GUÍA DE PRODUCCIÓN

Guía de producción y gestión de contenido digital para las XO's

Rev 0

La presente guía es el punto de entrada para los desarrolladores de cualquier tipo de contenido digital a ser consumido por usuarios de la XO. Aquí se resumen los pasos generales que deben seguir los desarrolladores para producir contenido compatible con los requerimientos del Plan Ceibal. La guía contiene 5 secciones: "Catalogación" describe los datos que debe proveer el desarrollador para poder catalogar adecuadamente el contenido; "Calidad" presenta un conjunto de reglas que deben seguir los desarrolladores para lograr una calidad adecuada del producto en el contexto de Ceibal; "Requerimientos de contenido para XO" describe los requerimientos generales que debe cumplir cualquier contenido para XO; "Sitios de juegos" da una serie de requerimientos mínimos para implementar sitios de juegos compatibles con el Motor de juegos. Por último "Entregables" da una lista de los elementos que debe proveer el desarrollador al entregar el contenido a Ceibal.

Ing. Eduardo Pérez Rico

Mayo de 2009

Contenido

Revisiones

Revisión	Fecha	Modificaciones
Rev 0	26-05-2009	Primera versión del documento.

CATALOGACIÓN

Introducción

El desarrollador deberá proveer una ficha de contenidos describiendo cada objeto a ser catalogado. El formato de dicho documento está especificado casi en su totalidad por el siguiente documento: http://www.relpe.org/relpe/DocumentoTecnico1.pdf. En la siguiente sección se provee un resumen de dicho documento con algunas modificaciones de índole práctica y algunos campos agregados.

Es responsabilidad del desarrollador asegurarse que la revisión de esta guía es la última disponible.

Especificación de la ficha de contenidos

Tipo de documento

La ficha de contenidos se deberá presentar como un documento de texto con formato XML. De estar disponible el editor de la fichas de contenidos provista por el CMS de Ceibal, el documento XML deberá ser generado mediante dicha herramienta para evitar inconsistencias.

Campos de la ficha

La **ceibal:edad**	Rango de edad sugerida del usuario	Obligatorio	Rango de edad objetivo del contenido, o edad para la cual el contenido es apropiado. El formato será NN-NN, en donde EE es la edad expresada en años como un número de dos dígitos. En caso de que el contenido sea apropiado para cualquier edad se escribirá "Cualquiera".

Tabla 1 especifica los campos de la ficha de contenidos. Los campos tienen algunas características que se presentan en la columna "Características" cuya descripción se puede ver en la *Tabla 2*. Tal como se puede ver en dichas tablas, algunos campos no deben ser provistos por el desarrollador, mientras que otros pueden dejarse en blanco si existen dudas de cómo llenarlos. En todos los casos, los campos pueden ser revisados y modificados por los contenidistas de Ceibal antes de la publicación del contenido.

Tabla de campos de la ficha

Nombre	Significado	Características	Descripción
dc:identifier	Identificador único	Obligatorio, Único, Automático	Dirección URL, por ejemplo http://www.portal.org/article-66609.html
dc:title	Título	Obligatorio,	Nombre del contenido dado por el creador. Si

		Simple	el contenido carece de título se le asignará uno representativo con menos de 50 caracteres.
dc:description	Descripción breve	Obligatorio, Simple	Resumen del contenido. La descripción no debe repetir el título in ningún otro campo, debe agregar información.
dc:creator *	Autores	Obligatorio, Múltiple	Personas, instituciones u organizaciones que poseen la responsabilidad primaria sobre el contenido intelectual. **Formato según autor**: Persona: "<Apellido1>[Apellido2], <Nombre1>[Nombre 2]". Institución u organización: "<entidad madre>, <repartición>[, <repartición>, …, <repartición>]". Desconocido: editor, distribuidor o hosteador. Sin datos: "autor anónimo". En donde: "< >" denota un dato obligatorio, "[ ]" denota un dato opcional y "…" denota repetición de datos.
dc:publisher	Publicador o Editor	Obligatorio, Automático	Nombre oficial del portal RELPE que hace accesible el contenido. Este campo siempre será "Plan Ceibal"
dc:source *	Fuente	Obligatorio, Múltiple	Referencia a los contenidos del cual deriva el contenido que se está catalogando. En caso de ser contenido original el desarrollador dejará el valor en blanco (sin embargo el campo debe aparecer en el documento XML de todas formas).
dc:rights	Derechos de uso	Obligatorio, Simple, Postergable	Condiciones del contenido en cuanto a los derechos de uso (política de acceso al contenido).
dc:type	Tipo de recurso	Obligatorio, Simple	Tipo de recurso digital: Collection, Dataset, Event, Image, InteractiveResource, MovingImage, Service, Software, Sound, Text, Links.
dc:format	Formato	Obligatorio, Simple	Formato MIME de representación de datos o tipo de archivo (jpg, mpg, etc). Ver http://www.iana.org/assignments/media-types/ o bien la documentación del creador del formato (por ejemplo Adobe TechNote tn_4151).
dc:language	Idioma	Obligatorio, Simple	Lenguaje del contenido según ISO 639 (ver tabla en el documento de RELPE mencionado arriba). Ejemplos: "es" para Español, "en" para Inglés, "No aplica" si no se le puede asignar un lenguaje particular al recurso.
dc:date	Fecha de publicación	Obligatorio, Simple, Postergable	Fecha de publicación en formato AAAA-MM-DD. Este campo podrá ser dejado en blanco hasta que se determine la facha de publicación.
dc:subject +	Temática	Obligatorio, Repetible, Postergable	Clasificación Decimal Dewey (http://www.oclc.org/americalatina/es/dewey/about/default.htm). Este campo podrá ser dejado en blanco por el desarrollador si no

			está claro cuál es la clasificación adecuada (los contenidistas deberán llenarlo antes de la publicación).
relpe:type +	Tipo de recurso educativo	Opcional, Repetible	Describe el recurso desde su formato para contextos educativos: actividades de aula, guías docentes, presentaciones, etc. Ver tabla de tipos en el documento de RELPE.
ceibal:edad	Rango de edad sugerida del usuario	Obligatorio	Rango de edad objetivo del contenido, o edad para la cual el contenido es apropiado. El formato será NN-NN, en donde EE es la edad expresada en años como un número de dos dígitos. En caso de que el contenido sea apropiado para cualquier edad se escribirá "Cualquiera".

Tabla 1

Tabla de características de los campos de la ficha

Característica	Descripción	Formato
Obligatorio	No se puede prescindir del campo, debe ser llenado.	Negrita
Opcional	No obligatorio, se puede prescindir de él.	Sin formato especial
Único	No se puede repetir entre todas las fichas de contenido existentes en el mundo. Todo campo único es a la vez simple.	Subrayado
Automático	El campo es generado automáticamente por el CMS, no debe ser provisto por el desarrollador.	Fondo gris
Postergable	El campo podrá ser dejado en blanco para ser llenado por los contenidistas en caso de no estar claro cuál es el valor adecuado.	Fondo gris.
Simple	El campo puede tener como máximo 1 valor.	Sin formato especial
Repetible +	El campo puede tener más de un valor, cada uno de los cuales caracteriza el campo. Por ejemplo, la temática de un recurso puede ser descrita por 200.23 y 204.35 (porque el tema del recurso está relacionado con dos clases distintas dentro del esquema Dewey); pero el conjunto de clases no definen el campo sino que más bien lo caracterizan.	+ a la derecha del nombre del campo.
Múltiple *	El campo puede tener más de un valor. La conjunción de todos los valores es lo que define el valor del campo. Por ejemplo, el creador de un recurso puede ser un conjunto de autores, y ese conjunto de autores como un todo es lo que define el creador de forma no ambigua.	* a la derecha del nombre del campo.

Tabla 2

Ejemplo de ficha de contenido en formato XML

El siguiente documento XML es un ejemplo de la ficha en su estado final, antes de ser consumida por el CMS. Los campos que no están en negrita podrán ser dejados en blanco por el desarrollador.

```
<?xml version="1.0"?>
<dc:title>El inspector marciano – misión cigarrillo</dc:title>
<dc:description>
Juego didáctico parte de una campaña de información anti-tabaco.
```

```
</dc:description>
<dc:creator>MTW Estudios</dc:creator>
<dc:source></dc:source>
<dc:rights>Sin restricciones</dc:rights>
<dc:type>InteractiveResource</dc:type>
<dc:format>swf</dc:format>
<dc:language>es</dc:language>
<dc:date>2008-09-08</dc:date>
<dc:subject>793</dc:subject>
<dc:subject>613</dc:subject>
<relpe:type>tc13</relpe:type>
<ceibal:edad>07-12</relpe:type>
```

Calidad

En esta sección se presentan una serie de criterios y prácticas que apuntan a mejorar la calidad de los contenidos. No pretende ser una guía exhaustiva ya que se parte de la base de que el desarrollador está familiarizado con los conceptos básicos de aseguramiento de la calidad (QA) e ingeniería de software, sobre todo cuando el contenido consta de aplicaciones.

Buenas prácticas de desarrollo

Se sabe que la aplicación de ciertas reglas durante el desarrollo del software aumenta enormemente la calidad del producto resultante. A continuación se presenta un resumen de estas prácticas llevadas al contexto que nos ocupa.

Ciclo de vida

A continuación se presenta un resumen del ciclo de vida ideal de un contenido digital. Este ciclo de vida está pensado para aplicaciones de software, pero hay conceptos generales que se pueden aplicar a cualquier recurso (documentos, imágenes, películas, etc). Tener en cuenta los pasos no se deben seguir estrictamente en la secuencia que se presenta aquí, sino que es muy conveniente hacer una o más iteraciones de la secuencia y/o volver a algún paso anterior para hacer refinamientos sucesivos del producto.

Requerimientos

La primera etapa de todo proyecto es determinar las características o requerimientos del producto. Los requerimientos se escribirán en un documento y se podrán ir refinando a medida que avanza el proyecto.

Diseño básico

Luego de que existe un documento de requerimientos adecuado, se procederá a hacer un diseño básico del producto. Este diseño se cotejará continuamente con los requerimientos, modificando uno u otro de ser necesario. Una vez obtenido un diseño básico se escribirá en un documento.

Diseño detallado

Algunas veces es posible hacer un diseño detallado a priori, pero generalmente es una mejor idea refinar el diseño detallado por medio de la prototipación. Normalmente se tendrá un

diseño detallado una vez que prototipo funciona de forma adecuada y se han corregido todos los posibles errores en el diseño y los requerimientos. Cuando esto ocurre se puede proceder a escribir formalmente el diseño detallado, ya que de hacerlo antes es muy probable que sufra muchos cambios lo cual retrasaría innecesariamente el desarrollo. Por otro lado, profundizar en un borrador del diseño detallado puede ser ventajoso ya que se pueden detectar problemas antes de que los mismos sean llevados a la implementación, con el consiguiente ahorro de tiempo.

Implementación: prototipación

Prototipar significa implementar el producto en refinamientos sucesivos. Se comenzará con una versión simplificada del producto, y se irán agregando prestaciones a medida que se avanza. Normalmente durante la prototipación se descubrirán problemas en el diseño (idealmente menores) que será necesario corregir. El prototipo irá evolucionando hasta convertirse eventualmente en el producto final para ser testeado formalmente.

Notar que al prototipar ya se está haciendo un testing básico (testing de desarrollo) constantemente. Esto es muy importante ya que alivia gran parte de la etapa de testing formal.

Para prototipar se pueden utilizar las herramientas y plataformas finales o bien utilizar otras alternativas si se prevé que esto pueda facilitar el desarrollo y disminuir los tiempos totales. Tener en cuenta que el pasaje a las herramientas y plataformas finales también consumirá tiempo, e incluso puede evidenciar errores en el propio diseño que no se habían expresado en la plataforma de desarrollo.

Por ejemplo, si se está desarrollando un juego en Flash, es posible hacerlo en Windows utilizando el navegador usual para probarlo. Pero finalmente habrá que pasarlo a una XO y ver si funciona correctamente. Este es un caso sencillo ya que Flash es de por sí multiplataforma. Otro ejemplo puede ser utilizar un emulador de XO para desarrollar una aplicación en Python (también multiplataforma). Incluso se podría comenzar el desarrollo directamente bajo Windows antes de llevarlo a un emulador. El pasaje a la plataforma definitiva es llevar la aplicación a la XO, con las probabilidades de que todo ande bien a favor por el hecho de ser Python multiplataforma. En el caso de una aplicación nativa de XO/Sugar este pasaje seguramente no será tan sencillo, por lo que de ser posible conviene evitar desarrollar aplicaciones nativas que puedan ser adecuadamente desarrolladas en Python.

Testing

El testing es una etapa fundamental en el ciclo de vida de un producto, ya que es la última oportunidad de descubrir errores antes de que el producto sea liberado. A pesar de que en la prototipación ya se efectuó testing, es importante mantener esta etapa como independiente ya que es de naturaleza diferente. En esta etapa es importante documentar los tests efectuados y que los mismos sean repetibles. Esto permite efectuar los mismos tests a las futuras revisiones del producto.

Entre los tipos de tests posibles se destacan los siguientes, todos igualmente importantes:

- **Tests de desarrollo**. Son los realizados durante la prototipación del producto. Aquí vale cualquier tipo de test informal.

- **Tests de requerimientos**. Son tests enfocados al cumplimiento de los requerimientos. Son tests de alto nivel que no incursionan en los detalles del diseño y mucho menos de la implementación. Por ejemplo, si un requerimiento es "el juego debe tener un chat" el test deberá asegurar que existe una ventana en donde el usuario pueda chatear y que dicho chat se comporta en forma correcta en términos generales, sin entrar en tests de fuerza para encontrar errores más sofisticados.

- **Tests de módulos**. Están enfocados a testear módulos independientes de la aplicación. En el sentido estricto, sería necesario aislar el módulo a testear del resto de la aplicación y comprobar el funcionamiento del mismo en forma independiente del resto. Muchas veces esto es muy difícil de lograr. De todas formas es posible testear módulos sin llegar a ser tan estrictos mediante una batería de tests que cubran principalmente características de un módulo particular. El test de módulos es especialmente importante cuando se trata de módulos críticos, centrales para el funcionamiento de la aplicación.

- **Tests de integración**. Se trata de tests que verifican el correcto funcionamiento de las interfaces entre módulos. Si el diseño fue hecho correctamente, será posible identificar estas interfaces fácilmente. Una interface siempre será un nexo entre dos módulos que ya fueron testeados independientemente. Testear la interface entonces será aplicar tests que cubran la interacción entre ambos módulos. Aquí también ocurre que suele ser difícil aislar ambos módulos y su interface, pero se pueden aplicar tests específicos que los cubran directamente sobre la aplicación completa.

- **Tests de fuerza**. Se trata de tests que no buscan verificar el correcto funcionamiento sino que intentan encontrar errores mediante la aplicación de un gran número de entradas inusuales. En los casos anteriores, se trata de que un módulo dado produzca una salida adecuada ante una cierta entrada. En un test de fuerza se trata de encontrar entradas no previstas en el diseño o la implementación que puedan generar salidas incorrectas (por ejemplo, la caída de la aplicación).

- **Tests de carga o escalabilidad**. Muchas veces una aplicación está pensada para funcionar en ambientes muy variables que no siempre son fáciles de producir en un ambiente de testing. Por ejemplo, una aplicación que despliega datos obtenidos de una base de datos puede funcionar muy bien con cierta cantidad o calidad de datos, pero fallar en el ambiente de producción, cuando los datos son reales, ya sea por su gran cantidad o porque contienen información no prevista por la aplicación. En este caso es imprescindible producir casos de testing que se aproximen lo más posible a una situación real, por ejemplo, generando artificialmente muchos registros de una tabla para probar el comportamiento de la aplicación en despliegue de una gran cantidad de datos. De todas formas el testing en producción no es sustituible.

- **Testing de producción o mantenimiento (beta)**. Es el testing que se hace luego de la liberación del producto. A esta etapa muchas veces se le llama beta testing. Aquí el equipo de test deberá observar el comportamiento de la aplicación muy de cerca mientras

está siendo utilizada por los usuarios finales. Siempre que sea posible es conveniente también estar atento y promover el feedback de los propios usuarios.

Finalmente, notar que durante la etapa de prototipación puede ser muy conveniente adelantarse al testing mediante la implementación de secciones especiales para el testing. Estas secciones pueden estar enfocadas a lo siguiente:

- Log de eventos y reporte de errores.
- Código específico (que se activa solamente con instrucciones especiales, como por ejemplo banderas de compilación o inclusiones condicionadas) para facilitar el aislamiento selectivo de módulos que permitan hacer un test de módulos efectivo.
- Contadores y otros indicadores del estado de la aplicación, como ser contadores de carga (cantidad de conexiones por ejemplo), cantidad de cuadros por segundo en una animación, cantidad consumida de un cierto recurso, etc.

Producción y mantenimiento

Esta es la etapa posterior a la liberación del producto, cuando se pone a disposición de los usuarios finales. En realidad esta etapa se puede dividir en beta testing y final, pero en ambos casos se trata de que la aplicación es utilizada por el usuario final, aunque probablemente convenga que en beta la cantidad de usuarios no sea muy grande para disminuir el impacto de posibles errores.

En esta etapa se produce el mantenimiento del producto, ya sea para corregir errores o bien para cambiar o agregar prestaciones. Eventualmente habrá que volver algunos pasos atrás en el ciclo de vida antes de poder poner en producción una nueva versión.

Estilo del código

En esta sección se presentan una serie de “tips” generales a aplicar durante la codificación de un producto de software. Luego se agregan algunos “tips” más específicos para Action Script 3.0 y Python.

Para cualquier lenguaje

- Siempre que el lenguaje lo permita utilizar técnicas de orientación a objetos.
- Cuando sea posible utilizar lenguajes fuertemente tipados, esto disminuye enormemente las posibilidades de errores en tiempo de corrida.
- Utilizar las características ventajosas del lenguaje en particular y evitar malas. No hay por qué utilizar una característica solamente porque está ahí, hay que hacerlo solamente si se adapta a nuestras necesidades.
- Evitar las características obsoletas que fueron declaradas “depercated” por el proveedor del lenguaje.
- Escribir claramente, evitar construcciones difíciles de leer.
- Utilizar nombres nemotécnicos.
- Utilizar siempre constantes con nombres claros en lugar de las constantes literales.
- Mantener el código lo más simple posible. Si se agrega código para modificar un comportamiento, integrarlo adecuadamente según el diseño evitando los “parches”. De ser necesario, cambiar el diseño (el diseño no es una roca rígida, debe ser flexible).
- Evitar a toda costa la redundancia. Es decir, cuando los datos o las estructuras de control se repiten ya sea en forma explícita o implícita, escribirlos de tal forma que aparezcan

una sola vez (por ejemplo utilizando métodos o funciones, tablas para describir los datos, clases, etc.). Esto facilitará mucho el mantenimiento en el futuro, ya que de tener que cambiar algo no habrá que buscar en muchos lugares.

- Evitar módulos que hagan demasiadas cosas al mismo tiempo. Un módulo debería idealmente hacer una cosa bien, más que muchas en forma mediocre.
- Cuando sea posible, escribir código que sea auto-explicativo, esto es, que los nombres de las funciones, variables, clases, miembros y métodos expliquen claramente su significado sin necesidad de comentarios superfluos.
- Cuando sea posible, escribir interfaces no ambiguas. Esto es, si se puede escribir un módulo de forma tal que exista una sola forma de utilizarlo, hacerlo. Esto evita errores y ahorra tiempo a la hora de utilizar ese módulo. Si no es posible hacer esto en tiempo de compilación, agregar código (idealmente de "debug") para detectar el mal uso del módulo. En resumen, más allá de que el uso de un módulo esté descrito por su documentación (comentarios, documento técnico, etc.) es buena práctica hacer obvio este uso mediante el código.
- Evitar comentarios superfluos, es decir, aquellos que el código es capaz de explicar por sí mismo. Utilizar comentarios que agreguen información relevante y evitar aquellos que simplemente relatan el código.
- Evitar comentarios "missleading". Esto es, comentarios que en poco tiempo quedan obsoletos. Este tipo de comentarios son muy confusos a la hora del mantenimiento, ya sea que éste lo haga el propio codificador u otra persona. En general los comentarios se deben colocar en lugares importantes, en donde la ausencia de un comentario hace difícil la lectura del código a pesar de los esfuerzos de hacerlo claro por sí mismo.
- Evitar comentarios sobre cómo se implementa cierto módulo (función, clase, etc) a menos que esto sea relevante. Los comentarios más bien deben apuntar a qué hace un módulo. Esto incrementa la vida del comentario antes de hacerse obsoleto. Siempre que se detecte un comentario obsoleto se debe actualizar inmediatamente.
- Si se detecta código inapropiado, mal diseñado o erróneo, no comentarlo para aclarar el problema, reescribirlo lo más pronto posible.
- Utilizar un estilo de codificación consistente. Es decir, sea cual sea el estilo que se lija para un proyecto (nomenclatura de las clases, funciones, variables, etc.) se debe utilizar el mismo estilo a lo largo de todo el proyecto.
- Programar defensivamente. Esto es, tener en cuenta el mal uso o las entradas "incorrectas" en el código. Dependiendo del caso, será conveniente que este código de "defensa" sea activado solamente durante la depuración (compilación condicional por ejemplo) o bien estar presente en la versión de "release", por ejemplo mediante el uso adecuado de las excepciones.
- Todo módulo debe prever todas las posibles entradas, es decir, debe tener un comportamiento definido para cualquier entrada (notar que cuando un módulo dispara una excepción o responde con un código de error, se está previendo correctamente la entrada que da lugar al error).
- Utlizar "patterns".
- No efectuar un diseño y/o codificación demasiado rígidos, es decir, siempre que sea razonable implementar módulos un poco más genéricos de lo estrictamente necesario para resolver un caso particular. Esto aumenta mucho las chances de que en el futuro el mantenimiento sea más sencillo ya que los módulos prevén muchos de los posibles cambios. Por otro lado, tener en cuenta que una abstracción exagerada puede llevar a un

esfuerzo desmesurado respecto al problema que se quiere resolver. Además, por más genérico que sea un módulo, siempre existe la posibilidad (nada remota) de que dicho módulo quede obsoleto al contrastarlo con los cambios de la realidad. Por eso habrá que encontrar un balance entre la generalidad y la resolución de casos concretos.

Para ActionScript 3.0

- Utilizar las nuevas características del lenguaje: excepciones, tipos, clases "sealed", expresiones regulares, "namespaces". De más está decir que se deben utilizar las características orientadas a objetos ya introducidas en ActionScript 2.0 (clases, herencia, etc.).
- Utilizar el modo de compilación estricto. Esto aumenta la cantidad de errores detectados en tiempo de compilación.
- Tener mucho cuidado con el pasaje de parámetros a las funciones. ActionScript 3.0 a pesar de haber introducido muchas mejoras en el lenguaje, sigue sin ser un lenguaje fuertemente tipado como lo es por ejemplo C#. Por ejemplo, si tenemos una función con la siguiente declaración: `function f(a:MiClase)`, podemos de todas formas llamar a esta función pasando un objeto de cualquier clase sin que el compilador de error. El error se producirá recién al intentar ejecutar dicha llamada.

Python

- Python es un lenguaje en constante desarrollo. Estar atento a las nuevas características para evitar la obsolescencia.
- Programar teniendo en cuenta que Python es un lenguaje interpretado y puede ser sensible a problemas de performance.
- Se trata de un lenguaje muy flexible que soporta múltiples paradigmas. No abusar de dicha flexibilidad ya que puede hacer que el código sea confuso y difícil de mantener.
- Python viene con una gran cantidad de módulos estándar que están documentados apropiadamente. Utilizarlos en lugar de implementar el mismo comportamiento otra vez. Esto no solamente puede ahorrar tiempo y facilitar la mantenibilidad del código, sino que aumenta la performance ya que dichos módulos fueron programados a bajo nivel y ya están compilados, no son interpretados.
- Python posee facilidades de programación funcional. Antes de utilizarlas, aprender a hacerlo si no se está familiarizado con dichas técnicas.

Performance

- **Hardware**. Las XO son máquinas muy limitadas con respecto a su capacidad de procesamiento y almacenamiento si las comparamos con los estándares actuales. Por ese motivo es muy importante tener en cuenta la performance durante el desarrollo. De nada servirá una aplicación que funciona muy bien en un ambiente de desarrollo que utiliza un PC potente si la misma no cumple su función en la XO por problemas de performance. Esto puede llevar incluso a la necesidad de un rediseño completo si no se tiene cuidado, con la consiguiente pérdida de tiempo y esfuerzo. Con respecto a la performance, se puede comparar a una XO con un PC con una CPU AMD K6-2 de 450 MHz, 256 MB de RAM con Windows 98. Puede ser una buena idea utilizar una máquina similar para parte del proceso de desarrollo para ver si todo anda bien

- **Optimización.** Dicho lo anterior, cabe destacar que apresurarse demasiado en optimizar un módulo puede ser peligroso, ya que es muy probable que dicho módulo deba sufrir cambios importantes muy pronto, lo cual hace que la optimización realizada quede obsoleta en forma inmediata. Además, muchas veces los verdaderos problemas de performance no son fáciles de predecir, y se encuentran en lugares inesperados, esquivando los esfuerzos de optimización. En resumen, lo deseable sería tener en cuenta la performance en el diseño, pero no hacer optimizaciones finas hasta que sea evidente que las mismas son necesarias.

- **Plataforma de testing de performance**. Lo ideal sería utilizar la XO lo más posible para verificar los aspectos relacionados a la performance, pero la XO no resulta muy cómoda para este fin dado su tamaño (sobre todo el teclado). Además el ambiente de desarrollo debe ser mucho más potente que una XO si se quieren utilizar las herramientas actuales. La conclusión es que en el proceso de desarrollo se deben intercalar frecuentemente pruebas sobre la misma XO (además de las pruebas en los emuladores, que pueden detectar problemas de funcionamiento, pero no tanto de performance).

A continuación se brindan una serie de "tips" relacionados con la performance.

Percepción de la performance

La percepción de la performance trata del nivel de satisfacción que le genera al usuario cierta característica de una aplicación en relación a los tiempos en que ocurre una secuencia de eventos. Es decir, se trata de utilizar técnicas que mejoren esta percepción del usuario más que de optimizar efectivamente la aplicación.

Este es un campo complejo, pero se pueden utilizar técnicas simples de sentido común que pueden mejorar mucho la experiencia del usuario sin necesidad de complicadas optimizaciones que muchas veces pueden ser imposibles. Por ejemplo, poner una barra de progreso, si bien no disminuye el tiempo que demora un proceso, puede disminuir significativamente la ansiedad del usuario y por lo tanto mejorar su experiencia con la aplicación.

Otro ejemplo un poco menos obvio es un "scroll" ya sea de una imagen o de texto. Si resulta que no se puede lograr que un "scroll" se haga de forma suave, la experiencia del usuario será muy probablemente mejor si simplemente se renuncia a hacer "scroll" en pasos pequeños y se implementa adecuadamente un "scroll" por páginas o por secciones convenientemente elegidas. Por ejemplo, en un juego en 2D en donde un personaje recorre un paisaje, al caminar hacia la derecha, pueden ir apareciendo en forma suave los elementos del paisaje (casas, árboles, etc.). Dependiendo de la complejidad de los gráficos, este "scroll" suave puede dar lugar a saltos y efectos desagradables. Puede ser conveniente entonces, sustituir el "scroll" por un salto único o paginado en donde el paisaje a la derecha aparece de golpe cuando el personaje alcanza una zona próxima al borde derecho de la pantalla.

Por último, se deberán evitar las estrategias para entretener al usuario durante los tiempos de carga o procesamiento, que impliquen un consumo excesivo de recursos, ya que el tiempo de procesamiento aumentará desmesurada e innecesariamente. Por ejemplo, es muy usual presentar contenido Flash durante la carga de un juego Flash (avisos por ejemplo). Si bien

esto cumple la función de distraer al usuario, en el caso de una máquina como la XO puede aumentar considerablemente el tiempo total de la carga. En general es mucho más efectivo utilizar una barra de progreso "liviana".

Tips generales

- Desplegar información siempre es mucho más lento que procesarla internamente en la memoria. Por lo tanto siempre será más económico averiguar si se debe desplegar o no un dato o gráfico antes de hacerlo. Por ejemplo, si un dato no ha cambiado se debe evitar desplegarlo comparando el dato actual con el anterior. De no hacerlo se producirían "flickers" indeseables en la pantalla.

- Cuando se despliegan imágenes en movimiento, se debe calibrar muy bien los cuadros por segundo con respecto al tamaño de la imagen. Cuanto más pixeles mayor es la demora y por lo tanto menor la cantidad de cuadros por segundo sin que aparezcan efectos indeseables.

- Cuando se trata de desplegar gráficos en movimiento, por ejemplo en un juego, restringir las áreas de despliegue a las estrictamente necesarias mejora en forma dramática la performance.

- Evitar el uso de gráficos complejos con efectos sofisticados. Se puede lograr una experiencia satisfactoria si los gráficos se diseñan de forma sencilla y con criterio. Por ejemplo, en lugar de intentar representar, digamos una fruta, en forma realista, es mucho mejor representar la fruta de forma estilizada según cierto criterio. Esto no solamente mejorará la performance sino que le dará a la aplicación un estilo propio.
- En caso de acceder a una propiedad o función varias veces en un trozo de código (y naturalmente, ese valor no será cambiado en dicho código), es mejor guardar previamente el valor en una variable.

Tips para Flash

- Utilizar scrollRect conjuntamente con cacheAsBitmap para reducir las áreas de despliegue a lo estrictamente necesario.

- Cuando se tiene un "movieclip" con muchas capas pero que no es animado es conveniente crear un "snapshot" de dicho "movieclip".

- Evitar el uso de demasiados "teweens" simultáneos.
- Utilizar selectivamente los vectores de imágenes y los bitmaps:
    - Para imágenes animadas o grandes usar vectores.
    - Para imágenes complejas o íconos usar bitmaps.
    - Optimizar los vectores de imágenes mediante Modify/Shape/Optimize

- Utilizar el tipo "int" siempre que sea posible. Evitar decimales.

- Asignar tipos a TODOS los objetos.

- Evitar conversiones implícitas de tipos numéricos, es decir, siempre que una función tome un tipo determinado, utilizar una variable de ese tipo directamente en lugar de efectuar una conversión de tipo. Por ejemplo:

```
var x:int = 10;
Math.sqrt(x);
```

Como Math.sqrt() toma un Number como tipo, sería mejor que x fuera directamente un Number.

- Evitar declaración de variables dentro de loops.

- Cuando se recorre un array con instancias de distintos tipos, no utilizarlas directamente. Es mejor guardarlas primero en una variable del tipo adecuado al objeto.

- En lugares críticos utilizar operadores de bits en lugar de operadores normales, son más eficientes. Por ejemplo, hacer 4 << 1 en lugar de 4*2.

- Borrar explícitamente los "listeners" de los eventos "enterFrame" de un "sprite". De lo contrario ActionScript 3.0 seguirá ejecutando el evento a pesar de haber quitado el "sprite" de la "display list" y anulado todas sus referencias, hasta que el "sprite" es borrado por el "garbage collector".

Tips para Python

- Utilizar los módulos estándar. Están implementados eficientemente y es mucho mejor que correr código propio por mejor que sea éste ya que el mismo es interpretado.

- Ordenamiento de listas: el método "sort" toma una función de comparación como parámetro opcional. De ser posible evitar su uso ya que la misma se llamará muchas veces. En su lugar usar el argumento "key" presente desde Python 2.4.

- Cuando sea posible utilizar "map" en lugar de "for". Esto mueve el loop dentro del módulo Python implementado en C en lugar de ser interpretado, lo cual es mucho más eficiente. Por ejemplo:

```
newlist = []
for word in oldlist:
   newlist.append(word.upper())
```

Se puede reescribir como:

```
newlist = map(str.upper, oldlist)
```

- Evitar los "puntos" en un loop. Por ejemplo:

```
newlist = []
for word in oldlist:
   newlist.append(word.upper())
```

puede ser reescrito como:

```
upper = str.upper
newlist = []
append = newlist.append
for word in oldlist:
    append(upper(word))
```

- Evitar el uso de import en lugares inapropiados. Por ejemplo, un import se puede hacer dentro de una función, sin embargo es más eficiente hacerlo en el "scope" global. De ser necesario hacer un import en un "scope" no global, utilizar una estrategia "lazy", es decir, condicionar el import para hacerlo solamente si efectivamente se utilizará.

REQUERIMIENTOS DE CONTENIDO PARA XO

En la presente sección se presentan una serie de requerimientos que debe cumplir cualquier contenido digital para XO. Cada requerimiento es acompañado con un test que verifica su cumplimiento. El desarrollador deberá entregar una ficha en donde consten los resultados de los tests realizados. En caso de que un requerimiento no aplique a un contenido particular, el mismo deberá constar de todas formas en el documento de resultados de test con la frase "No aplica".

Cuando en lugar de de Requerimiento se habla de Meta, significa que si bien no es una característica obligatoria, puede ser deseable en algunos casos.

Requerimiento: Tiempo de procesamiento

Los tiempos de procesamiento máximos según el tipo de contenido, así como las acciones realizadas durante el procesamiento, serán los que se presentan en la *Tabla de tiempos de procesamiento*, o bien acciones similares que cumplan los mismos objetivos.

Justificación

Mantener los tiempos de procesamiento acotados es muy importante para no perder el interés de los usuarios en un contenido particular.

Notas

- Este requerimiento aplica a una aplicación completa o cualquier parte de la misma que requiera más de 3 segundos de procesamiento antes de presentar un resultado al usuario.
- El tiempo de carga de una aplicación, página web o contenido Flash se considera tiempo de procesamiento.
- El tiempo de procesamiento se medirá desde el momento que el usuario realiza una acción hasta que la aplicación da una respuesta.

- En la tabla se incluyen acciones a tomar durante el procesamiento, así como tiempos de procesamiento "normales" como criterio de comparación. Ningún contenido debería sobrepasar el tiempo máximo de procesamiento. Asimismo, el desarrollador deberá hacer lo posible por mantener los tiempos en los niveles "normales".

- En caso de que el tiempo de procesamiento dependa de una conexión de red, el tiempo se deberá tomar utilizando un ancho de banda lo suficientemente grande como para que el "cuello de botella" sea la capacidad de procesamiento de la XO y no el ancho de banda (mayor o igual a 512 Kbps).

Tabla de tiempos de procesamiento

Tipo de procesamiento	Tiempo máximo	Tiempo normal	Acción durante el procesamiento
Carga de actividad Sugar	60 segundos	10 a 20 segundos	"Flashing icon" desplegado por Sugar.
Procesamiento dentro de actividad Sugar o aplicación Flash.	30 segundos	3 a 10 segundos	Entre 3 y 10 segundos utilizar un cursor de ratón indicador de espera. Para tiempos mayores utilizar una barra de progreso "liviana".
Carga de página web	40 segundos	5 a 20 segundos	Desplegar el contenido disponible cuanto antes.
Carga de aplicación Flash	60 segundos	15 a 30 segundos	Barra de progreso "liviana".

Testing

Se deberán recorrer todos los caminos de la aplicación o página web sobre la XO verificando y registrando los tiempos de respuesta y la acción durante los procesamientos. Se deberá llenar la *Ficha de testing*.

Ficha de testing

Tipo de procesamiento	Tiempo registrado	Acción durante el procesamiento	OK	Observaciones
Carga	<tiempo de carga>	<acción>	Si/No	<observaciones>
<Tipo de procesamiento>	<tiempo de procesamiento>	<acción>		<observaciones>
...	...	...	...	...

Requerimiento: cuadros por segundo

La cantidad de cuadros por segundo deberá ser definida en el diseño y verificada en la XO.

Nota: Este requerimiento aplica solamente en el caso de que una aplicación, presentación, o parte de las mismas, despliegue imágenes en movimiento utilizando la técnica de mostrar una serie de de cuadros consecutivos, ya sea en toda o parte del área de despliegue de la aplicación.

Justificación

Desplegar imágenes con una cantidad de cuadros por segundo errática afecta sustancialmente en forma negativa la experiencia del usuario.

Notas

- La cantidad de cuadros por segundo que deberá desplegar una parte de la aplicación deberá ser definida explícitamente en el diseño.

- De no ser posible cumplir con la cantidad de cuadros por segundo que especifica el diseño se deberá bajar la misma, simplificar los gráficos, bajar la resolución o bien rediseñar la aplicación para que no sea necesario tal cantidad de cuadros por segundo.

- La cantidad de cuadros por segundo efectiva que despliega la aplicación deberá ser estable, con no más de un 5% de variación. La medición se realizará en condiciones de uso exclusivo del procesador (naturalmente se excluye el uso de procesador del sistema operativo).

Testing

Se deberán implementar contadores que desplieguen la cantidad de cuadros por segundo en modo de "debug". En cada parte de la aplicación que despliegue imágenes en movimiento se observará el valor del contador y su estabilidad. Se registrará el resultado en la ficha de testing.

Nota: se abrevia "cuadros por segundo" como FPS (frames per second).

Ficha de testing

Tipo de imagen en movimiento	Módulo	FPS esperados	Rango de FPS observado	OK	Observaciones
< flash, video, gráficos Python, etc>	<zona de la aplicación a testear>	<fps>	<fps min> - <fps max>	Sí/No	<observaciones>
…	…	…	…		…

Requerimiento: Tiempo de respuesta

El tiempo de respuesta ante el uso del teclado o el mouse en el contexto de una aplicación con acción en tiempo real deberá ser lo suficientemente bajo como para que la experiencia del usuario sea satisfactoria.

Justificación

En una aplicación de tiempo real es fundamental la "simultaneidad" de la acción del usuario y la consiguiente respuesta de la aplicación. De no ocurrir así el usuario no podrá coordinar adecuadamente sus acciones, deteriorando seriamente su experiencia con la aplicación.

Notas

- Un ejemplo típico de este tipo de aplicaciones son los juegos de acción. En este caso las acciones del usuario se deben ver reflejadas de inmediato en la pantalla.

- En caso de obtener un tiempo de respuesta inadecuado se deberá tomar una o más de las siguientes acciones:
    - Optimizar y bajar el tiempo de respuesta.
    - Disminuir la calidad de los gráficos.
    - Aumentar la cantidad de píxeles entre dos posiciones consecutivas de un objeto en movimiento, siempre que esto no disminuya en forma significativa la experiencia del usuario.
    - Rediseñar la aplicación para disminuir la acción en tiempo real pero aún así logre su objetivo didáctico o de entretenimiento.

Testing

En el caso ideal los tiempos de respuesta deberían ser tales que el usuario no sea capaz de notarlos, y por consiguiente aparecerían como instantáneos. Un tiempo de respuesta bueno puede estar en el orden de los 100 ms. De 200 a 500 ms el tiempo de respuesta es mediocre, y más allá de eso es decididamente malo.

Para medir efectivamente estos tiempos de respuesta sería necesario aplicar técnicas de "profiling" en el código. Esto puede llegar a ser muy engorroso, y a veces innecesario. Por lo tanto, si bien serán bienvenidas las mediciones cuantitativas, será suficiente con un test cualitativo.

El test consiste en recorrer todas las zonas de la aplicación que requieran respuestas rápidas y registrar que tan perceptible es el tiempo de respuesta máximo observado. Se clasificarán los tiempos según los siguientes valores:

- "Muy bajo". El tiempo de respuesta no puede ser percibido.
- "Bajo". Si bien se puede notar una demora en la respuesta, la misma no afecta el desarrollo satisfactorio de la acción.
- "Medio". El tiempo de respuesta se nota claramente, pero aún es posible la coordinación con la acción.
- "Alto". Coordinar las acciones se hace difícil.
- "Muy alto". Es prácticamente imposible coordinar la acción, la interacción se vuelve extremadamente confusa.

Los tiempos de respuesta aceptables son "Muy bajo" y "Bajo".

Ficha de testing

Tipo de acción	Módulo	Tiempo de respuesta máximo observado	OK	Observaciones
<movimiento de objeto, selección de objeto, etc >	<zona de la aplicación a testear>	Muy bajo / Bajo / Medio / Alto / Muy Alto	Sí/No	<observaciones>
…	…	…		…

Requerimiento: fluidez de los movimientos

Los movimientos en una aplicación o presentación de cualquier tipo deberán ser fluidos.

Justificación

La presencia de movimientos erráticos o a saltos irregulares deteriora mucho la experiencia del usuario.

Notas

El movimiento fluido de un objeto tiene dos características:

- Distancia entre dos posiciones consecutivas en relación con la velocidad ("saltos").
- Uniformidad de los "saltos". Las distancias entre posiciones consecutivas del objeto para una velocidad dada es constante.

Para que el ojo humano perciba un movimiento fluido es mucho más importante la uniformidad de los saltos que su distancia. Esto permite aumentar dicha distancia por cuestiones de performance (para poder cumplir con el resto de los requerimientos) sin que afecte en forma dramática la percepción de la fluidez.

Notar que si bien este requerimiento está relacionado con el requerimiento "cuadros por segundo", éste último habla del redespliegue de toda una escena, mientras que el requerimiento "fluidez de los movimientos" se aplica a objetos dentro de una escena. Obviamente la cantidad y estabilidad de los FPS afectará la percepción de la fluidez, pero una cantidad de FPS adecuada no garantiza por sí sola la fluidez adecuada de los objetos de una escena.

Testing

El test de fluidez será cualitativo. Se recorrerán las partes de la aplicación en donde existan movimientos registrando la percepción de la fluidez con los siguientes valores:

- "Muy fluido". El movimiento es muy suave, tanto en la distancia entre saltos como su uniformidad.
- "Fluido". El movimiento es suave, aunque se perciben saltos. No se perciben saltos desparejos.
- "Poco fluido". El movimiento presenta algunos saltos desparejos.
- "Errático". El movimiento presenta saltos grandes e impredecibles.

Los valores aceptables serán “Muy fluido” y “Fluido”.

Ficha de testing

Tipo de movimiento	Módulo	Tiempo de respuesta máximo observado	OK	Observaciones
<movimiento de objeto, selección de objeto, etc >	<zona de la aplicación a testear>	Muy fluido / Fluido / Poco fluido / Errático	Sí/No	<observaciones>
…	…	…		…

Requerimiento: uso del gamepad

Si la naturaleza de la aplicación lo permite, la misma deberá ser compatible con el gamepad de la XO.

Justificación

La XO está diseñada para utilizar el gamepad, sobre todo en el caso de juegos (pero no exclusivamente en ellos). El gamepad de la XO hace mucho más cómodo utilizar aplicaciones que no requieren de mouse y son controladas por las 8 teclas correspondientes.

Notas

- El gamepad de la XO está mapeado por defecto con los siguientes botones del teclado de goma:
    - 4 botones de la izquierda: 4 flechas.
    - 2 botones verticales a la derecha: “fn + re pg” y “fn + av pg” (de arriba hacia abajo).
    - 2 botones horizontales a la derecha: “fn + inicio” y “fn + fin” (de izquierda a derecha).

- Este requerimiento aplica solamente si la aplicación, o una parte relativamente independiente de la misma, puede ser controlada satisfactoriamente utilizando únicamente el gamepad.

- El mousepad de la XO es bastante limitado. Jugar utilizando el mouse puede ser una experiencia un tanto frustrante dependiendo de la aplicación. Cuando sea posible es preferible utilizar el gamepad o el teclado.

Testing

Se deberá verificar que la aplicación puede ser controlada por el gamepad en forma satisfactoria.

Ficha de testing

Tipo aplicación	Módulo	Se prevé uso del gamepad?	OK	Observaciones
<juego de acción,	<zona de la	Sí/No	Sí/No	<observaciones>

juego por turnos, simulador, otro tipo de juego, otra aplicación, etc >	aplicación a testear>			
…	…	…		…

Requerimiento: resolución de la pantalla

El contenido debe estar diseñado para verse correctamente en una resolución de 800x600 a menos que la aplicación cambie la resolución automáticamente en forma temporal.

Justificación

Las XO se distribuyen con una resolución de 800 x 600.

Notas

- En lo posible el contenido deberá estar diseñado de forma tal que se pueda desplegar completo en 800x600. Las barras de scroll serán utilizas solamente si es estrictamente necesario.

- Tener en cuenta que el área utilizable del navegador de es menor que 600 pixeles (un poco menos de 550). Si se desea que un contenido entre completo en este caso se deberá restar la altura de la barra superior del navegador a la altura del contenido.

Testing

Se deberá recorrer la aplicación verificando que el contenido entra completo en la pantalla, o en su defecto que existan barras de scroll u otro método de scroll adecuados. Se verificará que el uso del scroll es justificado.

Ficha de testing

Tipo aplicación	Módulo	¿Contenido completo?	¿Scroll?	¿Se justifica scroll?	OK	Observaciones
<juego, editor de texto, libro electrónico, etc>	<zona de la aplicación a testear>	Sí/No	Sí/No	Sí/No <Motivo>	Sí/No	<observaciones>
…	…	…		…		…

Requerimiento: legibilidad del texto

El texto presente en el contenido deberá ser fácilmente legible en la XO.

Justificación

Es importante que el texto sea fácilmente legible para cualquier usuario, pero en especial para los niños, sobre todo los más chicos ya que puede que recién estén aprendiendo a leer.

Notas

- La pantalla de la XO es mucho más chica que la de una computadora normal. Si bien la definición pequeña de 800x600 aumenta la legibilidad, aún así puede ocurrir que un texto

perfectamente legible en una computadora normal se transforme en difícil de leer en la pantalla de la XO.

- Si bien la solución obvia al problema de un texto demasiado chico es agrandar la letra, esto puede afectar la correcta disposición del contenido en la pantalla. Considerar opciones como reducir la cantidad de texto, cambiar el tipo de letra o cambiar la disposición del contenido para que el texto entre mejor.

Testing

Se deberá recorrer el contenido verificando que el texto es razonablemente legible en la XO a una distancia de aproximadamente medio metro. Se pondrá especial atención a aquel texto que sea más relevante para el contenido en cuestión.

Ficha de testing

Tipo aplicación	Módulo	OK (¿Texto legible?)	Observaciones
<página web, juego, editor de texto, libro electrónico, etc>	<zona de la aplicación a testear>	Sí/No	<observaciones>
…	…	…	…

Requerimiento: ortografía, gramática y redacción

La ortografía, la gramática y la redacción del contenido escrito deberán ser correctas. Los conceptos deberán estar expresados en forma clara para el público al que apunta el contenido en cuestión.

Justificación

Evitar errores ortográficos y gramaticales es siempre deseable, pero en este caso es aún más importante porque se trata de un proyecto educativo para niños y es fundamental dar el ejemplo. Sumado a esto, es importante que el contenido exprese en forma correcta y simple lo que pretende transmitir.

Notas

Aplicar correctores ortográficos y gramaticales no es suficiente para garantizar la buena calidad de un escrito. Es necesario leer el contenido (en lo posible varias veces y por personas diferentes) para tener una buena posibilidad de que no se escaparon palabras que si bien son correctas ortográficamente, son erróneas en el contexto en donde aparecen. También es necesario leer el contenido cuidadosamente para asegurar que no se trasmiten ideas equivocadas o contradictorias, que no hay frases repetidas accidentalmente, etc.

Testing

El contenido escrito deberá ser leído por lo menos por dos personas diferentes. En caso de encontrar errores o sugerencias de mejora el corrector deberá anotarlas apropiadamente.

Obviamente las primeras lecturas se harán en el editor de texto, pero no se puede omitir una lectura del contenido final sobre la propia XO.

Ficha de testing

Tipo de texto	Módulo	Errores	OK	Observaciones
<página web, narración de juego, ayuda de aplicación, etc>	<zona del contenido a testear>	<anotar lugar y tipo de error>	Sí/No	<observaciones>
…	…	…		…

Requerimiento: contenido apropiado

El contenido deberá ser apropiado para el público al que está dirigido.

Justificación

Es importante que el público al que va dirigido el contenido obtenga algo que esté a su alcance, que pueda entender y dominar, sobre todo en el caso de los niños.

Notas

Contenido apropiado puede tener dos lecturas diferentes. Puede querer decir que sea apropiado desde el punto de vista emocional (relacionado con contenidos violentos o explícitos en algún sentido), o bien dese el punto de vista intelectual (la capacidad de una persona para entender o dominar el contenido). En el caso de Ceibal, obviamente, todos los contenidos que no estén restringidos a algún sector deben ser aptos para todo público, ya que los niños más chicos podrían acceder al contenido aunque éste no haya sido diseñado para ellos.

Por ejemplo, puede ser que un juego sea muy complicado para un niño de 6 años, pero apropiado para uno mayor que 10. También puede ocurrir que un texto sea comprensible para un niño de 11 años pero difícil de entender para uno de 8.

La edad no es el único criterio para clasificar el público objetivo. Puede ser que un contenido apunte a gente con una formación determinada, que lo hace inapropiado para el público general.

Siempre que sea posible, se deberá exhibir una leyenda que especifique el rango de edad al que apunta el contenido, o bien estar dentro de una sección específica para dicho rango de edades (por ejemplo, una rama dentro de un sitio web).

El público objetivo del contenido deberá constar en los requerimientos.

Testing

Se deberá recorrer el contenido para verificar que el mismo es adecuado para el público objetivo. Si existieran dudas se deberá consultar a profesionales de Ceibal o de Educación Pública. Este test no es necesario hacerlo directamente sobre la XO.

Ficha de testing

Tipo de contenido	Módulo	Público objetivo	OK (¿Contenido apropiado?)	Observaciones

<juego, texto, etc>	<zona del contenido a testear>	<rango de edad, maestras, padres, etc >	Sí/No	<observaciones>
	…	…	…	…

Estilo y estética

El contenido deberá seguir la Guía de Estilo del contenido de Ceibal. Asimismo, se deberán cuidar los aspectos estéticos del contenido.

Justificación

La uniformidad en el estilo le dará personalidad al contenido de Ceibal, haciendo que este sea identificable. Cuidar la estética no sólo es importante para capturar la atención del usuario, sino que aporta directamente a la calidad del producto.

Testing

Se deberá recorrer el contenido verificando que es consistente con la Guía de Estilo de Ceibal. Se deberá anotar cualquier discrepancia o desprolijidad estética.

Ficha de testing

Tipo de contenido	Módulo	Discrepancia o desproligidad	OK	Observaciones
<sitio web, juego, presentación, etc>	<zona del contenido a testear>	<anotar discrepancia y lugar exacto>	Sí/No	<observaciones>
…	…	…		…

Requerimiento: compatibilidad con Sugar

Las aplicaciones diseñadas para correr sobre el entorno Sugar (aplicaciones Python) deberán cumplir con la filosofía de Sugar, incluyendo los puntos siguientes:

- Empaquetamiento e instalación adecuada como actividad. Ver el sitio http://wiki.laptop.org/go/Activities para los detalles técnicos del empaquetamiento de actividades.
- Hacer uso adecuado del Diario.
- Poseer un ícono adecuado.
- Contener una barra de herramientas con los comandos adecuados a la aplicación. La barra debe incluir como mínimo el comando Parar (salir de la aplicación).

Justificación

Seguir las reglas de Sugar ayuda a uniformizar el manejo de las aplicaciones para que éste sea más fácil de entender.

Notas

La especificación completa del comportamiento bajo Sugar escapa al objetivo de esta guía. Consultar la documentación técnica de OLPC para más detalles sobre el SDK.

Sin embargo es especialmente importante el adecuado uso del Diario. Por uso adecuado se entiende que si una aplicación guarda información en el almacenamiento persistente de la XO, la misma deberá aparecer necesariamente en el Diario, y la aplicación deberá ser lanzada cuando el documento correspondiente sea abierto.

Testing

Se deberá verificar en forma general que la aplicación es compatible con Sugar. Se verificará especialmente la compatibilidad con el Diario.

Ficha de testing

Ítem	OK	Observaciones
La aplicación se instala correctamente haciendo click sobre el paquete de la actividad utilizando un pen drive desde el Diario.	Sí/No	<observaciones>
La información guardada aparece en el Diario con la fecha, hora, ícono y descripción correctas.	Sí/No	<observaciones>
La información guardada en el diario es abierta correctamente por la aplicación cuando se hace click sobre el ícono.	Sí/No	<observaciones>
El ícono de la aplicación aparece correctamente en la lista de aplicaciones.	Sí/No	<observaciones>
El ícono de la aplicación aparece correctamente y “flashea” al abrir la aplicación.	Sí/No	<observaciones>
La aplicación se despliega correctamente luego de abrirla desde la lista de aplicaciones.	Sí/No	<observaciones>
La aplicación sale correctamente.	Sí/No	<observaciones>
La aplicación reacciona como se espera ante los comandos del marco se Sugar.	Sí/No	<observaciones>

Requerimiento: lenguajes de programación

Las actividades Sugar deberán ser escritas en la última versión estable disponible de Python.

Las aplicaciones Flash deberán ser compatibles con Adobe Flash Player 10 y los scripts deberán ser compatibles con Action Script 3.0 o superior.

Justificación

Python es el lenguaje de programación estándar para el entorno de OLPC y Sugar. Utilizar Python, además de la ventaja de ser multiplataforma, estandariza la producción de software facilitando el intercambio con otros miembros de la comunidad de desarrollo.

Si bien no existe un estándar formal para el contenido flash, sí existe un estándar mundial de hecho, que es Adobe Flash Player 10. Action Script 3.0 es el último lenguaje de scripts

para Flash y posee características que lo hacen adecuadas para desarrollar software de buena calidad.

Testing

Es evidente que este requerimiento se cumplirá trivialmente implementando la aplicación en los lenguajes mencionados. Sin embargo se incluye un test por completitud y a modo de lista de verificación que además sirve para que un tercero pueda también correr los tests.

Actividad: verificar que el código fuente está escrito en Python.

Aplicación Flash: verificar que los scripts están escritos en Action Script 3.0.

Ficha de testing

Tipo de aplicación	Lenguaje de programación	Versión del lenguaje	OK	Observaciones
Actividad/Flash	<lenguaje >	<versión>	Sí/No	<observaciones>

Meta: compatibilidad con OLPC

Es deseable que el contenido instalable en XO desarrollado para Ceibal sea compatible con OLPC.

Justificación

La compatibilidad con los estándares de OLPC posibilita que los contenidos sean incorporados por otros países y eventualmente sean distribuidos en forma estándar por OLPC. Esto puede ser deseable si se pretende que una aplicación trascienda el Plan Ceibal y sea adoptado por otras organizaciones.

Notas

Ver el sitio http://wiki.laptop.org/go/Developers para obtener información de los requisitos de OLPC.

SITIOS DE JUEGOS

En esta sección se describe una propuesta para implementar un sitio de juegos para Ceibal. La misma se da como una serie de requerimientos que se verificarán por construcción y por inspección visual.

Introducción

El sitio de juegos propuesto tiene las siguientes características:

- Registro y autentificación de usuarios.
- Un conjunto de juegos multijugador en línea.
- Un conjunto de juegos de un solo jugador, ya sea del tipo solitario o para jugar contra la máquina.
- Un sistema de “ratings” para juegos competitivos.

- Una base de datos para guardar los perfiles de usuario que incluye su "rating" por cada juego.
- Un servidor de juegos con un protocolo de autentificación que se encargará de la distribución de la información entre los distintos clientes.
- Un módulo cliente que se utilizará como base de los juegos que se encargará de la comunicación con el servidor.
- Despliegue de los juegos en un navegador de internet con información sobre el "rating", cantidad de usuarios, etc.

En adición a los requerimientos detallados a continuación, el sitio y sus juegos, como cualquier contenido para XO, deberán respetar los criterios y requerimientos detallados en las secciones previas.

Algunos de los requerimientos son en realidad links a la información contenida en la documentación técnica del Servidor de Juegos o del Módulo Cliente.

Requerimientos

Requerimiento: autentificación de usuarios

El sitio de juegos deberá proveer una forma de autenticar a los usuarios del sitio basado en la usual combinación usuario/contraseña.

Justificación

Es imprescindible contar con un método de autentificación ya que los usuarios tendrán perfiles con información propia de cada uno. Es necesario prevenir que un usuario tome el lugar de otro.

Notas

La autentificación será exigida para cualquier juego que termine modificando el perfil del usuario. También será exigida para mostrar información del perfil.

Requerimiento: registro de usuarios

El sitio de juegos deberá proveer una manera en la que los usuarios puedan registrarse mediante usuario y contraseña.

Justificación

Los usuarios deben registrarse a sí mismos en el propio sitio de juegos.

Requerimiento: información desplegada

El sitio de juegos deberá desplegar por lo menos la siguiente información:

- Cantidad de usuarios conectados.
- Descripción de cada juego.
- Rating del usuario.
- Número de derrotas y victorias.
- Posición relativa del usuario con respecto a los otros usuarios.

- Cantidad de veces que el usuario jugó a un juego dado.

Requerimiento: base de datos

El sitio de juegos deberá interaccionar con un manejador de base de datos. La base de datos guardará los perfiles de usuario y los juegos disponibles, así como cualquier información necesaria para la correcta presentación de los mismos.

Justificación

Utilizar una base de datos es la forma estándar de implementar el "back-end" de un sitio web basado en usuarios. Tener los juegos y su información guardados en la base de datos permite implementar correctamente la consistencia de los datos.

Notas

- La base de datos podrá ser compartida o no con otras aplicaciones, por ejemplo, puede ser una parte de la base de datos de Ceibal.
- En caso de ser conveniente, y siempre que se asegure la consistencia de los datos, los juegos podrán ser manejados por el gestor de contenidos.

Requerimiento: despliegue automático

Los juegos se deberán desplegar en la pantalla de forma automática mediante páginas dinámicas.

Justificación

El despliegue automático a partir de los datos contenidos en la base de datos es una forma de asegurar la consistencia de los datos y la facilidad del mantenimiento. De esta forma se podrán agregar o quitar juegos en la base de datos y los cambios se verán reflejados automáticamente en la página web.

Requerimiento: compatibilidad con el Servidor de Juegos

La base de datos deberá ser compatible con el Servidor de Juegos:

- Las tablas y campos deberán ser los requeridos en el documento de diseño del servidor.
- El manejador de base de datos deberá ser MySQL.

Justificación

Dado que el Servidor de Juegos accede directamente a la base de datos, es necesario que la misma sea compatible.

Notas

- En caso de que la base de datos sea compartida y que la misma no sea MySQL, será necesario implementar una interfaz o bien modificar el Servidor de Juegos. Dicha modificación debería ser trivial ya que todos los accesos a la base de datos están aislados en un módulo especializado.

- Consultar la documentación técnica del Servidor de Juegos para los detalles de compatibilidad con la base de datos.

Requerimiento: compatibilidad del cliente

El cliente, en el caso de ser un juego multiusuario, deberá ser compatible con el Servidor de Juegos. Para ello deberá utilizar el Módulo Cliente para interaccionar con el servidor.

Justificación

Utilizar el Módulo Cliente en lugar de acceder directamente al Servidor de Juegos asegura la correcta implementación de la interfaz con el servidor además de facilitar el mantenimiento y disminuir el tiempo de desarrollo.

Notas

- La compatibilidad del cliente con el Módulo de Juegos implica directamente que los juegos multiusuario deberán estar programados utilizando ActionScript 3.0.
- Consultar la documentación técnica del Módulo Cliente para los detalles de utilización e integración de dicho módulo.

Requerimiento: plataforma tecnológica

La plataforma tecnológica tiene algunas herramientas obligatorias y otras opcionales a saber:

- Obligatorias
    - Implementación de juegos en línea: Flash, compatible con Adobe Flash Player 10 y ActionScript 3.0.
    - Manejador de base de datos: MySql (salvo necesidad de otra, en cuyo caso se adaptará el módulo de acceso a la base de datos del servidor).
- Opcionales
    - PHP como generador de páginas dinámicas y lenguaje para interactuar con la base de datos.
    - Linux o Windows como sistema operativo de los servidores.

Justificación

Los puntos obligatorios parten de la necesidad de compatibilidad con el Servidor de Juegos y el Módulo Cliente.

Los puntos opcionales parten de las recomendaciones hechas en el documento Plataforma Tecnológica.

Notas

El requerimiento de utilizar Flash como herramienta para implementar los juegos en línea es mucho más fuerte que el de utilizar MySQL como servidor de base de datos. Para más detalles sobre este tema ver el documento “Discusión sobre Flash”.

ENTREGABLES

Para cada contenido se deberá entregar el material descrito en la siguiente ficha. Los ítems obligatorios están marcados con negrita.

Ficha de entrega

Material	Obligatorio	Entregado	Observaciones
Esta ficha	Sí	Sí/No	<observaciones>
Versión final del contenido	Sí	Sí/No	<observaciones>
Ficha de contenidos	Sí	Sí/No	<observaciones>
Programas y procedimientos de instalación	Si corresponde	Sí/No	<observaciones>
Documento de requerimientos incluyendo público objetivo	Sí	Sí/No	<observaciones>
Documentación técnica: diseño global	Sí	Sí/No	<observaciones>
Documentación técnica: diseño detallado	Según propiedad intelectual	Sí/No	<observaciones>
Código fuente	Según propiedad intelectual	Sí/No	<observaciones>
Fichas de resultados de los tests de requerimientos del contenido para XO	Sí	Sí/No	<observaciones>
Protocolo de test de requerimientos específicos del contenido con sus resultados	Sí	Sí/No	<observaciones>
Manual de uso	Sí		

Notas:

- Aún cuando el contenido no sea una aplicación cabe la entrega de un diseño global. Por ejemplo, si se trata de un sitio web el diseño global será un mapa del sitio y una descripción general de su estructura. Si se trata de un artículo o de un libro el diseño global será un resumen del contenido. Si es una imagen o video el diseño global puede describir en forma general el contenido y algunos detalles de su producción.

- El manual de uso también aplica para contenidos que no son aplicaciones. En este caso será el "intended use" del material, por ejemplo, en qué ámbito o bajo qué condiciones se debería consumir el contenido o en qué lugar debería ser expuesto. El manual de uso debe incluir instrucciones específicas para la XO si las hay.

Ing. Eduardo Pérez Rico

Mayo de 2009