**CapÍtulo 2**

**Marco teórico de las rich internet applications**

En este capítulo se presentarán algunas definiciones de las *RIA* como así también, sus principales características, tales como: el almacenamiento de datos, la lógica de negocios en el cliente, la comunicación mejorada entre el cliente y el servidor, y a las presentaciones enriquecidas. Seguidamente se dará pié a las diferentes tecnologías y herramientas para implementar a las *RIA*, para finalmente presentar a algunos de los *widgets* (elementos de interfaz interactivos)más utilizados en la comunidad Web para el *front-end* de las aplicaciones.

**2.1 Las Rich Internet Applications (*RIA*)**

Desde el lanzamiento oficial del primer sitio web en 1991 por Tim Berners Lee hasta hoy en día, las aplicaciones Web que forman parte de la red de redes, Internet, han evolucionado de la Web 1.0, en la que los usuarios obtenían información estática representada en documentos hipertextuales, a la Web 2.0, en la cual la información de las páginas es generada de manera dinámica y en la que se combinan, no solamente información textual, sino también, características multimedia en las interfaces (audio, video *streaming*, *widgets* interactivos, entre otros). De igual forma, la evolución en la web también vino acompañada de cambios tecnológicos en los diferentes navegadores web y en los distintos protocolos de comunicación entre las aplicaciones cliente y servidor.

Muchos de estos avances se dieron para ir superando limitaciones de las aplicaciones web tradicionales en cuanto a la usabilidad e interactividad que ofrecen sus interfaces de usuario. En la Web 1.0, la comunicación síncrona existente entre el cliente y el servidor obliga a que por cada acceso a un enlace, el cliente deba esperar la respuesta del servidor, y una vez obtenida la respuesta, el cliente deba recargar la página completamente para actualizar una simple porción de página. Esto da lugar a retardos en el despliegue de las páginas en el cliente presentando de esta forma una interfaz poco interactiva. Con este mecanismo de comunicación, el cliente queda ocioso la mayor parte del tiempo, pudiéndose llevar a cabo actualizaciones de páginas únicamente, cuando ocurre un evento de navegación por parte de un usuario [<garrett>]. He ahí que surgen como alternativa, las denominadas Aplicaciones de Internet Enriquecidas (*Rich Internet Applications* - *RIA*) con la idea de mejorar las aplicaciones web tradicionales, agregando nuevas características que se encuentran presentes en las aplicaciones de escritorio. El término fue introducido en marzo de 2002 por la empresa Macromedia (actualmente Adobe) que en ese entonces abordaba problemas relacionados a las limitaciones en cuanto a la riqueza de las interfaces, medios y contenidos de las aplicaciones [].

Dado que las *RIA* poseen numerosas características innovadoras, es difícil ofrecer una definición formal que englobe todos sus atributos. Diversos autores las han caracterizado en un contexto particular y todas las descripciones presentadas a continuación resultan valederas.

“Las *RIA* emulan características de las aplicaciones de escritorio, mejorando la experiencia de los usuarios con nuevos efectos visuales, dándose principal realce a las características multimedia. El intercambio de los datos puede llevarse a cabo por medio de una comunicación asíncrona, de tal forma que el cliente permanece receptivo a eventos, mientras que continuamente recalcula o actualiza partes de la interfaz de usuario. Las *RIA* se caracterizan por poseer una variedad de controles interactivos de operación (*widgets*), y por dar la posibilidad de utilizar la aplicación con o sin conexión al servidor (uso offline de la aplicación), y también por ofrecer un uso transparente de las capacidades del cliente, del servidor y de la conexión de red” [].

“En las *RIA*, las aplicaciones se cargan de manera completa en el cliente, desde el inicio, realizándose la comunicación con el servidor solamente en caso de que sea necesario actualizar los datos desde una base de datos o bien desde un archivo externo. La navegabilidad de las aplicaciones web mejora de manera substancial, debido a que se evitan las recargas innecesarias de toda la página, actualizando solamente las porciones de ésta que son relevantes. Con esto se minimiza la cantidad de información que se transmite por la red a la par de mejorar la performance de la aplicación[[1]](#footnote-1)”.

“Las *RIA* son aplicaciones web que exhiben *widgets*, comportamientos y características que están presentes en las aplicaciones de escritorio. También, poseen una mayor capacidad de respuesta, son más seguras y presentan una interfaz más avanzada con respecto a las aplicaciones del modelo Web 1.0. Sus características principales incluyen: 1-) el paradigma de página única; 2-) un avanzado esquema de comunicación (con la inclusión de tecnologías *push* y comunicación asíncrona entre el cliente y el servidor y un manejo optimizado de los datos, reduciendo las solicitudes al servidor) y finalmente; 3-) la inclusión de un motor en el cliente (en la forma de máquina virtual o *plug-ins* instalados en el navegador) que administra la disposición gráfica de los elementos y la mayoría de las interacciones locales” [].

De las descripciones anteriores, puede notarse el hecho de que en las *RIA* se busca que las aplicaciones tiendan a comportarse lo más similarmente a las aplicaciones de escritorio, evitando el refrescado excesivo de las páginas, más precisamente, permitiendo el refrescado parcial de ciertas zonas que son relevantes o que necesitan actualizarse dado un cambio de estado o debido a una actualización en la fuente de datos. También puede resaltarse que el refrescado parcial de las páginas es posible debido a la comunicación asíncrona existente entre el cliente y el servidor, que mejora el intercambio de los datos que se transmiten por la red. Los *widgets* o elementos de interfaz interactivos son otra de las características que se encuentran presentes en las *RIA* y que ofrecen una mayor riqueza a las interfaces de usuario, como así también los elementos multimedia como audio y video *streaming*. El lado del cliente en las aplicaciones *RIA* funciona de una manera más independiente del lado servidor y en ocasiones es posible utilizar las aplicaciones de manera *offline*. Finalmente, todos estos objetivos son alcanzados agregando un motor en forma de *plug-in* en el cliente para la administración de las comunicaciones y para la gestión de las interacciones locales.

La definición propuesta por [] engloba la mayor cantidad de características que son comunes en las definiciones anteriormente presentadas y por ende, resulta ser la más completa. En este trabajo de fin de carrera se tendrán en cuenta características *RIA* presentes en esta definición.

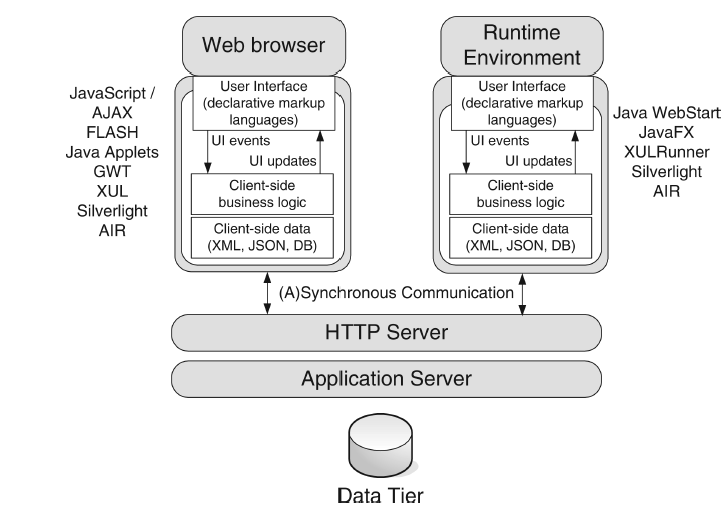


Figura 1 Arquitectura *RIA*. Fuente: Fraternali y otros, 2010.

Por otra parte, en [] se describe la arquitectura de las *RIA* de manera general como se muestra en la Figura 1. El sistema está compuesto de un servidor de aplicaciones web y un conjunto de aplicaciones de usuario corriendo en las máquinas clientes. Estas aplicaciones son implementadas de dos formas: en un navegador web o fuera de un navegador web. En un navegador web es posible implementarla utilizando una variedad de tecnologías como *Javascript*, animaciones Flash, código interpretado en *plug-ins* y *Java applets*. Fuera de un navegador web se implementa en términos de binarios descargados desde la web e interpretados en un ambiente específico de ejecución, por ejemplo, utilizando tecnologías como *Java Web Start*[[2]](#footnote-2) y *Adobe AIR*[[3]](#footnote-3).

Las *RIA* hoy en día juegan un papel preponderante. Según un estudio de mercado patrocinado por la empresa Adobe en 2011, dadas las mejoras con respecto a la interfaz de usuario y al comportamiento de las aplicaciones, las *RIA* han conseguido incrementar la productividad y la satisfacción de los usuarios que llevan a cabo operaciones en internet, debido, en gran medida, a la nueva experiencia de interacción que ofrecen [<rogowskimarch122007>]. Un estudio similar [<kiewe2011>], presenta datos cuantitativos con referencia a cómo una aplicación con características de las *RIA* puede mejorar las utilidades y disminuir los costes de desarrollo en una compañía.

**2.2 Características de las *RIA***

A continuación se presentan las características más distintivas de las *RIA* con respecto a las aplicaciones Web tradicionales, que fueron presentadas en la definición de [<martinez-ruiz2010>] en base a una clasificación propuesta en los trabajos de [] y []. Por cada una de estas características, se muestra un cuadro en donde se reflejan las ventajas y desventajas de cada característica.

**2.2.1 Almacenamiento de los datos**

En las *RIA* es posible almacenar datos en el lado cliente, con diferentes niveles de persistencia (temporalmente, mientras la aplicación está en ejecución, o persistentemente). También, los datos pueden distribuirse entre ambos pares, cliente y servidor.

En la Tabla 1 se presentan algunas ventajas y desventajas de llevar a cabo una distribución de datos entre el cliente y el servidor.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| * Es posible utilizar la aplicación sin necesidad de establecer una conexión con el servidor (uso offline). * Es factible la preparación y validación de los datos en el lado del cliente. | * Existe la posibilidad de replicación de datos en ambos pares. * Puede llegar a tornarse complejo establecer políticas para la asignación (distribución) de los datos. |

Tabla 1 Ventajas y desventajas de la distribución de los datos entre el cliente y el servidor

## 2.2.2 Lógica de negocio

En una aplicación web tradicional, la extracción de datos y la lógica de negocio se computan en el servidor. En las *RIA* es posible llevar a cabo operaciones complejas directamente en el cliente (por ejemplo: efectuar navegaciones, filtrados y ordenamiento de los datos con múltiples criterios; operaciones de dominio específico para sistemas complejos; y validación local de datos). También es factible distribuir la lógica de negocios entre el cliente y el servidor para, por ejemplo, validar algunos campos de un formulario en el cliente y otros en el servidor. Por lo tanto, el diseño conceptual debe responder a la decisión de cómo asignar la computación tanto de las páginas como así también de los componentes de éstas [].

En la Tabla 2 se presentan algunas ventajas y desventajas de una computación distribuida de la lógica de negocios entre el cliente y el servidor.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| * Validación de datos en vivo. * La posibilidad de utilizar la aplicación sin necesidad de una conexión (uso offline de la aplicación). | * La lógica de la aplicación en conjunto se complica. * Puede ser confuso definir si una funcionalidad en particular debe computarse en el cliente o en el servidor. * El restablecimiento de la comunicación entre el cliente y el servidor luego de la utilización offline de la aplicación es una acción propensa a errores. |

Tabla 2 Ventajas y desventajas de una computación distribuida de páginas

## 2.2.3 Comunicación entre el cliente y el servidor

Con las *RIA* se crean mecanismos para reducir al mínimo la transferencia de los datos migrando las capas de interacción y presentación del servidor al cliente. Las *RIA* soportan comunicaciones asíncronas entre el cliente y el servidor para la distribución de objetos de dominio, datos y la computación.

En la Figura 2 se puede ver una comparativa con respecto a la comunicación entre los pares cliente y servidor, de las aplicaciones de la web 1.0 (lado izquierdo) y las actuales basadas en *RIA* (lado derecho). En las aplicaciones web tradicionales, los datos residen en el servidor, y los clientes a medida que necesitan alguna actualización de página, llevan a cabo la solicitud de actualización por medio de la activación de algún enlace navegacional (que puede ser algún hipervínculo, botón de solicitud de registro de usuario, etc.). Seguidamente, en respuesta a la solicitud del cliente, el servidor devuelve la página con la actualización correspondiente. La comunicación es llevada a cabo de una manera síncrona, en donde un evento del usuario es necesariamente el elemento disparador de una solicitud al servidor. Con las *RIA*, un motor instalado en el cliente es el encargado de gestionar las solicitudes de transferencia de los datos al servidor y de gestionar los cambios en la disposición de los distintos elementos en la interfaz del usuario. Las solicitudes al servidor, al ser gestionadas asíncronamente por un motor (o *plug-in*) instalado en el cliente, permiten a la aplicación llevar a cabo diversas acciones en paralelo, como por ejemplo, actualizar distintas porciones de una misma página en un momento dado.

Figura 2 Comunicación síncrona versus comunicación asíncrona. Fuente: E Palacios y otros, 2008.

En la Tabla 3 se muestran algunas ventajas y desventajas de una comunicación asíncrona entre el cliente y el servidor.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| * Es posible llevar a cabo el refrescado parcial de las páginas, abarcando solamente las zonas de interés. * Se mejora la interacción del usuario con la aplicación. * *Server-push[[4]](#footnote-4)*. | * Se incrementan los esfuerzos de desarrollo de las aplicaciones. * El *testing* de las aplicaciones se vuelve más complejo. |

Tabla 3 Ventajas y desventajas de una comunicación asíncrona en entre el cliente y el servidor

## 2.2.4 Presentaciones enriquecidas

Las interfaces de usuario ofrecen una mayor riqueza con el manejo de eventos en el lado del cliente y con la inclusión de *widgets* interactivos. Los *widgets* son micro programas empotrados dentro de las páginas web y son administrados por un motor de *widgets* (que podría ser un *plug-in* instalado en el navegador). Los *widgets* presentan funciones bien específicas que por lo común resultan de utilidad a los usuarios tales como: presentar el estado del tiempo, la hora de diversos países, la cotización de las monedas extranjeras, calculadoras, entre otros. Los elementos multimedia dentro de las páginas como la intrusión de audio y video de alta calidad, a la par de animaciones también son características típicas de las *RIA*. Así también, la capacidad de arrastrar y soltar elementos dentro de la interfaz, las auto-sugerencias de datos a medida que se va escribiendo un patrón en un campo y el refrescado automático de las páginas (o porciones de esta), son otras de las características interesantes que pueden encontrarse.

En la Tabla 4 se presentan algunas ventajas y desventajas de un comportamiento más sofisticado en la interfaz de usuario.

|  |  |
| --- | --- |
| **Ventajas** | **Desventajas** |
| * Funcionamiento como una aplicación de una sola página, evitando de esta forma perderse en la navegación del sitio web. * Se presenta al usuario una interfaz mucho más enriquecida y reactiva a eventos. | * Pueden presentarse problemas de rendimiento. * Es posible que se tengan incompatibilidades en el navegador web. |

Tabla 4 Ventajas y desventajas de un comportamiento más sofisticado en la interfaz de usuario

De todas las características subyacentes a las *RIA* anteriormente descriptas, las presentaciones enriquecidas (que abarcan el manejo de eventos en el lado del cliente, los *widgets* interactivos, el paradigma de una sola página y el contenido multimedia) son las que representan el *look and feel* final de las aplicaciones. Por lo tanto, esta característica resulta ser percibida en primera instancia por parte de los usuarios finales. He ahí su importancia y el por qué son numerosos los *frameworks* de desarrollo existentes en la actualidad que los contemplan.

Dada la amplitud que conllevan las *RIA* en lo que concierne a características y funcionalidades, y con el afán de mostrar algunas de sus bondades principales de una manera concreta, este trabajo de fin de carrera se enfocar principalmente en los aspectos concernientes a presentaciones enriquecidas.

## 2.3 Tecnologías para la implementación de las *RIA*

Actualmente, las capacidades de las *RIA* se pueden implementar en diferentes tecnologías cliente que pueden clasificarse en tres categorías principales, de acuerdo con el entorno de ejecución:

• **Basadas en *Javascript*:** en este caso, la lógica del lado cliente está implementada en *Javascript* (el enfoque también es conocido como "*Ajax*", *Asynchronous Javascript y XML* [<garrett>] y las interfaces de usuario se basan en una combinación de *HTML* y *CSS*.

Figura 3 A la izquierda el Modelo de aplicación web clásico. A la derecha, el modelo de aplicación web Ajax. Fuente: J J, Garret, 2005.

La principal ventaja de este enfoque es que se basa en el *Javascript* incorporado en el navegador y soporta los estándares de *W3C*. En la Figura 3 se presenta el modelo de aplicación Ajax en comparación con el modelo de aplicación web clásico. Como puede apreciarse para el caso del modelo Ajax, el motor Ajax es el encargado de orquestar la disposición de los elementos en la interfaz de usuario en el lado del cliente por medio de *HTML* y *CSS*. El manejo de las interacciones entre la interfaz de usuario y el motor *Ajax*, que representan a la lógica de negocio en el lado del cliente, son implementadas por medio de *Javascript*.

El motor *Ajax* gestiona la comunicación entre el cliente y servidor, por medio de solicitudes *HTTP* o *HTTPS*, obteniendo las respuestas del lado servidor, interpretando los datos utilizando lenguajes de marcado como *XML* o *JSON*. Los principales inconvenientes son el soporte multimedia insuficiente, limitaciones en las cajas de arena (*sandboxes*) del navegador, por ejemplo, el acceso al sistema de archivos o almacenamiento persistente, y la inconsistencia en el comportamiento del navegador. Debido a este último aspecto, un gran número de bibliotecas se han propuesto para permitir a los desarrolladores abstraerse de las idiosincrasias del navegador.

• **Basadas en *plug-ins*:** en este caso, la representación avanzada y el procesamiento de eventos se encomienda a los *plug-ins* del navegador por medio de la interpretación de lenguajes específicos de scripting, *XML* o archivos multimedia. Una ventaja de los *plug-ins* es que generalmente soportan la interacción multimedia de forma nativa, permitiendo la persistencia en el lado del cliente y ofrecen un mejor desempeño que *Javascript* interpretado. Algunos *plug-ins* vienen ya instalados en los navegadores, pero otros requieren de la intervención del usuario administrativo. Sin embargo, en algunos casos no proveen el acceso a servicios del sistema operativo (por ejemplo, al sistema de archivos).

• **Basadas en entornos de ejecución:** en este caso, las aplicaciones se descargan de la Web, pero se ejecutan fuera del navegador, utilizando un ambiente de escritorio en tiempo de ejecución. Estas soluciones ofrecen lo máximo en términos de capacidades de cliente y el uso *off-line*, con pleno acceso al sistema operativo subyacente. Sin embargo, se basan en un ambiente especializado en tiempo de ejecución, lo que obliga a los usuarios a que lo instalen (y podría no estar disponible en todas las plataformas, como por ejemplo en teléfonos móviles). Muchas de las tecnologías *RIA* se pueden utilizar para desarrollar aplicaciones de este tipo.

En la Tabla 5 puede apreciarse las capacidades y las limitaciones de cada una de las tecnologías con respecto a las características descritas en la sección 2.2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tecnología cliente vs**  **Características de las *RIA*** | **Presentaciones enriquecidas** | **Almacenamiento de los datos en el cliente** | **Lógica de negocio en el cliente (o distribuida entre el cliente y el servidor)** | Comunicación entre el cliente y servidor |
| **Basados en JavaScript** | Limitada: sin multimedia | Limitada: no hay persistencia de datos | Si | Si |
| **Basados en *Plug-ins*** | Si | Si, con *plug-ins* adicionales | Si | Si |
| **Basados en ambientes en tiempo de ejecución** | Si | Si | Si | Si |

Tabla 5 Tecnología cliente vs Características de las *RIA*

Las tecnologías basadas en *Javascript* son las más populares y más ampliamente adoptadas para el desarrollo de *RIA*. Sus limitaciones actuales se suelen resolver utilizando extensiones de Flash para el procesamiento de vídeo (y *Google Gears* o *Flash Shared Objects* cuando en el lado del cliente el almacenamiento persistente es necesario). Algunas de las razones por las cuales las tecnologías *Javascript* son las más utilizadas son: 1-) Ajax es asumido por muchos desarrolladores como el conjunto más abierto y estándar de tecnologías y la más cercana a la especificación *HTML5*; 2-) no requiere acciones administrativas (por ejemplo, la instalación de software) de los usuarios, 3-) se pueden combinar fácilmente con *plug-ins* que se construyen para superar sus limitaciones.

**2.4 Herramientas para el desarrollo de las *RIA***

En la sección anterior se presentaron las diferentes tecnologías para el desarrollo e implementación de las *RIA*. Para este trabajo de fin de carrera, se optó analizar en mayor detalle las herramientas y *frameworks* de desarrollo de uso abierto y que son de amplia utilización en la comunidad web. Se ha señalado el hecho de que las implementaciones basadas en *Javascript* o librerías Ajax son las más utilizadas en la actualidad, debido a que utiliza tecnologías de uso abierto estandarizado como lo son *Javascript*, *HTML* y *CSS*. Además, esta forma de implementar las *RIA* es la más cercana al estándar *HTML5*.

Son numerosas las librerías *Javascript* existentes en la actualidad. Estas librerías tienen como objetivo abstraer a los desarrolladores de tener que lidiar directamente con el *DOM* (*Document Object Model*) para la disposición de los elementos en las páginas web, ofreciendo capas de software amigable, reduciendo notablemente los tiempos de desarrollo y mejorando la productividad. En la Figura 4 se puede apreciar algunas librerías *Javascript* de uso extendido.



Figura 4 Algunas librerías *Javascript* de uso común

Estas librerías también buscan explotar el lado del cliente en las aplicaciones y minimizar las interacciones con el lado servidor, para que de esta forma se obtenga un mejor rendimiento. A la par de permitir a los desarrolladores implementar aplicaciones a un alto nivel de abstracción, las librerías ofrecen una gran variedad de *widgets* interactivos que son de uso común en las aplicaciones web.

Los *widgets* representan elementos enriquecidos para la interfaz de usuario, que tienen como objetivo ofrecer una mayor interactividad, dada sus características dinámicas y un comportamiento general, similar a los patrones de comportamiento. Los *widgets* son microprogramas que cumplen una función predeterminada. Sus propiedades pueden ser modificadas para expresar comportamientos personalizados por el usuario. Una vez modificada las propiedades del *widget*, éste es introducido dentro de la aplicación para cumplir una función en particular.

**2.4.1 *Widgets* más utilizados**

En estudios llevados a cabo en 2009[[5]](#footnote-5) y 2010[[6]](#footnote-6) , se presenta un análisis de los *widgets* más utilizados por las aplicaciones web. La presenta tales *widgets*. De todos los *widgets* identificados, resulta interesante determinar cuáles son los más utilizados hoy en día, por lo que en el marco de este trabajo se ha realizado un análisis de portales web populares (Facebook, Gmail, Youtube y Amazon) para determinar qué *widgets* son comunes en estos sitios. El análisis determinó que los cuatro portales utilizan los siguientes *widgets*:

***Accordion:*** Muestra paneles de contenido plegable para presentar la información en una cantidad limitada de espacio.

***Tabs:*** Ofrece una sola área de contenido con múltiples pestañas, cada uno asociado con una cabecera en una lista.

***Autocomplete:*** Permite a los usuarios seleccionar un texto de interés rápidamente de una lista. A medida que se escribe en un campo de entrada, algunas sugerencias son presentadas al usuario, en base al patrón actual de caracteres ingresados. El usuario puede elegir una de las palabras sugeridas en un momento dado o bien seguir ingresando caracteres para refinar la búsqueda.

***Tooltip:*** Ofrece mensajes personalizados de sugerencia sobre los elementos de interfaz, reemplazando los mensajes nativos.

Figura 5 Elementos de interfaz de usuario enriquecidos (*widgets*) más utilizados. Fuente: Designing Web Interfaces: <http://designingwebinterfaces.com/essential_controls>, 2015.

***Datepicker:*** Permite seleccionar una fecha de un calendario emergente o *inline*.

***Live validation:*** Ofrece validaciones en vivo de los campos en los formularios.

**2.4.2 Las librerías *Javascript* *jQuery*UI[[7]](#footnote-7) y *jQuery Validation Plugin*[[8]](#footnote-8)**

Estudios de mercado recientes han presentado a *jQuery*[[9]](#footnote-9) como la librería *Javascript* más utilizada a nivel global[[10]](#footnote-10). *jQuery* es de código abierto y ha tenido un crecimiento notable en términos de evolución hasta hoy en día desde su aparición en el año 2005. La librería *jQuery* propone una manera robusta y confiable para desarrollar código *Javascript*. Posee extensiones para los dominios de aplicación Web (*jQueryUI)* y móviles (*jQuery Mobile*[[11]](#footnote-11)). En ambos dominios de aplicación, se encuentran numerosos *widgets* interactivos idóneos para las interfaces de usuario enriquecidas.

De todas las características citadas en la sección anterior, *Live validation* es el único *widget* no soportado por *jQueryUI* de manera nativa. Sin embargo con *jQuery Validation Plugin* (extensión basada en *jQuery)*, es factible llevar a cabo validaciones locales sobre los campos de un formulario de una manera bastante intuitiva.

Con *jQueryUI* y *JQuery Validation Plugin*, es posible dar cobertura a todos los *widgets* que serán tenidos en cuenta en este trabajo de fin de carrera.

**2.5 Resumen del Capítulo**

En este capítulo se han visto las diversas características que ofrecen las *RIA*, como así también los enfoques tecnológicos para explotar el lado del cliente en este tipo de aplicaciones. Estos enfoques son: las implementaciones basadas en librerías *Javascript,* las basadas en la instalación de *plug-ins,* o las que se enfocan en ambientes en tiempo de ejecución. La primera de ellas es la más utilizada en la comunidad web, debido a que la aplicación se implementa por medio de un compendio de estándares abiertos trabajando conjuntamente como lo son *HTML* y *CSS* (para la representación de los elementos y el posicionamiento), *Javascript* (para la lógica de la aplicación en el lado cliente) y *XML* o *JSON* (para la comunicación entre el cliente y el servidor).

Son varias las librerías *Javascript* existentes en la actualidad[[12]](#footnote-12). Algunas de ellas permiten la representación de ciertos elementos de interfaz interactivos (*widgets*) que son comunes en las interfaces enriquecidas actuales y a la vez ofrecen la posibilidad de agregar cierta lógica en el lado cliente, como validaciones locales de campos de entrada en los formularios. Según el análisis llevado a cabo considerando importantes portales web, entre los *widgets* más utilizados se encuentran los *tooltips*, *tabs, accordion, datepicker y autocomplete,* como así también diversas validaciones locales en los campos de entrada (por ejemplo, validaciones de tipo numérico, email, claves, etc.). Actualmente, *jQuery* es la librería *Javascript* más popular. Además, utilizando sus versiones *jQueryUI* y *JQuery Validation Plugin* se brindacobertura a todas estas características enriquecidas.

1. **Wikipedia- *Rich internet applications*:** <http://en.wikipedia.org/wiki/Rich_Internet_application> 2015 [↑](#footnote-ref-1)
2. **Oracle:** <http://www.oracle.com/technetwork/java/javase/javawebstart/index.html> 2015 [↑](#footnote-ref-2)
3. **Adobe AIR:** <http://www.adobe.com/products/air.html> 2015 [↑](#footnote-ref-3)
4. Describe un estilo de comunicación sobre Internet donde la petición de una transacción se origina en el servidor. [↑](#footnote-ref-4)
5. **Designing Web Interfaces:** <http://designingwebinterfaces.com/essential_controls> 2009 [↑](#footnote-ref-5)
6. **UX BOOTH:** [http://www.uxbooth.com/articles/essential-controls-for-Web-applications/](http://www.uxbooth.com/articles/essential-controls-for-web-applications/) 2010 [↑](#footnote-ref-6)
7. **jQuery user interface:** <http://jqueryui.com/> 2015 [↑](#footnote-ref-7)
8. **jQuery Validation Plugin:** <http://jqueryvalidation.org/> 2015 [↑](#footnote-ref-8)
9. **jQuery:** <http://jquery.com/> 2015 [↑](#footnote-ref-9)
10. **Usage of JavaScript libraries for websites** <http://w3techs.com/technologies/overview/javascript_library/all> 2015 [↑](#footnote-ref-10)
11. **jQuery Mobile:** <http://jquerymobile.com/> 2015 [↑](#footnote-ref-11)
12. **List of Ajax frameworks:** http://en.wikipedia.org/wiki/List\_of\_Ajax\_frameworks 2015 [↑](#footnote-ref-12)