%	0,8
				75%		75%		34%		59%		44%		75%		69%		50%

7.1.1.3 Tabla de resultados

Área	Angularjs	Backbone	Capuccino	Ember	Ext JS	JQuery	Knockout	SproutCore
Ejecución	**68%**	**61%**	**61%**	**69%**	**69%**	**47%**	**50%**	**69%**
Componentización	50%	50%	50%	75%	75%	75%	0%	75%
Navegación	75%	75%	75%	75%	75%	25%	50%	75%
Integración otros frontales	50%	50%	50%	50%	50%	50%	50%	50%
Manejo de Eventos	75%	75%	75%	75%	75%	25%	50%	75%
Seguridad	75%	50%	50%	50%	50%	50%	50%	50%
Gestores comunes (errores, sesiones,configuración)	75%	50%	50%	75%	75%	25%	75%	75%
Multidispositivo/Multinavegador	75%	75%	75%	75%	75%	75%	75%	75%
Desarrollo	**73%**	**77%**	**50%**	**67%**	**83%**	**77%**	**71%**	**71%**
Herramientas de desarrollo	75%	75%	25%	50%	100%	75%	50%	50%
Herramientas de test	75%	75%	50%	75%	75%	75%	75%	75%
Herramientas de despliegue	75%	75%	25%	75%	75%	75%	75%	75%
Facilitadores (Plantillas, proyectos demo)	75%	75%	50%	50%	75%	75%	75%	75%
Analisis de la calidad (automática)	25%	50%	25%	50%	50%	50%	50%	50%
Independencia del servidor	100%	100%	100%	100%	100%	100%	100%	100%
Operación	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**	**0%**
Monitorización de aplicación	0%	0%	0%	0%	0%	0%	0%	0%
Tareas de mantenimiento (Jobs, backups y explotación de logs)	0%	0%	0%	0%	0%	0%	0%	0%
Soporte	**75%**	**75%**	**34%**	**59%**	**44%**	**75%**	**69%**	**50%**
Soporte proveedor	75%	75%	25%	50%	25%	75%	50%	50%
Licencia producto	75%	75%	50%	75%	75%	75%	75%	75%
Evolución	75%	75%	25%	50%	25%	75%	75%	25%
Media	**54,03%**	**53,26%**	**36,37%**	**49,03%**	**48,97%**	**49,79%**	**47,48%**	**47,65%**

7.1.2 Tablas detalladas de los *frameworks* Server MVC

7.1.2.1 Tabla de análisis

7.1.2.1.1 Vaadin

		Vaadin		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	A	**75%**	Proporciona un gran conjunto de componentes visuales, que permiten customización.
	Navegación	A	**75%**	Dispone de un enrutador capaz de manejar flujos y parametros.
	Integración otros frontales	M	**50%**	Se puede integrar con otros *Frameworks* mediante iFrame's
	Manejo de Eventos	A	**75%**	Dispone de un despachador de eventos basado en colas internas.
	Seguridad	M	**50%**	Necesita desarrollarse como una aplicación Java.
	Gestores comunes (errores, sesiones,configuración)	A	**75%**	Al tratarse de un *Framework* Server MVC se diponene de todas las posibilidades de las que dispone Java.
	Multidispositivo/Multinavegador	A	**75%**	Soportado todos los navegadores modernos: Internet Explorer 6+, Firefox, Chrome, Safari, iOS Safari, Opera. La versión depende de la platilla elegida.
Desarrollo	Herramientas de desarrollo	A	**75%**	Existe un *plugin* para Eclipse y para Netbeans. Dispone de una integración con Maven.
	Herramientas de test	M	**50%**	Herramientas de Test Java. El Testeo del Html se hace mas dificil al ser códio generado.
	Herramientas de despliegue	A	**75%**	Al usar Maven se puede inegrar con multitud de herramientas de despliegue.
	Facilitadores (Plantillas, proyectos demo)	M	**50%**	Mediante Maven, permite el uso de Spring.
	Analisis de la calidad (automática)	M	**50%**	Existen añalisis de java, pero no hay un análisis específico de Vaadin.
	Independencia del servidor	N	**0%**	Depende de un servidor java.
Operación	Monitorización de aplicación	B	**25%**	Existe una herramienta para monitorizar el código Java peor es difícil de analizar.
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	B	**25%**	Permite la reutilización de servicios Java.
Soporte	Soporte proveedor	M	**50%**	Comunidad activa con versiones actualizadas cada año.
	Licencia producto	A	**75%**	Licencia Apache, permite modificar y distribuir el código, únicamente se exige que se informe a los receptores que en la distribución se ha usado código con la Licencia Apache.
	Evolución	A	**75%**	Desde 2006 ha tenido una versión estable cada año con importantes novedades.

7.1.2.1.2 ZK

		ZK		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	A	**75%**	Proporciona un gran conjunto de componentes visuales, que permiten customización.
	Navegación	T	**100%**	Dispone de un enrutador capaz de manejar flujos y parametros.
	Integración otros frontales	M	**50%**	Se pueden integrar otros *Frameworks* mediante iFrame's.
	Manejo de Eventos	A	**75%**	Dispone de un despachador de eventos basado en colas internas.
	Seguridad	M	**50%**	Necesita desarrollarse como una aplicación Java.
	Gestores comunes (errores, sesiones,configuración)	A	**75%**	Soportado mediante Java.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por todos los navegadores modernos: Internet Explorer 6+, Firefox, Chrome, Safari, iOS Safari, Opera.
Desarrollo	Herramientas de desarrollo	A	**75%**	Existe un *plugin* de ZK para eclipse.
	Herramientas de test	M	**50%**	Herramientas de Test Java. El Testeo del Html se hace mas dificil al ser códio generado.
	Herramientas de despliegue	A	**75%**	Al usar Maven se puede integrar con multitud de herramientas de despliegue.
	Facilitadores (Plantillas, proyectos demo)	A	**75%**	Mediante Maven, permite el uso de Spring.
	Analisis de la calidad (automática)	M	**50%**	Existen herramientas de análisis mediante Java, pero no existe una análisis específico para ZK.
	Independencia del servidor	N	**0%**	Depende de un servidor java.
Operación	Monitorización de aplicación	B	**25%**	Existen herramientas Java para la monitorización de la aplicación Java monitoring tools as Dynatrace but user experence is difficult to analyxe
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	B	**25%**	Permite la reutilización de servicios Java.
Soporte	Soporte proveedor	M	**50%**	Posee versiones estables con una alta frecuencia.
	Licencia producto	M	**50%**	GNU (Licencia Pública General, ofrece la libertad de usar, estudiar, compartir y modificar el software).
	Evolución	M	**50%**	Desde sus inicios ha experimentado una gran evolución debido a su gran funcionalidad y al estar escrita en Java.

7.1.2.1.3 JavaFX

		JavaFX		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	A	**75%**	Proporciona un rico conjunto de componentes visuales y proporciona mecanismos de customización de los componentes.
	Navegación	T	**100%**	Proporciona mecanismos de enrutamiento.
	Integración otros frontales	B	**25%**	Permite integrarse con otros forntales mediante la herramienta Embed Browser.
	Manejo de Eventos	A	**75%**	Proporciona un despachador de eventos basado en colas.
	Seguridad	M	**50%**	Debe ser desarrollado como una aplicación Java.
	Gestores comunes (errores, sesiones,configuración)	A	**75%**	Soportado mediante Java.
	Multidispositivo/Multinavegador	N	**0%**	No tiene soporte nativo.
Desarrollo	Herramientas de desarrollo	A	**75%**	Eclipse, Netbeans mediante el *plugin* del SDK.
	Herramientas de test	M	**50%**	Herramientas de Test Java. El Testeo del Html se hace mas dificil al ser códio generado.
	Herramientas de despliegue	B	**25%**	No proporciona herramientas específicas, necesita de *plugin* para su despliegue.
	Facilitadores (Plantillas, proyectos demo)	A	**75%**	Mediante herramientas Java como maven y Spring.
	Analisis de la calidad (automática)	M	**50%**	Proporciona herramientas de análisis de la calidad, aunque los análisis son poco detallados.
	Independencia del servidor	N	**0%**	Depende de un servidor java.
Operación	Monitorización de aplicación	B	**25%**	Existen herramientas de monitorización de terceros como Dynatrace.
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	B	**25%**	No proporciona herramientas por defecto, auqnue permite reusar servicios Java.
Soporte	Soporte proveedor	A	**75%**	Desarrollado por Oracle, implicado activamente en el desarrollo del *framework*.
	Licencia producto	M	**50%**	Licencia EULA (En este tipo de licencias el propietario establece los criterios de distribución, por lo general no permiten que el software sea modificado, desensamblado, copiado o distribuido de formas no especificadas en la propia licencia)
	Evolución	A	**75%**	Actualizado con cada versión de Java, actuamelmete disponible la versión javaFx 8.

7.1.2.2 Tablas de pesos

Área	Característica	Coeficiente de Peso	Vaadin		ZK		JavaFX	
			Puntuación		Puntuación		Puntuación	
Ejecución	Componentización	3	75%	2,3	75%	2,3	75%	2,3
	Navegación	3	75%	2,3	100%	3,0	100%	3,0
	Integración otros frontales	2	50%	1,0	50%	1,0	25%	0,5
	Manejo de Eventos	2	75%	1,5	75%	1,5	75%	1,5
	Seguridad	2	50%	1,0	50%	1,0	50%	1,0
	Gestores comunes (errores, sesiones,configuración)	3	75%	2,3	75%	2,3	75%	2,3
	Multidispositivo/Multinavegador	3	75%	2,3	75%	2,3	0%	0,0
				69%		74%		58%

Área	Característica	Coeficiente de Peso	Vaadin		ZK		JavaFX	
			Puntuación		Puntuación		Puntuación	
Desarrollo	Herramientas de desarrollo	3	75%	2,3	75%	2,3	75%	2,3
	Herramientas de test	2	50%	1,0	50%	1,0	50%	1,0
	Herramientas de despliegue	1	75%	0,8	75%	0,8	25%	0,3
	Facilitadores (Plantillas, proyectos demo)	2	50%	1,0	75%	1,5	75%	1,5
	Analisis de la calidad (automática)	2	50%	1,0	50%	1,0	50%	1,0
	Independencia del servidor	3	0%	0,0	0%	0,0	0%	0,0
				46%		50%		46%

Área	Caracterítica	Coeficiente de Peso	Vaadin		ZK		JavaFX	
			Puntuación		Puntuación		Puntuación	
Operación	Monitorización de aplicación	1	25%	0,3	25%	0,3	25%	0,3
	Tareas de mantenimiento (Jobs, backups y explotación de logs)	1	25%	0,3	25%	0,3	25%	0,3
				25%		25%		25%

Área	Caracterítica	Coeficiente de Peso	Vaadin		ZK		JavaFX	
			Puntuación		Puntuación		Puntuación	
Soporte	Soporte proveedor	2	50%	1,0	50%	1,0	75%	1,5
	Licencia producto	3	75%	2,3	50%	1,5	50%	1,5
	Evolución	3	75%	2,3	50%	1,5	75%	2,3
				69%		50%		66%

7.1.2.3 Tabla de resultados

Área	Vaadin	ZK	JavaFX
Ejecución	69%	74%	58%
Componentización	75%	75%	75%
Navegación	75%	100%	100%
Integración otros frontales	50%	50%	25%
Manejo de Eventos	75%	75%	75%
Seguridad	50%	50%	50%
Gestores comunes (errores, sesiones,configuración)	75%	75%	75%
Multidispositivo/Multinavegador	75%	75%	0%
Desarrollo	46%	50%	46%
Herramientas de desarrollo	75%	75%	75%
Herramientas de test	50%	50%	50%
Herramientas de despliegue	75%	75%	25%
Facilitadores (Plantillas, proyectos demo)	50%	75%	75%
Analisis de la calidad (automática)	50%	50%	50%
Independencia del servidor	0%	0%	0%
Operación	25%	25%	25%
Monitorización de aplicación	25%	25%	25%
Tareas de mantenimiento (Jobs, backups y explotación de logs)	25%	25%	25%
Soporte	69%	50%	66%
Soporte proveedor	50%	50%	75%
Licencia producto	75%	50%	50%
Evolución	75%	50%	75%
Media	**52,34%**	**49,65%**	**48,78%**

7.1.3 Tablas detalladas de los *frameworks* visuales analizados

7.1.3.1 Tablas de análisis

7.1.3.1.1 Dojo

		Dojo		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	A	**75%**	Ofrece una gran cantidad de componentes robustos, aunque de dificil customización.
	Integración otros *frameworks*	T	**100%**	Se puede integrar con otros *frameworks* visuales.
	Manejo de Eventos	A	**75%**	Dispone de multitud de api's para manejar eventos de sus componentes.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por los navegadores más modernos. Soportado por IE7 con algunas restricciones.
Desarrollo	Herramientas de desarrollo	A	**75%**	Soportado por Aptana y WebStorm.
	Facilitadores (Plantillas, proyectos demo)	M	**50%**	Dispone de plantillas.
Soporte	Soporte proveedor	M	**50%**	La empresa encargada del desarrollo Dojo Foundation, participa activamente en el desarrollo con versiones estable con mucha frecuencia.
	Licencia producto	T	**100%**	Licencia BSD, esta licencia tiene menos restricciones en comparación con otras como la GPL estando muy cercana al dominio público.
	Evolución	M	**50%**	Desde 2005 ofrece versiones estables cada pocos meses, aunque las mejoras son escasas.

7.1.3.1.2 JQueryUI

		Bootstrap		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	T	**100%**	Dispone de una gran cantidad de componentes robustos.
	Integración otros *frameworks*	T	**100%**	Se puede integrar con otros *frameworks* visuales.
	Manejo de Eventos	A	**75%**	Dispone de multitud de api's para manejar eventos de sus componentes.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por todos los navegadores modernos. Soportado por IE7 con algunas restricciones.
Desarrollo	Herramientas de desarrollo	M	**50%**	Existe un *plugin* para Eclipse y Netbeans.
	Facilitadores (Plantillas, proyectos demo)	A	**75%**	Dispone de temas.
Soporte	Soporte proveedor	A	**75%**	Sus desarrolladores están muy implicados en el desarrollo y la evolución del *framework*.
	Licencia producto	A	**75%**	GNU (Licencia Pública General, ofrece la libertad de usar, estudiar, compartir y modificar el software).
	Evolución	M	**50%**	Desde 2008 han aparecido numerosas versiones, aunque se centran mas en la compatibilidad con navegadores que en añadir nuevas funcionalidades.

7.1.3.1.3 Bootstrap

		Bootstrap		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Componentización	A	**75%**	Gran cantidad de robustos componentes.
	Integración otros *frameworks*	T	**100%**	Se puede integrar con otros *frameworks* visuales.
	Manejo de Eventos	M	**50%**	Dispone de manejadores de eventos en algunos de sus componentes.
	Multidispositivo/Multinavegador	A	**75%**	Soportado por los navegadores mas modernos. Soportado por IE8 con algunas restricciones.
Desarrollo	Herramientas de desarrollo	A	**75%**	Existen multitud de herramientas y *plugin*'s, como: Jetstrap, Divshot, Bootply, Paintstrap, Bootstrap Multiselect, NOD, Flatstrap, etc.
	Facilitadores (Plantillas, proyectos demo)	A	**75%**	Incluye temas.
Soporte	Soporte proveedor	A	**75%**	Desarrollado por Twitter, con una alta implicación en su desarrollo y utilizada en su red social.
	Licencia producto	A	**75%**	Licencia MIT, permite usar, copiar, modificar, publicar, sublicenciar o vender el código, incluyendo el copyright.
	Evolución	A	**75%**	Desde su creación en 2012 ha experimentado una gran evolución, con una gran comunidad de usuarios debido a si facilidad y potendia.

7.1.3.2 Tablas de pesos

Área	Característica	Coeficiente de Peso	Bootstrap		JQueryUI		Dojo	
			Puntuación		Puntuación		Puntuación	
Ejecución	Componentización	3	75%	2,3	100%	3,0	75%	2,3
	Integración otros *frameworks*	2	100%	2,0	100%	2,0	100%	2,0
	Manejo de Eventos	2	50%	1,0	75%	1,5	75%	1,5
	Multidispositivo/Multinavegador	3	75%	2,3	75%	2,3	75%	2,3
				75%		88%		80%

Área	Característica	Coeficiente de Peso	Bootstrap		JQueryUI		Dojo	
			Puntuación		Puntuación		Puntuación	
Desarrollo	Herramientas de desarrollo	3	75%	2,3	50%	1,5	75%	2,3
	Facilitadores (Plantillas, proyectos demo)	2	75%	1,5	75%	1,5	50%	1,0
				75%		60%		65%

Área	Característica	Coeficiente de Peso	Bootstrap		JQueryUI		Dojo	
			Puntuación		Puntuación		Puntuación	
Soporte	Soporte proveedor	2	75%	1,5	75%	1,5	50%	1,0
	Licencia producto	3	75%	2,3	75%	2,3	100%	3,0
	Evolución	3	75%	2,3	50%	1,5	50%	1,5
				75%		66%		69%

7.1.3.3 Tabla de resultados

Área	Bootstrap	JQueryUI	Dojo
Ejecución	**75%**	**88%**	**80%**
Componentización	75%	100%	75%
Integración otros *frameworks*	100%	100%	100%
Manejo de Eventos	50%	75%	75%
Multidispositivo/Multinavegador	75%	75%	75%
Desarrollo	**75%**	**60%**	**65%**
Herramientas de desarrollo	75%	50%	75%
Facilitadores (Plantillas, proyectos demo)	75%	75%	50%
Soporte	**75%**	**66%**	**69%**
Soporte proveedor	75%	75%	50%
Licencia producto	75%	75%	100%
Evolución	75%	50%	50%
Media	**75,00%**	**71,04%**	**71,25%**

7.1.4 Tablas detalladas de los *frameworks* gráficos analizados

7.1.4.1 Tablas de análisis

7.1.4.1.1 d3js

		d3js		
Área	**Características**	**Cobertura**		**Comentarios**
Ejecución	Riqueza de gráficos	M	**50%**	D3.js es una librería de dibujado, y no sólo una biblioteca gráfica. Se puede realizar prácticamente cualquier gráfico.
	Resizing	M	**50%**	Se debe lanzar una funcion javascript y capturar el evento de reescalado de la pantalla.
	Zoom/scrolling	M	**50%**	Proporciona una api de zoom, auqnue es compleja de manejar.
	Exportación	M	**50%**	No proporciona ninguna api, pero se puede crear una imagen o un pdf del gráfico.
	Mecanismo de randerización	A	**75%**	SVG (Gráficos Vectoriales Redimensionables).
	Eventos	T	**100%**	Los eventos se capturan mediente funciones Javascript.
	Actualización dinámica	A	**75%**	Usando la API se puede modificar cualquier elemento que se desee.
	Compatibilidad navegadores	A	**75%**	Todos los navegadores modernos de escritorio y móviles, incluido IE8+.
Desarrollo	Curva de aprendizaje	B	**25%**	Pronunciada (multiples de ficheros de configuración).
Operación	Gestión de librerias	A	**75%**	Se puede descargar el fuente de las librerias y realizar tus propias modificaciones, proporciona gran flexibilidad de edición.
Soporte	Licencia producto	T	**100%**	BSD (impone restricciones mínimas en la redistribución del software)

7.1.4.1.2 Google Charts

		googlecharts		
Área	**Característica**	**Cobertura**		**Comments**
Ejecución	Riqueza de gráficos	A	**75%**	Proporciona un amplio conjunto de gráficos y soporte 3D.
	Resizing	M	**50%**	No se proporciona una api específica, se debe resolver añadiendo código javaScript adicional.
	Zoom/scrolling	M	**50%**	Proporcionado por api aunque no para todos los gráficos.
	Exportación	M	**50%**	Proporciona exportacion a html, csv y formato excel, pero no en formato imagen o pdf.
	Mecanismo de randerización	T	**100%**	Proporciona: SVG (Gráficos Vectoriales Redimensionables) y VML (Vector Marktup Language, lenguaje XML destinado a la creación de gráficos).
	Eventos	T	**100%**	Mediante funciones javaScript.
	Actualización dinámica	M	**50%**	No se proporciona una api específica, se debe redibujar mediante javaScript.
	Compatibilidad navegadores	T	**100%**	Soportado por todos los navegadores modernos, soportado a partir de IE6+.
Desarrollo	Curva de aprendizaje	A	**75%**	Suave, existe multitus de documentación de calidad.
Operación	Gestión de librerias	M	**50%**	Los ficheros javaScript necesarios se cargan direcametne desde los servidores de Google. Tu aplicación necesita disponer de conexión a internet
Soporte	Licencia producto	T	**100%**	Se proporciona de forma gratuita.

7.1.4.1.3 Highcharts

		Highcharts		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Riqueza de gráficos	T	**100%**	Proporciona un amplio conjunto de gráficos de gran calidad y soporte 3D.
	Resizing	A	**75%**	Proporciona resize mediante atributos.
	Zoom/scrolling	A	**75%**	Proporcionado para algunos gráficos.
	Exportación	A	**75%**	Proporcina exportación en los formatos jpeg, png y pdf.
	Mecanismo de randerización	T	**100%**	Proporciona: SVG (Gráficos Vectoriales Redimensionables) y VML (Vector Marktup Language, lenguaje XML destinado a la creación de gráficos).
	Eventos	T	**100%**	Permite la captura de eventos con funciones javaScript.
	Actualización dinámica	A	**75%**	Mediante la API puedesanañadir, modificar y remover puntos y modificar los ejes en cualquier momento desde la creación del gráfico.
	Compatibilidad navegadores	T	**100%**	Soprtado por todos los navegadores modernos.
Desarrollo	Curva de aprendizaje	M	**50%**	Media, se debe aprender mucha configuración con su propia sintaxis.
Operación	Gestión de librerias	A	**75%**	Permite descargar el código y ralizar tus propias modificaciones. Lo que permite gran flexibilidad.
Soporte	Licencia producto	B	**25%**	Existe una versión gratuita y una de pago. La versión comercial es de pago.

7.1.4.1.4 JqPlot

		Jqplot		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Riqueza de gráficos	A	**75%**	Proporciona un amplio conjunto de gráficos y soporte 3D.
	Resizing	M	**50%**	Se debe resolver añadiendo código javaScript.
	Zoom/scrolling	M	**50%**	Se debe resolver añadiendo código javaScript.
	Exportación	M	**50%**	No dispone de una api dedicada, pero se puede realizar serializando mediante la api jqplotToImageStr.
	Mecanismo de randerización	T	**100%**	Proporciona: SVG (Gráficos Vectoriales Redimensionables) y VML (Vector Marktup Language, lenguaje XML destinado a la creación de gráficos).
	Eventos	T	**100%**	Dispone de un escuchador de evenos mediante funciones javaScript.
	Actualización dinámica	M	**50%**	Se debe desarrllar en javaScript. Se debe re-llamar a la funcion de dibujado.
	Compatibilidad navegadores	T	**100%**	Compatible con todos los navegadores modernos. Disponible a partir de IE7+.
Desarrollo	Curva de aprendizaje	M	**50%**	Media, requiere de mucha configuración.
Operación	Gestión de librerias	A	**75%**	Permite descargar el código y ralizar tus propias modificaciones. Lo que permite gran flexibilidad.
Soporte	Licencia producto	T	**100%**	Licencia MIT.

7.1.4.2 Tablas de pesos

			Jqplot		Highcharts		googlecharts		d3js	
Área	Característica	Coeficiente	Puntuación		Puntuación		Puntuación		Puntuación	
Ejecución	Riqueza de gráficos	3	75%	2,3	100%	3,0	75%	2,3	50%	1,5
	Resizing	3	50%	1,5	75%	2,3	50%	1,5	50%	1,5
	Zoom/scrolling	2	50%	1,0	75%	1,5	50%	1,0	50%	1,0
	Exportación	2	50%	1,0	75%	1,5	50%	1,0	50%	1,0
	Mecanismo de randerización	3	100%	3,0	100%	3,0	100%	3,0	75%	2,3
	Eventos	2	100%	2,0	100%	2,0	100%	2,0	100%	2,0
	Actualización dinámica	2	50%	1,0	75%	1,5	50%	1,0	75%	1,5
	Personalización	3	100%	3,0	100%	3,0	100%	3,0	75%	2,3
	Compatibilidad navegadores	3	50%	1,5	50%	1,5	75%	2,3	25%	0,8
			71%		84%		74%		60%	

			Jqplot		Highcharts		googlecharts		d3js	
Área	Característica	Coeficiente	Puntuación		Puntuación		Puntuación		Puntuación	
Desarrollo	Curva de aprendizaje	3	50%	1,5	50%	1,5	75%	2,3	25%	0,8
			50%		50%		75%		25%	

			Jqplot		Highcharts		googlecharts		d3js	
Área	Característica	Coeficiente	Puntuación		Puntuación		Puntuación		Puntuación	
Operación	Gestión de librerias	2	75%	1,5	75%	1,5	50%	1,0	75%	1,5
				75%		75%		50%		75%

			Jqplot		Highcharts		googlecharts		d3js	
Área	Característica	Coeficiente	Puntuación		Puntuación		Puntuación		Puntuación	
Soporte	Licencia producto	3	100%	3,0	25%	0,8	100%	3,0	100%	3,0
				100%		25%		100%		100%

7.1.4.3 Tabla de resultados

Area	Jqplot	Highcharts	googlecharts	d3js
Ejecución	**71%**	**84%**	**74%**	**60%**
Riqueza de gráficos	75%	100%	75%	50%
Resizing	50%	75%	50%	50%
Zoom/scrolling	50%	75%	50%	50%
Exportación	50%	75%	50%	50%
Mecanismo de randerización	100%	100%	100%	75%
Eventos	100%	100%	100%	100%
Actualización dinámica	50%	75%	50%	75%
Personalización	100%	100%	100%	75%
Compatibilidad navegadores	50%	50%	75%	25%
Desarrollo	**50%**	**50%**	**75%**	**25%**
Curva de aprendizaje	50%	50%	75%	25%
Operación	**75%**	**75%**	**50%**	**75%**
Gestión de librerias	75%	75%	50%	75%
Soporte	**100%**	**25%**	**100%**	**100%**
Licencia producto	100%	25%	100%	100%
Media	**69,71%**	**73,91%**	**71,59%**	**60,65%**

7.1.5 Tablas detalladas de las herramientas de desarrollo analizadas

7.1.5.1 Tablas de análisis

7.1.5.1.1 Webstorm

		Webstorm		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Previsualización	T	**100%**	Permite edición en vivo, muestra los cambios automaticamente según se edita. Trabaja sobre chrome.
	Autocompletado y corrector sintáctico	T	**100%**	Soporta ECMAScript, especificación de lenguaje de programación, para el autocompletado y la correción sintáctica.
	Productividad	T	**100%**	Análisis de código, búsquedas avanzadas, refactorización, integración transparente con SVN,
	Test (debug, performance)	T	**100%**	Herramientas de debug y rendimiento mediante spy.js
	Cobertura de *frameworks*	A	**75%**	Cubre la mayría de *Frameworks* MVC javaScript.
	Licencias	N	**0%**	89$ por desarrollador para su uso comercial.

7.1.5.1.2 SublimeText

		SublimeText		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Previsualización	T	**100%**	Permite edición en vivo, muestra los cambios automaticamente según se edita.
	Autocompletado y corrector sintáctico	T	**100%**	Soporta ECMAScript, especificación de lenguaje de programación, para el autocompletado y la correción sintáctica.
	Productividad	A	**75%**	Análisis de código, búsquedas avanzadas, refactorización, integración transparente con SVN,
	Test (debug, performance)	M	**50%**	Incorpora herramientas de debug.
	Cobertura de *frameworks*	A	**75%**	Cubre la mayría de *Frameworks* MVC javaScript.
	Licencias	M	**50%**	50-70$ por desarrrollador en su uso comercial.

7.1.5.1.3 Aptana

		Aptana		
Área	**Característica**	**Cobertura**		**Comentarios**
Ejecución	Previsualización	A	**75%**	Es necesario guardar el fichero, para ver los cambios.
	Autocompletado y corrector sintáctico	T	**100%**	Soporta ECMAScript, especificación de lenguaje de programación, para el autocompletado y la correción sintáctica.
	Productividad	A	**75%**	Análisis de código, búsquedas avanzadas, refactorización, integración transparente con GIT.
	Test (debug, performance)	A	**75%**	Incorpora herramientas para el debug.
	Cobertura de *frameworks*	A	**75%**	Cubre la mayría de *Frameworks* MVC javaScript.
	Licencias	T	**100%**	GNU (Licencia Pública General, ofrece la libertad de usar, estudiar, compartir y modificar el software).

7.1.5.2 Tabla de pesos

			Webstorm		SublimeText		Aptana	
Área	**Caracterítica**	**Coeficiente de Peso**	**Puntuación**		**Puntuación**		**Puntuación**	
Ejecución	Previsualización	**2**	100%	2,0	100%	2,0	75%	1,5
	Autocompletado y corrector sintáctico	**3**	100%	3,0	100%	3,0	100%	3,0
	Productividad	**3**	100%	3,0	75%	2,3	75%	2,3
	Test (debug, performance)	**3**	100%	3,0	50%	1,5	75%	2,3
	Cobertura de *frameworks*	**2**	75%	1,5	75%	1,5	75%	1,5
	Licencias	**3**	0%	0,0	50%	1,5	100%	3,0
				78%		**73%**		**84%**

7.1.5.3 Tabla de resultados

Área	Webstorm	SublimeText	Aptana
Previsualización	100%	100%	75%
Autocompletado y corrector sintáctico	100%	100%	100%
Productividad	100%	75%	75%
Test (debug, performance)	100%	50%	75%
Cobertura de *frameworks*	75%	75%	75%
Licencias	0%	50%	100%
Media	78%	73%	84%

7.2 MATERIAL DEL PROYECTO

El material entregado para este proyecto, incluye:

- Memoria en formato PDF.
- Código fuente de la prueba de concepto realizada.
- Excel donde se han realizado las comparativas del estudio.