1. Introducción

La industria del software ha pasado por diferentes etapas y épocas basado en lo que la tecnología del momento le ha posibilitado. En cuanto a las aplicaciones empresariales, durante la última década y media, las aplicaciones basadas en tecnología web han copado el mayor espacio. Esto sin duda, ha sido posible dadas las bondades que presentan las tecnologías basadas en la web para distribuir las aplicaciones. Cabe recordar, que antes de esta era de aplicaciones web, el estilo arquitectural cliente-servidor era el usado por la mayor parte de las soluciones. Con los problemas propios de dicho estilo, la llegada del concepto de aplicaciones web, disparó las posibilidades de distribuir las aplicaciones a escalas antes no imaginadas. El concepto de cliente ligero basado en un browser web, permite que la misma aplicación pudiera ser accedida casi que sin restricciones desde diferentes posiciones geográficas, desde diferentes entornos de ejecución, desde diferentes sistemas operativos y desde diferentes tipos de navegador.

Actualmente las aplicaciones basadas en la web son usadas en casi cualquier sector de la industria. Esto ha generado también que surjan diversas tecnologías que soportan su desarrollo. Finalmente, con la masificación de las tecnologías móviles, la construcción de aplicaciones web ha tenido un nuevo impulso originado en la posibilidad de desplegar la misma aplicación hacia navegadores convencionales como hacia dispositivos móviles.

En este ecosistema de aplicaciones web, el browser siempre ha jugado el rol de cliente liviano, desplegando el contenido generado por un servidor encargado de procesar y generar datos.

Acompañando al browser en su labor de desplegar la interfaz gráfica de las aplicaciones web, el lenguaje de programación JavaScript ha sido ampliamente usado como complemento a HTML en su rol de despliegue de interfaz de usuario. Sin embargo, su uso ha estado siempre enmarcado en el “enriquecimiento” visual de la aplicación y ayudando en todo cuanto sea posible en la capa de presentación. Su diseño como lenguaje nunca fue concebido para escribir lógica de negocio, de hecho, tradicionalmente ha sido considerado como un lenguaje lento en términos de tiempos de ejecución debido a su naturaleza de ser un lenguaje interpretado y no compilado.

Este esquema tradicional de construcción de aplicaciones web, en su capa de presentación, está dando un giro paulatino hacia el concepto de SPA (Single Page Application). Bajo este esquema, la interfaz de usuario sigue siendo generada a través de HTML pero JavaScript comienza a ser usado como el lenguaje para definir la lógica de negocio.

Esto rompe los paradigmas con los cuales han venido siendo construidas las aplicaciones web desde siempre. Primero, JavaScript ya no es un lenguaje lento (dependiendo del browser) ya que gracias a proyectos como V8 (máquina virtual de google), el código JavaScript es compilado y optimizado para cada sistema operativo en donde debe ser ejecutado. Segundo, la aparición de la última generación de HTML más conocido como HTML5 permite la visualización de todo tipo de contenido a través de un navegador web a la vez que permite desplegar el mismo contenido desde diferentes dispositivos (browser, móviles, tabletas, etc.) de manera transparente.

Este nuevo esquema de construcción de aplicaciones web tiene mucha similitud con el tradicional esquema cliente-servidor. Durante muchos años se ha desperdiciado poder de computación al delegar toda la responsabilidad computacional hacia los servidores y dejando los tradicionales PCs como clientes livianos, cuando en realidad los equipos de cómputo actuales tienen muchísimo poder computacional como para limitarse a ser simples “pintadores” de HTML. Con este nuevo esquema web, que como se dijo antes, retoma los conceptos del esquema cliente-servidor, se le puede sacar provecho a la capacidad de cómputo local y delegar a los servidores a ser repositorios de información.

Este enfoque de construcción de software presenta ciertos desafíos, entre ellos, algunos relacionados con la diversidad de tecnologías que se pueden emplear para la creación de las aplicaciones y la poca o baja estandarización que se tiene en la industria para la construcción de las mismas. Entre otras cosas, esto ocasiona que las labores de mantenimiento sobre una aplicación sean tareas complejas y costosas de realizar.

Este trabajo se centra en solucionar algunos de los problemas relacionados con la construcción de software, especializándose en aplicaciones transaccionales construidas con el enfoque mencionado al comienzo de esta introducción.

Se ha encontrado que este tipo de aplicaciones comparten entre sí muchos requerimientos por lo cual utilizando enfoques guiados por modelos, se pueden construir generadores de código para automatizar las labores de desarrollo. Sin embargo, esto genera nuevos retos y problemas. Tradicionalmente la generación de código bien sea con los enfoques de modelos o herramientas tipo CASE, está amarrada a una tecnología en particular lo cual dificulta la evolución, actualización o modernización de una aplicación.

También es claro que las aplicaciones generadas requieren ser completadas manualmente por lo cual, la arquitectura definida por la herramienta CASE o el generador de código se pierde con facilidad dado que no hay posibilidad de tener control sobre el código adicionado.

Una forma de solucionar este inconveniente es modelando no solo el comportamiento normal de o básico de la aplicación sino también modelar las extensiones que se le pueden hacer al mismo, de este modo la intervención de un programador para adicionar nuevas características a un software estarán gobernadas por la definición de dicho modelo de extensiones. De este modo, se logra controlar el crecimiento de una solución sin perder la arquitectura definida por la estrategia de generación de código.