Capítulo III. Metodología y Requerimientos

En este capítulo se analizan, en primer lugar, las distintas tecnologías relacionadas con el desarrollo de este proyecto, incluyendo los estándares de empaquetado y de manejo y de metadatos. También se hace un análisis de la aplicación de distintas tecnologías de voz en el desarrollo de contenido educativo y se elige una para desarrollar un sub-módulo del sistema. Después se analizan las arquitecturas de las diferentes opciones de implementación para los distintos módulos del software desarrollado, tomando en cuenta la importancia de la disponibilidad de la aplicación a través de Internet como requerimiento imperativo.

3.1. Requerimientos tecnológicos

3.1.1. SCORM: El estándar principal usado

El estándar elegido para este proyecto es SCORM, debido a que es un estándar abierto, enfocado al área de la educación, y formado a partir de otros estándares y recomendaciones. Este estándar, en su libro *Content Aggregation Model* (CAM), disponible en su página Web, establece los lineamientos para la construcción de recursos educativos reutilizables e interoperables. Entre otras cosas, este libro establece las características de las unidades de contenido educativo a desarrollar (*SCORM Content Model*), sugiriendo el seguimiento del estándar IMS *Content Packaging* para el empaquetado de recursos, y del estándar IEEE LOM para el etiquetado o descripción con metadatos de los recursos desarrollados. A continuación se examinan con más detalle estos requerimientos.

3.1.1.1. SCORM Content Model: unidades de contenido educativo

El *SCORM Content Model* describe a los distintos tipos de componentes usados para desarrollar experiencias de aprendizaje, y cómo estos componentes pueden formar unidades más grandes y complejas de contenido educativo. Los componentes en torno a los cuales se desarrollan contenidos, de acuerdo a este estándar, son los que se describen a continuación [ADL, 2004].

Recurso educativo

Los recursos son los materiales didácticos en cuestión que, de acuerdo a este estándar, son digitales. Hay dos tipos de recursos educativos, los *assets* y los *shareable content objects*.

Asset

Constituyen la forma más básica de recurso educativo. Son archivos de contenido en sí, como archivos de texto, imágenes, audio, etc. Varios de estos componentes se pueden unir para formar otro del mismo tipo, como por ejemplo, una página Web puede requerir de más que sólo un archivo HTML. Por estas características, los *asset* pueden ser vistos como objetos de aprendizaje fundamentales o combinados cerrados (ver sección 2.2.2).

Shareable Content Object (SCO)

Este tipo de componente está formado por uno o varios *assets*, y representa el nivel más bajo de granularidad de un recurso educativo capaz de comunicarse con un LMS. Por estas características, este tipo de componentes pueden ser vistos como objetos de aprendizaje combinados cerrados (ver sección 2.2.2).

Organización de Contenido *(Content Organization)*

Este tipo de componente consta de un mapa jerárquico que representa el uso deseado de los recursos educativos a través de unidades estructuradas de instrucción o actividades. Ejemplos de organizaciones de contenido pueden ser los temarios de cursos o los planes de estudio. Por estas características, las organizaciones de contenido son objetos de aprendizaje combinados abiertos (ver sección 2.2.2).

Actividades *(items)*

Las actividades son los nodos de las jerarquías que conforman las organizaciones de contenido. Pueden verse como los temas en un temario de un curso, o las materias en un plan de estudios. Cada actividad puede constar de otras actividades, que se pueden anidar indefinidamente, hasta que una de ellas tenga un recurso asociado, sea un *Asset* o un *SCO*. Por estas características, las actividades pueden ser vistas como son objetos de aprendizaje combinados cerrados (ver sección 2.2.2).

Paquetes de contenido educativo *(Content Package)*

Son componentes que constan de uno o varios recursos educativos y, opcionalmente, distintas organizaciones de contenido y metadatos para todos los niveles de granularidad. Los paquetes de contenido educativo pueden ser vistos como objetos de aprendizaje combinados abiertos (ver sección 2.2.2). En este trabajo se consideran sinónimos los términos “objeto de aprendizaje” y “paquete de contenido educativo”, tal como lo definen los estándares IMS CP y SCORM CAM.

3.1.1.2. IMS Content Packaging: empaquetado de recursos

El propósito del empaquetado de recursos educativos es el de estructurar y permitir el intercambio del contenido con otros sistemas. En el Anexo A y en capítulos anteriores se describió el contenido de los paquetes de contenido educativo de IMS. Sólo resta mencionar los requerimientos técnicos más importantes marcados por este estándar y por SCORM [ADL, 2004]:

- El manifiesto debe ser un archivo XML llamado `imsmanifest.xml`
- El paquete puede o no incluir una o varias organizaciones de contenido *(content organization)*, especificadas en forma de metadatos incluidos en el archivo de manifiesto. Estas organizaciones de contenido agrupan los recursos del paquete en unidades de instrucción o actividades.
- El paquete puede o no incluir información acerca de secuenciación y navegación (*sequencing and navigation*) asociada al contenido educativo, incluida en forma de metadatos en el archivo de manifiesto. Esta información indica a los LMS's a qué recursos del paquete pueden tener acceso sus usuarios, en qué orden y en qué momento.
- Los recursos educativos incluidos en el paquete pueden o no estar descritos por metadatos, pudiendo estas descripciones ser parte del archivo de manifiesto principal o ser archivos separados, en cualquier estándar de metadatos, preferiblemente LOM.
- Un recurso educativo puede estar formado por uno o varios archivos, *assets* o *shareable content objects*. Los archivos físicos de contenido también pueden o no estar descritos por metadatos.

- Todo el paquete, incluyendo archivos de metadatos y archivos de contenido, deben ser reunidos en lo que se denomina un archivo de intercambio de paquetes o PIF (*Package Interchange File*), que debe ser un archivo en formato ZIP.

Estos requerimientos están especificados con más detalle en el libro *SCORM Content Aggregation Model (CAM) Book* [ADL, 2004], que dedica una sección importante a las características técnicas recomendadas para la construcción de paquetes de contenido educativo, como es la descripción de las etiquetas del manifiesto, *namespaces* y otras cosas. Esta sección es una referencia importante para el desarrollo del software de este proyecto.

Para este proyecto se realizarán paquetes de contenido educativo con información sobre organizaciones de contenido, y con metadatos para la descripción del paquete como un todo, así como a nivel de archivos físicos, recursos, actividades y organizaciones de contenido. La generación de valores de metadatos será sólo a nivel de archivos físicos.

Queda fuera del alcance de este proyecto la inclusión de información sobre secuenciación y navegación, debido a que el manejo de organizaciones de contenido constituye en sí misma una forma de navegación y secuenciación utilizada por defecto en LMS's basados en SCORM [ADL, 2004].

3.1.1.3. IEEE LOM: manejo de metadatos

El estándar de metadatos para recursos educativos adoptado por SCORM es, como se había comentado anteriormente, IEEE LOM. Este estándar es usado para describir cada tipo de componente del *Content Model* de SCORM, desde los *Assets* hasta los tipos más complejos. El total de etiquetas que proporciona este estándar para la descripción de recursos educativos es 64, y aun así está abierto a extensiones [ADL, 2004]. En el CAM Book de SCORM se especifica en detalle cómo describir a cada tipo de componente y en qué parte del paquete de contenido educativo debe estar su archivo de descripción.

Es importante para este proyecto analizar la definición de metadatos de LOM para determinar qué valores pueden generarse programáticamente a nivel de archivos físicos. En primer lugar es importante distinguir aquellas categorías que, por la naturaleza no técnica o pedagógica de sus valores, es conveniente que se llenen a mano. La tabla 3.1 expone para cada categoría de metadatos de LOM una descripción, su naturaleza y sus opciones de tipo de generación de metadatos.

Categoría	**Descripción**	**Naturaleza**	**Tipo de Generación de metadatos**
<general>	Información general que describe al recurso como un todo.	Pedagógica Técnica	Manual Programática
<lifeCycle>	Historia y el estado actual del recurso.	Técnica	Manual
<metaMetadata>	Información del estándar de metadatos usado para describir al recurso.	Técnica	Programática
<technical>	Información de las características y requerimientos técnicos del recurso	Técnica	Manual Programática
<educational>	Características clave del recurso de tipo pedagógico o educacional.	Pedagógica	Manual

<rights>	Describe los derechos de propiedad y condiciones de uso del recurso.	Pedagógica Técnica	Manual Programática
<relation>	Describe la relación del recurso con otros recursos.	Pedagógica	Manual Programática
<annotation>	Información sobre comentarios hechos acerca del recurso.	Pedagógica	Manual
<classification>	Indica en qué categoría cae el recurso en base a determinado sistema de clasificación.	Pedagógica	Manual

Tabla 3.1. Análisis de las categorías de LOM

De este análisis surgen las categorías sobre las que se a trabajó en la generación de valores de metadatos: <general>, <metaMetadata>, <rights>, <relation> y <technical>. A continuación se presenta para cada categoría una tabla con el nombre de los elementos cuyo valor se puede generar automáticamente, la forma de obtener dicho valor, y si se implementó como parte de este proyecto la generación de dichos valores de metadatos.

Categoría <general>

Elemento	**Forma de obtención**	**Implementación**
<title>	A partir de títulos ingresados en el módulo central.	X
<language>	Español por defecto.	X
<keyword>	Mediante el uso de indexadores y buscadores de texto.	X
<structure>	Para archivos físicos el valor será "*atomic*", para *assets*, *SCO's* y paquetes será "*networked*" y para organizaciones de contenido será "*hierarchical*".	✓
<aggregationLevel>	"1": archivos físicos, *assets* y *SCO's* "2": actividades dentro organizaciones de contenido "3": organizaciones de contenido como un todo "4": paquete de contenido como un todo	✓

Tabla 3.2. Generación de metadatos de la categoría <general>

Categoría <metaMetadata>

Elemento	Forma de obtención	Implementación
<identifier>	Se usarán los definidos por el usuario en el módulo de empaquetado.	✓
<contribute>	Se incluirá el nombre del software desarrollado y se establecerá con el rol de creador de metadatos.	✓
<metadataSchema>	El valor especificado por el estándar	✓
<language>	El valor será "*sp*" por defecto (español), con la opción a cambiarlo.	✓

Tabla 3.3. Generación de metadatos de la categoría <metaMetadata>

Categoría <rights>

Elemento	Forma de obtención	Implementación
<cost>	Por defecto se asumirá que no tiene costo el uso de recursos educativos.	✓
<copyRightAndOtherRestrictions>	Por defecto se asumirá que los recursos sí tienen este tipo de restricciones.	✓

Tabla 3.5. Generación de metadatos de la categoría <rights>

Categoría <relation>

Elemento	Forma de obtención	Implementación
<kind>	"*ispartof*": valor para archivos físicos, que generalmente son parte de *assets* y *SCO's*. "*haspart*": valor para *assets* y *SCO's*, que generalmente están formados de otros *assets* y archivos físicos.	✓
<resource>	Obtenido a partir de la información generada en el módulo principal del sistema desarrollado. Constituye el recurso con el que se hace la relación.	✓

Tabla 3.6. Generación de metadatos de la categoría <relation>

Categoría <technical>

Elemento	Forma de obtención	Implementación
<format>	*MIME type* del archivo físico, obtenido a partir de métodos de la API del lenguaje de programación a usar.	✓
<size>	Tamaño en bytes del archivo físico, obtenido a partir de métodos de la API del lenguaje de programación a usar	✓
<location>	Ruta donde se encuentra el recurso. Valor obtenido del módulo principal del sistema desarrollado.	✓
<duration>	Duración de archivos de audio, obtenida a través de métodos de la API del lenguaje de programación a usar.	✓
<requirement>	Se añadirá por defecto el requerimiento de contar con una versión mínima de navegador (Explorer 5, Netscape 4.7), con posibilidad de edición. Además el valor por defecto para sistema operativo será "*multi-os*".	X
<instalationRemarks>	Se analizará la extensión del archivo para determinar si es necesaria la instalación se software adicional, como *plug-ins*.	X
<otherPlatformRequirements>	Se indicará la necesidad de contar con tarjeta de sonido en caso de tratarse de un archivo de audio. En caso de ser una imagen, se analizará su tamaño y composición para recomendar una resolución de pantalla.	X

Tabla 3.4. Generación de metadatos de la categoría <technical>

3.1.2. Uso de tecnologías de voz

En esta sección se hace un análisis del uso de las distintas tecnologías de voz aplicables en el contexto educativo, con la finalidad de decidir cuál de ellas incluir dentro del presente proyecto. En el Anexo B de este documento se incluye un panorama general de las tecnologías de voz, presentando información general sobre grabación, codificación, síntesis y reconocimiento de voz, así como procesamiento de lenguaje natural. La tabla 3.5 muestra el análisis comparativo de las características de estas tecnologías.

Los parámetros a considerar en este análisis son los siguientes:

- Disponibilidad de herramientas - indica si hay herramientas disponibles de determinada tecnología de voz que puedan usarse para el desarrollo de sistemas de contenido educativo.
- Calidad - este parámetro indica si determinada tecnología funciona de manera adecuada o si todavía no se perfecciona lo suficiente, llega a fallar o a funcionar de manera errónea.
- Requerimientos sencillos - este parámetro indica si una tecnología no requiere de recursos adicionales a los comunes, como micrófono y bocinas, ya sea de software o hardware, para poder ser usada.
- Facilidad de desarrollo - determina qué tan complejo es el uso de determinada tecnología desde el punto de vista del desarrollador de software
- Facilidad de uso - indica si el uso de la tecnología resulta sencillo desde el punto de vista del usuario final del sistema.

- Aplicabilidad - este parámetro indica si determinada tecnología es implementable en objetos de aprendizaje sin necesitar de ambientes especiales adicionales.

Parámetro	**Grabación de voz**	**Síntesis de voz**	**Reconocimiento de voz**	**Procesamiento de Lenguaje Natural**
Disponibilidad de herramientas	✓	✓	✓	X
Calidad	✓	✓	✓	✓
Requerimientos sencillos	✓	✓	X	X
Facilidad de desarrollo	X	X	X	X
Facilidad de uso	✓	✓	X	✓
Aplicabilidad	✓	X	X	X

Tabla 3.5. Análisis de las características de distintas tecnologías de voz

Una vez hecho este análisis, se elige a la grabación de voz como el tipo de tecnología a implementar. Una segunda opción era la síntesis de voz, pero se descartó por dos razones principales:

- En primer lugar, era necesario contar no sólo con un sistema de creación y empaquetado de objetos de aprendizaje, sino también con un ambiente de ejecución de objetos de aprendizaje, de tal forma que en ambas herramientas se implemente la configuración de las herramientas de síntesis de voz.
- En segundo lugar, la calidad de la síntesis de voz es menor a la de la obtenible por medio de grabaciones de voz, y en el contexto educativo es más deseable una mejor calidad de voz que un sistema que genere voz a partir de texto.

3.2. Las opciones de implementación

En esta sección se describen de manera general las tres opciones distintas de implementación para los tres módulos principales del sistema desarrollado, tomando en cuenta la importancia de la disponibilidad del mismo a través de Internet. Para cada opción se analizan las diferencias en la arquitectura, y sus ventajas y desventajas.

3.2.1. Implementación como un *applet*

Como es ampliamente conocido, un *applet* es un subprograma que se carga y se ejecuta en un Navegador Web como parte de una página HTML [Ceballos, 2000]. El Servidor Web donde se encuentra la página que contiene al *applet*, contiene también los archivos que necesita el *applet* para ejecutarse, ya sean JAR o CLASS. El Navegador descarga estos archivos e inicia la ejecución del *applet*, que se lleva a cabo en su totalidad del lado del cliente, como se puede observar en la figura 3.1.

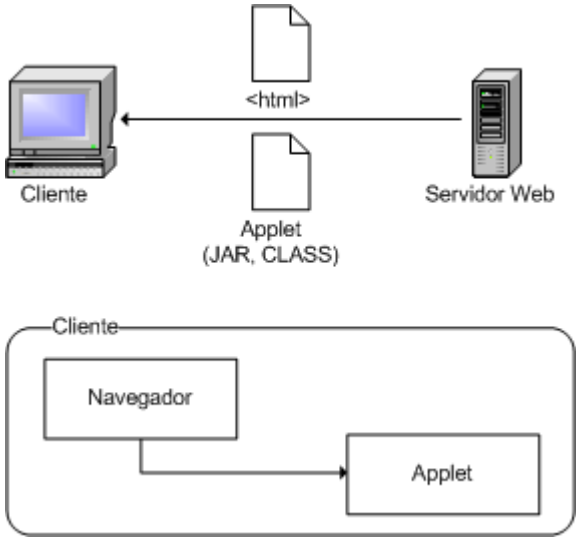

Figura 3.1. Arquitectura de un sistema con *applets*

Desarrollar *applets* tiene ventajas importantes. Se pueden crear interfaces gráficas de usuario muy sofisticadas gracias a que esta tecnología soporta el uso de paquetes como *AWT* y *Swing*. Además, los *applets* permiten mostrar imágenes, reproducir sonidos y desarrollar animaciones de una manera relativamente sencilla.

Sin embargo, esta opción tiene desventajas importantes. La primera de ellas es el desempeño. Los *applets* tienden a ser programas de lenta ejecución, sobre todo si se trata de aplicaciones grandes o que manejan muchos recursos, o recursos muy pesados. La segunda ventaja es la de la seguridad. Los *applets* tienen restricciones de seguridad importantes directamente relacionadas con este proyecto, entre las que se encuentran el no poder leer y escribir archivos en el sistema de archivos del cliente [Ceballos, 2000].

3.2.2. Hacer un portal basado en *Servlets* y JPS

Un segundo escenario de implementación es desarrollar un portal en el que el procesamiento se lleve a cabo del lado del servidor, gracias a tecnologías como *Servlets* y JSP. De esta manera, el usuario cargaría en el Servidor Web sus archivos de contenido. El servidor se encargaría de analizarlos, generar sus metadatos, empaquetarlos, y entregar al cliente el objeto de aprendizaje o paquete generado.

Es importante recordar que un *Servlet* es un programa que acepta peticiones de un cliente, procesa la información en las peticiones, y devuelve una respuesta al cliente que puede ser de distintos tipos de contenido, como texto, HTML, *applets*, etc., todo esto bajo el protocolo HTTP [Ceballos, 2000].

Asimismo, JSP (*Java Server Pages*) es una tecnología en la que se incluyen *scripts* de Java en códigos HTML. Estos *scripts* son compilados de manera automática la primera vez que el navegador del cliente solicita la página que los contiene, siendo el resultado de esta compilación un *Servlet*. La figura 3.2 muestra la arquitectura de los sistemas basados en *Servlets* y JSP.

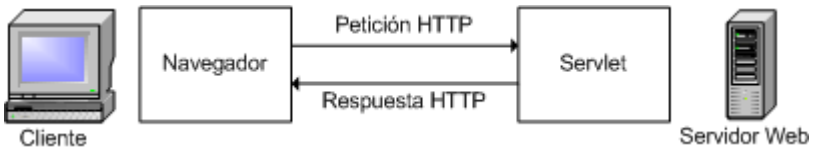

Figura 3.2. Arquitectura de portales basados en *Servlets* y JSP

Trabajar con estas tecnologías tiene ciertas ventajas. En primer lugar, generalmente su ejecución es más rápida que la de otras tecnologías, como la de los *applets*. Los *Servlets* y las páginas JSP se cargan la primera vez que se solicitan, las siguientes peticiones crean hilos de ejecución [Ceballos, 2000]. Además, sólo se requiere soporte de Java del lado del servidor. Del lado del cliente no es necesario, debido a que los *servlets* y las páginas JSP se ejecutan en su totalidad del lado del servidor. Otra ventaja importante está en que, en el caso del módulo de metadatos, se podría sacar mucho provecho de las hojas de transformación para crear formas de llenado para metadatos, mismas que se podrían actualizar de una manera relativamente sencilla en caso de que los estándares cambien.

Pero a pesar de las ventajas anteriores, estas tecnologías tienen desventajas importantes en el contexto de este proyecto. En primer lugar, es importante considerar el riesgo de sobrecarga del servidor, sobre todo tomando en cuenta que varios usuarios podrían hacer uso del sistema de manera simultánea.

Además de lo anterior, está la desventaja del tiempo de espera para el usuario. Los archivos de contenido que constituyen la entrada al sistema desarrollado pueden ser de distintos tamaños. Cargarlos al Servidor Web para que puedan ser procesados tiene la desventaja del tiempo invertido, que puede ser variable dependiendo del tamaño de los contenidos a cargar y de la velocidad de la conexión a Internet del cliente.

3.2.3. Implementación con *Java Web Start*

Java Web Start es una tecnología más reciente que los *applets*, los *Servlets* y las páginas JSP, basada en el protocolo JNLP (*Java Network Launching Protocol*). En este caso, los archivos de la aplicación a ejecutar se encuentran empaquetados inicialmente en un conjunto de archivos JAR en un Servidor Web, el cual también proporciona una página Web en la que se incluye un vínculo para iniciar la ejecución de la aplicación.

En primer lugar, el cliente debe hacer clic en este link, que apunta a un archivo especial en el servidor (con extensión JNLP, pero escrito en XML), que contiene la información de la aplicación a ejecutar.

Una vez hecho lo anterior, el navegador ejecuta *Java Web Start*, que automáticamente descarga y ejecuta la aplicación especificada del lado del cliente. Este proceso se lleva a cabo generalmente sin requerir intervención adicional al clic inicial por parte del usuario [Vallejo, 2004]. La figura 3.3 muestra este proceso.

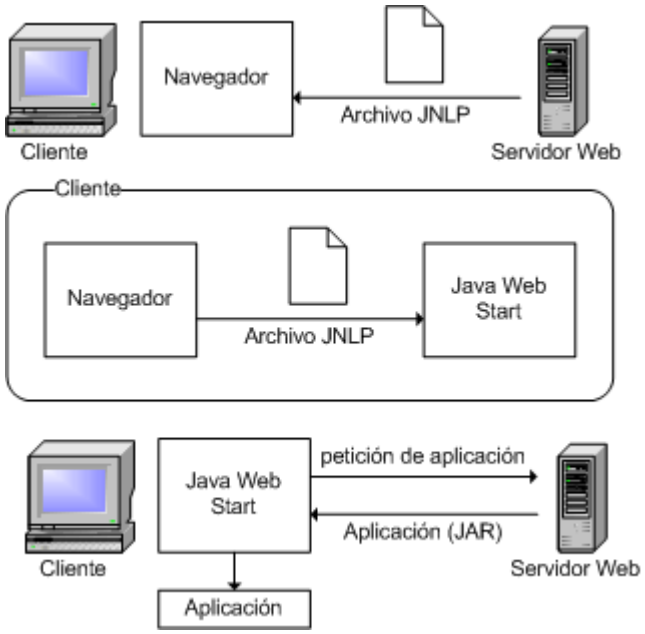

Figura 3.3. Java Web Start

Java Web Start puede verse como una tecnología muy parecida a los *Applets* en muchos sentidos, pero tiene ventajas importantes. Una de ellas es que permite la ejecución de aplicaciones independiente del Navegador y aun sin conexión a Internet, lo cual ofrece un mejor desempeño. Al ser aplicaciones Java, también se cuenta con soporte para la creación de interfaces gráficas de usuario usando AWT o Swing. Pero la característica más importante de esta tecnología en el contexto de este proyecto es que permite leer y escribir archivos del lado del cliente modificando las restricciones de seguridad con las que está configurada esta tecnología por defecto.

No se encuentran desventajas significativas en esta tecnología que pudieran afectar directamente el presente proyecto. Del lado del servidor, lo único que hay que hacer es configurarlo para soportar un nuevo tipo de archivo *MIME type*, que es JNLP. Del lado del cliente sólo es necesario contar con una versión reciente de Java RTE.

3.2.4. La opción elegida

Una vez analizadas las tres opciones de implementación más importantes, es claro que la opción más conveniente es *Java Web Start*. Esta tecnología, al ser la que facilita más el acceso al sistema de archivos del cliente, es la más adecuada para la generación de metadatos y el empaquetado de archivos. Al realizarse estas tareas del lado del cliente, la aplicación está disponible desde Internet sin correr el riesgo de tener una sobrecarga del lado del servidor con el aumento del número de usuarios.

La tabla 3.6 muestra un resumen de las características estudiadas de las distintas opciones de implementación.

	Applets	**Servlet's & JSP's**	**Java Web Start**
Facilidad de desarrollo	✓	X	✓
Ejecución local	✓	X	✓
Desempeño	X	✓	✓
Diversidad de GUI's	✓	X	✓
Seguridad configurable	X	X	✓

Tabla 3.6. Comparación de las opciones de implementación

Los parámetros en base a los cuales se comparan estas opciones son los siguientes:

- Facilidad de desarrollo - este parámetro indica la facilidad de uso de la tecnología desde el punto de vista del desarrollador. Es preferible usar tecnologías sencillas que permitan concentrarse más en el proyecto que en detalles técnicos.

- Ejecución local - este parámetro indica si la aplicación se ejecuta del lado del cliente, y por lo tanto, si éste requiere soporte de Java. Si la aplicación se ejecuta del lado del cliente, constituye una ventaja, ya que se evitan cuellos de botella y no se tiene que manejar concurrencia.
- Desempeño - indica si la velocidad de ejecución de las aplicaciones basadas en cierta tecnología es aceptable. Lógicamente se busca un buen desempeño.
- Diversidad de GUI's - indica si la tecnología permite la creación de GUI's elaboradas a partir de una gran diversidad de componentes gráficos. Para el proyecto, es preferible contar con una tecnología que permita crear GUI's a partir de elementos como árboles.
- Seguridad configurable - este parámetro indica si la tecnología permite configurar el acceso al sistema de archivos del cliente, característica ideal para la generación de metadatos.