Evaluación de Grails

Gabriel Centurion
gccomputos@gmail.com

Maximiliano Felix
maxi.felix@gmail.com

RESUMEN

Cuando nos enfrentamos a un desarrollo web nos cuestionamos cual seria la mejor herramienta para utilizar. Cual nos dará mejores ventajas tomando en cuenta la velocidad de desarrollo y la integración con otras tecnologías. Aún limitándonos en la plataforma Java son muy variadas las opciones.

En este articulo nos concentraremos en evaluar el framework Grails utilizando además tecnologías que están fuera del mismo, integradas y no a través de sus plugins.[2]

Para evaluar esta herramienta se realizo una aplicación que la llamamos Scrumme la cual permite gestionar proyectos que apliquen la metodología Ágil Scrum.

Durante el desarrollo fuimos encontrando distintas bondades del framework que nos facilitaron el trabajo y otras prometedoras que no cumplieron las expectativas. En el articulo se presenta el diseño de la aplicación como así la evaluación de las tecnologías utilizadas.

Palabras Clave

Grails, Groovy, Java, J2me, Spring Source Tookit, JQuery, Quartz, Scrum [14],

1. INTRODUCCIÓN

Grandes esfuerzos se han hecho en el campo de las aplicaciones web basadas en Java, pero crear aplicaciones aun parece un montón de trabajo. La propuesta de Grails, es batir con todos los problemas que se enfrenta un desarrollador al momento de crear una aplicación basada en Java, mejorando principalmente la meta a la que apuntan las empresas de hoy día, la productividad. Grails es un framework de desarrollo web basado en Java, que toma "lo mejor" de las herramientas, técnica y tecnologías existentes de los frameworks Java y los combina con el poder y la innovación de los lenguajes dinámico. El resultado es un framework que ofrece la estabilidad de las tecnologías que ya conoces, pero te protege de la engorrosa configuración, la complejidad de los diseños, y el rehúso excesivo de código que hacen del desarrollo web en java tan tedioso. Grails te permite invertir tu tiempo desarrollando nuevas funcionalidades, y evitar la edición de XML's. Grails es un framework Java de la próxima generación, que genera una alta productividad en los desarrolladores a través de la confluencia de un lenguaje dinámico, la filosofía "Convención sobre Configuración", poderosas herramientas de soporte y extensión, y una perspectiva ágil subyacente de los mejores y emergentes paradigmas de desarrollo web.

El trabajo realizado se presenta en el marco de la edición 2010 de la materia Taller de Sistemas de Información.

El resto del documento se organiza de la siguiente manera. La sección 2 presenta el marco conceptual describiendo el mismo algunos conceptos que consideramos fundamentales para la comprensión del documento, En la sección 3 describimos en problema planteado así como la motivación para realizar el mismo. En la sección 4 describimos la solución al problema planteado como así detalles del diseño de la misma. En la sección 5 se presenta la arquitectura del sistema. En la sección 6 se describe los artefactos utilizados para la implementación, las diferentes herramientas los puntos fuertes y débiles de cada una como también los problemas encontrados. En la sección 7 evaluamos la solución. . En la sección 8 comentamos sobre como maneamos los tiempos durante el proyecto. Por último, en la sección 9 se presentan las conclusiones del trabajo y el trabajo a futuro.

2. MARCO CONCEPTUAL

En esta sección se describen varios conceptos que pueden facilitar la comprensión del documento y la aplicación planteada.

2.1 *Convención sobre Configuración*

Una de las propiedades mas interesantes de Grails, es que se necesitan muy pocos archivos de configuración. Grails toma la mayoría de las decisiones basándose en convenciones extraídas directamente del código fuente.
Por ejemplo: si creas un controlador "UsuarioController" y una acción "perfil", automáticamente Grails lo expondrá bajo la URL: /app/usuario/perfil, entre otras interesantes aplicaciones de este principio.
Pero Grails es Convención sobre configuración, no convención envés de configuración, esto significa que si es necesario cambiar los estándares se puede hacer a través de archivos de configuración, de una manera muy cómoda, sin tener que usar XML. Y aun mas, si se quisiera utilizar archivos de configuración en XML ya existentes, Grails es totalmente compatible.

2.2 *Filosofía ágil*
Grails brinda una forma muy ágil de desarrollo, aprovechando al máximo las bondades del lenguaje dinámico Groovy, lo cual hace al framework conciso y expresivo, haciendo las tareas de desarrollar una aplicación mucho más fácil de extender y mantener.

2.3 *Fundamentos solidos*
Grails está basado en tecnologías como Spring y Hibernate, tecnologías solidas altamente probadas en el mercado, y aceptadas, brindando altos niveles de confiabilidad. Esto también es bueno, pues la magia de Grails esta tras bambalinas, mejorando mucho la curva de aprendizaje. Esta característica se extiende también a los plugins[2], por ejemplo el plugin de agendado de procesos esta basado en Quartz, un sistema muy conocido y usado en el mundo empresarial.

2.4 *Generación automática*
El framework brinda mecanismos muy rápidos para rápidamente crear aplicaciones que funcionen apenas creadas, a esto llama "scaffolding" brinda una estructura básica sobre la cual se pueden apoyar las aplicaciones para seguir creciendo a medida que se las desarrolla, además de brindar otras funcionalidades como generación de "templates" muy útiles para testeo, para modelar el dominio de la aplicación, desarrollo de librerías de tags, etc.

2.5 *Integración con Java*
Al correr sobre una maquina virtual java, toda aplicación es compatible con todo tipo de librerías desarrolladas enteramente en Java (teniendo en cuenta por supuesto versiones de Java y niveles de compilación), además de otra de las bondades de Groovy como ser que la sintaxis puede parecerse tanto a Java como el desarrollador quiera, esto ultimo también ayuda muchísimo a que un desarrollador Java pueda pasarse a Groovy rápidamente, aunque no funciona muy bien al revés.

2.6 *Comunidad*
Otra de las cosas que hacen grande a Grails es su increíble comunidad, muy activa y de gran participación, tanto de expertos desarrolladores como de novatos.

2.7 *Productividad*
La gran productividad que brinda el framework permite poder enfocarse en detalles que pueden hacer muy importante una aplicación web, que antes debías dejar de lado porque los tiempos apremiaban.

2.8 Scrum
Metodología que define un marco de trabajo ágil, al cual se le incorporan un conjunto de buenas prácticas.

Su filosofía se basa en involucrar a los actores en el proceso de desarrollo

2.9 Scrum Master
Actor que se encarga de que se cumplan las pautas de Scrum o sea que la metodología se cumpla sirve como guía hasta que en este caso el grupo de desarrollo comprenda y apliqué (dentro de las posibilidades) la metodología

2.10 Product Owner
Actor que se encarga de definir los requerimientos y participa en los avances del proyecto, siendo el responsable de que los requerimientos se cumplan.

2.11 Sprint
Espacio en el tiempo definido por el equipo para finalizar un subconjunto de tareas las cuales les brinden al proyecto valor.

2.12 Product Backlog
Conjunto de tareas necesarias para finalizar el proyecto.

2.13 *Planning Poker*
Juego de cartas que se utiliza para estimar en este caso la duración de las tareas. Consiste en que cada jugador elija el costo en tiempo para cierta tarea planteada. Se presenta la elección al mismo tiempo que el resto, no permitiendo de esta manera que los jugadores vean influenciadas sus estimaciones por el resto de los jugadores.

3. DESCRIPCIÓN DEL PROBLEMA

Como forma de Evaluar Grails se plantea realizar un sistema web, para dar soporte al desarrollo de proyectos que utilicen la metodología Scrum. Dicho sistema proveerá las siguientes funcionalidades.

Dashboard de Scrum en donde para cada sprint, se mostraran las tareas, el estado de las mismas y estimaciones. ABM de usuarios, proyectos, sprints, tareas y su clasificación. Revisión de avance de proyecto presentando tablas, gráficas.

Estadísticas como ser horas hombre por tipo de tarea Manejo de product backlog y versionado de entregables.

Los principales objetivos de este desarrollo además de la evaluación de Grails son la evaluación de Groovy como lenguaje scripting, evaluación de integración con componentes ricos como ser librerías js y Flex. Además se busca evaluar las ventajas del framework en la exposición de WS. Para ello se plantea implementar una aplicación cliente en J2ME que consuma los mismos.

4. SOLUCIÓN PLANTEADA

El sistema implementado provee soporte a los diferentes actores de Scrum. Estos son los desarrolladores, el Scrum master el product owner.

Al momento que el usuario ingresa al sistema a través del login se identifica que tipo de usuario es y de esta manera se cambia las vistas por defecto. Dentro de las funcionalidades implementadas se destacan el dashboard de tareas por sprint, planning poker, vista resumen de proyecto, soporte para mensajería, manejo de reuniones como ser acta y sistema de alarmas para indicar el inicio de las mismas.

Para consumir WS expuestos se realizo una aplicación Cliente en J2ME la cual a través de usuario y password se presenta los los proyectos a los que pertenece. Grails no propone nada

particular para el desarrollo en J2ME, la aplicación se desarrollo por fuera del framework utilizando las ventajas de Wireless Tool Kit

En la Figura 1 presentamos la parte del modelo de domino del sistema que representa lo relativo a las tareas proyectos y sprints. En dicha figura se puede notar que una tarea está relacionada con el proyecto tanto directamente como a través de los Sprints del Proyecto. Cabe mencionar que una tarea esta relacionada a un Sprint que pertenece a el Proyecto que se encuentra relacionada a través del product backlog.

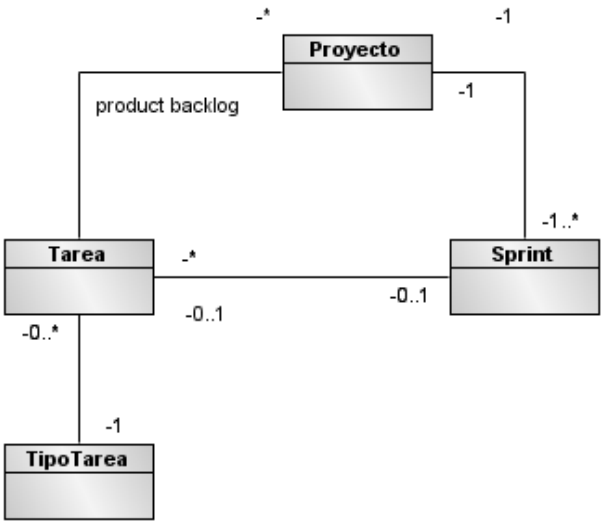


Figura 2 – Modelo Tareas

En la Figura 2 podemos observar cómo se modeló lo concerniente al planning poker. En cada mano se estima una tarea y es el creador del juego quien tiene la última palabra en la estimación.

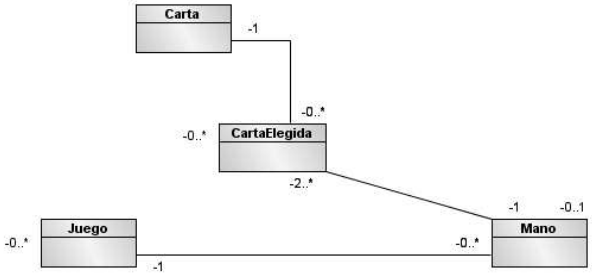


Figura 2 –Modelo Planning Poker

5. ARQUITECTURA DEL SISTEMA

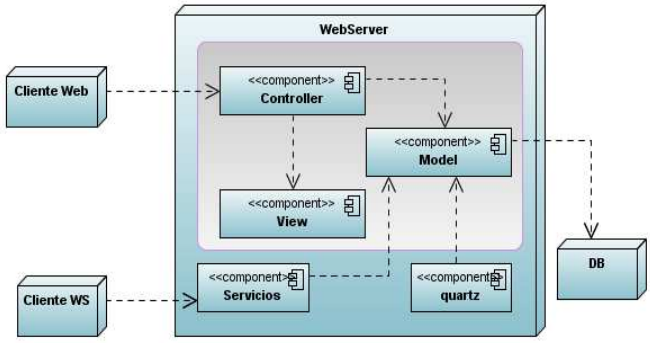


Figura 2 – Arquitectura del sistema

Grails como Framework, propone una arquitectura a utilizar para el desarrollo de los sistemas, la misma está basada en el modelo MVC (model-view-controller), esto es evidente porque uno se limita a implementar las clases del dominio (model), los controladores (controller), y las gsp que son las vistas.

Basándose en el Spring Framework, además nos brinda los servicios, que son registrados en la aplicación como Spring Beans, estos funcionan de manera muy similar a un EJB, exponen métodos que pueden ejecutarse en un contexto transaccional entre otras cosas. Este tipo de servicios sirven mucho para implementar procesos de negocio transaccionales o incluso operaciones complicadas sobre bases de datos.

Los servicios también son explotados por plugins para exponer web services, de manera muy sencilla y fiable.

Otro componente importante es el Quartz, que nos permite agendar procesos, que corren en forma paralela a los servicios proveídos vía http. Pudiendo realizar así trabajos asincrónicos, respecto a la interfaz de usuario, utilizados en nuestro caso para dar soporte al sistema de alarmas, que avisan al usuario de invitaciones a reuniones, juegos, etc.

6. IMPLEMENTACIÓN

Para implementar la aplicación utilizamos Spring Source Tool Suite STS

Para preparar el IDE, desde el dashboard se instalaron las extensiones de groovy a través de los puntos, grails tools , y grails framework.

Pudimos notar que posee una buena importación de proyectos, resuelve las dependencias de plugins automáticamente al ejecutar la aplicación siempre y cuando estén en los repositorios

En los casos que los plugins aportan librerías, los proyectos antes de instalarlos no compilan, para STS se debe de forzar la ejecución para que instale los plugins automáticamente. Para estos mismos casos luego de instalado el plugin si aún no compila el proyecto se debe de ir a botón derecho sobre el proyecto-> grails Tool -> "Refresh Dependencies"

Primer Controller, a diferencia de netbeans STS no provee una opción para la creación de un controller , por lo que o hay que crear una clase groovy o correr un comando desde la consola

para dicha creación. Lo bueno de STS para este punto es que provee una consola integrada para ejecutar los comandos de grails, a la cual se accede (en la versión 2.3.2) sobre una clase groovy con Ctrl+Shift+Alt+g lo que nos despliega la consola. La consola además tiene soporte para autocompletar con la lista de comandos, paquetes y clases (si aplica).

En particular se creó un controller incluyendo en el comando la estructura de paquetes o sea create-controller com.algo.paquete.Nombre, creado la estructura de paquetes y la clase NombreController sin inconvenientes.

Para conectarnos a la BD se agregó el driver de la base (en este caso mysql) a la carpeta lib del proyecto y se configura el archivo DataSource que provee el framework

Algo que resulto interesante es que Grails soporte especial para los campos date con los nombres dateCreated y last-Updated que solo con indicarlos en cada clase de dominio se guardan los timestamps correspondientes , lo que utilizamos para guardar dichas fechas en las tareas y de esta manera poder generar la grafica de BurnDown.

Para comenzar a desarrollar creamos las clases de nuestro dominio con el comando create-domain-class indicando com.algo.paquete.NombreDeLaClase. Luego de creadas las clases del dominio comenzamos a agregarle los atributos y las asociaciones con el resto de las clases del Dominio.

Por ejemplo:

```
class Proyecto {

          String nombre
          Date fechaIni
          Date fechaFin
          EstadoProyecto estado

          static hasMany = [sprints : Sprint, backlog: Tarea,
usuarios: UsuarioProyecto]
          static constraints = {
                    nombre(nullable:false)
                    fechaIni(nullable:false)
                    fechaFin(nullable:false)
          }

          String toString(){
                    nombre+":"+estado.nombreEstado
          }
}
```

Llendo línea por línea podemos identificar claramente el nombre de la clase “Proyecto” y los siguientes atributos nombre , fecha ini y fecha fin. Luego llega un atributo que identifica el estado del proyecto, este tipo no es una clase es un enumerado y se crea en una carpeta distinta al de las clases de dominio (por convención sobre configuración) seguido vemos las asociaciones con las clases Sprint, Tarea, UsuarioProyecto que indican que son 1 a n a travez de hasMany.
Luego definimos los contratos (constraints) que indican que características debe de cumplir cada atributo. A partir de dichos contratos se definen restricciones tanto sobre la base de datos como la generación de controles de las vistas, por lo que tomarse el trabajo de definirlos de manera completa nos evita un montón de trabajo a la hora de realizar controles por la integridad de los datos de la aplicación.
Sobre el final redefinimos el método toString al mejor estilo Java, lo que nos sirve como ejemplo de uso indiscriminado de Java sobre Groovy haciendo uso de las bondades de este último como ser los “;” y “return” opcionales.

Una vez finalizada la creación de todas las clases de dominio y sus asociaciones ejecutamos el comando automágico generate-all que para todas las clases del dominio genera los controladores y las vistas con las correspondientes navegaciones entre las asociaciones. Lo generado nos permite crear objetos editarlos , listarlos asociarlos. La pagina principal de la aplicación (recién generada) nos muestra una lista de todos los controladores de nuestra aplicación. Ingresando a través de dicha lista se puede acceder a la vista para realizar la creación listado etc de cada objeto.
Luego de toda esta generación nos ponemos a trabajar en las vistas definiendo el estilo de la aplicación, el flujo y diseño de páginas.
Para cada funcionalidad comienzan a ser necesarias ciertas tecnologías, que en la investigación inicial ya las habíamos tomado en cuenta. Por ejemplo a las Reuniones les queríamos dar la posibilidad de redactar un acta con texto rico , para ello habíamos investigado que nos proveía Grails y de las opciones nos decidimos por CKEditor. Por otra parte teníamos en vista las Fusion Charts y queríamos lograr integrarlas a una aplicación hecha en Grails ya que son componentes muy agradables y Flash. Todo el misterio resulto en

```
<%String laGrafica = FusionChartsCreator.createChartHTML(
          "/scrumMe/swf/Pie3D.swf", // tipo de grafica

"${createLink(action:'pieTareasTipo',
controller:'chart',id:proyecto.id) }", // Grails nos construye la
URL al controller que nos retorna el xml necesario para la
construccion de la Grafica !!!

"","tsi 3 2010", 300, 150, false);%> // tamaño y demas parametros
 <%= laGrafica %> // La renderizacion
```

El Xml necesario que utiliza el componente flash para presentar los datos, lo genera un controller utilizando los builders que provee Grails.

En cuanto a los WS se utilizó el plugin Xfire que permite anotar el servicio a exponer de forma sencilla.
Para la creación de Feeds se utilizo el plugin Feeds que provee builders para RSS, que permite rápidamente y de forma muy sencilla generar el xml necesario.
Para ejecutar las alarmas se utilizó el plugin de Quartz que simplifica significativamente el trabajo al utilizar esta tecnología. Provee un comando que genera los jobs, estos son una clase que hereda de Job, la cual tiene un método execute donde se pone el código a ser ejecutado, además en una forma similar a las constraints, se define el trigger, se muestra un ejemplo:

```
static triggers = {
cron name: 'myTrigger',
cronExpression: "0 0/1 * * * ?"
}
```

Por defecto este plugin se ejecuta en memoria, pero es fácilmente configurable para usar la base de datos, combinando la configuración de quartz con el sistema de propiedades de Grails.
Y así con el resto de las tecnologías que se indican en la siguiente sub sección.

6.1 Productos y Herramientas

Tabla 1. Evaluación de Plugins de Grails

Producto	Puntos Fuertes	Puntos Débiles	Evaluación General
Spring security acegi [13]	Sencillo, confiable.	Detalles en la generación de código	Funciona correctamente, provee funcionalidades como login, registro, envío de mail, maneja sesión y permisos.
XFire [11]	Sencillo, conocido.	No tiene buen soporte de documentación	Expone servicios, combina con anotaciones.
Feeds [10]	Sencillo	No se observaron	Sirve para generar feeds a través de los builders de grails
Quartz [12]	Sencillo, robusto. Disminuye el trabajo significativamente. Buena Documentación	Acoplamiento entre el trigger y el job	Se utiliza para agendar procesos.
CKEditor [3]	Nos brinda taglibs tanto para ubicar el componente como para configurarlo	Documentación básica que puede mejorar	Nos permite mediante taglibs utilizar CKEditor como un componente más.
JQueryUI [5]	Basado en JQuery	No tienen taglibs que faciliten o minimicen la utilización de Java Script. Incompleto puede mejorar	Contiene un taglib que realiza la importación de js necesarios

Tabla 2. Evaluación de librerías js

Producto	Puntos Fuertes	Puntos Débiles	Evaluación General
JQuery UI [5]	Poco código, fácil de aprender	Pocos componentes comparado	Muy bueno , nos brinda un aserie de
	, buena documentación y ejemplos	con YUI	componentes que además de ser fiables tiene una linda estética
JQuery /TimePicker [7]	De fácil integración	Utiliza componentes básicos que complican a la hora de configurar los css en la zona de las horas.	Componente que nos permite seleccionar una fecha y hora la cual se carga en un campo de texto.
JQuery /Round About [6]	Fácil de usar buena documentación.	Mala estética por defecto, se necesita saber mucho de configuración css para mejorarla.	Componente que hace de Carrusel que utilizamos para mostrar los proyectos.
CKEditor [3]	Completo, permite configurar los elementos y su distribución en la barra de herramientas	El soporte para el guardado de archivos no nos funcionó	Componente Js que nos permite editar textos ricos

Tabla 3. Evaluación de Otros Productos y Herramientas

Producto	Puntos Fuertes	Puntos Débiles	Evaluación General
Fusioncharts [4]	Muy integrable, muy estética por defecto, lo mejor que hemos visto en lo que respecta a generación de graficas	El browser cliente debe de tener instalado Flash, pero hoy en día no se si debe de considerar una debilidad.	Librería Flash que nos permite renderizar graficas consumiendo su configuración a través de xml.
Spring Source Tool Suite – STS [9]	Gratis, completo, basado en Eclipse. Debugger para Grails que funciona. Los actuales dueños de Grails son los creadores de STS	Algunos Bugs. El soporte para Grails no es tan bueno como el de NetBeans o IntelliJ	IDE aceptable para el desarrollo de Proyectos en Grails que promete mejorar a futuro. Para quienes prefieren Eclipse no

	SpringSource		hay otra opción
Wireless Tool Kit [8]	Muy útil, facilita el trabajo	No se encontraron	Buen set de herramientas para el desarrollo de J"ME , se utilizo para generar los stubs que consumen el Ws que lista los proyectos y login

6.2 Problemas Encontrados

STS

Genera los imports sobre la declaración de package, por lo que da error de compilación. Se soluciona reubicando los imports

Grails

En cuanto a las Imágenes como opciones podemos guardar las imágenes en un archivo y en la base de datos.

Lo correspondiente e al upload Grails lo genera automáticamente si la imagen es un atributo de la clase de dominio pero no así la renderización. Ese atributo es de tipo byte[] por lo que grails no sabe exactamente que es, por lo que puede generar el uplodad (como cualquier archivo)

Para el caso de que la imagen sea un atributo del dominio por lo que se guardara en la base de datos, surge la complicación de que hay que traer a dicha imagen desde la acción de un controlador generando de esta forma una llamada extra a la BD. Además por seguridad hay que desarrollar algún tipo de sistema que nos permita ocultar la url a la imagen, puesto que si la misma queda expuesta se podría sufrir un denial of service (en posible ataque). En definitiva Grails no nos provee por defecto un manejo seguro y homogéneo para el manejo de imágenes.

En cuanto a Hostings no son económicos, para poder hostear una aplicación mediana que utilice JEE generalmente se necesita hosteo dedicado.

Manejo de Plugins es básico en los siguientes aspectos:

- Plugins que dependen de plugins, se anidan, aunque se obliga al proyecto a instalar todas las dependencias, lo cual genera un aumento desmedido del tamaño del proyecto con un montón de archivos inútiles.

- Los plugins deben volver a instalarse si un desarrollador descarga un proyecto desde el svn o se hace un upgrade de la versión de Grails.

En cuánto a los Builders no se puede usar set como tag del xml en los builder. No interpreta correctamente el set de g:set. Para el caso particular de la integración con FusionChart se utilizó Set que funciona y Fusión lo interpreta.

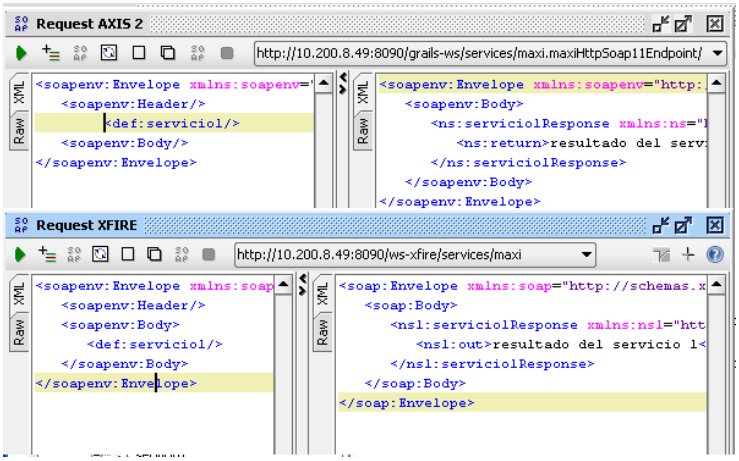
WS

Probando un cliente que usa groovyws, contra un servidor xfire, funciona sin problemas pero contra un servidor axis2 se encontraron los siguientes problemas:

- el servidor levantado con axis2 en localhost, toma como ip de los endpoint, la ip de otras interfaces de red en windows, en nuestro caso por ejemplo tenía 2 generadas por vmware, en otra sub red, y esto provoca que luego el cliente no se pueda conectar, lo que no ocurre con xfire, y aunque suceda este comportamiento soap ui funciona correctamente.

-Ver en la siguiente imagen que el xml generado por ambos sistemas es diferente para la misma operación, la de xfire es interpretada correctamente por el cliente de GroovyWS, devolviendo el resultado de la operación.

En el caso de Axis el cliente retorna un objeto de clase:org.apache.ws.axis2.Servicio1Response, que en el atributo: return tiene un objeto de clase:javax.xml.bind.JAXBElement, el cual tiene el resultado de la operación en el atributo value.



JPICKER

Color picker en javascript, se descarto puesto que los css no compatibilizaron con los css del template de la aplicación por lo que verificamos que la librería no cumple con estándares de desarrollo web a nivel de css.

Por tal motivo eliminamos la funcionalidad de seleccionar los colores de las tareas desde un color picker (JPICKER)

JQUERY UI

No soporta Datetime picker, Solo Date picker, pero utilizamos http://addyosmani.com/blog/the-missing-date-time-selector-for-jquery-ui/ que lo agrega al componente de jquery-ui

Grails UI

Extiende de YUI para brindar una muy bien lograda interfaz de usuario basada en javascript.

Se descarto porque nos pareció que Jquery-ui era más fácil de utilizar.

JSecurity

Muy completo y flexible, pero complicado de utilizar, solo provee manejo de usuarios, roles y permisos.

Basic security(gsec)

Implementación propia de las funcionalidades de seguridad, le falta mucho para ser útil, y no es muy confiable.

Stark

Basado en Spring security, no requiere base de datos para guardar los permisos, sino que se hace en el código o por configuración, tiene problemas en la instalación. No funcionó correctamente

Apache Shiro

Muy potente y flexible, pero los permisos deben ser implementados por la aplicación, ya que son expresados en conceptos de esta.

WS JSR 311

Soporte para ws rest, permite combinar Grails con el api de java para ws sobre rest

Apache Axis2

Muy fácil de usar, falta investigar que tan adaptable y extensible es. Solo expone servicios

Spring ws

Consume y expone, requiere contratos en xml

CXF

Expone WS formato SOAP, muy sencillo, combina con anotaciones para mayor control sobre el WSDL generado.

Remoting

Expone servicios en diferentes protocolos como ser RMI por ejemplo

Simple Blog

Además de RSS, también tiene creado de post, tagging, comentarios

Charts

Eastwood servlet basado en jfreechart que emula a googlechart.

Provee solo imágenes estáticas nos pareció que Fusión es mas interesante y con mejor out of box. Además para no exponer el servicio y evitar así por ejemplo un Denial-of-service hay que consumir las imágenes desde la misma aplicación para luego enviarla al browser por lo que implica más trabajo a la hora de implementar.

J2ME

Los tipos de datos en J2ME son muy limitados y no soportan todos los tipos de datos existentes en Java. Por Ejemplo java.util.Map los cual nos obligo a simplificar los WS ofrecidos.

7. EVALUACIÓN DE LA SOLUCIÓN

La aplicación provee interesantes funcionalidades las cuales se podrían mejorar en cantidad y usabilidad.

Hay más de una forma de acceder a ciertos puntos por lo que no la hace tan clara como pretendíamos a la hora de utilizar.

Los accesos son ilimitados o sea que sin importar el rol se puede acceder a cualquier información y modificar lo que se desee. No es casualidad, creemos que de esa forma se respeta parte de filosofía de Scrum en cuanto a eliminación de poderes dentro del grupo de desarrollo y fomenta la auto-organización.

La estética no fue la esperada aunque de cierta manera nos conforma el utilizar colores distintos a las “típicas” aplicaciones, pretendiendo de esta manera que el usuario pueda disfrutar de utilizar la aplicación.

En cuanto a la implementación de algunas funcionalidades podría ser conveniente medir el consumo de memoria cpu y bd con anterioridad a una eventual puesta en producción ya que estos temas no se midieron ni fueron previstos.

8. DESARROLLO DEL PROYECTO

En el inicio del proyecto se dedicaron más que las 8 hs planteadas por la materia hasta la primera presentación en la cual se tenía desarrollado el 40 % del sistema. Durante la segunda parte se llego a trabajar no más de las 8 hs propuestas logrando así el desarrollo del otro 60 %. Esto fue gracias a una buena investigación en la primera etapa evitando agregar nuevas tecnologías y componentes para no tener desviaciones, aunque nos vimos tentados en más de una ocasión probar algún que otro componente o tratar de agregar alguna funcionalidad para aplicar alguna tecnología más.

Cabe destacar que los tiempos de implementación estuvieron por debajo de lo esperado gracias a las bondades del framework Grails.

9. CONCLUSIONES Y TRABAJO A FUTURO

9.1 Conclusiones

Grails nos provee más de lo necesario para realizar aplicaciones web. Es muy ágil desarrollar en este framework y el hecho de poder embeber java indiscriminadamente nos permite ir avanzando en el conocimiento de groovy a demanda. Descubriendo las bondades de Groovy paulatinamente, evitando de esta manera perder ritmo en el desarrollo.

Además del buen out the box que provee el sistema de plugins es muy bueno y además en las nuevas versiones están mejorando su manejo. Es interesante tener más de un plugin sobre el mismo tema lo que nos permite decidir sobre la tecnología que nos resulte mas cómoda para realizar esa funcionalidad especifica.

9.2 Trabajo a Futuro

Dash Board

Ajusta las tareas a travez de posiciones absolutas quedando fuera del dash board en ciertas resoluciones de pantalla.

J2ME

Si bien se llego a consumir un servicio de la aplicación como ser el listado de proyectos y además utilizando Login. Faltaría agregar una serie de servicios que permita por ejemplo consultar un resumen de lo que sucede para un proyecto específico o revisar mensajes, alarmas e incluso poder cambiar una tarea de estado.

Charts

Faltaría agregar Graficas como ser cantidad de tareas por desarrollador, cantidad de horas por desarrollador. BurnDown combinando tipos de tareas.

Falto desarrollar un versionado de entregables para dar soporte a los Potentially Shippable.

Reportes

Uno de los puntos que pretendimos abordar eran los Reportes. Si bien son necesarios en casi cualquier aplicación, tuvimos que descartarlo para dedicarle más tiempo a otras funcionalidades que nos resultaron más interesantes. La siguiente es una lista de plugins que fueron investigados

Jasper

Basado en jasper reports, fácil de usar y de extender pero no soporta jasper reports en su totalidad. Por ejemplo no tiene soporte para subreportes

Export

Utiliza librerías gratuitas y conocidas como poi e iText, también es fácil de usar

Birt

Basado en Birt para generar reportes. Existe la herramienta Eclipse BIRT Report Designer para su diseño

Testing

No se utilizo ninguna herramienta de Testing aunque se estuvieron evaluando algunas de ellas no nos fue posible por temas de tiempo aplicarlas. He aquí una lista de los distinto plugins que estuvimos investigando.

Auto test

Ejecuta automáticamente los tests definidos, con cada cambio en los archivos fuentes, de esta forma agiliza el lanzado de tests unitarios y de integración

Selenium

Testing funcional, complementado con firefox, es muy buena herramienta

Test templates

Genera templates más completos para unit tests

jsUnit

Permite integrar el framework de testing jsUnit en Grails, para probar javascript de forma muy similar a jUnit en Java.

dbUnit

Basado en junit permite crear varios test para cada configuración del datasource

Build test data

Permite crear datos de prueba basándose en las clases de dominio, respetando las constraints definidas

Test code coverage

Testing por cobertura de código usando Cobertura

Webdriver func testing

Integra Webdriver para testing funcional

Canoo web test

Testing funcional usando Canoo

Clover

Integra clover que es una herramienta para testing por cobertura de código

Spock

Integra spock en Grails, para test unitarios, sintaxis muy sencilla para desarrollar tests

10. REFERENCIAS

[1] http://www.grails.org/

[2] http://blog.springsource.com/2010/06/01/whats-a-plugin-oriented-architecture/?utm_source=feedburner&utm_medium=feed&utm_campaign=Feed%3A+Interface21TeamBlog+%28SpringSource+Team+Blog%29

[3] http://ckeditor.com/ , http://www.grails.org/plugin/ckeditor

[4] http://www.fusioncharts.com

[5] http://jqueryui.com/ , http://www.grails.org/plugin/jquery-ui

[6] http://fredhq.com/projects/roundabout/

[7] http://jonathonhill.net/2009-03-27/jquery-datetime-picker/

[8] http://java.sun.com/products/sjwtoolkit/

[9] http://www.springsource.com/developer/sts

[10] http://www.grails.org/Feeds+Plugin

[11] http://www.grails.org/XFire+plugin

[12] http://www.grails.org/plugin/quartz

[13] http://grails.org/plugin/acegi

[14] http://www.scrumalliance.org/