**ARTÍCULO ORIGINAL**

# Propuesta de arquitectura cliente de la aplicación de interfaz de usuario del sistema xavia ris 2.0

## Proposal for client architecture for user interface application of systems ris xavia 2.0

**Ing. Eddy Yanier Duque García**

Ingeniero en Ciencias Informáticas, Centro de Informática Médica, Universidad de las Ciencias Informáticas. Carretera San Antonio de los Baños Km 2 ½, Boyeros, La Habana, Cuba. E-mail: eyduque@uci.cu

**RESUMEN**

El Sistema de Información Radiológica (RIS, por sus siglas en inglés) es el encargado de la gestión de la información del Departamento de Radiología en una Institución Sanitaria, así como la Gestión de los Informes médicos y la creación de listas de trabajo para equipos DICOM compatibles.[1] Dicho sistema está concebido como un sistema web por las ventajas de accesibilidad que brinda la Web y el indetenible crecimiento y desarrollo de las tecnologías que la sustentan. La presente investigación tiene como objetivo proponer las tecnologías, herramientas, estructuras y finalmente la arquitectura cliente para la aplicación de Interfaz de Usuario del Sistema XAVIA RIS, acorde a las tendencias actuales del desarrollo Web.

**Palabras clave:** interfaz de usuario, radiología, tecnologías, web, arquitectura, marco de trabajo.

**ABSTRACT**

Radiological Information System is in charge of information management in the radiology department in a health institution, and the management of medical records and the creation of work lists for equipment DICOM compatible. This system is designed as a web system due to the benefits of accessibility offered by the Web and the unstoppable growth and development of technologies that support it. This research aims to propose technologies, tools, structures and ultimately the

client architecture for User Interface Application of the XAVIA RIS System, according to current trends in Web development.

**Key words:** user interface, radiology, technologies, web, architecture, framework.

## INTRODUCCIÓN

La medicina es uno de los sectores de la sociedad que más se ha beneficiado con el desarrollo y potenciamiento de la Web, aumentando considerablemente en las instituciones sanitarias la capacidad de procesamiento de información para la generación de estadísticas y reportes, así como la comunicación entre los especialistas y profesionales del sector.

El Departamento de Imagenología, es el encargado de la realización de los estudios médicosimagenológicos en los cuales se utilizan equipos de gama alta que generan imágenes del cuerpo humano de relevancia para la creación del Diagnósticomédico.

Las Imágenes Médicas obtenidas son distribuidas y almacenadas a través de la red Imagenológica por los Sistemas PACS[2] usando el estándar DICOM,[1] mientras que la informacióndemográfica de los pacientes, personal médico y listas de trabajo es gestionada por los Sistemas RIS.[2]

Los sistemas web son aplicaciones informáticas que se despliegan en servidores Web a las cuales se accede mediante el Protocolo de Transferencia de Hipertexto (HTTP, por sus siglas en inglés) haciendo uso de un Navegador Web o cliente Web con soporte para dicho protocolo.

Las aplicaciones de interfaz de usuario (UI) de los Sistemas Web se desarrollan con tecnologías soportadas por el cliente Web; algunas de estas tecnologías se encargan de la Presentación como por ejemplo: las Hojas de estilos en Cascada y Lenguaje de Marcado de Hipertexto (CSS y HTML, por sus siglas en inglés respectivamente); mientras que otras como JavaScript y AJAX[3] se encargan de manejar la interacción de la aplicación con el usuario, la lógica de negocio cliente y la comunicación asincrónica con él Back-end.[4]

El auge y popularidad de la Web ha contribuido al desarrollo acelerado de las tecnologías clientes que la sustentan; evidenciadocon el surgimiento de Framework[5] JavaScript que implementan patrones como MVC, MVVM[6]; librerías para el soporte modular y la abstracciónde cliente Web o el dispositivo.

El Sistema XAVIA RIS en su versión 1.0 fue desarrollado sobre el Framework .NET usando en enfoque de desarrollo ASP .NET Web Forms; entorno que provee una magnífica productividad, administración de memoria, manejo de excepciones y todas las fortalezas del Lenguaje de Programación Orientado a objetos C#. Sin embargo, este entorno de desarrollo no explota al máximo las potencialidades de los lenguajes y herramientas nativos del cliente web, debido a que las páginas son escritas en lenguajes de servidor que se especializan más en la lógica de negocio de la página que en la forma en que esta se muestra.

**OBJETIVOS**

Proponer las tecnologías, herramientas, estructuras y finalmente la arquitectura cliente para la aplicación de Interfaz de Usuario del Sistema XAVIA RIS, acorde a las tendencias actuales del desarrollo Web.

## MATERIALES Y MÉTODOS

Durante el desarrollo de la investigación se utilizaron los siguientes métodos científicos:

**Histórico Lógico:** al realizar un análisis crítico valorativo de la información contenida en las fuentes bibliográficas consultadas, con el objetivo de conocer cuáles son las tendencias y cómo se ha comportado el desarrollo de software en torno a la identificación de nódulos pulmonares solitarios en imágenes de tomografía de tórax.

**Analítico Sintético:** para descomponer el problema de la investigación en elementos concretos de la solución.

**Inductivo Deductivo:** teniendo en cuenta el funcionamiento del proceso de diagnóstico por imágenes médicas se evalúa la problemática existente, para determinar aspectos particulares o característicos y desarrollar una propuesta de solución.

**Experimento:** para realizar la validación de los resultados obtenidos se realizaron pruebas de caja blanca a la aplicación base obtenida.

### Herramienta y tecnologías informáticas empleadas

**C# 5.0:** es un lenguaje de programación orientado a objetos y permite a los desarrolladores crear una amplia gama de aplicaciones que se ejecuten en la plataforma .NET Framework, su sintaxis es sencilla, fácil de utilizar y permite los conceptos de encapsulación, herencia y polimorfismo.[3]

**Visual Studio 2013:** es un Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés), que soporta varios lenguajes de programación incluido Visual C#. Incluye herramientas que simplifican todo el proceso de desarrollo de aplicaciones, de principio a fin. Permite realizar una administración del ciclo de vida de las aplicaciones e incorpora otras pruebas que ayudan a garantizar la calidad del código en todo momento.[4]

**.NET Framework 4.5:** es un marco de trabajo desarrollado por Microsoft que incluye una amplia Librería de Clases y provee interoperabilidad de lenguajes a través de varios lenguajes de programación. Los programas escritos en .NET se ejecutan en un ambiente de software manejado, conocido como Entorno de Ejecución Manejado (CLR, por sus siglas en inglés) y que constituye una Máquina Virtual que provee servicios de seguridad, administración de memoria y manejo de excepciones.[5]

**ASP .NET MVC 5:** es un marco de trabajo para la construcción escalable y basada en estándar de aplicaciones web, usando patrones de diseño bien establecidos y todas las características de ASP .NET y el marco de trabajo .NET.[6]

**ASP .NET Web API:** es un marco de trabajo para el desarrollo sencillo de servicios HTTP que incluye un amplio rango de cliente, incluyendo navegadores y dispositivos móviles. Brinda grandes facilidades para la creación de aplicaciones RESTFul sobre .NET.[7]

## XAVIA RIS 1.0

Está desarrollado sobre el Framework ASP .NET 3.5 siguiendo el enfoque ASP .NET Web Forms. Dicho marco de trabajo está diseñado para lograr una alta productividad haciendo uso de las ventajas de los lenguajes .NET orientados a objetos y la experiencia en el desarrollo con Windows Forms para sistemas de escritorio. Los sistemas desarrollados usando este enfoque generalmente presentan:

- Fuerte acoplamiento entre la Capa de Presentación y la Capa de Negocio.
- Bajo rendimiento del sistema con el uso del Renderizado Parcial[7] y el Mantenimiento de Estado de la Vista.[8]
- Soporte limitado para plataformas Unix.
- Incompatibilidad de controles antiguos en navegadores y dispositivos modernos.
- Complejo ciclo de vida de aplicación que dificulta la integración con tecnologías clientes de terceros.
- Esquema poco flexible para exponer y proveer servicios de datos de Transferencia de Estado Representacional (REST o RESTFul, por sus siglas en inglés).

### Aplicación en un sola Página

En las aplicaciones tradicionales, el cliente inicia la comunicación con el servidor solicitando una página. El servidor por su parte procesa la solicitud y envía el HTML de la página al cliente. En las siguientes iteraciones con la página, ej.: el usuario navega a un enlace o somete un formulario con datos, una nueva solicitud es enviada al servidor, y el flujo comienza nuevamente: el servidor procesa la solicitud y envía la nueva página al cliente en respuesta a la nueva acción solicitada.

Se puede definir una Aplicación en un sola Página (SPA, por sus siglas en inglés) como una aplicación web que se ejecuta en una única página, logrando así una experiencia de usuario más cercana a una aplicación de escritorio.[8] En una SPA el usuario no navega por un engorroso sistema de enlaces tradicionales si no que, en su lugar, mediante el uso cada vez más extendido de JavaScript, AJAX, HTML5 o una combinación de las anteriores, se actualiza lo que el usuario ve siempre desde la misma página (sin cambiar de URL ni refrescar el contenido entero).

En una SPA, la página es cargada en el cliente después de la solicitud inicial, pero las siguientes solicitudes tienen lugar a través de solicitudes AJAX, como se muestra en la figura 1. Esto significa que el cliente solo tiene que actualizar la porción de la página que tiene que ser cambiada; sin necesidad de recargar toda la página. El enfoque SPA reduce el tiempo tomado por la aplicación en responder a las acciones de los usuarios, lo que resulta en una experiencia más fluida.


**Fig.1.** Flujo básico de la interacción de una SPA y el servidor

## El patrón MVVM

Este patrón intenta brindar una limpia separación de responsabilidades entre los controles de la interfaz de usuario y su lógica. Hay tres componentes principales cada uno de los cuales sirve a un rol distinto y separado. El modelo, la vista y el modelo de la vista, en la figura 2.

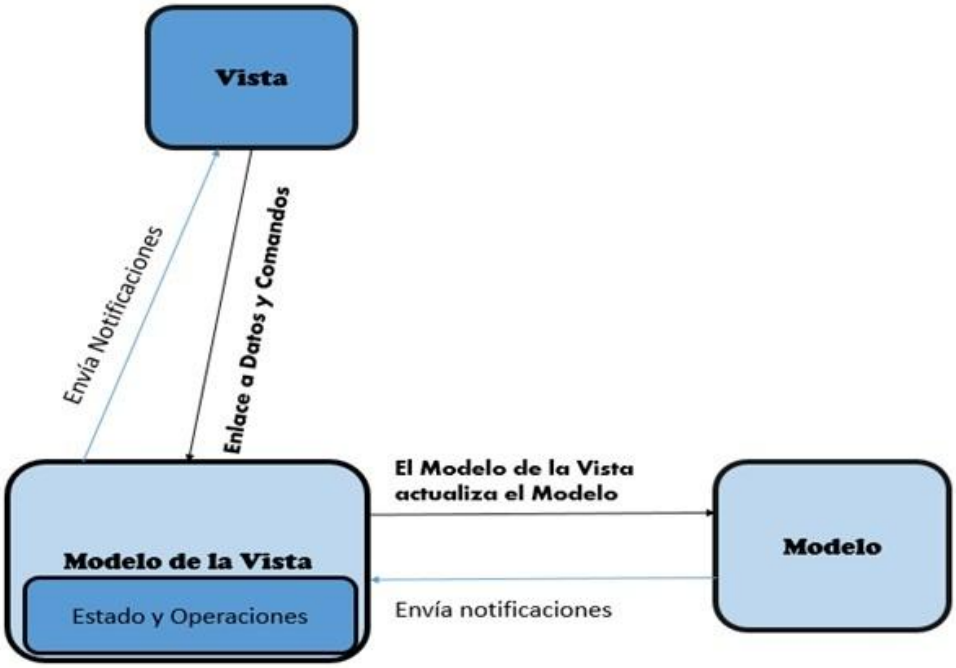


**Fig. 2.** Patrón Arquitectónico MVVM

Los componentes están desacoplados unos de otros, permitiendo:

- Los componentes pueden ser intercambiados.
- La implementación interna puede ser cambiada sin que afecte a los demás.
- Componentes que trabajan de forma independiente.
- Pruebas unitarias aisladas.

**Vista**

Es la encargada de definir la estructura, diseño y apariencia de lo que el usuario ve en la pantalla y no implementa lógica de negocio, solo de presentación.

**Modelo**

Es una implementación del modelo de dominio de la aplicación que incluye un modelo de datos junto con la lógica de negocio y validación.

**Modelo de la Vista**

Actúa como intermediario entre la vista y el modelo, y es responsable de manejar la lógica de la vista. Típicamente, interactúa con el modelo invocando funciones en el modelo de clases y provee los datos a la vista. Recupera los datos del modelo y los pone a disposición de la vista, y puede reformatear los datos de alguna manera que haga su manipulación más simple a la vista. También provee implementación de comandos que un usuario de la aplicación inicia con su interacción en la vista.

**Beneficios de MVVM**

MVVM permite un gran flujo de trabajo de desarrollador-diseñador, proporcionando los siguientes beneficios:

- Durante el proceso de desarrollo, los desarrolladores y diseñadores pueden trabajar de manera más independiente y al mismo tiempo en sus componentes. Los diseñadores pueden concentrarse en la vista, mientras que los desarrolladores puedan trabajar en los componentes de modelo de vista y modelo.
- Los desarrolladores pueden crear pruebas unitarias para el modelo de vista y el modelo sin utilizar la vista. Las pruebas unitarias para el modelo de vista pueden ejercer exactamente la misma funcionalidad como si fueran usadas por la vista.
- Es fácil de rediseñar la interfaz de usuario de la aplicación sin tocar el código porque la vista se implementa generalmente con lenguaje de marcado. Una nueva versión de la vista debe trabajar con el modelo de vista existente.
- Si hay una implementación existente del modelo que encapsula la lógica de negocio existente, puede ser difícil o arriesgado cambiar. En este escenario, el modelo de vista actúa como un adaptador para las clases del modelo y le permite evitar hacer cambios importantes en el código del modelo.

**Módulos JavaScript**

Los módulos son una parte integral de la arquitectura de cualquier aplicación robusta y por lo general ayudan a mantener las unidades de código para un proyecto de manera limpia separada y organizada.

Opciones para implementar módulos en JavaScript

- El patrón de Módulo.
- Notación de Objeto Literal.
- Definición Asincrónica de Módulos (AMD, por sus siglas en inglés).
- Módulos CommonJS.
- Módulos ECMAScript Harmony.

En JavaScript, el patrón del módulo se utiliza para emular aún más el concepto de clases de tal manera que puede incluir ambos métodos públicos y privados y variables dentro de un único objeto, protegiendo de este modo partes particulares del ámbito global. Lo que esto resulta en una reducción en la probabilidad de quelosnombres de funciones entren en conflicto con otras funciones definidas en script adicionales en la página.

El patrón del módulo encapsula "privacidad", el estado y la organización utilizando cierres. Proporciona una forma de envolver una mezcla de métodos y variables públicas y privadas, evitando que las partes se filtren en el ámbito global y colisionen accidentalmente con una interfaz de otro desarrollador. Con este patrón, solamente se devuelve una API pública, manteniendo todo lo demás dentro del ámbito privada.

Cabe señalar que no existe un concepto explícito de verdadero sentido de "privacidad" dentro de JavaScript porque a diferencia de algunos lenguajes tradicionales, no tiene modificadores de acceso. Las variables no pueden ser declaradas técnicamente como públicoso privadas y por eso usan ámbito de función para simular este concepto. Dentro del modelo de módulo, las variables o métodos declarados sólo están disponibles dentro del propio módulo gracias a su cierre. Las variables o métodos definidos dentro del objeto retornado, sin embargo, están disponibles en el ámbito global.

## RESULTADOS Y DISCUSIÓN

Se propone una arquitectura basada en la implementación SPA y el soporte MVVM de DurandalJS, se muestra en la figura 3. En la base de la arquitectura se anida RequireJS como sustento para la carga asincrónica de módulos y dependencias; KnockoutJS para realizar el Databinding[9] y la composición de vistas de Interfaz de Usuario y JQuery para el manejo del DOM.[10] El Bootstraper es el encargado de instanciar e inicializar las librerías base, que incluye además i18next responsable de la localización de las vistas y toastrJS para el manejo de notificaciones. El núcleo de DurandalJS brinda un ciclo de vida de aplicación simple pero muy flexible que permite al desarrollador insertar código personalizado en cada una de las fases del ciclo, en la tabla 1. Uno de los objetos del núcleo de DurandalJS es el Binder, que se encarga de unir una instancia de objeto y un árbol de elementos del DOM[9] aplicando Databinding e invocando llamadas de retorno del ciclo de vida de Databinding. Esto permite insertar lógica personalizada una vez que el Databinding está completo permitiendo a i18next localizar cada vista una vez que esté totalmente compuesta. El Router es otro de los objetos del núcleo y se encarga de conectar el historial de URL de módulo y el soporte del seguimiento del historial al motor de activación y composición del framework, brindando además la capacidad de extender la lógica base del enrutamiento; ya sea restringiendo el acceso a ciertos módulos, la redirección o evitando la navegación.

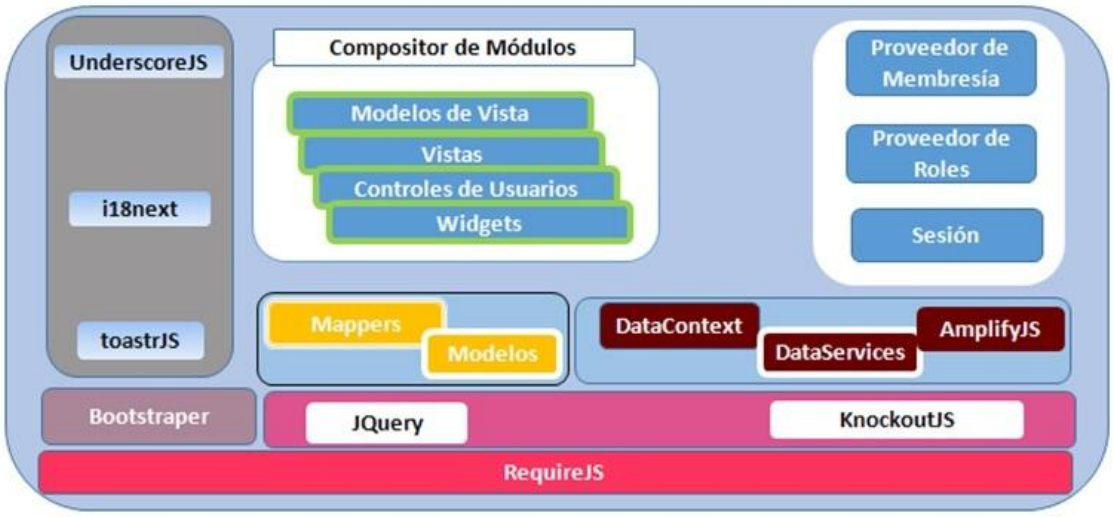


**Fig. 3.** Arquitectura propuesta

**Tabla 1**. Ciclo de vida de Composición y Activación de Vistas en DurandalJS

| Llamada de Retorno | Ciclo de Vida | Propósito |
|---|---|---|
| getView()<br>viewUrl | Composición | Permite al nuevo objeto retornar una Vista personalizada. |
| canDeactivate() | Activador | Permite al objeto previo cancelar la desactivación. |
| canActivate() | Activador | Permite al nuevo objeto cancelar la activación. |
| deactivate() | Activador | Permite al objeto previo ejecutar lógica personalizada de desactivación. |
| activate() | Composición y Activador | Permite al nuevo objeto ejecutar lógica personalizada de activación. |
| binding() | Composición | Notifica al nuevo objeto inmediatamente antes que ocurra en Enlace a Datos. |
| bindingComplete() | Composición | Notifica al nuevo objeto inmediatamente después que ocurra en Enlace a Datos. |
| attached() | Composición | Notifica al nuevo objeto cuando su Vista ha sido adjuntada a su Nodo del DOM padre. |
| compositionComplete() | Composición | Notifica al nuevo objeto cuando la composición en la que participa está completa. |
| detached() | Composición | Notifica al objeto compuesto cuando su vista es removida del DOM. |

## DurandalJS

Es un framework JavaScript ligero diseñado para hacer simple y elegante la construcción de una SPA. Está desarrollado como un conjunto de módulos AMD sobre las librerías JQuery, KnockoutJS y RequireJS.

## Arquitectura

Brinda fuerte soporte para MVC, MVP y MVVM. Con RequireJS como base y una fina capa de convenciones, provee una increíble productividad y ayuda a mantener prácticas de codificaciónsólidas. Aparejado a esto también provee una rica composición de Interfaz de Usuario (UI, por sus siglas en inglés), diálogos modales, eventing/messaging,[11] widgets,[12] transiciones, enrutamiento etc.

Características fundamentales:

1. Arquitectura MV* limpia.
2. Modularidad JavaScript y HTML.
3. Ciclo de vida de Aplicación Simple.
4. Eventing, Diálogos Modales, Cuadros de Mensaje, etc.
5. Administración de Navegación y Estado de la Pantalla.
6. Programación asincrónica consistente con Promesas.
7. Optimización de la aplicación.
8. Uso de cualquier tecnología backend.
9. Construido sobre jQuery, Knockout y RequireJS.
10. Integración con otras librerías como Bootstrap.

**Compatibilidad**
IE 6+
Firefox 2+
Safari 3.2+
Chrome 3+
Opera 10+

**RequireJS**
Es una librería JavaScript para la definición, resolución de dependencias y carga asincrónica de módulos AMD, ver la figura 4.

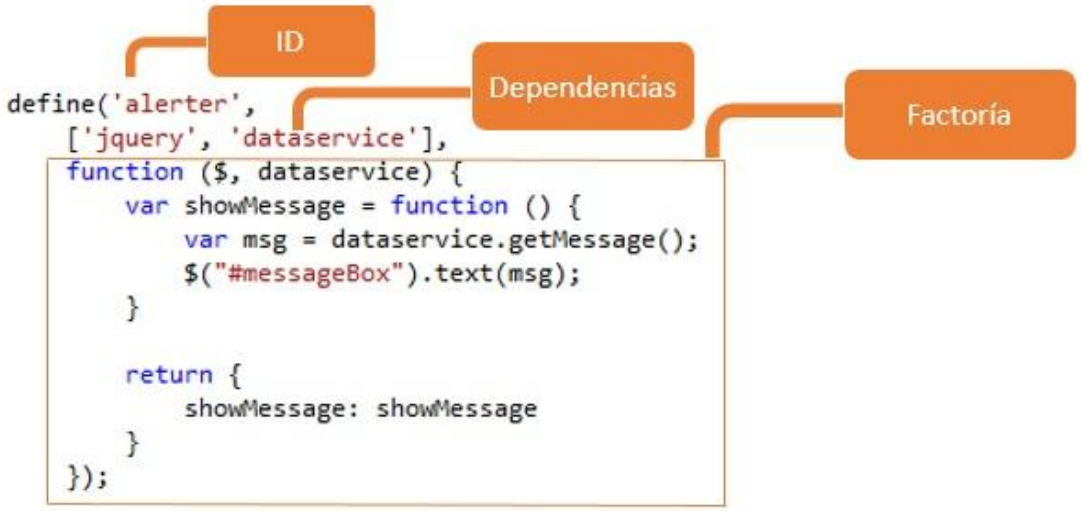


**Fig. 4.** Módulo AMD implementado con RequireJS

**KnockoutJS**
Es una librería JavaScript que permite crear ricas y sensibles interfaces de usuario con un limpio modelo de datos subyacente.

Principales características

1. Librería puramente JavaScript - funciona con cualquier tecnología cliente o servidor.
2. Puede ser adicionada a una Aplicación Web existente sin mucho impacto.
3. Elegante seguimiento de dependencias - automáticamente actualiza las partes apropiadas de la Interfaz de Usuario siempre que cambie el modelo de datos.
4. Bindings declarativos - una vía simple y obvia de conectar partes de la

Interfaz de Usuario al modelo de datos. Se pueden construir UI dinámicas complejas fácilmente haciendo uso de bindings arbitrariamente anidados.
5. Trivialmente extensible - implementar nuevos comportamientos, widgets y binding declarativos de fácil reutilización, ver la figura 5.


**Fig. 5.** Implementación de Binding Personalizado con KnockoutJS

## Seguridad

Cuando el núcleo de DurandalJS está listo se intenta obtener la información del usuario localmente en el cliente, en caso de no estar disponible se redirecciona al módulo de Login; la Web API[13] proveerá un token de seguridad[14] para el usuario que se almacenará localmente para ser usado en cada una de la solicitudes posteriores, ver la figura 6. Una vez autenticado se instancia la Sesión del usuario de manera global y única para toda la aplicación lo que garantiza la unicidad de la información, así como el Proveedor de Roles y el Proveedor de Membresía con la información obtenida de la Web API.

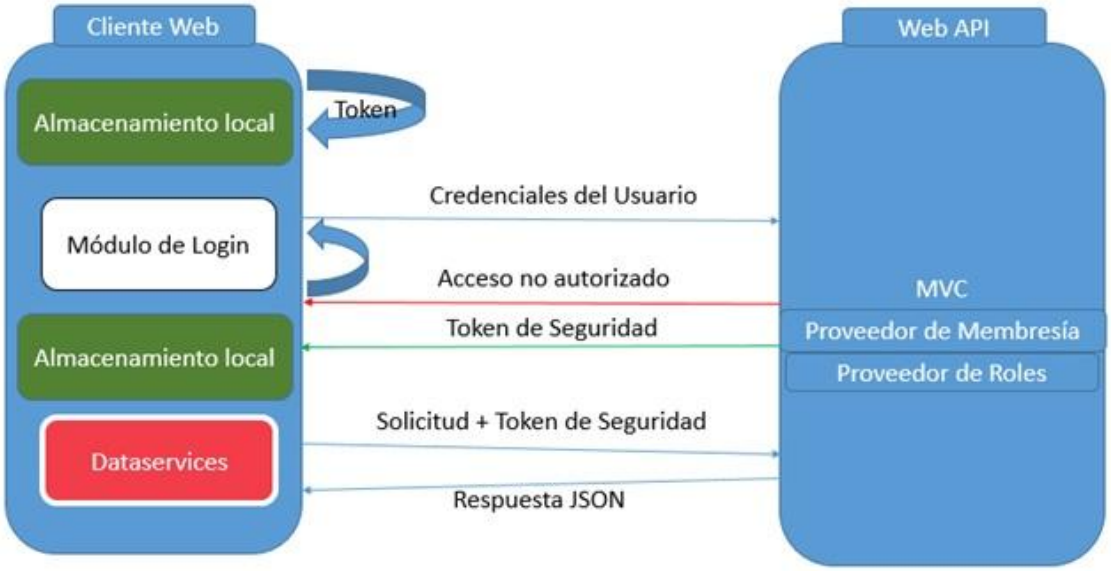


**Fig. 6.** Flujo del Proceso de Autenticación de Usuario

## Gestión de la Información

Se propone la implementación del Patrón Repositorio queestá íntimamente relacionado con el acceso a datos y permite tener una abstracción de la implementación de acceso a datos en las aplicaciones, de modo que la lógica de negocio no conozca ni esté acoplada a la fuente de datos, ve la figura 7. En pocas palabras esto quiere decir que el repositorio actúa como un intermediario entre la lógica de negocio y la lógica de acceso a datos para que se centralice en un solo

punto, y de esta forma se logre evitar redundancia de código. Y como se mencionó anteriormente, al ser una abstracción del acceso a datos permite desacoplar y probar de una forma más sencilla el código mediante pruebas unitarias. Adicionalmente, al estar centralizado en un solo punto de acceso, se puede reutilizar la lógica de acceso a datos desde cualquier cliente, es decir desde otras aplicaciones presentes en el entorno corporativo.

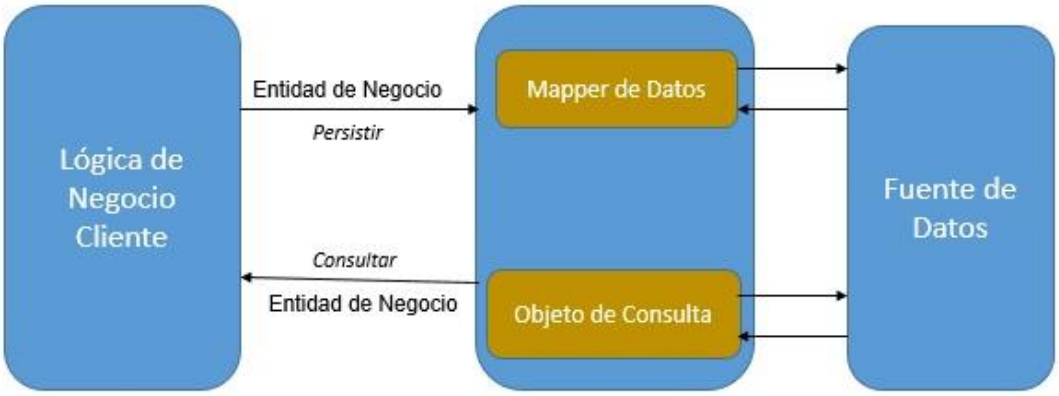


**Fig. 7.** Estructura del Patrón Repositorio

Uno de los escenarios donde más se suele usar el patrón repositorio, es cuando se tienen múltiples fuentes de datos. Por ejemplo, se obtiene gran parte de la información de una base de datos relacional, otros datos desde un servicio web o RESTFul y algunos desde una base de datos No SQL (Lenguaje de Consulta Estructurado).

En el entorno SPA, los datos se solicitan al servidor y pueden ser almacenados localmente de manera inteligente para su uso posterior. Esta arquitectura propone el uso de Dataservices y la implementación AJAX de AmplifyJS para el acceso de los datos remotos y el caching[15] inteligente de la información. Los Dataservices deberán implementar los Métodos HTTP: POST, PUT, GET y DELETE. La Web API envía las respuestas en formato JSON, los Dataservices invocan a los Mappers[16] para instanciar los objetos de negocio y se crea el Contexto de Datos disponible globalmente para toda la aplicación.

A continuación, se brindan algunas de las principales funcionalidades de la arquitectura base obtenida en la figura 8, figura 9, figura 10 y ver Anexo:


**Fig. 8.** Vista de Autenticación de Usuario

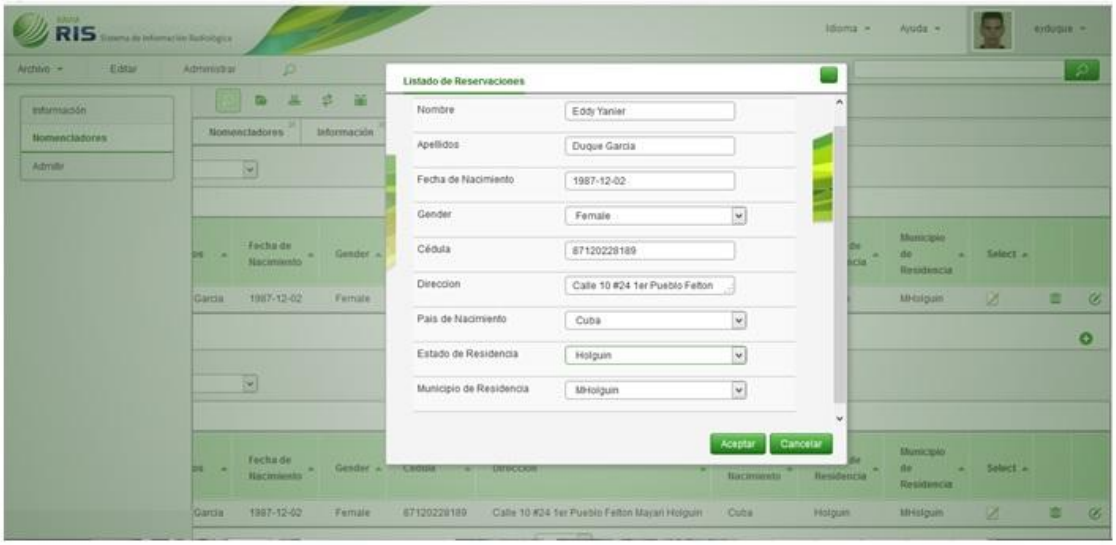


**Fig. 9.** Vista de Módulo de Admisión, que muestra los Widgets: GridView y EditView

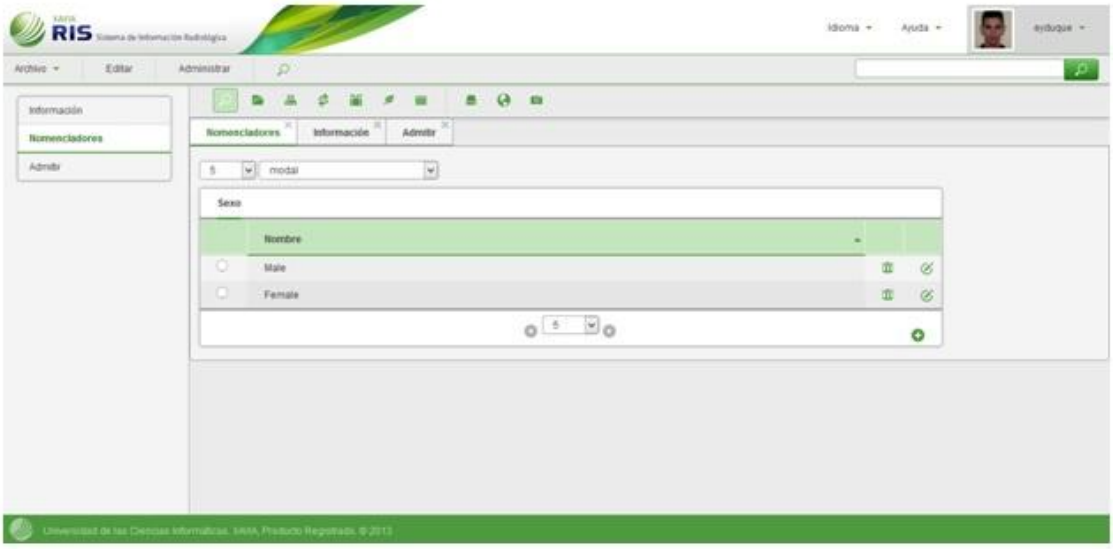


**Fig. 10.** Vista del Módulo de Nomencladores

- Infraestructura de Localización e Internacionalización.
  - Mecanismo de gestión de Notificaciones y Mensajes entre módulos.
  - Carga dinámica de módulos basado en convenciones y composición de vistas.
  - Mecanismo de gestión de datos remotos basado en el Patrón Repositorio y Binding de Modelo.
  - Controles de Usuario, Comportamientos y DataBindings personalizados; y Widgets comunes: GridView, DropDownList, ListView, DetailView y EditView etc.
  - Patrón de Navegación y Enrutamiento.
  - Estructura de directorios y archivos.
  - Infraestructura de Autenticación y Gestión de Roles.
  - Prototipo de Interfaz de Usuario.

## CONCLUSIONES

Al utilizar la arquitectura propuesta se evidencia gran productividad en el desarrollo, el uso de los controles de usuarios, los bindings personalizados e Widgets permitió desarrollar vistas de alta complejidad con mucha facilidad. La infraestructura de Localización automáticamente acopla los recursos de las vistas obtenidas, textos e imágenes al local deseado por el usuario. El compositor de módulos dispone los módulos, menús y barra de herramientas solo con la implementación de un grupo de convenciones. El prototipo de Interfaz de Usuario aprovecha las fortalezas de JQuery UI y Bootstrap brindándole al usuario una experiencia al interactuar con el software. La estructura de archivos y carpetas brindará una organización y estructura lógica y sugerente a los desarrolladores.

## REFERENCIAS BIBLIOGRÁFICAS

1. DICOM Standards Committee. DICOM PS3.1 2015c - Introduction and Overview [Internet]. Digital Imaging and Communication in Medicine [Internet]. Citado: 08 de Septiembre de 2015. Disponible en: http://dicom.nema.org/medical/dicom/current/output/docx/part01.docx

2. Nance Jr JW, Meenan C, Nagy PG. The Future of the Radiology Information System [Internet]. AJR. 2013 May; 200(5). Citado: 04 de Septiembre de 2015 Disponible en: http://www.ajronline.org/doi/pdf/10.2214/AJR.12.10326

3. MVP Award Program. An Introduction to New Features in C# 5.0 [Internet]. 2012. Citado: 12 de Septiembre de 2015. Disponible en: http://blogs.msdn.com/b/mvpawardprogram/archive/2012/03/26/introduction-of-new-features-in-c-5-0.aspx

4. Bienvenido a Visual Studio 2015 [Internet]. 2015. Citado: 08 de Septiembre de 2015. Disponible en: https://msdn.microsoft.com/es-es/library/dd831853.aspx

5. Microsoft .NET Framework 4.5 [Internet]. Citado: 14 de Septiembre de 2015. Disponible en: https://www.microsoft.com/es-es/download/details.aspx?id=30653

6. Scott G, Scott H, Anderson R. Getting Started with ASP.NET MVC 5 [Internet]. 2015. Citado: 15 de Septiembre de 2015. Disponible en: http://www.asp.net/mvc/overview/getting-started/introduction/getting-started

7. Wasson M. Getting Started with ASP.NET Web API 2 [Internet]. 2015.Citado: 07 de Septiembre de 2015. Disponible en:http://www.asp.net/web-api/overview/getting-started-with-aspnet-web-api/tutorial-your-first-web-api

8. Takada M. Single page apps in depth [Internet]. Citado: 16 de Septiembre de 2015. Disponible en: http://singlepageappbook.com/

9. Document Object Model (DOM) [Internet]. World Wide Web Consortium. 2009. Citado: 17 de Septiembre de 2015. Disponible en: http://www.w3.org/DOM/

## ANEXO

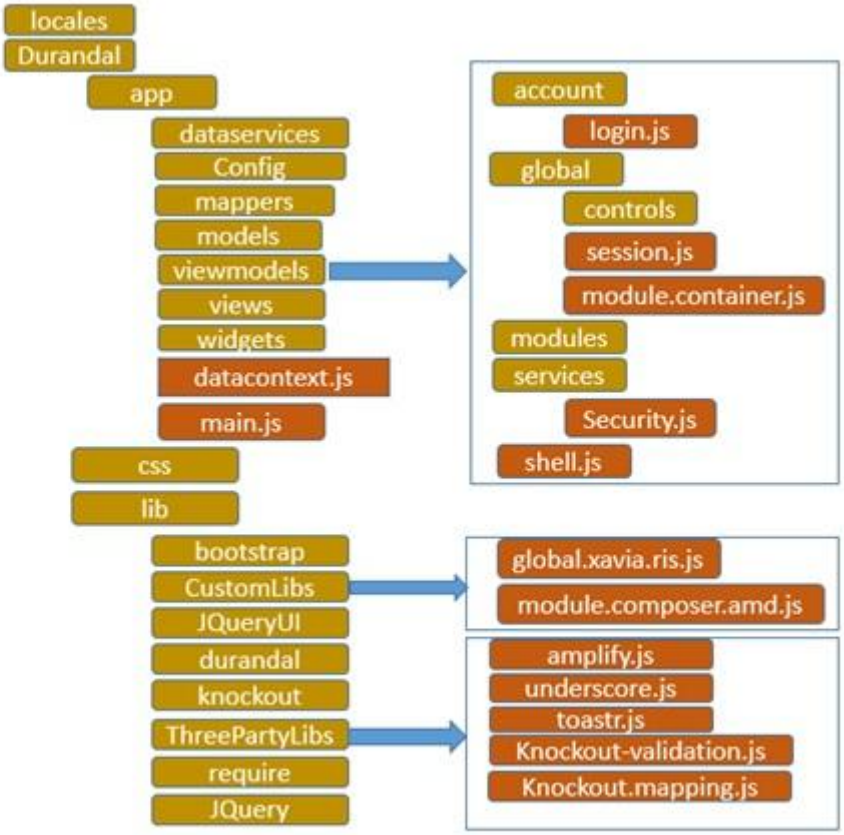


**Fig. 11.** Estructura de Archivos propuesta

Recibido: 3 de diciembre de 2015.
Aprobado: 6 de mayo de 2016.

[1]Se refiere a equipos que implementan parcial o totalmente el Estándar DICOM.
[2]Se refiere a los Sistemas de Archivado y Comunicación de Imágenes.
[3]Se refiere a la tecnología JavaScript que permite la comunicación asincrónica con el servidor web.
[4]Se refiere al actor encargado la proveer servicios de datos, restricciones de seguridad y que accede a la fuente de datos subyacente.
[5]Se refiere a un marco de trabajo, que generalmente es una pieza de software subyacente y brinda un Kit de Herramientas de Desarrollo.
[6]Se refiere a los Patrones Arquitectónicos Modelo-Vista-Controlador y Modelo-Vista-Vista-Modelo respectivamente.
[7]Se refiere a la actualización parcial de la página que generalmente ocurre como consecuencia de una solicitud asincrónica al servidor.
[8]En ASP .NET se conoce como ViewState (Estado de la Vista) y se usa para simular mantenimiento de estado de la vista propio de aplicación de escritorio.
[9]Se refiera a la capacidad de una pieza de software de enlazar o rellenar objetos desde otra estructura de datos. En la Web muy a menudo se usa esto para crear objetos con la información obtenida del servidor en formato JSON o enlazar piezas

de marcado HTML con objetos, resultando la vista final.
[10]Se refiere a la interfaz dela plataforma yde lenguaje neutroque permitirá alos programas yscripts accederdinámicamentey actualizar elcontenido, la estructuray el estilo delos documentos. El documentose puede procesar adicionalmentey los resultados deque el procesamientose puede incorporarde nuevo enla páginapresentada.
[11]Se refiere al manejo de eventos y la comunicación entre piezas de software mediante mensajes asincrónicos.
[12]Se refiere a una pequeña pieza de software queentre sus objetivos están dar fácil acceso a funciones frecuentemente usadas y proveer de información visual.
[13]Se refiere al Back-end que provee servicios de datos y seguridad.
[14]Se refiere a una clave que otorga al Back-end a un usuario autenticado para su identificación en transacciones posteriores.
[15]Se refiere a la capacidad de una pieza de software para almacenar y administrar datos localmente por cierto tiempo y bajo circunstancias específicas antes de volver a solicitarlos al servidor.
[16]Se refiere a una pieza de software capaz de obtener de una estructura de datos, una instancia de un objeto.