Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Aplicación para la creación de animaciones web con HTML5

Autores: Wendy Reyes Jiménez,

Orelbis Lago Vasallo

Tutores: Ing. Yunior Orosa Velázquez,

Ing. Javier Bandomo Ruiz

Facultad 4

La Habana, Cuba

Junio 2014

Declaración de Autoría

Declaramos que somos los únicos autores del trabajo “Aplicación para la creación de animaciones web con HTML5” y autorizamos a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ___ días del mes de _____ del año _____.

Autor: Wendy Reyes Jiménez

Autor: Orelbis Lago Vasallo

Tutor: Ing. Yunior Orosa Velázquez

Tutor: Ing. Javier Bandomo Ruiz

Agradecimientos

Wendy:

A mis padres por apoyarme y confiar en mí.

A mi Yau por desde el primer día retarme a ser una mejor profesional y una mejor persona, por ver en mi lo que nadie más, por estar siempre a mi lado apoyándome y por muchas muchas cosas más. Te amo.

A mis pequeñas molestias Meli, Albert y Gise porque son un pedacito de mí, y forman parte de todo cuanto he alcanzado.

A mis abuelos, Amparo, Isela, Alberto y Jiménez, que de cerca o lejos siempre están ahí para mí.

A mi segunda mamá que aunque esté lejos físicamente nunca se ha apartado de mi lado, junto con mi tío Rey y mis otras hermanas Glen y Vane.

A mis buenos amigos de todos estos años que me han soportado y hasta me quieren a pesar de todo, Yile, Ariel, Odenys, Nany y todos los demás.

A mi compañero de tesis que trabajó conmigo por alcanzar esta meta y que ni buscándolo lo hubiese encontrado mejor y mi tutor Yunior por guiarme en todo este año aunque a veces lo volviera loco.

Y a todos los que de una forma u otra me ayudaron a llegar hasta aquí. Gracias!

Orelbis:

A mis padres que son los que lograron darme el apoyo que necesitaba en el momento que lo necesitaba, a mi hermano que ha confiado en mí y en mis decisiones, que le debo parte de la motivación a querer ser universitario, a mi novia que también ha depositado su confianza en mí, a mis tíos, primos, y demás familiares que hacen difícil escribir estos agradecimientos.

A mis amigos que vienen conmigo desde primer año y con los que he compartido todos los momentos buenos y malos de la universidad, a Nao, Flaunder, Marlonete, Robertón, Wendiola y todos los demás que me faltan por mencionar.

Agradecerle al killer Yausolo que fue el principal guía del desarrollo y de la calidad del software.

Y a mi compañera de tesis que sin ella no hubiera podido lograr este reto que tenía en mi vida que era graduarme de universitario, aunque a veces pensaba que estaba en el servicio militar otra vez, sin su exigencia no lo hubiera logrado. A todos ustedes les dedico mi título ¡cuando me lo den!, aunque muchos no sepan programar. Gracias.

Resumen

En la Universidad de las Ciencias Informáticas el proceso de creación de animaciones web para software educativo se realiza sin el uso de herramientas especializadas para ello. El presente trabajo consiste en el desarrollo de una aplicación web multiplataforma, de código abierto y libre para la creación de animaciones web en 2D. Su propósito fundamental es crear una interfaz visual para facilitar el proceso de creación de animaciones web, además de ayudar a los desarrolladores y diseñadores a aprovechar las ventajas del estándar HTML5 y la Web. El desarrollo del software fue guiado por la metodología XP. La implementación se realizó con los lenguajes HTML5, CSS3 y JavaScript en el IDE NetBeans. UML se utilizó como lenguaje de modelado en la herramienta Visual Paradigm for UML. Fueron empleados para el desarrollo los frameworks jQuery, Paper.js, Bootstrap y qUnit, además se empleó Node.js como entorno de programación en la capa del servidor. En su primera versión, la aplicación permite el diseño de animaciones mediante un conjunto de herramientas de dibujo y transformación, controladas a través de una línea de tiempo con capas y fotogramas. Su principal funcionalidad es exportar la animación final como código JavaScript, de esta manera puede ser visualizada en la Web sin necesidad de plugins o frameworks. La investigación finaliza con las pruebas funcionales del software, las cuales demostraron la correcta implementación de todas sus funcionalidades.

Palabras clave: animación web, aplicación web, canvas, HTML5, JavaScript

Índice

Índice de imágenes

Índice de tablas

Introducción

Iniciando el siglo XXI surge un nuevo modelo para el intercambio de información que acuñó la Web 2.0, este logró que la Web (*World Wide Web*) se volviera fácil de utilizar y que a la vez ganara prestigio, organización y popularidad. Nuevas ideas como blogs, wikis, redes sociales y otros sitios web, favorecieron que tecnologías existentes tales como HTML (*Hypertext Markup Language*), CSS (*Cascading Style Sheet*), PHP (*Hypertext Preprocessor*) y JavaScript evolucionaran a nuevas versiones más potentes, propiciando la aparición de nuevos *frameworks* especializados en estos lenguajes, haciendo más fácil su desarrollo y propiciando el incremento de su uso a nivel global.

Con la evolución de la Web y debido a la necesidad que existe de atraer a los consumidores, crece la variedad de recursos utilizados para aumentar la calidad de las aplicaciones web. Uno de ellos son las animaciones web, que tienen la capacidad de atraer e impresionar a un usuario cuando visita un sitio web. Estos componentes visuales han evolucionado desde rústicas formas de desarrollarlos hasta avanzadas herramientas que facilitan y simplifican esta labor.

El primer tipo de animación en hacerse popular en la Web fue la animación GIF (*Graphics Interchange Format)* que aún es utilizada por su simplicidad para trabajar, además de ser reconocida automáticamente en casi todos los navegadores web, su principal limitación es que necesita mantener las animaciones sencillas por el elevado peso de los archivos, y esto afecta la fluidez de las animaciones. Los desarrolladores y diseñadores web consideran su uso inadecuado para comunicar ideas complejas y añadir sensación de movimiento real a los sitios web.

Otra forma de proporcionar una animación web es usando DHTML (*Dynamic* HTML), al igual que las animaciones GIF es reconocido automáticamente por la mayoría de los navegadores, pero es mucho más complicado programar animaciones usando DHTML que funcionen correctamente en todos los navegadores web, aunque existen herramientas que facilitan su desarrollo, algunas de las más conocidas son: Adobe Flash, Adobe Edge y Sencha.

Flash es una herramienta que evolucionó el mundo web y se convirtió rápidamente en el estándar multimedia de Internet. Sin embargo, HTML5 ya se considera su sustituto natural y el nuevo estándar para multiplicar la interactividad de los proyectos web.

Grandes empresas ya utilizan HTML5, tales como Facebook, LinkedIn, Google y Apple. Con la incorporación de la etiqueta <canvas>, HTML5 abre la posibilidad de definir un área para incluir *scripts*,

dibujar y realizar representaciones de formas en dos dimensiones (2D) sobre un lienzo o *canvas*. Esta nueva funcionalidad aparentemente complementaria abre la Web al mercado del videojuego, el diseño y la animación nativa, sin necesidad de *plugins* de terceros, eliminando las dependencias de plataformas propietarias como Flash. HTML5 está disponible en una multitud de plataformas y funciona muy bien en computadoras, teléfonos móviles y tabletas. El desarrollo de aplicaciones con HTML5 proporciona una experiencia de usuario integrada con el dispositivo (móvil, tableta, computadora de escritorio, portátil), además de ser más fáciles de desarrollar y mantener que las aplicaciones clásicas. HTML5 se ha convertido en el estándar multimedia de la Web por ser adaptable, flexible, escalable y multiplataforma.

La Universidad de las Ciencias Informáticas (UCI) está llamada a respaldar la economía nacional mediante la producción de software tanto para el mercado nacional como internacional. La industria de software cubana cuenta con muchos retos sociales y comerciales para su avance en convertir al sector en una de las principales fuentes de ingreso al país. Cuba, por encontrarse embargada, no puede hacer uso de la mayoría del software privativo que se produce. A pesar de las muchas limitaciones comerciales que afectan la economía, se han trazado planes de migración hacia nuevas tecnologías que garanticen libertad de comercio de software con el mundo así como independencia tecnológica.

Dentro de la infraestructura productiva de la UCI se encuentran los centros de producción especializados en distintos productos y servicios informáticos. Uno de ellos es el Centro de Tecnologías para la Formación (FORTES) que contribuye al desarrollo de aplicaciones educativas en una línea de desarrollo libre. FORTES cuenta con una serie de productos que utilizan la Web como plataforma de desarrollo y estos tienen la necesidad de crear componentes visuales, y dar vida gráfica al entorno del software para lograr sus objetivos. Las animaciones web son parte imprescindible de estos productos como complemento educativo para captar la atención de los usuarios y lograr un resultado más atractivo e interesante.

Los proyectos productivos del Centro FORTES no cuentan con una aplicación para la creación de animaciones web. Las principales herramientas para el desarrollo de estos componentes visuales se encuentran incluidas en las prohibiciones del bloqueo económico de Estados Unidos de América a Cuba. Concebir una animación web sin hacer uso de herramientas especializadas para ello es un trabajo intenso y tedioso.

A partir de entrevistas realizadas a desarrolladores de los proyectos productivos del Centro FORTES se pudo detectar que: las formas rústicas utilizadas para el desarrollo de animaciones web y la no existencia de una interfaz de usuario para su creación en los proyectos productivos del Centro, provoca una planificación de períodos de desarrollo prolongados, empleando gran cantidad de recursos computacionales, personal y tiempo. Además afecta la estructura organizada del trabajo, los desarrolladores tienen que lidiar con exceso de código que dificulta su desenvolvimiento y trabajo efectivo. Limita la creación de animaciones web sólo a los desarrolladores, estos para crear una animación web deben tener conocimiento de varios lenguajes de programación y *frameworks* disponibles, provocando que un diseñador u otra persona capacitada para desempeñarse en esa función deban poseer conocimientos de programación. Asimismo los programas requeridos para crear las animaciones, dígase entorno integrado de desarrollo, editores de imágenes y navegador web, consumen considerables recursos de la estación de trabajo del desarrollador, y disminuyen su rendimiento. Se suman a estas restricciones el hecho de que al emplear más tiempo durante la creación de dichos componentes visuales se pierdan oportunidades para adquirir otros compromisos de trabajo, que significarían mayor ganancia para el Centro.

Lo mencionado anteriormente de forma general provoca falta de motivación para crear las animaciones web, limita la creatividad de los desarrolladores, restringe a personas capacitadas para crear estos recursos, trae consigo extensos períodos de desarrollo, trabajo excesivo, y pérdida de ganancias en el desarrollo de los productos del Centro FORTES.

A partir de lo fundamentado anteriormente se define como **problema de investigación:** ¿Cómo facilitar el proceso de creación de animaciones web en el Centro FORTES? Para brindarle una solución al problema planteado se declara como **objeto de estudio** el proceso de desarrollo de aplicaciones para la creación de animaciones web. Donde surge como **objetivo general de la investigación** desarrollar una aplicación web que facilite el proceso de creación de animaciones web en el Centro FORTES. El objetivo general planteado será desglosado en **objetivos específicos**, los cuales se enuncian a continuación:

- Definir los requerimientos de la aplicación.
- Diseñar la solución de software propuesta.
- Implementar la solución de software propuesta.
- Probar las funcionalidades de la solución desarrollada.

El **campo de acción** se centra en el proceso de desarrollo de aplicaciones web para la creación de animaciones web con HTML5. Por lo tanto la **idea a defender** de la investigación es que el desarrollo de una aplicación web facilitará el proceso de creación de animaciones web con HTML5, sin necesidad de utilizar herramientas privativas en el Centro FORTES.

Para cumplir los objetivos propuestos, se definen las siguientes **tareas de investigación**:

- Realización de entrevistas a los desarrolladores del Centro FORTES para obtener información sobre el proceso de creación de las animaciones web.
- Elaboración del marco teórico de la investigación a partir del estado del arte referente a las aplicaciones para la creación de animaciones web.
- Levantamiento de aspectos funcionales y no funcionales que debe tener el sistema.
- Realización de las actividades y artefactos que tributen al diseño de la propuesta de solución, de acuerdo a la metodología de desarrollo utilizada.
- Realización de las actividades y artefactos que tributen a la implementación de la propuesta de solución, de acuerdo a la metodología de desarrollo utilizada.
- Realización de pruebas a las funcionalidades implementadas.

Los **métodos científicos de investigación** utilizados en la elaboración de este trabajo fueron los siguientes:

En el nivel teórico:

Para conformar el marco teórico se utilizó el método **analítico-sintético**, que permitió realizar el análisis de la situación problemática, las herramientas para la creación de animaciones web y la documentación existente, para llegar a conclusiones sobre las características que debe tener la solución propuesta, así como la selección de las tecnologías, herramientas y la metodología para el desarrollo de la solución.

En el nivel empírico:

La **entrevista** con la finalidad de obtener información acerca de la creación de animaciones web en los proyectos del Centro. La entrevista de forma no estructurada, aplicada a especialistas, para obtener criterios de expertos sobre el tema.

El presente trabajo está constituido por tres capítulos, los cuales se resumen a continuación:

Capítulo 1: Fundamentación teórica. En este capítulo se hace un análisis de los principales conceptos relacionados con el dominio del objeto de estudio de la investigación y se realiza un estudio del estado del arte de herramientas para la creación de animaciones web. Además se seleccionan las herramientas, tecnologías y la metodología que guiará el desarrollo.

Capítulo 2: Descripción de la solución propuesta. En este capítulo se describe la solución de software propuesta. Abarca las fases de Exploración y Planificación de la metodología de desarrollo XP en la solución de software. Se generan las Historias de usuario, el Plan de iteraciones y el Plan de entrega, donde queda definida la velocidad del proyecto y cuándo el cliente obtendrá cada entregable del producto.

Capítulo 3: Diseño, implementación y pruebas. En este capítulo se describe la arquitectura del sistema, con los patrones aplicados a la solución. Además se generan las Tareas de ingeniería en la etapa de implementación y se realiza el proceso de pruebas generando los casos de pruebas necesarios.

Capítulo 1. Fundamentación teórica

Debido a la constante evolución tecnológica, hoy día, las animaciones son esenciales para enriquecer el diseño de los sitios web, puesto que es muy importante contar con elementos gráficos que lo hagan atractivo para los usuarios. Un sitio web ahora se puede valer de estas herramientas visuales que lo hacen agradable y apreciado por quienes lo visitan. Existen varias tecnologías y herramientas que los desarrolladores y diseñadores web utilizan para crear estos recursos.

En el Capítulo 1 se describen los principales conceptos asociados a las animaciones y las herramientas para su creación. Asimismo se expone el estado del arte de las soluciones existentes que resuelven el problema planteado por la investigación y se analizan sus características positivas y negativas. Se sustenta teóricamente el desarrollo de una aplicación para la creación de animaciones web mediante el estudio crítico de las herramientas, tecnologías, tendencias y metodologías de software en pos de seleccionar las más adecuadas para el desarrollo.

1.1 Conceptos asociados al dominio del problema

Animación, cualquier cambio visual que se produce a lo largo del tiempo. Se pueden modificar varios aspectos de un elemento gráfico para animarlo: posición, tamaño, color, transparencia, entre otros. Es la forma de generar secuencias gráficas en diferentes soportes para representar, a través de la alternación de unas y otras de modo continuo, la ilusión de movimiento aunque en realidad sean todas imágenes estáticas. Al generar cambios en la imagen, se produce en el usuario la sensación de dinamismo y movilidad. Una **animación es 2D** cuando los elementos que interactúan se encuentran en un sistema de coordenadas de dos dimensiones, en el plano XY. (Digital, 2013)

La animación es una buena forma para llamar la atención y agregar interés a una página web, toda animación creada para ser utilizada en la Web es una **animación web**, la cual define un modelo de apoyo a la animación sobre esta plataforma. Las animaciones web suelen utilizarse para publicidad, juegos, *banners*, detalles de diseño, efectos, botones animados, entre otros.

La animación de gráficos bidimensionales en general se basa en el concepto de **fotograma**. La animación se compone por tanto de una secuencia de fotogramas que son mostrados al usuario en orden y uno detrás del otro. Por lo tanto un fotograma es un estado de los elementos que componen la animación en un instante concreto de tiempo. La sucesión de estos fotogramas produce la sensación de movimiento. (Digital, 2013)

En la animación y los gráficos de software, una **capa** se refiere a los diferentes niveles en los cuales se pueden realizar los dibujos y objetos, apiladas una por encima de la otra. Las capas de encima taparán capas inferiores. Trabajar con capas permite componer las escenas de manera que no se necesita que todo encaje a la perfección en cada cuadro, empleando capas se alcanza mayor libertad en la edición, y evita la fusión de los objetos en una escena cuando se necesita que estén separados de las demás. (Sanders, 2014)

La **línea de tiempo**, muestra una sucesión temporal de fotogramas. En cada uno de ellos se agregan los distintos elementos que compondrán la animación. Igualmente, en esta línea de tiempo se decide qué elementos se modifican y cuándo. La línea de tiempo organiza y controla el contenido de una animación a través del tiempo en capas y fotogramas. Por tanto, también se puede entender la línea de tiempo como el lugar en el que se establece la secuencia de fotogramas en el tiempo que compondrán la animación. (Adobe Flash, 2013)

Existen dos métodos principales que define Adobe Flash para crear animaciones: la animación fotograma a fotograma y la animación interpolada. La **animación fotograma a fotograma** cambia el contenido del escenario en cada fotograma y es ideal para las animaciones complejas en las que la imagen cambia en cada fotograma en lugar de moverse por el escenario. La **animación interpolada** se utiliza cuando se va a realizar una animación más simple, donde se selecciona el fotograma inicial y el fotograma final y todos los fotogramas intermedios se crean solos. (Adobe Flash, 2013)

Aplicación web, es un sistema informático que los usuarios utilizan accediendo a un servidor web a través de Internet o de una intranet. Las aplicaciones web son populares debido a la practicidad del navegador web que actualmente está disponible en equipos de escritorio, *notebooks*, celulares y tabletas. La facilidad para actualizar y mantener aplicaciones web sin distribuir e instalar software en la gran cantidad de potenciales clientes es otra razón de su popularidad. Entre sus principales ventajas destacan las siguientes (Desarrollos Web, 2000):

- Multiplataforma: una misma versión de la aplicación puede ejecutarse sin problemas en múltiples plataformas tales como Windows, Linux, Mac, Android, etc.
- Actualización: las aplicaciones web siempre se mantienen actualizadas y no requieren que el usuario deba descargar actualizaciones y realizar tareas de instalación.

- Portabilidad: acceso inmediato y desde cualquier lugar, las aplicaciones basadas en tecnologías web no necesitan ser descargadas, instaladas o configuradas. Son independientes del ordenador donde se utilicen, pueden ser accedidas desde cualquier computadora conectada a la red.
- Menos requerimientos de hardware: pueden funcionar en cualquier equipo que disponga de un navegador web. Esto aplica tanto a celulares, tabletas como computadoras u otros dispositivos modernos.
- Seguridad en los datos: los datos se alojan en servidores ubicados en centros de datos con toda la infraestructura necesaria para asegurar la protección de datos y el funcionamiento constante de las aplicaciones.

1.2 Análisis de soluciones existentes

La creación de animaciones web es uno de los aspectos más importantes para el desarrollo de un software educativo. En la actualidad, existen aplicaciones de alta calidad para la producción de estos componentes visuales, pero en su mayoría son software privativo, además de que no venden licencias legales a Cuba, y aún, en caso de existir la posibilidad de compra, los precios son muy elevados. Existen varias aplicaciones libres y gratuitas para la creación de animaciones web, pero en su mayoría poseen funcionalidades básicas que no responden a la necesidad de crear animaciones web de calidad.

1.2.1 Adobe Flash

Adobe Flash es un paquete de diseño para sitios web, su última versión es *Creative Cloud* (CC), utiliza la tecnología en la nube. Permite integración con otros programas de Adobe como Photoshop o Illustrator. Flash admite la animación interpolada y fotograma a fotograma. Su licencia es privativa. (Adobe, 2014)

Adobe Flash ofrece el lenguaje de *scripts,* ActionScript para crear aplicaciones interactivas, juegos, efectos, interfaces para web, etc. Entre las características que posee Adobe Flash se encuentran herramientas de dibujo vectorial, efectos con vectores, soporte de audio en MP3, transiciones de movimiento, transiciones de forma, entre otras. (Flash Professional , 2002)

Flash presenta muchas desventajas que se ven agudizadas por las nuevas tendencias de emplear HTML5 para el desarrollo de una Web interactiva. Algunos de estos problemas son (Clic, 2002):

- El tiempo de carga. Mientras que una página HTML puede ocupar 10-20 KB como media, una animación Flash ocupa mucho más. Evidentemente depende del contenido que tenga, pero suelen

superar los 100 KB con facilidad. Al ocupar más espacio, el tiempo que tarda en estar visible el contenido de la animación es mayor.

- Flash requiere de plugins para poder visualizarse, y el hecho de no tenerlos instalados impedirá la visualización del Flash.
- Compatibilidad con distintos dispositivos. Cada vez es más frecuente acceder a la web con teléfonos móviles, teléfonos inteligentes y tabletas, y algunos de ellos no soportan Flash (como los que emplean el sistema operativo Android anterior a la versión 2.2).
- Tendencia a su desuso en la web. Con la llegada de HTML5, se solventan muchas de las carencias de las páginas tradicionales que obligaban a usar Flash. Por lo que su uso va en disminución.

1.2.2 Adobe Edge Animate

Adobe Edge Animate es un software HTML con herramientas integradas para la creación de animaciones, es compatible con otros programas de Adobe como Illustrator CC y Photoshop CC. Permite crear animaciones compatibles con móviles y tabletas iOS y Android así como navegadores web de ordenadores (Chrome, Safari, Firefox e Internet Explorer 9+). En Adobe Edge Animate se puede crear contenido o animar elementos en HTML5, CSS3 y JavaScript para páginas web. Su licencia es propietaria y costosa. Se ejecuta sobre las plataformas Windows y Mac. No exporta a *canvas* las animaciones, para esto utiliza los elementos HTML clásicos. (Adobe Edge, 2013)

1.2.3 Sencha Animator

Sencha Animator es una herramienta que permite trabajar en una línea de tiempo y realizar animaciones sin tener que escribir código. La línea de tiempo es similar a la de programas como Flash. Con Sencha Animator, se pueden crear anuncios CSS3 y animar texto e imágenes con transiciones, botones de diseño con degradados y brinda la posibilidad de obtener el código CSS3 de todos los elementos del diseño realizado. El resultado es compatible con la mayoría de los estándares web. Se ejecuta sobre las plataformas Windows, MacOS y Linux. Sin embargo solo utiliza CSS3 para crear y exportar las animaciones lo que restringe la complejidad que puede alcanzar la animación. No es libre y su precio es uno de los más elevados entre las herramientas actuales para la creación de animaciones. Los *demos* ofrecidos hasta el momento por la página oficial, deben ser vistos en navegadores que utilicen WebKit,

como Chrome, debido a los aún notables problemas de compatibilidad con HTML5 y CSS3. (Sencha, 2013)

1.2.4 HTML5 Maker

HTML5 Maker es una herramienta en línea para crear contenido animado. Permite producir con las tecnologías JavaScript y HTML5 animaciones, diapositivas y presentaciones. Así como crear animaciones directamente desde la nube, editar y descargar. La subscripción para el uso de la herramienta no es gratuita, los costos varían de acuerdo al plan escogido al subscribirse. Las animaciones generadas son pesadas, como la herramienta no permite el dibujo y edición, entonces las animaciones solo pueden ser creadas a través de plantillas o imágenes y un conjunto de transiciones que limitan al usuario sólo a ese tipo de movimiento. Si se quiere diseñar una animación personalizada esta no es la herramienta más conveniente. (HMTL5 Maker, 2013)

1.2.5 KoolMoves

KoolMoves es a la vez una herramienta de Flash y HTML5 para la creación de texto y efectos de imagen, juegos, botones de navegación, presentaciones de diapositivas, reproductores multimedia, personajes animados y sitios web. KoolMoves permite programar en ActionScript y exportar como SWF, HTML5, SVG, y marcos para GIFs animados. Cuenta con una biblioteca de efectos y componentes, permite adjuntar audio, formas de relleno con gradientes de color o imágenes, y añadir acciones a los botones y marcos. Proporciona un conjunto de herramientas de dibujo, puede reproducir la animación en varios entornos y genera el código HTML para incluirlo en una página web. Su licencia tiene un costo alto. (KoolMoves, 2008)

1.2.6 Google Web Designer

Google Web Designer es una aplicación web de diseño HTML5 profesional. Por el momento se encuentra en formato beta, esta herramienta forma parte de los esfuerzos de la compañía por ayudar a los desarrolladores y diseñadores a aprovechar las ventajas del estándar HTML5. Permite crear contenido animado al tiempo que el código puede verse y editarse, viendo los cambios de manera casi instantánea. Los usuarios también pueden crear y manipular contenido en 3D, al tiempo que rotan objetos y diseños en 2D a lo largo de cualquier eje. Además se pueden importar componentes de cualquier otra suite creativa o utilizar herramientas de ilustración incorporadas, evitando la duplicación de trabajo creado anteriormente. (Google, 2014)

Desde Cuba no se puede acceder a la herramienta por causa de las restricciones del bloqueo económico.

1.2.7 Synfig Studio

Synfig Studio es un software de animación 2D libre y de código abierto, basado en vectores, perfilado para la creación de animaciones pero con una larga curva de aprendizaje. Las animaciones no son creadas fotograma a fotograma sino por interpolación, además de que son exportadas como GIF, aspectos que limitan la complejidad y el rendimiento de la animación final. Synfig Studio está disponible para Windows, Linux y MacOS y está desarrollado en C++. El soporte de la herramienta no es gratuito. (Robert Quattlebaum, 2014)

1.2.8 Moonbase

Moonbase es una herramienta en línea para crear animaciones sencillas en HTML5. Es necesario que el usuario tenga una cuenta creada en el sitio web de la herramienta para poder guardar y gestionar sus proyectos. Permite incluir imágenes arrastrándolas directamente desde una carpeta. Es una herramienta que aún necesita evolucionar dado que sus opciones son muy básicas. El servicio es gratuito actualmente, pero su creador tiene la intención de cobrar por las herramientas destinadas a los profesionales. La empresa quebró y está en oferta para compradores por lo que el sitio oficial no ofrece soporte en la actualidad para la herramienta. (Moonbase, 2012)

1.2.9 Análisis de las soluciones

Después de hacer un estudio del estado del arte de las aplicaciones para el desarrollo de las animaciones web se llegó a las siguientes conclusiones:

Las soluciones analizadas con mayor número de características que se ajustan a la solución del problema de la investigación son aplicaciones propietarias, con licencias costosas, a las que ni pagando su precio, Cuba puede acceder por restricciones comerciales. Además de que el Centro FORTES sigue una línea de desarrollo de software libre por lo que soluciones privativas no tributarían a las tendencias del Centro ni de la UCI. Las soluciones libres estudiadas en la investigación carecen de opciones avanzadas, y solo permiten crear animaciones básicas que no satisfacen las necesidades de los desarrolladores de animaciones del Centro.

Las aplicaciones estudiadas que utilizan HTML5, crean las animaciones con la tecnología para crear gráficos vectoriales SVG (*Scalable Vector Graphics*) y no *canvas,* que tiene mejor rendimiento para este

fin. Pocas de estas soluciones son multiplataforma, por lo que no funcionan sobre sistemas operativos basados en Linux y no tributan a la independencia tecnológica del país.

De las herramientas abarcadas por la investigación, falta una solución completa que permita diseñar animaciones web a la medida, que integre un entorno de dibujo y edición, donde las animaciones no sean sólo creadas por imágenes previamente concebidas en otras aplicaciones, sino que permita su desarrollo en una única herramienta, con opciones avanzadas, desde el graficado hasta la generación de la animación. Que sea de licencia libre y código abierto, multiplataforma y centrada en las necesidades reales de los proyectos del Centro FORTES.

Por estas razones las soluciones estudiadas serán utilizadas solo como referencia para el desarrollo de la solución de software que se implemente. Mediante su análisis, se precisó que la principal tendencia de todas las herramientas es utilizar la plataforma web y encontrarse en la nube. Además se definieron las principales tendencias de diseño y funcionalidades para ser puestas en práctica en la solución.

1.3 Herramientas y tecnologías

Luego de investigar acerca de las soluciones similares y además de tener en cuenta aspectos relevantes para el ciclo de desarrollo, se seleccionan las distintas herramientas y tecnologías para apoyar y regir el proceso de construcción de la aplicación. Se comienza con el análisis de los lenguajes para el desarrollo.

1.3.1 Lenguajes utilizados

Para la implementación de la solución se utilizará como lenguaje de programación JavaScript, que unido con los lenguajes CSS3 y HTML5 es el más utilizado por las soluciones actuales y para el desarrollo de animaciones web. A continuación se describen las características esenciales, que se tuvieron en cuenta para su selección.

1.3.1.1 JavaScript 1.8.5

JavaScript es un lenguaje de script multiplataforma basado en objetos. Es un lenguaje pequeño y ligero; no es útil como un lenguaje independiente, más bien está diseñado para una fácil integración con otros lenguajes, tales como HTML. JavaScript puede ser conectado a los objetos de su entorno para proveer un control programable sobre estos. (Zeldman, 2004)

Técnicamente, es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. Los programas escritos con JavaScript se pueden probar directamente en

cualquier navegador sin necesidad de procesos intermedios. Se define como basado en prototipos y dinámico. (Pérez, 2011)

Las funciones en JavaScript se consideran objetos que contienen métodos y propiedades. *Prototype* es una de estas propiedades. JavaScript utiliza el objeto *prototype* para lograr la herencia, es el que permite manejar el concepto de clases en el lenguaje. Todos los métodos que JavaScript trae definidos son funciones desde el momento que se ejecutan. En JavaScript las funciones pueden manipularse dentro del mismo lenguaje, lo que proporciona la ventaja de que además de ser creadas pueden utilizarse como dato.

1.3.1.2 CSS3

CSS (*Cascading Style Sheets*) es un lenguaje para definir el estilo o la apariencia de las páginas web escritas con HTML. CSS se creó para separar el contenido de la forma, a la vez que permite a los diseñadores mantener un control mucho más preciso sobre la apariencia de las páginas. CSS3 es la versión más reciente de CSS, la novedad más importante que aporta CSS3, de cara a los desarrolladores web, consiste en la incorporación de nuevos mecanismos para mantener un mayor control sobre el estilo con el que se muestran los elementos de las páginas, sin tener que recurrir a trucos, que a menudo complicaban el código de las web. (Vega, 2011)

1.3.1.3 HTML5

HTML5 es la más reciente actualización de HTML, el lenguaje en el que es creada la web. HTML5 también es un término para agrupar las nuevas tecnologías de desarrollo de aplicaciones web: HTML5, CSS3 y nuevas capacidades de JavaScript. La versión anterior y más usada de HTML, HTML4, carece de características necesarias para la creación de aplicaciones modernas basadas en un navegador. El uso fuerte de JavaScript ha ayudado a mejorar esto, gracias a *frameworks* como jQuery. (Vega, 2011)

HTML5 es una tecnología creada para modernizar la web y el desarrollo de aplicaciones web, en línea y sin conexión. HTML5 es en sí mismo una colección de estándares para el diseño y desarrollo de páginas web. En esta versión se ha prestado especial atención a la definición de criterios para mejorar la interoperabilidad. HTML5 ofrece nuevas herramientas y posibilidades para abordar los problemas comunes a los que se enfrentan los desarrolladores web en la Internet más interactiva de hoy en día. (Hickson, 2010)

1.3.2 Análisis de las características de HTML5 para crear las animaciones web

Con HTML5, el desarrollador o diseñador puede usar tecnologías web basadas en estándares para crear aplicaciones y sitios interactivos ricos en gráficos sin tener que usar tecnologías especializadas ni escribir código específico del navegador. Esto mejora, en gran medida, la experiencia del usuario, ya que elimina la instalación de complementos, que se asocia con el 50% de las causas de abandono de los sitios. Actualmente, el navegador proporciona los gráficos en forma nativa. (David, 2010)

HTML5 incluye un conjunto de etiquetas nuevas que potencian sus ventajas entre las que se encuentra <canvas> y <svg>. Esta es característica fundamental que tributa al desarrollo de animaciones con el lenguaje HTML5. Hasta hace poco, no existía ninguna forma estándar de realizar dibujos directamente en el navegador, las alternativas eran muy pocas. Si había que generar gráficos en el navegador, existían opciones pero generalmente estas acarreaban algún tipo de problema, por ejemplo:

Utilizar Flash: *canvas* y SVG son elementos estándares de HTML5 y cualquier navegador debe ser capaz de manejarlos, mientras que Flash es una tecnología de una empresa que requiere instalar en el navegador un *plugin*. A pesar de ser una de las opciones más populares, el soporte en distintas plataformas y arquitecturas es muy variable. Incluso, es uno de los grandes problemas para la estabilidad de los navegadores ya que cada uno requiere *plugins* diferentes.

Usar Java: en particular las *applets*, resulta otra opción problemática, porque requiere un *plugin* de Java (*Java Virtual Machine*), que no está por defecto en el navegador. El tiempo de arranque de los *applets* suele ser mayor que el de Flash.

Crear el dibujo en el servidor, mostrando la imagen resultante en el navegador, y usar Ajax o un mecanismo similar para realizar algún tipo de actualización: es la opción más universal, pero aún ineficiente, porque limita mucho las posibilidades de animación. (Clic, 2002)

1.3.2.1 SVG

SVG (*Scalable Vector Graphics)*, es un modelo de gráficos de modo retenido que persiste en un modelo con memoria que se puede manipular mediante los resultados de códigos en el reprocesamiento. SVG está integrado en el documento con elementos, atributos y estilos. Cuando el elemento <svg> se introduce por primera vez en el documento, se comporta de manera similar a un y es parte de HTMLDocument, pero incluye SVGDocument de interfaz adicional (SVGDocument proporciona una

interacción más profunda y enriquecida con los gráficos vectoriales). SVG es basado en formas y está compuesto por múltiples elementos gráficos, que forman parte del DOM (*Document Object Model*).

SVG contiene atributos de presentación, que son utilizados para modificar los elementos. Es posible asignar un estilo a los atributos de presentación según las reglas de estilo CSS. Otro factor diferencial importante de SVG es la capacidad para codificar la interacción sin complejidad. Así como SVG tiene un DOM programable como HTML, también tiene un modelo de eventos. Su rendimiento es mejor con un número pequeño de objetos, una superficie mayor, o ambos. (Microsoft, 2013)

1.3.2.2 Canvas

Otro enfoque para aportar una experiencia gráfica más rica para los usuarios, es la etiqueta <canvas>. *Canvas* es una poderosa API (*Application Programming Interface*) de bajo nivel que permite que los desarrolladores proporcionen experiencias gráficas nuevas. La manipulación de los gráficos en *canvas* se realiza mediante código JavaScript. Expone una experiencia más programática para dibujar gráficos de modo inmediato, incluidos el rectángulo, la ruta y las imágenes, similar a SVG. La representación de gráficos de modo inmediato es un modelo "*fire and forget*" (dispara y olvida) que muestra gráficos directamente en la pantalla y que, posteriormente, no tiene contexto de lo que se hizo.

A diferencia del modo retenido, no se guardan los gráficos representados; un desarrollador debe volver a invocar todos los comandos de dibujo que se necesitan para describir la escena completa cada vez que se requiere un nuevo marco, independientemente de los cambios reales (por el contrario, SVG es conocido por tener un "gráfico de escenas"). *Canvas* está compuesto por un único elemento HTML, al definir un *canvas* en una página web se crea un lienzo de dibujo o área de dibujo rectangular. Después de crear el dibujo lo que queda es un "mapa de bits", es decir, cada coordenada del lienzo tiene asignado un color, no queda ninguna estructura de lo que contiene el lienzo. *Canvas* no crea objetos vectoriales al estilo de otros entornos como SVG o Flash, sino mapas de bits como una imagen fotográfica. Su rendimiento es mejor con una superficie menor, un número grande de objetos, o ambos. (Microsoft, 2013)

1.3.2.3 Selección de la tecnología para crear gráficos vectoriales

Es posible dibujar prácticamente cualquier gráfico vectorial con las tecnologías *canvas* y SVG, pero, según la tarea, el desarrollador o el equipo deberán trabajar mucho más. Se debe observar que tanto SVG como *canvas* pueden lograr resultados casi idénticos.

Para el desarrollo de la solución de software propuesta se seleccionó la tecnología *canvas*. Se basó la decisión en que el conocimiento, el conjunto de habilidades y los recursos existentes del equipo de desarrollo juegan un papel importante en la elección de tecnologías, y el equipo tiene amplia experiencia trabajando con esta tecnología, y un vasto conocimiento de la API de gráficos de bajo nivel. *Canvas* maneja los gráficos mediante JavaScript, lenguaje de programación dominado por el equipo, lo que disminuye la curva de aprendizaje y garantiza un trabajo más cómodo y profesional.

Asimismo, el rendimiento es de gran importancia en sitios web con muchos gráficos. Es necesario comparar las características de rendimiento de las dos tecnologías. SVG requiere añadir elementos DOM para cada forma que se agregue, por lo que el documento puede rápidamente sobrecargarse con dibujos complejos, y la aplicación se ralentiza o incluso podría bloquearse. *Canvas* es mucho más eficiente y considerada de alto rendimiento. Las animaciones web contienen muchos elementos, y deben mantener su calidad aunque sean complejas. Cuando se observa una gran cantidad de formas diferentes, en imágenes o animación, esto generalmente indica que *canvas* es la tecnología que se debe usar. Dado que SVG comienza a disminuir su tiempo para representar los gráficos a medida que se agregan muchos elementos.

1.3.3 Frameworks

Un *framework* es un producto que sirve como base para la programación avanzada de aplicaciones, que aporta una serie de funciones o códigos para realizar tareas habituales. Es un conjunto de bibliotecas de código que contienen procesos o rutinas ya listos para usar. Los programadores utilizan los *frameworks* para no tener que desarrollar ellos mismos las tareas más básicas, puesto que en el propio *framework* ya hay implementaciones que están probadas, funcionan y no se necesitan volver a programar. (Álvarez, 1999)

Se analizaron los principales *frameworks* existentes en los lenguajes seleccionados para el desarrollo, HTML5, CSS3 y JavaScript. Dentro de los cuales los siguientes fueron los seleccionados por poseer características afines a la propuesta de solución.

1.3.3.1 JQuery 1.9.2

JQuery es un *framework* para el lenguaje JavaScript, ofrece una infraestructura con la que se tiene mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Este *framework* permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos,

desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. Simplemente se deben conocer las bibliotecas del *framework* y programar utilizando las clases, sus propiedades y métodos. JQuery es el *framework* de JavaScript más utilizado, de licencia gratuita para uso en cualquier tipo de plataforma, personal o comercial. (Álvarez, 1999)

1.3.3.2 Bootstrap 3.1.1

Bootstrap es un *framework* de software libre, intuitivo y de gran alcance para el diseño de sitios y aplicaciones web más rápido y fácil. En la actualidad se ha convertido en uno de los proyectos de código abierto más populares en el mundo. Con Bootstrap 2, se añaden funcionalidades adaptables a toda la estructura como una hoja de estilos opcional. Sobre esa base, con Bootstrap 3, se reescribe el *framework* una vez más para hacerlo adaptativo por defecto con un primer acercamiento móvil. Permite crear interfaces que se adapten a los diferentes navegadores, tanto de escritorio como tabletas y móviles a distintas escalas y resoluciones. Se integra perfectamente con las principales librerías JavaScript, como JQuery. (Thornton, 2010)

1.3.3.3 Selección del *framework* para *canvas*, HTML5

Canvas no proporciona una API para enlace de eventos, algo que puede resultar complicado si no se tiene un amplio dominio de esta tecnología. Para mitigar este problema se han creado *frameworks*, según las tendencias actuales los más destacados son EaselJS, Paper.js, Fabric.js, y Processing.js.

Processing.js, creado por John Resig, tuvo un comienzo temprano en 2008. En 2011, se incluyen EaselJS, Paper.js y Fabric.js. A partir de 2013, EaselJS es uno de los *frameworks* más usados, y el empleo de Paper.js está aumentando progresivamente, mientras que el interés por otros como Processing.js ha declinado. (Drowell, 2013)

EaselJS proporciona soluciones para trabajar con gráficos de forma interactiva con *canvas*. Provee una API que es familiar a los desarrolladores de Flash, y abarca las sensibilidades de JavaScript. Provee una lista jerárquica (*Display List)*, igual que Flash, para poder controlar lo que se dibuja en el lienzo. Incluye un modelo de interacción del núcleo, y las clases de ayuda que hacen el trabajo con *canvas* mucho más fácil. Tiene una licencia MIT (*Massachusetts Institute of Technology*), esta licencia permite reutilizar el software así licenciado tanto para ser software libre como para ser software privativo, permitiendo no liberar los cambios realizados al programa original. (CreateJS, 2014)

Por su parte Paper.js es un *framework* de licencia MIT para gráficos vectoriales que se ejecuta en *canvas*, HTML5. Ofrece un escenario gráfico limpio y muchas funcionalidades de gran alcance para crear y trabajar con gráficos vectoriales y curvas de Bézier, todo cuidadosamente incluido en una interfaz de programación consistente y bien diseñada. Paper.js es fácil de aprender para los principiantes y tiene muchas funcionalidades a dominar por los usuarios intermedios y avanzados. Entre las que se encuentran: (Puckey, 2013)

- Un escenario para gráficos vectoriales: trabaja con capas anidadas, grupos, trazados, trazados compuestos, símbolos, etc.
- Una API bien diseñada donde la manipulación y el dibujo de componentes gráficos es automático y optimizado, lo que le permite construir o modificar sus elementos y estilos.
- Paper.js hace el trabajo con vectores y geometría tan simple como sea posible a través de sus tipos básicos tales como punto, forma y rectángulo, donde son posibles las operaciones matemáticas directas utilizando la sintaxis normal de operador en este tipo de objetos como si fueran números planos.
- Permite la mayoría de los modos de mezcla conocidos de las herramientas Adobe Illustrator y Adobe Photoshop apoyados a través de emulación en JavaScript: multiplicidad, pantalla, superposición, luz suave, luz fuerte, oscuros, claros, diferencia, exclusión, tono, saturación, luminosidad, color, sumar, restar, media y negación.

EaselJS y Paper.js son los *frameworks* que más se ajustan a las necesidades de la solución de software. Ambas permiten la creación (dibujo), animación y edición básica (trasladar, escalar, rotar) de cualquier tipo de elemento, así como la interacción con estos. Pero Paper.js brinda mayor volumen de información acerca de los elementos creados, admitiendo una edición más detallada. Ejemplo de esto es que Paper.js permite el cambio de posición de los puntos, puntos de control, o cualquier propiedad que posea el elemento. Por su parte EaselJS se centra más en la creación y animación de los elementos. Además Paper.js facilita la exportación de los elementos creados a SVG y a JSON (*JavaScript Object Notation*), dando así mayores posibilidades para los módulos de cargar, salvar y generar el código, que es de interés para la solución.

Por las características explicadas de Paper.js, se decide utilizar esta biblioteca en su versión 0.9.15 para el manejo de gráficos 2D en la solución de software. Por ser de las bibliotecas estudiadas, la que sus prestaciones mejor se ajustan a las necesidades de la solución de software. Además, el equipo de desarrollo tiene experiencia en el trabajo de esta biblioteca, lo que minimiza la curva de aprendizaje, y acelera el proceso de desarrollo, con una calidad superior.

1.3.3.4 QUnit 1.1

QUnit es un potente framework para realizar pruebas unitarias con JavaScript. Es desarrollado por el equipo de jQuery, su flexibilidad, basada en el propio framework jQuery, facilita la integración de plataformas de terceros que pueden aportar nuevas funcionalidades. Por lo tanto es la mejor opción si este es el framework de JavaScript empleado. (QUnit, 2014)

1.3.4 Selección del Entorno Integrado de Desarrollo

Un Entorno Integrado de Desarrollo (*Integrated Development Environment,* IDE), es un programa informático compuesto por un conjunto de herramientas que facilita el trabajo de los desarrolladores, puede estar orientado a uno o varios lenguajes de programación y brinda facilidades como: resaltado de sintaxis y completamiento de código.

Para la selección del IDE a utilizar, se analizaron los principales IDEs de licencia libre existentes para el desarrollo en los lenguajes de programación antes mencionados.

1.3.4.1 NetBeans 7.3

El proyecto NetBeans está formado por un IDE de código abierto y una plataforma de aplicación disponible para Windows, Mac, Linux y Solaris, que permite a los desarrolladores crear con rapidez aplicaciones web, empresariales, de escritorio y móviles utilizando la plataforma Java, PHP, JavaScript y Ajax, entre otras. El proyecto de NetBeans está apoyado por una comunidad de desarrolladores dinámica y ofrece documentación y recursos de formación exhaustivos.

A partir de la versión 7.3, el soporte de JavaScript ha sido reescrito desde cero, se introdujeron nuevas características para apoyar y mejorar la experiencia de desarrollo con aplicaciones web del lado del cliente que utilizan la familia de tecnologías de HTML5. El apoyo incluye coloración específica de sintaxis, autocompletado de código, refactorización, sugerencias de código y otras características disponibles. También permite controlar las opciones de formato para el lenguaje JavaScript en sí. El IDE proporciona

herramientas que ayudan a depurar y probar archivos de JavaScript. NetBeans permite configurar fácilmente y ejecutar pruebas unitarias en archivos JavaScript utilizando algún *framework* para ello. (NetBeans, 2010)

1.3.4.2 Eclipse 4.0

Eclipse provee un conjunto de herramientas para administrar espacios de trabajo; construir, ejecutar y depurar aplicaciones. Está construido sobre un mecanismo para el descubrimiento, integración y ejecución de módulos llamados *plugins*. Muchos *plugins*, comúnmente sin relación alguna, pueden ser instalados en una misma instancia de Eclipse, y convivir y cooperar sin problemas para ejecutar una tarea. El IDE contiene un tipo de proyecto específico para JavaScript, así como una serie de puntos de vista, editores, asistentes y constructores para este tipo de proyectos. (The Eclipse Foundation, 2001)

1.3.4.3 Selección del IDE a utilizar

El IDE que se decidió utilizar es NetBeans por ser un reconocido entorno de desarrollo integrado libre y de código abierto, apoyado por una amplia comunidad de desarrolladores y mucha documentación. Asimismo incluye resaltado de sintaxis para los lenguajes de programación HTML, JavaScript, CSS, que son los utilizados para la solución de software, brindando depuración y un potente completamiento de código. Ofrece soporte para *frameworks* como jQuery, que es uno de los utilizados, agilizando el desarrollo de la solución. Además posee integración con *plugins* para probar y refactorizar el código JavaScript. Es un IDE, que a diferencia de otros como Eclipse, es fácil de usar con una interfaz de usuario agradable y simple. Además los desarrolladores poseen años de experiencia trabajando en este entorno de desarrollo, aspecto que facilita su desenvolvimiento al implementar la solución de software.

1.3.5 Entorno de programación en la capa del servidor Node.js 0.10.28

Node.js es un entorno de programación en la capa del servidor basado en el lenguaje de programación JavaScript. Fue creado con el enfoque de ser útil en la creación de aplicaciones web altamente escalables, como por ejemplo, servidores web. Node.js utiliza una arquitectura orientada a eventos y es basado en el motor JavaScript V8, que lo hace ligero y eficiente, ideal para aplicaciones en tiempo real de datos que se ejecutan a través de dispositivos distribuidos. Por estas características es seleccionado para publicar la aplicación propuesta, además logra una homogenización en cuanto a lenguaje de programación, al ser este el mismo en el del cliente que en el servidor, aumenta la velocidad de desarrollo

y se logra un mejor dominio por parte de los desarrolladores al no necesitar conocer un lenguaje nuevo para completar los requerimientos del software. (Node.js, 2014)

1.3.6 Herramienta CASE para el modelado Visual Paradigm for UML 5.0

Las herramientas CASE (*Computer Aided Software Engineering*) son un conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas informáticos. Estas herramientas son de ayuda para ingenieros de software, analistas y arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo. Permiten además el modelado de los sistemas mediante diferentes diagramas. (Meza, 2011)

Se seleccionó para el modelado la herramienta Visual Paradigm for UML porque es una herramienta creada para proyectos de software ágiles, potente, multiplataforma y fácil de usar. Soporta los principales estándares de la industria como el Lenguaje de Modelado Unificado (UML). Ofrece un conjunto completo de herramientas, brindando a los equipos de desarrollo de software todo lo necesario para la captura de requisitos, planificación de software, planificación de controles, modelado de clases y modelado de datos. (Visual Paradigm, 2014)

1.3.7 Herramienta para generar el manual de usuario HelpNDoc 4

HelpNDoc es una herramienta gratuita con contiene un potente entorno de creación de ayuda, que puede generar varios formatos de documentación de una sola fuente. Proporciona todas las herramientas necesarias para escribir archivos completos de ayuda, manuales, documentación y libros electrónicos con una mínima curva de aprendizaje. HelpNDoc puede generar archivos estándar de ayuda Windows CHM, documentación basada en Web, sitios web específicos, PDF (*Portable Document Format*) y documentos de Word, entre otros. (HelpNDoc, 2014)

1.4 Metodologías de desarrollo de software

Las metodologías de desarrollo de software son un conjunto de procedimientos y técnicas que apoyan el proceso de desarrollo de productos de software. Surgen por la necesidad de elevar la calidad del software y la eficacia y eficiencia de su proceso de desarrollo. No existe una metodología universal, sino una amplia gama de estas que permite seleccionar la que más se ajuste a los requerimientos del software a implementar.

1.4.1 Metodologías tradicionales

Teniendo en cuenta la filosofía de desarrollo de las metodologías, aquellas con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de metodologías tradicionales o pesadas. Estas metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del producto software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar. Además son poco flexibles y generan gran cantidad de documentación. Se aconseja solo a proyectos de larga duración y gran cantidad de recursos, de lo contrario retrasaría el proceso de desarrollo del software. (Marcela, 2007)

1.4.2 Metodologías ágiles

Las metodologías ágiles o “ligeras” constituyen un nuevo enfoque en el desarrollo de software, mejor aceptado por los desarrolladores de proyectos que las metodologías convencionales (ISO-9000, CMM, etc.) debido a la simplicidad de sus reglas y prácticas, su orientación a equipos de desarrollo de pequeño tamaño, su flexibilidad ante los cambios y su ideología de colaboración. (Hernán, 2004)

1.4.2.1 Programación Extrema (Extreme Programming, XP)

Es la metodología más popular de las relativamente recientes metodologías ágiles. Entre sus principales características se encuentran:

Retroalimentación con el cliente: conceptualmente, al menos uno de los miembros del equipo de trabajo del proyecto, es un cliente. Esto propicia una constante interacción del mismo con el producto en desarrollo.

Cortas iteraciones: en cada iteración, se obtiene un producto listo para entregar y que tiene valor para el cliente. La entrega continua de resultados compromete a ambas partes en la evolución del proyecto e indirectamente influye de forma positiva en la calidad del producto final.

Muy flexible a cambios: una de las características más reconocidas de XP. La constante retroalimentación con los clientes permite prever futuros cambios y evita llegar a momentos que paralizan el desarrollo del producto.

Cuenta con una serie de prácticas destinadas a aumentar la productividad del equipo de trabajo como la programación en pares, reuniones diarias y planes de entrega a corto plazo, por mencionar algunos. “XP es una metodología aplicable a proyectos de corta duración, en el cual efectuar un cambio es más importante que respetar la planificación realizada”. (Walden, 2007)

El ciclo de vida ideal de XP consta de seis fases: Exploración, Planificación de la Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. (Beck, 1999)

1.4.2.2 SCRUM

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. (Canós, 2003)

1.4.2.3 Comparación de las metodologías ágiles

La siguiente tabla (obtenida de (Highsmith, 2002)), compara dos distintas aproximaciones ágiles en base a tres parámetros: vista del sistema como algo cambiante, tener en cuenta la colaboración entre los miembros del equipo y características más específicas de la propia metodología como son simplicidad, excelencia técnica, resultados, adaptabilidad. El autor hace una valoración de uno a cinco basado en el cumplimiento de cada aspecto comparativo por cada metodología, donde la puntuación mínima es uno y cinco la máxima.

Tabla 1. Comparación de metodologías ágiles

	XP	**Scrum**
Sistema como algo cambiante	5	5

Colaboración	5	5
Características de la metodología (CM)		
Resultados	5	5
Simplicidad	5	5
Adaptabilidad	3	4
Excelencia técnica	4	3
Práctica de colaboración	5	4
Media de CM	4.4	4.2
Media total	4.8	4.7

Después del análisis de los diferentes aspectos medidos en la tabla se concluyó que SCRUM y XP varían en cuanto a la adaptabilidad, la excelencia técnica y la práctica de colaboración. XP se destaca como la de mejor promedio según la puntuación por los parámetros proporcionados.

1.4.3 Selección de la metodología de desarrollo

El desarrollo de la solución de software está centrado en satisfacer al cliente mediante entregas continuas, además los requisitos no están bien definidos por lo que pueden variar en el transcurso del tiempo y es necesario aceptar esos cambios sin demasiado costo, las metodologías pesadas no son óptimas para estas condiciones. Además, el equipo de desarrollo es pequeño, compuesto por dos personas y la solución debe ser terminada en un período de sólo cinco meses. Por las características explicadas se determinó que las metodologías tradicionales no se ajustan a las particularidades del desarrollo. Por lo tanto se seleccionó una metodología ágil que presenta aspectos propicios para dirigir el desarrollo.

De las metodologías ágiles estudiadas en la investigación XP es la escogida para guiar el proceso de desarrollo de la aplicación. El equipo tiene experiencia utilizando esta metodología, por lo que se puede trabajar de forma ágil y eficiente en menor tiempo. El equipo de desarrollo es de dos personas permitiendo cumplir con la programación en pares que es una de las prácticas propuestas por la metodología.

Al igual que SCRUM su desarrollo es iterativo e incremental, sin embargo XP no regula las iteraciones a *sprints* de un tiempo fijo como SCRUM permitiendo mayor flexibilidad al equipo. Además, según la

comparación con SCRUM, XP es la mejor calificada de acuerdo a la vista del sistema como algo cambiante, a la colaboración entre los miembros del equipo y la simplicidad, excelencia técnica, resultados y adaptabilidad.

El cliente está disponible para asesorar a los programadores, está incorporado al equipo para ser consultado y observar rápidamente los posibles cambios, aspecto que garantiza gran parte del éxito del proyecto, como lo define XP en su práctica: cliente *in-site*. Al emplear XP se busca la simplicidad de código, una mejor comunicación y una alta calidad en el producto en un mínimo período de tiempo, ajustándose al período de desarrollo que es sólo de 5 meses. Los estándares de código mediante los que se programa con XP garantizan un código entendible, apoyando la poca documentación generada. En la UCI, XP es empleada en muchos proyectos productivos, con profesionales altamente capacitados, y es la mejor documentada comparada con otras metodologías ágiles y esto garantiza una mejor preparación para el desarrollo con esta metodología mediante el estudio de la documentación o la consulta a profesionales.

1.5 Conclusiones del Capítulo 1

En el Capítulo 1 se realizó el análisis del estado del arte de las soluciones existentes, donde se pudo constatar que existen en la actualidad herramientas especializadas para la creación de animaciones web, sin embargo, las que tienen prestaciones deseadas son de licencia privativa, que no pueden ser compradas por la Universidad por causa de resrtricciones comerciales o por su alto precio, además no tributan a la independencia tecnológica y la línea de desarrollo libre que sigue la misma, imposibilitando de esta manera su uso en los proyectos del Centro FORTES.

Como resultado de este análisis se decidió desarrollar una aplicación web especializada para la creación de animaciones web, multiplataforma, de código abierto y licencia libre, que apoye de esta forma la socialización del conocimiento y la independencia tecnológica del país. La implementación se realizará empleando el IDE NetBeans, los lenguajes JavaScript, HTML5 y CSS3 y los *frameworks* jQuery, Bootstrap, Paper.js y qUnit, asegurando de esta forma velocidad en el desarrollo por presentar las características propicias para la solución propuesta, el cual será dirigido y controlado por la metodología XP.

Capítulo 2. Descripción de la solución propuesta

Después de lo expuesto en el capítulo anterior se propone desarrollar una aplicación para la creación de animaciones web en el Centro FORTES. El presente capítulo aborda una propuesta de solución de la aplicación. Se realiza la descripción de las características fundamentales del sistema así como los aspectos no funcionales y funcionales que darán paso a los artefactos generados en las primeras fases de la metodología XP, Exploración y Planificación. El objetivo principal de estas fases es definir el alcance general del proyecto mediante la redacción de Historias de Usuarios, estimar los tiempos de desarrollo en base a esta información para obtener una visión general del sistema, y un plazo total estimado.

2.1 Propuesta de aplicación para la creación de animaciones web

Como propuesta de solución que fundamenta esta investigación se plantea el desarrollo de una aplicación multiplataforma, de código abierto y libre, para la creación de animaciones web en 2D. La misma utilizará la plataforma web, dado que luego del análisis de las herramientas para crear animaciones se determinó que la principal tendencia es a utilizar los beneficios de la Web. La aplicación tendrá como nombre Flavas.

La herramienta debe estar publicada en un servidor para ser accedida y utilizada por los desarrolladores del Centro FORTES desde sus estaciones de trabajo. Estando disponible desde cualquier sitio de la universidad, sin consumir recursos de la computadora, ya que el software no está alojado en cada estación de trabajo. Además una misma versión de la aplicación pueda ejecutarse en varias plataformas y las actualizaciones de futuras versiones, o la corrección de problemas solo se tienen que hacer a la aplicación publicada, y ya instantáneamente todos los usuarios tendrán acceso a la aplicación actualizada. Además la animación se desarrolla en el mismo entorno donde será utilizada, o sea es una aplicación WYSIWYG (*What You See Is What You Get*), "lo que ves es lo que obtienes", por tanto el rendimiento que tiene durante el proceso de desarrollo será el mismo que tendrá una vez producida. Su propósito fundamental será proveer una interfaz visual para facilitar el proceso de creación de animaciones web, además de ayudar a los desarrolladores y diseñadores a aprovechar las ventajas del estándar HTML5.

El sistema debe permitir que los usuarios diseñen animaciones web en el escenario, ya sea mediante imágenes o con herramientas para el dibujo, y la edición de las propiedades de los elementos dibujados. La animación será basada en línea de tiempo, que se encargará de controlar y organizar todo el contenido del proyecto a través de capas y fotogramas. Las animaciones serán creadas fotograma a fotograma que es el método ideal para la creación de animaciones complejas. Una vez concluido el diseño, la animación

será generada en un fichero JavaScript que solo se necesita incluir en la web donde se quiere utilizar. El proyecto puede ser guardado sin terminar y cargado posteriormente para continuar el trabajo en él. A continuación se describen las áreas de trabajo y las opciones que brindan.

Panel de herramientas

A la izquierda tendrá un panel de herramientas el cual contará con las siguientes opciones de dibujo:

- Línea: trazar una línea de un punto a otro.
- Rectángulo: trazar un rectángulo en el área seleccionada.
- Círculo: trazar un círculo en el área seleccionada.
- Pluma: trazar formas.
- Texto: insertar un texto en la posición seleccionada.
- Imagen: insertar imágenes desde una carpeta.
- Relleno: selección del color de relleno antes de dibujar el elemento.
- Borde: selección del color de trazo antes de dibujar el elemento.

Además contiene herramientas para la edición de las propiedades de transformación de los elementos insertados en el escenario.

- Seleccionar: escoger un elemento.
- Mover punto: trasladar un punto de un elemento seleccionado.
- Adicionar/eliminar punto: agregar o excluir puntos en el elemento seleccionado.
- Bézier: editar curva de Bézier del elemento seleccionado.
- Eliminar: borrar un elemento del lienzo.
- Rotar: girar un elemento.
- Mover: trasladar un elemento.
- Escalar: redimensionar un elemento.

Las herramientas de dibujo funcionarán de la siguiente forma: se trazarán al hacer clic en el escenario y arrastrar el cursor por el mismo. Para crear los dibujos, la herramienta centrará su funcionamiento en el *framework* gráfico libre Paper.js.

Panel de propiedades

En la parte superior derecha de la aplicación se mostrará el panel de propiedades en el cual se editarán las siguientes características del gráfico seleccionado:

- Relleno: selección del color de relleno del elemento.
- Borde: selección del color de borde del elemento.
- Ancho de línea: selección del ancho de línea en píxeles del elemento.
- Edición de texto: selección de la fuente y tamaño del texto.
- Cambiar orden de visualización: enviar un elemento al fondo del escenario o traerlo al frente.
- Cambiar dimensiones del escenario: selección en píxeles del ancho y alto del escenario.

Panel de elementos

En la parte inferior derecha se encontrará un listado con los elementos dibujados por capas para su selección o eliminación.

Línea de tiempo

En la parte superior se encontrará la línea de tiempo, encargada de gestionar los fotogramas por capas, que es donde se irá creando la animación. Contiene una cabeza lectora, que es la encargada de mostrar en el escenario los elementos de los fotogramas donde esté situada. Por defecto siempre existen una capa y un fotograma creados. La línea de tiempo permite las siguientes opciones:

- Nueva capa: crear una nueva capa.
- Duplicar capa: clonar la capa seleccionada.
- Nombrar capa: renombrar la capa seleccionada.
- Eliminar capa: borrar la capa seleccionada.

- Visualizar/ocultar capa: esconder todos los elementos de la capa y permitir su visualización nuevamente.
- Bloquear/desbloquear capa: la capa bloqueada no puede ser modificada, se puede desbloquear permitiendo su edición nuevamente.
- Nuevo fotograma: crear fotograma nuevo al final de los fotogramas existentes.
- Duplicar fotograma: clonar el fotograma seleccionado.
- Limpiar fotograma: elimina todos los elementos del fotograma seleccionado.
- Eliminar fotograma: eliminar el fotograma seleccionado.
- Cambiar velocidad de reproducción: variar la cantidad de fotogramas por segundos de la reproducción.
- Reproducir/detener animación: reproducir a la cantidad de fotogramas por segundos seleccionada la animación creada hasta el momento. Detener cuando se desee.

Menú de archivo

Una vez que los usuarios terminen el trabajo con los gráficos, tendrán la opción de abrir la ventana de generación de código JavaScript. Se mostrará una vista previa del código JavaScript de la animación creada que es guardado en un archivo .js, listo para ser incluido en el proyecto web para el que se diseñó la animación, sin necesidad de utilizar ningún *plugin* o *framework*. En la parte superior de la aplicación aparecerá el menú de la misma con las siguientes opciones:

- Nuevo: crear nuevo proyecto.
- Abrir: abrir un proyecto existente.
- Guardar: salvar el proyecto actual como un fichero XML.
- Exportar: generar la animación y exportar el código como un fichero JavaScript.

2.2 Usuarios del sistema

Los usuarios del sistema son todas aquellas personas o sistemas que interactúan con el mismo con el objetivo de obtener un resultado específico. La aplicación web no presenta restricciones de acceso a las funcionalidades, todos los usuarios tienen los mismos privilegios sobre las funcionalidades del software.

Tabla 2. Usuarios del sistema

Usuarios del Sistema	**Justificación**
Diseñador o desarrollador de animaciones web	Puede ser un diseñador de animaciones web sin conocimientos de programación o un desarrollador que utiliza la aplicación para crear animaciones web.

2.3 Planificación del proyecto por roles

Con el objetivo de llevar a cabo el proceso de desarrollo utilizando XP de forma organizada, a continuación se definen los roles de la metodología que serán utilizados en el equipo con sus respectivos responsables.

Tabla 3. Roles de XP

Rol	**Responsabilidad**	**Nombre**
Cliente	Escribe las Historias de Usuario y las Pruebas de Aceptación. Decide cuáles Historias de Usuario se implementan en cada iteración.	Ing. Osmany Montes de Oca Rodríguez
Programadores	Elabora el código del sistema y las pruebas unitarias.	Wendy Reyes Jiménez Orelbis Lago Vasallo
Encargado de seguimiento	Verifica el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones.	Wendy Reyes Jiménez
Jefe (*Big boss*)	Ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas.	Ing. Yunior Orosa Velázquez

2.4 Modelo de dominio

Un modelo de dominio asocia todas las entidades o conceptos que se manejan en el entorno en el que trabaja el sistema mediante un diagrama de clases UML. (Jacobson, 2004)

Son usados frecuentemente en ambientes donde no están bien definidos los procesos de negocio, siendo este el caso de la presente investigación.

2.4.1 Análisis de los conceptos del dominio

Usuario: desarrollador o diseñador que interactúa con el sistema.

Proyecto: es la configuración de la aplicación sobre la que se está trabajando.

Animación: representación de movimiento a imágenes o dibujos mediante su transformación en el escenario en cada fotograma. Es el producto final obtenido.

Herramienta: medio para realizar la animación en el escenario.

Dibujo: las herramientas de dibujo permiten trazar elementos en el escenario.

Transformación: las herramientas de transformación permiten alterar el tamaño, posición, rotación, y otros, de los elementos en el escenario.

Escenario: es el espacio de trabajo donde se trazan e insertan los elementos para crear la animación.

Elemento: objetos que son trazados o insertados en el escenario.

Geométrico: los elementos geométricos básicos son: línea, círculo, rectángulo y formas (trazo libre), que se trazan en el escenario mediante su respectiva herramienta de dibujo.

Gráfico: los elementos gráficos son: imagen y texto, que se insertan en el escenario mediante su respectiva herramienta de dibujo.

Línea de tiempo: organiza y controla el contenido del proyecto a través del tiempo en capas y fotogramas, dando lugar a la animación.

Capa: permiten trabajar la animación en distintos planos independientes, cada capa tiene su propia línea de fotogramas.

Fotograma: representa el contenido de la animación en un instante de tiempo.

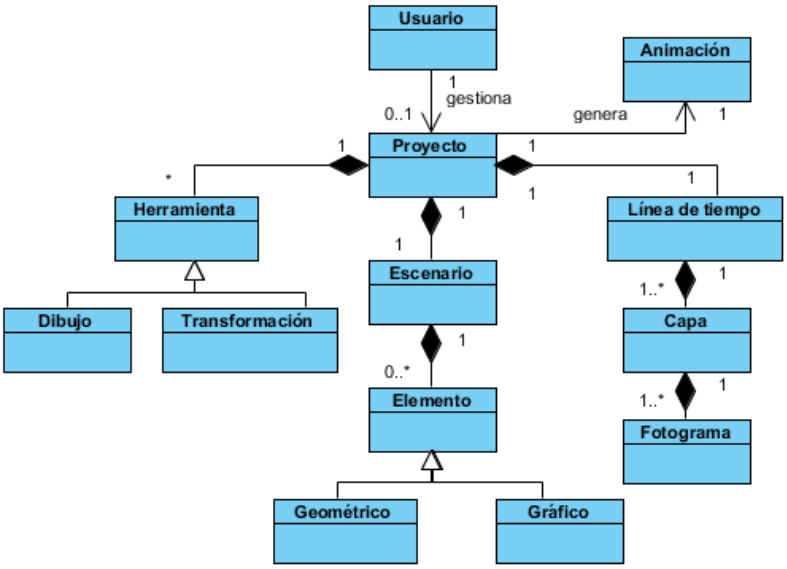

Ilustración 1. Modelo de dominio

2.5 Lista de reservas del producto

Según define la metodología XP la Lista de reservas del producto está compuesta por los requisitos funcionales que debe cumplir la aplicación para satisfacer con eficiencia las necesidades requeridas por el cliente. Es por ello que, redactar un listado de requerimientos bien detallados y que posteriormente puedan ser probados, es tan importante. A continuación se muestra la Lista de reservas del producto.

R1 Dibujar elementos geométricos básicos en el escenario

R1.1 Dibujar línea.

R1.2 Dibujar rectángulo.

R1.3 Dibujar círculo.

R1.4 Dibujar forma.

R1.5 Seleccionar color de borde para el elemento a dibujar.

R1.6 Seleccionar color de relleno para el elemento a dibujar.

R2 Insertar elementos gráficos en el escenario

R2.1 Insertar texto en el escenario.

R2.2 Insertar imagen en el escenario.

R3 Editar propiedades de transformación de los elementos

R3.1 Seleccionar un elemento del escenario.

R3.2 Rotar el elemento seleccionado.

R3.3 Trasladar el elemento seleccionado.

R3.4 Escalar el elemento seleccionado.

R3.5 Editar curva de Bézier del elemento seleccionado.

R3.6 Borrar un elemento.

R3.7 Modificar orden de visualización del elemento seleccionado.

R4 Editar puntos de los elementos

R5.1 Mover un punto del elemento seleccionado.

R5.2 Agregar un punto al elemento seleccionado.

R5.3 Eliminar un punto al elemento seleccionado.

R5 Cambiar dimensión del escenario

R5.1 Cambiar ancho del escenario.

R5.2 Cambiar alto del escenario.

R6 Editar propiedades de estilo de los elementos geométricos básicos

R6.1 Cambiar color de borde del elemento seleccionado.

R6.2 Cambiar color de relleno del elemento seleccionado.

R6.3 Cambiar ancho de línea del elemento seleccionado.

R7 Editar propiedades de estilo del elemento texto

R7.1 Cambiar tipo de fuente del elemento texto seleccionado.

R7.2 Cambiar tamaño de fuente del elemento texto seleccionado.

R7.3 Cambiar contenido del texto seleccionado.

R8 Editar los elementos

R8.1 Copiar un elemento seleccionado.

R8.2 Cortar un elemento seleccionado.

R8.3 Pegar un elemento copiado o cortado.

R9 Gestionar capas

R9.1 Crear capa nueva.

R9.2 Eliminar capa seleccionada.

R9.3 Duplicar capa seleccionada.

R10 Editar propiedades de las capas

R10.1 Visualizar capa seleccionada.

R10.2 Ocultar capa seleccionada.

R10.3 Bloquear capa seleccionada.

R10.4 Desbloquear capa seleccionada.

R10.5 Renombrar capa seleccionada.

R11 Gestionar fotogramas

R11.1 Crear fotograma nuevo.

R11.2 Eliminar fotograma seleccionado.

R11.3 Duplicar fotograma seleccionado.

R11.4 Limpiar fotograma seleccionado.

R12 Controlar animación

R12.1 Reproducir animación.

R12.2 Detener animación.

R12.3 Cambiar cantidad de fotogramas por segundo de la animación.

R13 Gestionar elementos mediante listado

R13.1 Visualizar listado de elementos dibujados por capas.

R13.2 Seleccionar elemento dibujado del listado.

R13.3 Eliminar elemento seleccionado del listado.

R14 Gestionar proyecto

R14.1 Crear nuevo proyecto.

R14.2 Abrir proyecto existente en XML.

R14.3 Salvar proyecto como XML.

R15 Generar código

R15.1 Generar código JavaScript de la animación.

2.6 Aspectos no funcionales del sistema

Los aspectos no funcionales del sistema son propiedades o cualidades que el producto debe tener. Son características para hacer que el software sea más agradable, usable, rápido y seguro.

Apariencia o interfaz externa

- El tamaño de la fuente debe ser mayor de 8 píxeles.
- Los íconos deben ser elementos representativos que identifiquen cada una de las herramientas para el diseño.
- La interfaz de la aplicación debe estar compuesta por cuatro ventanas principales. En el centro el área de trabajo, en la parte superior la línea de tiempo, a la izquierda la barra de herramientas y por último a la derecha las propiedades y composición de las capas.
- La barra de herramientas debe estar organizada por categorías, así como la ventana de propiedades.
- Las herramientas del software deben ser accedidas mediante un acceso directo.

Portabilidad

- Será posible el acceso desde la Web usando cualquier navegador compatible con *canvas*, HTML5. A continuación se mencionan las versiones a partir de las cuales comienzan a tener soporte para el elemento *canvas* los navegadores web más difundidos, tomado de (StatCounter, 2014).

- Firefox versión 2 o superior.
- Chrome versión 4 o superior.
- Opera versión 9 o superior.
- Internet Explorer versión 9 o superior.

Usabilidad

- Existirá un manual de usuario disponible que describa el funcionamiento y uso del sistema al usuario final.
- La interfaz de usuario debe ser intuitiva y sencilla, facilitando el trabajo al usuario.

Rendimiento

- La aplicación deberá ser capaz de importar proyectos salvados en un lapso de tiempo inferior a 30 segundos.
- La aplicación deberá ser capaz de guardar proyectos en un lapso de tiempo inferior a 15 segundos.

Aspectos legales

- La herramienta una vez desarrollada será de código abierto y licencia GPL (*General Public License*).
- Se utilizarán para el desarrollo tecnologías libres y gratis.

Hardware

- Soporte de video que admita resolución de al menos 1024x768.
- Dispositivo de red de al menos 100 Mbits.

2.7 Exploración

La fase de Exploración es el punto de partida de la metodología XP, en ella los clientes plantean a grandes rasgos las Historias de Usuario que son de interés para la primera entrega del producto. De igual manera el equipo de desarrollo en este período se familiariza con las herramientas y tecnologías que serán utilizadas en el proyecto. El tiempo que toma esta fase depende en gran medida de la experiencia de los programadores con la tecnología, que puede llegar a ser de varios meses.

2.7.1 Historias de Usuario

Las Historias de Usuarios (HU) son una descripción de las funcionalidades del sistema, escritas por el cliente. Los desarrolladores realizan una estimación del tiempo que llevará su desarrollo y deben organizarse de forma tal que prioricen las necesidades del cliente, permitiendo al equipo de trabajo identificar las más significativas a desarrollar durante el proceso de implementación de la solución. A continuación se muestran las descripciones de cada una de las HU propuestas por el cliente. No existe un consenso de la información específica que debe contener una HU, se propone utilizar la información que se expone en la siguiente tabla.

Tabla 4. Parámetros de las Historias de Usuario

Historia de Usuario	
No.: Número sucesivo a partir de uno.	**Nombre:** Identifica la Historia de Usuario.
Usuario: El usuario del sistema que utiliza o protagoniza la historia.	
Prioridad en el negocio: Define la relevancia e impacto (baja, media, alta) de la Historia de Usuario para el negocio de acuerdo a las necesidades del usuario.	**Nivel de complejidad:** Define la dificultad técnica (baja, media, alta) que supone desarrollar la Historia de Usuario desde el punto de vista del programador.
Tiempo de desarrollo estimado: Estima la duración de la implementación en semanas.	**Iteración asignada:** Precisa la iteración a la que pertenece la Historia de Usuario.
Descripción: Explica en qué consiste la Historia de Usuario, teniendo en cuenta las acciones realizadas por el usuario y la respuesta brindada por el sistema.	
Observaciones: Información extra que se estime agregar para hacer más comprensible la Historia de Usuario. Por ejemplo: conceptos, pos condiciones, etc.	

Tabla 5. HU1: Dibujar elementos geométricos básicos en el escenario

Historia de Usuario	
No.: 1	**Nombre:** Dibujar elementos geométricos básicos en el escenario
Usuario: Todos	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Alta

Tiempo de desarrollo estimado: 1 semana	**Iteración asignada:** 1
Descripción: El usuario selecciona uno de los 4 posibles tipos de elementos geométricos básicos del panel de herramientas (línea, rectángulo, círculo y forma) y los traza en el escenario seleccionando el color de borde y de relleno con que serán dibujados.	
Observaciones: Da cumplimiento al requisito R1: Dibujar elementos geométricos básicos en el escenario.	

Tabla 6. HU2: Insertar elementos gráficos en el escenario

Historia de Usuario	
No.: 2	**Nombre:** Insertar elementos gráficos en el escenario
Usuario: Todos	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Tiempo de desarrollo estimado: 1 semana	**Iteración asignada:** 1
Descripción: El usuario selecciona el elemento que desea insertar en el escenario, texto o imagen. Si es un elemento texto, selecciona su tipo y tamaño de fuente antes de insertarlo. Si es una imagen busca la carpeta de su localización y la inserta en el escenario.	
Observaciones: Da cumplimiento al requisito R2: Insertar elementos gráficos en el escenario**.**	

Tabla 7. HU3: Editar propiedades de transformación de elementos dibujados

Historia de Usuario	
No.: 3	**Nombre:** Editar propiedades de transformación de elementos dibujados
Usuario: Todos	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Alta
Tiempo de desarrollo estimado: 1 semana	**Iteración asignada:** 1
Descripción: El usuario selecciona uno de los elementos dibujados y puede realizar sobre este las acciones de seleccionar, trasladar, escalar, rotar, eliminar y editar curva de Bézier. Además puede cambiar el orden de visualización del elemento seleccionado (enviarlo hacia el fondo o traerlo al frente de los demás elementos).	

Observaciones: Da cumplimiento al requisito R3: Editar propiedades de transformación de elementos dibujados.

Tabla 8. HU4: Editar puntos de los elementos dibujados

Historia de Usuario	
No.: 4	**Nombre:** Editar puntos de los elementos dibujados
Usuario: Todos	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Media
Tiempo de desarrollo estimado: 0.6 semanas	**Iteración asignada:** 1
Descripción: El usuario selecciona uno de los elementos dibujados y puede realizar sobre este las acciones de mover, agregar o eliminar un punto.	
Observaciones: Da cumplimiento al requisito R4: Editar puntos de los elementos dibujados**.**	

Tabla 9. HU5: Cambiar dimensión del escenario

Historia de Usuario	
No.: 5	**Nombre:** Cambiar dimensión del escenario
Usuario: Todos	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Baja
Tiempo de desarrollo estimado: 0.1 semanas	**Iteración asignada:** 1
Descripción: El usuario puede cambiar el alto y ancho del escenario que por defecto es de 400 píxeles de alto por 600 píxeles de ancho.	
Observaciones: Da cumplimiento al requisito R5: Cambiar dimensión del escenario**.**	

Tabla 10. HU6: Editar las propiedades de estilo de los elementos geométricos básicos

Historia de Usuario	
No.: 6	**Nombre:** Editar las propiedades de estilo de los elementos geométricos básicos

Usuario: Todos	
Prioridad en el negocio: Media	**Nivel de complejidad:** Media
Tiempo de desarrollo estimado: 0.5 semanas	**Iteración asignada:** 2
Descripción: El usuario selecciona uno de los elementos dibujados y puede cambiar sus propiedades (color de relleno, color de línea, ancho de línea).	
Observaciones: Da cumplimiento al requisito R7: Editar las propiedades de estilo de los elementos.	

Tabla 11 HU7: Editar propiedades de estilo del elemento texto

Historia de Usuario	
No.: 7	**Nombre:** Editar propiedades de estilo del elemento texto
Usuario: Todos	
Prioridad en el negocio: Media	**Nivel de complejidad:** Baja
Tiempo de desarrollo estimado: 0.2 semanas	**Iteración asignada:** 2
Descripción: El usuario selecciona un elemento texto y puede cambiar su contenido, tipo y tamaño de fuente.	
Observaciones: Da cumplimiento al requisito R7: Editar propiedades de estilo del elemento texto.	

Tabla 12. HU8. Editar los elementos

Historia de Usuario	
No.: 8	**Nombre:** Editar los elementos
Usuario: Todos	
Prioridad en el negocio: Baja	**Nivel de complejidad:** Baja
Tiempo de desarrollo estimado: 0.2 semanas	**Iteración asignada:** 4
Descripción: El usuario selecciona un elemento puede copiar o cortar este y pegarlo en otro	

fotograma y capa.
Observaciones: Da cumplimiento al requisito R8: Editar los elementos.

Tabla 13. HU9: Gestionar capas

Historia de Usuario	
No.: 9	**Nombre:** Gestionar capas
Usuario: Todos	
Prioridad en el negocio: Media	**Nivel de complejidad:** Alta
Tiempo de desarrollo estimado: 1 semana	**Iteración asignada:** 2
Descripción: El usuario puede crear nuevas capas, además puede realizar las acciones de eliminar y duplicar capas creadas.	
Observaciones: Da cumplimiento al requisito R9: Gestionar capas.	

Tabla 14. HU10: Editar propiedades de las capas

Historia de Usuario	
No.: 10	**Nombre:** Editar propiedades de las capas
Usuario: Todos	
Prioridad en el negocio: Media	**Nivel de complejidad:** Media
Tiempo de desarrollo estimado: 1 semana	**Iteración asignada:** 2
Descripción: El usuario puede renombrar, ocultar y visualizar las capas creadas, también puede bloquear una capa para que no pueda ser editada.	
Observaciones: Da cumplimiento al requisito R10: Editar propiedades de las capas.	

Tabla 15. HU11: Gestionar fotogramas

Historia de Usuario	
No.: 11	**Nombre:** Gestionar fotogramas
Usuario: Todos	

Prioridad en el negocio: Media	**Nivel de complejidad:** Media
Tiempo de desarrollo estimado: 1 semanas	**Iteración asignada:** 3
Descripción: El usuario puede realizar las acciones de eliminar, duplicar y limpiar un fotograma seleccionado. Puede también crear nuevos fotogramas.	
Observaciones: Da cumplimiento al requisito R11: Gestionar fotogramas.	

Tabla 16. HU12: Controlar animación

Historia de Usuario	
No.: 12	**Nombre:** Controlar animación
Usuario: Todos	
Prioridad en el negocio: Baja	**Nivel de complejidad:** Media
Tiempo de desarrollo estimado: 2 semanas	**Iteración asignada:** 3
Descripción: El usuario puede visualizar la animación hasta el fotograma que esté creada, puede detener la animación y cambiar la cantidad de fotogramas por segundo a la que se visualiza la animación.	
Observaciones: Da cumplimiento al requisito R12: Controlar animación.	

Tabla 17. HU13: Gestionar los elementos mediante listado

Historia de Usuario	
No.: 13	**Nombre:** Gestionar los elementos mediante listado
Usuario: Todos	
Prioridad en el negocio: Baja	**Nivel de complejidad:** Media
Tiempo de desarrollo estimado: 2 semanas	**Iteración asignada:** 5
Descripción: El usuario puede visualizar los elementos dibujados en el escenario en un listado, así como seleccionarlos y eliminarlos.	
Observaciones: Da cumplimiento al requisito R13: Gestionar los elementos mediante	

listado.

Tabla 18. HU14: Gestionar proyecto

Historia de Usuario	
No.: 14	**Nombre:** Gestionar proyecto
Usuario: Todos	
Prioridad en el negocio: Baja	**Nivel de complejidad:** Media
Tiempo de desarrollo estimado: 2 semanas	**Iteración asignada:** 5
Descripción: El usuario es capaz de realizar las operaciones clásicas de trabajo con un editor: abrir un proyecto existente, crear uno nuevo y salvar el proyecto actual.	
Observaciones: Da cumplimiento al requisito R14: Gestionar proyecto.	

Tabla 19. HU15: Generar código

Historia de Usuario	
No.: 15	**Nombre:** Generar código
Usuario: Todos	
Prioridad en el negocio: Alta	**Nivel de complejidad:** Alta
Tiempo de desarrollo estimado: 3 semanas	**Iteración asignada:** 4
Descripción: El usuario selecciona la opción generar código, y visualiza el código JavaScript en un formulario, que puede guardar como un fichero .js.	
Observaciones: Da cumplimiento al requisito R15: Generar código.	

2.8 Planificación

La fase de planificación XP la plantea como un permanente diálogo entre las partes involucradas en el proyecto, el cliente, programadores y gerente. Una vez definidas las HU los actores del proyecto establecen un Plan de entregas, que debe ser cambiado en caso necesario, el plan es una estimación de la realidad que si no coincide con esta es totalmente inútil. Una vez acordado este cronograma comienza un Plan de iteraciones donde las HU seleccionadas para cada entrega son desarrolladas y probadas

iterativamente, de acuerdo al orden establecido. Otro aspecto a tener en cuenta durante esta fase es la velocidad del proyecto, la cual se estima utilizando el tiempo empleado en el desarrollo de las iteraciones terminadas.

2.8.1 Estimación de esfuerzo por HU

En la fase de planificación de la entrega se lleva a cabo una estimación del esfuerzo que realizará el equipo de desarrolladores para efectuar cada HU planteada. Se expresa esta estimación utilizando como medida el tiempo en cantidad de semanas. Esta estimación organiza las HU de acuerdo a su prioridad para el cliente y permite mayor claridad al establecer el Plan de iteraciones, además del tiempo estimado se reflejó quién será el programador responsable en la implementación de las HU.

Tabla 20. Estimación de esfuerzo por Historia de Usuario

Asignado a	**No.**	**HU**	**Estimación**	**Estimado por**
Prioridad Alta				
Orelbis Lago Vasallo	1	Dibujar elementos geométricos básicos en el escenario	1.3	Orelbis Lago Vasallo
	2	Insertar elementos gráficos en el escenario	0.4	
Wendy Reyes Jiménez	3	Editar propiedades de transformación de los elementos	1	
	4	Editar puntos de los elementos	0.6	
	5	Cambiar dimensión del escenario	0.1	
Orelbis Lago Vasallo	6	Generar código JavaScript	2.5	Wendy Reyes Jiménez
Prioridad Media				
Wendy Reyes Jiménez	7	Editar propiedades de estilo de los elementos geométricos básicos	0.5	Wendy Reyes Jiménez

	8	Editar propiedades de estilo del elemento texto	0.2	
Orelbis Lago Vasallo	9	Gestionar capas	1	Orelbis Lago Vasallo
	10	Editar propiedades de las capas	1	
Orelbis Lago Vasallo	11	Gestionar fotogramas	1	
Prioridad Baja				
Wendy Reyes Jiménez	12	Controlar animación	2	Orelbis Lago Vasallo
Wendy Reyes Jiménez	13	Gestionar elementos mediante listado	2	Wendy Reyes Jiménez
Orelbis Lago Vasallo	14	Gestionar proyecto	2	
	15	Editar los elementos	0.2	

2.8.2 Plan de iteraciones

XP enfatiza en el carácter iterativo e incremental del desarrollo. Cada HU se traduce en tareas específicas de programación. Las iteraciones generan un resultado de software con un valor para el cliente, agrupan un conjunto de HU a implementar en un período de tiempo. No se recomienda que excedan las 4 semanas en su desarrollo. La entrega con rapidez de módulos al cliente aumenta la retroalimentación y resulta más provechoso en términos de calidad del producto que una entrega a largo plazo.

La primera iteración debe ser un esqueleto de funcionamiento del sistema como un todo. Se escogen una serie de HU sencillas, que obliguen a crear la arquitectura completa. Luego se deben implementar las HU de la forma más simple que puedan funcionar. (Beck, 1999)

Tabla 21. Plan de iteraciones

Iteraciones	**Orden de las HU a implementar**	**Cantidad de tiempo de trabajo**
Iteración 1	HU1. Dibujar elementos geométricos básicos en el escenario HU2. Insertar elementos gráficos en el escenario	4 semanas

	HU3. Editar propiedades de transformación de los elementos HU4. Editar puntos de los elementos HU5. Cambiar dimensión del escenario	
Iteración 2	HU6. Editar propiedades de estilo de los elementos geométricos básicos HU7. Editar propiedades de estilo del elemento texto HU9. Gestionar capas HU10. Editar propiedades de las capas	3 semanas
Iteración 3	HU11. Gestionar fotogramas HU12. Controlar animación	3 semanas
Iteración 4	HU8. Editar los elementos HU15. Generar código	3 semanas
Iteración 5	HU12. Gestionar elementos mediante listado HU13. Gestionar proyecto	4 semanas

2.8.3 Plan de entregas

El Plan de entregas es un documento que especifica las HU que serán implementadas en cada entrega de software y sus prioridades. Debe ser elaborado durante las reuniones de planificación de entregas por el cliente y el equipo de desarrollo en conjunto. Se busca hacer entregas frecuentes para obtener una mayor retroalimentación.

Tabla 22. Plan de entregas

Entregable: Aplicación para la creación de animaciones web con HTML5.					
Final de iteración	Primera	Segunda	Tercera	Cuarta	Quinta
Fecha	10 de febrero del 2014	3 de marzo del 2014	24 de marzo del 2014	14 de abril del 2014	12 de mayo del 2014
Versión	0.1	0.2	0.3	0.4	1.0

2.9 Conclusiones del Capítulo 2

En el Capítulo 2 de la investigación se desarrollaron las dos primeras fases de la metodología XP. Se generaron los artefactos de las HU, Plan de iteraciones y de entrega, correspondientes a la primera etapa del ciclo de vida de la solución propuesta. Concluidas estas fases se cuenta con una descripción de las funcionalidades del sistema y la prioridad de estas definidas por el cliente, centrado en aportar un mayor valor de negocio. Con el Plan de entregas definido el cliente tiene conocimiento de en qué fecha obtendrá cada versión de la aplicación.

Capítulo 3. Diseño, implementación y pruebas

En el presente capítulo se presenta la descripción de la arquitectura de la aplicación siguiendo los buenos principios de emplear patrones. Además se detallan las iteraciones realizadas durante la etapa de codificación del proyecto mediante las tareas en las que se desglosó cada Historia de Usuario y las pruebas aplicadas al sistema. La metodología XP no restringe los artefactos a utilizar, dejando en manos del equipo su selección en pos de facilitar el proceso de desarrollo, los artefactos generados son las Tarjetas CRC, las Tareas de Ingeniería, los Casos de prueba de aceptación y Casos de prueba unitaria.

3.1 Diseño

El diseño en XP no es algo que se hace solo en un momento, sino en toda la vida del proyecto. XP hace énfasis en diseños simples, si alguna parte del sistema es de desarrollo complejo, se divide, un sistema simple se implementará con mayor rapidez que uno complejo. XP lleva a cabo un proceso de mejora continua del diseño, la refactorización, donde se perfecciona y modifica la estructura y codificación del código sin cambiar su funcionalidad. La metáfora del sistema es un término definido por XP, que no es más que el concepto global del sistema a implementar, debe ser clara y entendible para el cliente y además servir de guía a la arquitectura del proyecto. Se generan como artefactos las Tarjetas CRC.

3.1.1 Patrón arquitectónico N-Capas

Los patrones arquitectónicos representan el nivel más alto dentro del sistema de patrones y expresan el esquema de la estructura fundamental de la organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos, específican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos. (Almeira, 2007)

El patrón arquitectónico N-Capas, se define como una organización jerárquica tal que cada capa proporciona servicios a la capa inmediatamente superior y se sirve de las prestaciones que le brinda la inmediatamente inferior. El mantenimiento de mejoras en una solución que utilice N-Capas será mucho más fácil porque las funciones están localizadas y además las capas deben estar débilmente acopladas entre ellas y con alta cohesión internamente, lo cual posibilita variar de una forma sencilla diferentes implementaciones/combinaciones de capas. (Llorente, 2010)

3.1.1.1 Aplicación del patrón arquitectónico N-Capas a la solución

Para el diseño de la arquitectura de la solución se definieron tres capas: Presentación, Lógica de aplicación y Servicios de aplicación.

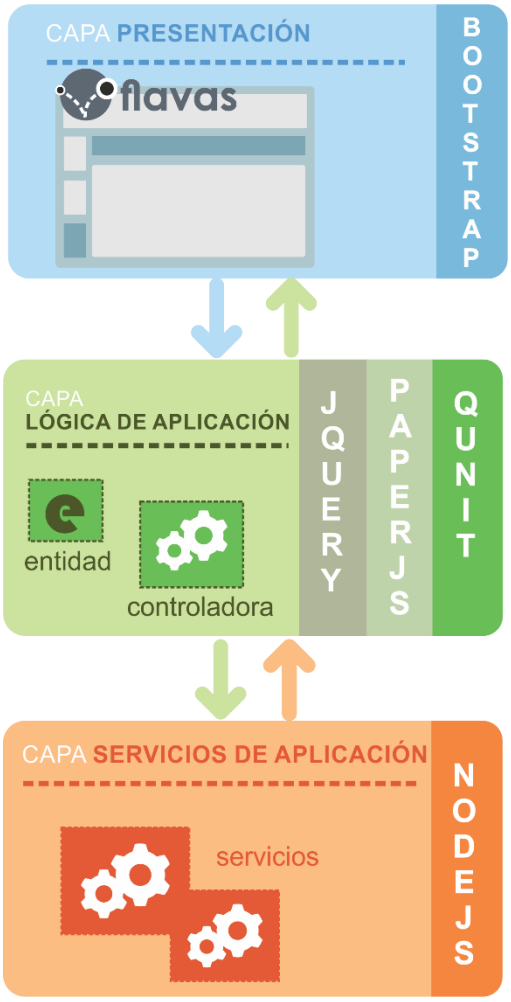

Ilustración 2. Patrón arquitectónico N-Capas

Capa de Presentación

La capa de Presentación es la referente a la interacción entre el usuario y la aplicación. La responsabilidad de esta capa es presentar los conceptos del dominio al usuario mediante una interfaz de usuario. Es necesario facilitar la explotación de la aplicación al usuario final, porque desde su punto de

vista esta capa es la aplicación en sí. La interfaz de usuario necesita para su desarrollo conocimientos de diseño, accesibilidad y usabilidad logrando así una interfaz intuitiva y simple. (Llorente, 2010)

Esta capa se comunica con la capa inmediata inferior, la capa de Lógica de aplicación. La capa Presentación agrupa la página HTML *Index,* que contiene toda la interfaz de usuario de la aplicación y los formularios mediante los que este interactúa con el sistema. Utiliza además el *framework* Bootstrap para dar estilo a la página web.

Capa de Lógica de aplicación

La capa de Lógica de aplicación maneja el funcionamiento de la aplicación. Es la responsable de representar los conceptos del sistema. En esta capa se implementan las reglas del dominio, y se define todo el comportamiento de la aplicación, además contiene la información sobre la situación de los procesos del dominio. Esta capa es el corazón del software. Recibe la información de la capa Presentación y actúa de acuerdo a lo pedido. (Llorente, 2010)

El sistema tiene como característica que la lógica de la aplicación ocurre en el lado del cliente, mediante clases JavaScript. Esta capa contiene las clases controladoras *ControlTimeLine*, *ControlDraw*, *ControlListElement y ControlDocument*. También componen esta capa las entidades fotograma y capa, definidos por las clases del diseño Photogram y Layer. Estos objetos pertenecen a la capa de Lógica de aplicación porque son entidades del dominio independientes de cualquier tecnología de infraestructura de persistencia de datos, como ORMs. Esta capa utiliza los frameworks jQuery, Paper.js y qUnit.

La capa de Lógica de aplicación envía solicitudes a la capa de Servicios de aplicación y esta ejecuta las acciones correspondientes para retornar una respuesta. En caso de que esta respuesta sea requerida para alguna tarea visual el controlador responsable se encarga de manejar los cambios visuales necesarios en la capa de Presentación. A continuación se explican las clases correspondientes a la capa Lógica de aplicación.

Layer es la clase encargada de almacenar y gestionar todos los fotogramas que contiene el proyecto. Permite adicionar nuevos fotogramas, duplicarlos, eliminarlos y controlar la cantidad de fotogramas que contiene cada capa. Mediante los métodos *get* y *set* se acceden a los atributos de las capas.

La clase ***Photogram*** se encarga de almacenar y modificar los elementos que son dibujados en el escenario. Para ello contiene las funcionalidades eliminar o agregar, traer adelante y enviar hacia atrás los elementos. Mediante ella se acceden a los atributos de los fotogramas.

ControlDraw es la clase que controla todas las herramientas de la aplicación, las de dibujo, de transformación y de estilo. Para ello se utiliza el framework Paper.js. Contiene métodos para el dibujo y la edición de elementos, además de funcionalidades para el reajuste de todos los elementos del lienzo una vez cambiado su tamaño. En ella se encuentran los eventos para controlar el mouse cuando interactúa con el lienzo.

La clase ***ControlTimeLine*** se ocupa de la gestión de la línea de tiempo, controlando el comportamiento de las capas, los fotogramas que contiene cada capa y los elementos que contiene cada fotograma, utiliza todas las funcionalidades de las clases *Layer* y *Photogram* además de gestionar la sección formas. Contiene los métodos necesarios para el dibujo, modificación y actualización de la línea de tiempo. Esta clase controla las animaciones reproducidas dentro de la aplicación. En ella se encuentran los eventos para controlar el *mouse* cuando interactúa con la línea de tiempo.

La clase ***ControlListElement*** contiene las funcionalidades para actualizar los elementos dibujados en el escenario en el listado de los elementos que se encuentran en el fotograma y capa seleccionados. Además de los eventos para seleccionar y eliminar elementos a través del listado.

ControlDocument es la clase que gestiona el proyecto. Lo hace mediante las funcionalidades de crear un nuevo proyecto, guardar un proyecto en la cual se genera un archivo XML con los datos necesarios para su posterior carga, y cargar proyecto a partir del archivo XML guardado en el cual se recupera la información necesaria para seguir trabajando en el proyecto cargado.

Capa de Servicios de aplicación

La capa Servicios de aplicación se encarga del funcionamiento de las clases JavaScript que se encuentran en el lado del servidor. Contiene las funcionalidades para proveer operaciones de lectura y escritura en el servidor que son solicitadas por la capa superior. Esta capa utiliza la tecnología Node.js.

3.1.2 Patrones de asignación de responsabilidades

Los patrones GRASP (*General Responsibility Assignment Software Patterns*) constituyen patrones generales de software para asignación de responsabilidades. La calidad de diseño de la interacción de los

objetos y la asignación de responsabilidades presentan gran variación. Las decisiones poco acertadas dan origen a sistemas y componentes frágiles y difíciles de mantener, entender, reutilizar o extender. (Larman, 1999)

3.1.2.1 Aplicación de patrones de asignación de responsabilidades a la solución

Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos. La intención básica del patrón Creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación. (Larman, 1999)

En la solución de software se evidencia el uso de este patrón en el diseño de las clases *ControlTimeLine, Layer* y *Photogram*. La clase *ControlTimeLine* tiene la responsabilidad de crear instancias de la clase *Layer*, mediante el método *createLayer()*, *ControlTimeLine* contiene objetos de *Layer*. Así mismo sucede con las clases *Layer* y *Photogram*, la clase *Layer* instancia un fotograma en su constructor *Layer(),* y además cada vez que se adiciona un fotograma mediante el método *addPhotogram()*.

Bajo acoplamiento: impulsa la asignación de responsabilidades de manera que su localización no incremente el acoplamiento hasta un nivel que lleve a los resultados negativos que puede producir un acoplamiento alto. No se puede considerar de manera aislada a otros patrones como el de Alta cohesión, sino que necesita incluirse como uno de los diferentes principios de diseño que influyen en una elección al asignar una responsabilidad. (Larman, 1999)

El patrón Bajo acoplamiento fue utilizado en el diseño de todas las clases del software, logrando el diseño que mantiene el acoplamiento global más bajo entre clases. Además como se aplica el patrón arquitectónico N-Capas, este exige que exista bajo acoplamiento entre las capas.

Alta cohesión: una clase tiene una responsabilidad moderada en un área funcional y colabora con otras clases para llevar a cabo las tareas. Una clase con alta cohesión tiene un número relativamente pequeño de métodos, con funcionalidad altamente relacionada, y no realiza mucho trabajo. Colabora con otros objetos para compartir el esfuerzo si la tarea es extensa. (Larman, 1999)

En el diseño de las clases de la solución se buscó que cada clase tuviera las responsabilidades específicas de un área sin que se sobrecargaran de tareas, manteniendo alta la cohesión. Existe una clase controladora especializada en cada parte del sistema. Se realizó el diseño logrando que las clases

colaboren entre sí para llevar a cabo las tareas, teniendo responsabilidades moderadas en espacios funcionales con una alta cohesión.

Controlador: un Controlador es un objeto que no pertenece a la interfaz de usuario, responsable de recibir o manejar un evento del sistema que define el método para la operación del sistema. Un Controlador recibe la solicitud del servicio desde la capa de Presentación y coordina su realización, normalmente delegando a otros objetos. (Larman, 1999)

En la solución, en la capa Lógica de aplicación se encuentran las clases de control, estas clases reciben los eventos del sistema de la capa Presentación y en la capa Lógica de aplicación el controlador responsable define el método para esta operación del sistema. Así, la lógica de la aplicación está separada de la interfaz de usuario, esto permite aumentar la reutilización de código y a la vez tener un mayor control de la aplicación. Los eventos del sistema están divididos en varias clases controladoras para poder aumentar la cohesión y disminuir el acoplamiento.

3.1.3 Patrones de diseño

Los patrones de diseño son soluciones bien documentadas que los desarrolladores emplean para dar solución a nuevos problemas apoyados en la experiencia de haberlas utilizado con éxito en el pasado. Dentro de los patrones de diseño existen variaciones según su nivel de granularidad y abstracción, lo que permite clasificarlos bajo dos criterios: ámbito, específica si un patrón se aplica primariamente a una clase o a un objeto; y propósito, refleja qué hace un patrón teniendo en cuenta si es de creación, estructural o de comportamiento.

Creación, abstrae el proceso de instanciación de objetos, su misión es permitir construir sistemas independientes de la forma de creación, composición o representación de objetos.

Estructura, controla como se componen las clases u objetos en la construcción de estructuras mayores.

Comportamiento, se relaciona con algoritmos, la forma en la que interactúan las clases u objetos y la asignación de responsabilidades entre ellos. (Almeira, 2007)

3.1.3.1 Aplicación de los patrones de diseño a la solución

JavaScript no es un lenguaje de Programación Orientada a Objetos, sin embargo por la necesidad de beneficiarse en la implementación de las ventajas de aplicar diferentes patrones de diseño, estos se han

adaptado al lenguaje para su uso. A continuación se explican algunos de los patrones de diseño utilizados en la solución.

Constructor (Builder): Es de tipo creación, a nivel de objetos. Separa el proceso de construcción de un objeto complejo de su representación, para que el mismo proceso de construcción pueda crear diferentes representaciones. JavaScript no admite el concepto de clases, pero es compatible con funciones constructoras especiales que trabajan con objetos. Anteponiendo una llamada a una función constructora con la palabra clave "new", se logra que la función se comporte como un constructor y una instancia de un nuevo objeto con los parámetros definidos por esa función. (Osmani, 2012)

Es usado en los objetos fotograma y capas donde sus constructores se utilizan para crear tipos específicos de estos objetos. Se optimiza con la utilización del patrón Prototipo. A continuación se muestra la creación de un objeto fotograma.

```
60  Layer.prototype.addPhotogramPos = function(p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, pos) {
61      value = new Photogram(p1, p2, p3, p4, p5, p6, p7, p8, p9, p10);
62      if (pos + 1 === this.photogram_total) {
63          this.addPhotogram(value);
```

Ilustración 3. Patrón Constructor

Prototipo (Prototype): Creación, a nivel de objetos. Crea objetos nuevos copiándolos, clonando una instancia creada previamente. Las funciones en JavaScript tienen una propiedad llamada *prototype*. Cuando se llama a un constructor de JavaScript para crear un objeto, todas las propiedades del prototipo del constructor se ponen a disposición para el nuevo objeto. De esta manera, varios objetos se pueden crear que provienen del mismo prototipo. (Osmani, 2012)

Es empleado en todos los objetos de la solución. A continuación se muestra uno de los métodos de la clase Photogram en el que se utiliza el patrón.

```
19  Photogram.prototype.getTag = function() {
20      return this.tag;
21  }
```

Ilustración 4. Patrón Prototipo

Decorador (Decorator): Estructura, a nivel de objetos. El patrón Decorador tiene como principal objetivo promover la reutilización de código. Ofrece la posibilidad de añadir comportamientos a los objetos existentes en un sistema de forma dinámica, no se centra en cómo se crean los objetos sino en el

problema de ampliar su funcionalidad. Añadir nuevos atributos a los objetos en JavaScript es un proceso muy sencillo, y una forma de aplicar el patrón. (Osmani, 2012)

Cuando se crean los objetos capa y fotograma se extienden sus funcionalidades mediante los métodos *setter* y *getter*. A continuación se muestra un método set que permite cambiar la cantidad de fotogramas total al objeto capa cada vez que se instancie.

```
87  Layer.prototype.setPhotogramTotal = function(value) {
88      this.photogram_total = value;
89  }
```

Ilustración 5. Patrón Decorador

Fachada (Facade): Estructura, a nivel de objetos. Proporciona una abstracción de un cuerpo de código complejo mediante una interfaz simple. Se puede analizar como la simplificación de la API que se presenta a otros desarrolladores. Fachada es un patrón que se puede encontrar con frecuencia en bibliotecas de JavaScript como jQuery, que proporciona a los desarrolladores un acceso fácil a las implementaciones para la manipulación del DOM. (Osmani, 2012)

```
83      $(canvas_tl).bind('mouseup', clickMouse);
```

Ilustración 6. Patrón Fachada

3.1.4 Tarjetas CRC

La metodología XP representa las clases del sistema mediante tarjetas CRC (Clase, Responsabilidad y Colaboración). Las cuales ayudan al equipo a definir actividades durante el diseño del sistema. Las tarjetas CRC tienen la siguiente estructura.

Tabla 23. Parámetros de las tarjetas CRC

Tarjeta CRC	
Clase: es cualquier persona, cosa, evento, concepto, pantalla o reporte.	
Responsabilidades: es lo que debe hacer la clase, sus atributos y métodos.	**Colaboradores:** son el resto de las clases con las que interactúa para llevar a cabo sus responsabilidades.

Tabla 24. Tarjeta CRC: ControlTimeLine

Tarjeta CRC	
Clase: ControlTimeLine	
Responsabilidades: Gestionar capas. Editar propiedades de las capas. Controlar animación.	**Colaboradores:** Layer ControlListElements ControlDocument

Tabla 25. Tarjeta CRC: ControlListElements

Tarjeta CRC	
Clase: ControlListElements	
Responsabilidades: Gestionar los elementos mediante listado.	**Colaboradores:**

Tabla 26. Tarjeta CRC: ControlDraw

Tarjeta CRC	
Clase: ControlDraw	
Responsabilidades: Dibujar elementos geométricos básicos en el escenario. Insertar elementos gráficos en el escenario. Editar propiedades de transformación de elementos. Editar puntos de los elementos. Editar las propiedades de estilo de elementos geométricos básicos. Editar propiedades de estilo de elemento texto. Editar los elementos.	**Colaboradores:** ControlListElements ControlDocument

Tabla 27. Tarjeta CRC: ControlDocument

Tarjeta CRC	
Clase: ControlDocument	
Responsabilidades: Gestionar proyecto. Generar código JavaScript. Cambiar dimensión del escenario.	**Colaboradores:**

Tabla 28. Tarjeta CRC: Photogram

Tarjeta CRC	
Clase: Photogram	
Responsabilidades: Contiene las propiedades de un fotograma, y su comportamiento.	**Colaboradores:**

Tabla 29. Tarjeta CRC: Layer

Tarjeta CRC	
Clase: Layer	
Responsabilidades: Gestionar fotogramas. Contiene las propiedades de las capas, y su comportamiento.	**Colaboradores:** ControlTimeLine Photogram

3.2 Implementación

En esta fase se lleva a cabo la codifición de cada una de las HU por iteración. Al principio de cada una, se realiza una revisión del plan de iteraciones y se modifica en caso de ser necesario. Estas HU se descomponen en tareas de programación o ingeniería, que se asignan a un equipo de desarrollo o persona. Estas tareas no tienen que necesariamente ser entendidas por el cliente, pueden ser escritas en lenguaje técnico y son para el uso estricto de los programadores. XP define como principio en esta fase

que el cliente esté disponible no solo para ayudar al equipo de desarrolladores sino formando parte del mismo, y mediante esta constante comunicación se logra la retroalimentación que hace factible la evolución del sistema. Otro principio consistente de esta fase lo constituye la estandarización del código para facilitar la lectura y la modificación del mismo por parte de cualquier miembro del equipo. Tomando como referencia la planificación realizada anteriormente, se llevaron a cabo cinco iteraciones de desarrollo sobre el sistema, obteniéndose como finalidad un producto con todos los requisitos cumplidos.

3.2.1 Primera iteración

El objetivo de esta iteración es desarrollar HU1: Dibujar elementos geométricos básicos en el escenario, HU2: Insertar elementos gráficos en el escenario, HU3: Editar propiedades de transformación de elementos, HU4: Editar puntos de los elementos, HU5: Cambiar dimensión del escenario. Esta iteración finaliza con varios componentes visuales mediante los que el usuario puede dibujar en el escenario o insertar imagen y texto, además de permitir la edición de estos elementos dibujados. Estas son algunas de las HU con mayor prioridad del negocio para el cliente, y sirven de base para las próximas iteraciones. Para el desarrollo de la iteración se trazaron las tareas que se enuncian a continuación. Las tablas que representan estas tareas se encuentran descritas en el Anexo 1.

HU1. Dibujar elementos geométricos básicos en el escenario

Tarea No.1: Establecer el estándar de código y comentarios.

Tarea No.2: Implementar las clases principales del núcleo de la aplicación.

Tarea No.3: Diseñar interfaz de dibujo.

Tarea No.4: Dibujar elementos línea, rectángulo, círculo y forma.

Tarea No.5: Seleccionar color de borde y de relleno para el elemento a dibujar.

HU2. Insertar elementos gráficos en el escenario

Tarea No.6: Insertar elemento texto.

Tarea No.7: Insertar elemento imagen.

HU3: Editar propiedades de transformación de elementos

Tarea No.8: Seleccionar, borrar, trasladar, escalar y rotar un elemento seleccionado.

Tarea No.9: Editar curva de Bézier de los elementos geométricos básicos.

Tarea No.10: Modificar orden de visualización de los elementos.

HU4. Editar puntos de los elementos

Tarea No.11: Agregar, mover y eliminar un punto de los elementos geométricos básicos.

HU5. Cambiar dimensión del escenario

Tarea No.12: Cambiar alto y ancho del escenario.

3.2.2 Segunda iteración

La segunda iteración tiene como objetivo desarrollar las HU6: Editar propiedades de estilo de los elementos geométricos básicos, HU7: Editar propiedades de estilo del elemento texto, HU9: Gestionar capas, HU10: Editar propiedades de las capas. Obteniéndose al final de estas una primera versión del producto que permite el trazado de gráficos por capas y su edición. Las tablas que representan las tareas de esta iteración se encuentran descritas en el Anexo 2.

HU6. Editar las propiedades de estilo de los elementos geométricos básicos

Tarea No.13: Diseñar interfaz del panel de propiedades.

Tarea No.14: Cambiar color de borde y color de relleno de los elementos geométricos básicos.

Tarea No.15: Cambiar ancho de línea de los elementos geométricos básicos.

HU7. Editar propiedades de estilo del elemento texto

Tarea No.16: Cambiar contenido, tipo y tamaño de fuente del elemento texto.

HU9. Gestionar capas

Tarea No.17: Diseñar interfaz de la línea de tiempo.

Tarea No.18: Crear, eliminar y duplicar capa.

HU10. Editar propiedades de las capas

Tarea No.19: Renombrar, visualizar, ocultar y bloquear capa.

3.2.3 Tercera iteración

Al concluir la tercera iteración se logra uno de los objetivos fundamentales de la aplicación, la visualización de la animación. Se cumplen la HU11: Gestionar fotogramas y la HU12: Controlar animación. Las tablas que representan las tareas de esta iteración se encuentran descritas en el Anexo 3.

HU11. Gestionar fotogramas

Tarea No.20: Crear y eliminar fotograma.

Tarea No.21: Duplicar y limpiar fotograma.

HU12. Controlar animación

Tarea No.22: Reproducir y detener animación.

Tarea No.23: Cambiar cantidad de fotogramas por segundo de la animación.

3.2.4 Cuarta iteración

La cuarta iteración tiene como propósito fundamental generar el código de la animación, que es el requisito fundamental del software. Se desarrollan para ello la HU8: Editar los elementos y la HU15: Generar código. Las tablas que describen las tareas de esta iteración se encuentran descritas en el Anexo 4.

HU8. Editar los elementos

Tarea No.24: Copiar, cortar y pegar un elemento seleccionado.

HU15. Generar código

Tarea No.25: Generar y visualizar código de la animcación.

Tarea No.26: Descargar la animación.

3.2.5 Quinta iteración

La quinta y última iteración tiene como objetivo la gestión del proyecto y de los elementos dibujados en el escenario. Al final de esta iteración se obtiene una versión del producto donde se satisfacen todos los requisitos del cliente. Se desarrollan las tareas necesarias para implementar la HU13: Gestionar elementos mediante listado y la HU14: Gestionar proyecto. Las tablas que representan las tareas de ingeniería de esta iteración se encuentran descritas en el Anexo 5.

HU13. Gestionar los elementos dibujados.

Tarea No.27: Diseñar la interfaz del panel de elementos dibujados.

Tarea No.28: Visualizar listado de elementos dibujados y seleccionar un elemento.

Tarea No.29: Eliminar un elemento seleccionado del listado.

HU14. Gestionar proyecto

Tarea No.30: Crear nuevo proyecto.

Tarea No.31: Establecer estándar XML del proyecto.

Tarea No.32: Guardar proyecto en XML.

Tarea No.33: Abrir proyecto en XML.

3.2.6 Estrategia de codificación. Estándares y estilos a emplear.

Las convenciones o estándares de codificación son pautas de programación que no están enfocadas a la lógica del programa, sino a su estructura y apariencia física para facilitar la lectura, comprensión y mantenimiento del código. Las normas de codificación para el lenguaje JavaScript que se emplean están basadas en el documento *Convenciones de código para el lenguaje de programación JavaScript* de Douglas Crockford. A continuación se presentan algunas de estas convenciones para la programación en el lenguaje JavaScript usadas para la programación de la aplicación implementada.

Comentarios

Se incluirán como apoyo a variables y llamadas a funciones. Deben contener autor y descripción.

Nombre de identificadores

Se considera como identificador a los nombres de variables, funciones, así como cualquier tipo de dato definido por el usuario (estructura, clase). Dichos identificadores deberán seguir las siguientes normas:

- Tener un nombre significativo para que por su simple lectura, pueda conocerse su función, sin tener que consultar manuales o hacer demasiados comentarios.
- Los nombres deben estar en el idioma inglés y no deben contener caracteres extraños, excluyendo el guión bajo.

- Para nombres que se usen con frecuencia o para términos largos, se recomienda usar abreviaturas estándar para que estos tengan una longitud razonable.

Identificadores de variables

Para distinguir palabras dentro del nombre deberá emplearse un guión bajo (_). Preferiblemente no usar más de 3 palabras por identificador siempre y cuando en el nombre quede implícita la acción. Ejemplo: news_variable.

Identificadores de clases y métodos

Los identificadores de clases comenzarán escritos con letra mayúscula. En caso de tener más de una palabra, la segunda comenzará de nuevo con mayúscula. Los identificadores de los métodos comenzarán en letra minúscula. En caso de tener más de una palabra, la segunda comenzará con mayúscula. Ejemplo clase: NewsClass, ejemplo método: newsFunction.

Líneas, sangrías y espacios en blanco

- No manejar en los programas más de una instrucción por línea.
- Declarar variables del mismo tipo en una línea.
- Evitar líneas de más de 100 caracteres.
- La unidad de la sangría es de cuatro espacios
- Insertar una línea en blanco antes y después de una declaración de datos que aparezca entre instrucciones ejecutables.
- Las declaraciones de datos dentro de una función, deberán ir al inicio y separadas de las instrucciones ejecutables de la función por medio de una línea en blanco.

3.3 Pruebas

Para la realización de pruebas, una de las técnicas más comunes son las de caja blanca y caja negra. Las pruebas de caja blanca están estrechamente ligadas al código fuente, se realizan sobre las funciones internas del software. Mientras, las pruebas de caja negra no necesita conocer los detalles internos de funcionamiento, su objetivo fundamental es verificar el cumplimiento de la lista de reservas del producto desde la perspectiva del usuario final.

Los programadores escriben pruebas unitarias para aumentar su confianza en el funcionamiento del software. Asimismo los clientes escriben pruebas funcionales para aumentar su confianza en el funcionamiento del software. El resultado es un software que se vuelve más y más confiable con el tiempo y se hace más capaz de aceptar el cambio, no menos. (Beck, 1999)

3.3.1 Pruebas unitarias

Para comprobar el funcionamiento de los códigos que se van implementando, uno de los principales pilares de la metodología XP son las pruebas unitarias o también conocidas como TDD (*Test Driven Development*). El código debe ser probado gradualmente y obtener un resultado, lo que permite trabajar con la seguridad que el código funciona. Las pruebas deben ser definidas antes que el código, y todo código debe pasar correctamente las pruebas unitarias antes de ser liberado. Al detectar un error este debe ser corregido inmediatamente, y se generan nuevas pruebas para comprobar que el error fue resuelto. TDD es particularmente relevante para JavaScript, con sus incompatibilidades entre navegadores, siendo la mejor manera de capturar la mayoría de los errores de programación. Entre las ventajas más notables de TDD, analizadas para JavaScript se encuentran:

Refactorización: consiste en hacer el código más simple y entendible. Requiere cambiar la implementación sin alterar el comportamiento del programa, conservando intacta la funcionalidad. En Internet las aplicaciones tienen que ser lo más eficientes posible con la menor cantidad de código, además es una práctica que mejora el diseño.

Compatibilidad de navegadores: la aplicación debe funcionar correctamente en un elevado número de plataformas y dispositivos. Las pruebas manuales serían poco eficientes, deberían repetirse en cada uno de los entornos para los que se quiere ofrecer compatibilidad. Disponer de todas las pruebas necesarias optimiza esta tarea.

En el Anexo 11 se muestran los Casos de prueba unitaria realizadas al software.

3.3.2 Pruebas de aceptación

Las pruebas de aceptación son consideradas como pruebas de caja negra. Es el cliente quien las realiza y es el responsable de que sus resultados sean correctos. Estas pruebas se centran en comprobar las funcionalidades del sistema y son creadas en base a las HU. Una HU debe pasar correctamente las

pruebas de aceptación para considerarse terminada. A continuación se muestra los parámetros a medir en los casos de pruebas de aceptación.

Tabla 30. Parámetros Caso de prueba de aceptación

Caso de prueba de aceptación	
Código: Muestra un identificador para cada prueba realizada.	**HU:** Indica la HU a la que pertenece la prueba.
Nombre: Indica el nombre de la prueba.	
Condiciones de ejecución: Indica cuáles son las condiciones que se tienen que cumplir para que el componente realice correctamente la funcionalidad que se va a medir.	
Entrada/Pasos de ejecución: Indica los pasos a seguir, o las entradas para que el componente realice correctamente la funcionalidad que se va a medir.	
Resultado esperado: Indica el resultado que se obtendría de un correcto funcionamiento de la prueba.	
Evaluación de la prueba: Indica el estado de la prueba si es satisfactoria o insatisfactoria.	

3.3.2.1 Casos de pruebas de aceptación para la versión 0.1

Para esta entrega se realizaron veintiún Casos de prueba de aceptación, identificando cinco no conformidades. Debido a la complejidad mínima de las no conformidades, fueron todas resueltas y no quedaron Casos de pruebas pendientes para la siguiente versión. Las tablas que representan los Casos de prueba de aceptación para esta versión se encuentran descritas en el Anexo 6.

HU1. Dibujar elementos geométricos básicos en el escenario

Caso de pruebas de aceptación (CPA) No.1: Dibujar línea.

CPA No.2: Dibujar rectángulo.

CPA No.3: Dibujar círculo.

CPA No.4: Dibujar forma.

CPA No.5: Seleccionar color de borde para el elemento a dibujar.

CPA No.6: Seleccionar color de relleno para el elemento a dibujar.

HU2. Insertar elementos gráficos en el escenario

CPA No.7: Insertar elemento texto.

CPA No.8: Insertar elemento imagen.

HU3: Editar propiedades de transformación de los elementos

CPA No.9: Seleccionar un elemento del escenario.

CPA No.10: Rotar un elemento seleccionado.

CPA No.11: Trasladar el elemento seleccionado.

CPA No.12: Escalar el elemento seleccionado.

CPA No.13: Editar curva de Bézier.

CPA No.14: Borrar un elemento.

CPA No.15: Enviar elemento hacia el fondo.

CPA No.16: Traer elemento al frente.

HU4. Editar puntos de los elementos

CPA No.17: **:** Mover un punto de los elementos línea, rectángulo, círculo y forma.

CPA No.18: Agregar un punto a los elementos línea, rectángulo, círculo y forma.

CPA No.19: Eliminar un punto a los elementos línea, rectángulo, círculo, y forma.

HU5. Cambiar dimensiones del escenario

CPA No.20: Cambiar ancho del escenario.

CPA No.21: Cambiar alto del escenario.

3.3.2.2 Casos de prueba de aceptación para la versión 0.2

Para esta entrega se realizaron trece Casos de prueba de aceptación, identificando tres no conformidades de las cuales dos no procedieron. Se le dio solución a las no conformidades, por lo que no quedaron Casos de pruebas pendientes para la siguiente versión. Las tablas que describen los Casos de prueba de aceptación para esta versión se encuentran descritas en el Anexo 7.

HU6. Editar las propiedades de estilo de los elementos geométricos básicos

CPA No.22: Cambiar color de borde del elemento seleccionado.

CPA No.23: Cambiar color de relleno del elemento seleccionado.

CPA No.24: Cambiar ancho de línea del elemento seleccionado.

HU7. Editar propiedades de estilo del elemento texto

CPA No.25: Editar contenido del elemento texto.

CPA No.26: Editar tipo de fuente del elemento texto.

CPA No.27: Editar tamaño de fuente del elemento texto.

HU9. Gestionar capas

CPA No.28: Crear capa nueva.

CPA No.29: Eliminar capa seleccionada.

CPA No.30: Duplicar capa seleccionada

HU10. Editar propiedades de las capas

CPA No.31: Nombrar capa seleccionada.

CPA No.32: Visualizar capa seleccionada

CPA No.33: Ocultar capa seleccionada

CPA No.34: Bloquear capa seleccionada

CPA No.35: Desbloquear capa seleccionada

3.3.2.3 Caso de prueba de aceptación para la versión 0.3

Para esta entrega se realizaron siete Casos de prueba de aceptación, identificando una no conformidad. Esta no conformidad se solucionó, por lo que no quedó ningún Caso de prueba pendiente para la siguiente versión. Las tablas que representan los Casos de prueba de aceptación para esta versión se encuentran descritas en el Anexo 8.

HU11. Gestionar fotogramas

CPA No.36: Crear fotograma nuevo

CPA No.37: Eliminar fotograma seleccionado

CPA No.38: Duplicar fotograma seleccionado

CPA No.39: Limpiar fotograma seleccionado

HU12. Controlar animación

CPA No.40: Reproducir animación

CPA No.41: Detener animación

CPA No.42: Cambiar cantidad de fotogramas por segundo de la animación

3.3.2.4 Casos de prueba de aceptación para la versión 0.4

Para esta entrega se realizaron cuatro Casos de prueba de aceptación, donde se identificaron dos no conformidades. Las tablas que describen los Casos de prueba de aceptación para esta versión se encuentran descritas en el Anexo 9.

HU8. Editar elementos

CPA No.43: Copiar un elemento

CPA No.44: Cortar un elemento

CPA No.45: Pegar un elemento

HU15. Generar código

CPA No.46: Generar código JavaScript

3.3.2.5 Casos de prueba de aceptación para la versión 1.0

Para esta entrega se realizaron siete Casos de prueba de aceptación, identificando dos no conformidades. Estas no conformidades se solucionaron, dando fin al proceso de pruebas de aceptación obteniendo resultados satisfactorios. Las tablas que representan los Casos de prueba de aceptación para esta versión se encuentran descritas en el Anexo 10.

HU13. Gestionar los elementos mediante listado

CPA No.47: Visualizar listado de elementos dibujados por fotograma

CPA No.48: Seleccionar elemento dibujado del listado

CPA No.49: Eliminar elemento seleccionado del listado

HU14. Gestionar proyecto

CPA No.50: Crear nuevo proyecto

CPA No.51: Abrir proyecto existente en XML

CPA No.52: Guardar proyecto como XML

3.3.3 Resultados generales de las pruebas de aceptación

Para la entrega final de la aplicación se reaizaron un total de cincuenta y tres Casos de prueba de aceptación, se detectaron trece no conformidades, de ellas dos no procedieron y el resto fueron resueltas. El siguiente gráfico muestra el resumen del resultado de las pruebas realizadas.

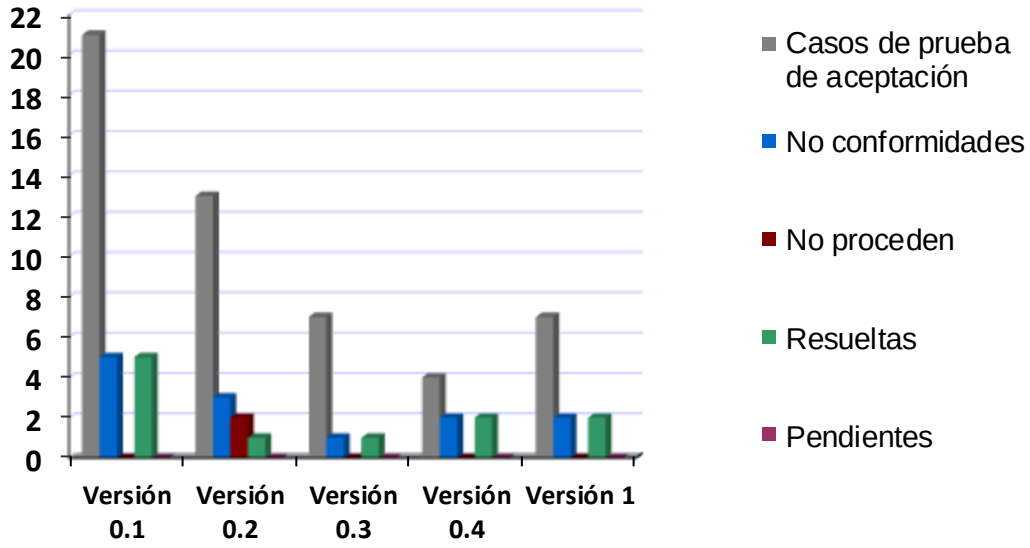

Ilustración 7. Resultado de las pruebas de aceptación

3.4 Conclusiones del Capítulo 3

En el presente capítulo se abordaron los temas referentes al diseño, implementación y pruebas de la solución propuesta. Se aplicaron patrones arquitectónicos y de diseño al sistema, por su importancia para el desarrollo. Además se realizó el proceso de implementación y prueba generando los artefactos que propone la metodología XP. Se demostró que el desarrollo dirigido por pruebas apoyado por la realización de las pruebas de aceptación, es una estrategia íntegra para obtener un software probado y funcional, que garantiza la confiabilidad en el software tanto del cliente como del desarrollador.

Conclusiones generales

Al término de la presente investigación se concluye lo siguiente:

1. Se desarrolló una aplicación para la creación de animaciones web en el Centro FORTES, siguiendo las tendencias de las principales herramientas y tecnologías actuales.
2. Fue utilizada para dirigir el proceso de desarrollo la metodología de software XP, como lenguajes para la implementación JavaScript, CSS3 y HTML5 en el entorno de desarrollo NetBeans, y lenguaje de modelado UML en la herramienta Visual Paradigm for UML. Estas tecnologías en conjunto con los *frameworks* jQuery, Bootstrap y Paper.js permitieron obtener un software que cumple con todos los requerimientos del cliente.
3. Se elaboró un manual de usuario que apoya la utilización de Flavas como aplicación para la creación de animaciones web.
4. El lenguaje HTML5 con su nueva etiqueta <canvas> es muy potente para crear contenido interactivo, ya sea para software educativo u otro tipo de aplicaciones de alto impacto en el usuario final.
5. Se obtuvo una aplicación libre y de código abierto, contribuyendo con la independencia tecnológica del país.

Recomendaciones

Después de analizados los resultados obtenidos por la presente investigación se recomienda:

- Extender el uso de la herramienta a todos los proyectos del Centro FORTES así como a otras áreas de la Universidad.
- Continuar el desarrollo de la aplicación en el lado del servidor para permitir la gestión de usuario y guardar los proyectos en la nube explotando sus ventajas.
- Desarrollar las funcionalidades requeridas para:
    - Agregar el método de creación de animaciones por interpolación de movimiento.
    - Desarrollar las animaciones mediante la programación.
    - Deshacer una acción.
    - Agrupar elementos.

Referencias bibliográficas

Adobe. 2014. Adobe Help. *Adobe Flash.* [En línea] Adobe Systems Inc, 2014. [Citado el: 23 de 04 de 2014.] http://help.adobe.com/es_ES/flash/cs/using/WS4C0E4220-5C0C-44c0-B58D-496A5424C78B.html.

Adobe Edge. 2013. Adobe Edge Animate CC. *Adobe & HTML.* [En línea] Adobe, 2013. [Citado el: 08 de 11 de 2013.] http://html.adobe.com/edge/animate/.

Adobe Flash. 2013. Adobe Help. *Adobe Flash.* [En línea] Adobe Systems Inc, 2013. [Citado el: 08 de 11 de 2013.] http://help.adobe.com/es_ES/flash/cs/using/WSd60f23110762d6b883b18f10cb1fe1af6-7f84a.html.

Albuera, Pablo. Software Propietario. *Software Propietario.* [En línea] [Citado el: 08 de 05 de 2014.] http://www.slideshare.net/pabloalbuera/software-propietario-1569982.

Algesa. 2014. Diccionario de informática. *Algesa.* [En línea] Algesa, 2014. [Citado el: 08 de 05 de 2014.] http://www.alegsa.com.ar/Dic/applet.php.

—. 2014. Diccionario de informática. *Algesa.* [En línea] Algesa, 2014. [Citado el: 08 de 05 de 2014.] http://www.alegsa.com.ar/Dic/plugin.php.

Almeira, Adriana Sandra. 2007. *Arquitectura de Software:Estilos y Patrones.* Argentina : s.n., 2007.

Álvarez, Miguel Angel. 1999. Desarrolloweb.com. *Desarrolloweb.com.* [En línea] 12 de 1999. [Citado el: 04 de 11 de 2013.] http://www.desarrolloweb.com/articulos/introduccion-css3.html.

Aula Clic. 2010. Aula Clic. *Aula Clic.* [En línea] Aula Clic, 2010. [Citado el: 08 de 05 de 2014.] www.aulaclic.es/coreldraw-x5/b_1_1_1.htm.

Beck, Kent. 1999. *Extreme Programming Explained. Embrace Change.* s.l. : ISBN, 1999. 0201616416.

Bqto. 2009. Definiciones, Conceptos y Evolución de la Tecnología. *Bqto.* [En línea] 2009. [Citado el: 08 de 05 de 2014.] http://bqto.unesr.edu.ve/pregrado/Gestion%20de%20Tecnologia/gtr_unid1/dependencia_e_independencia_tecnolgica.html.

Canós, José H. 2003. *Métodologías Ágiles en el Desarrollo de Software.* Valencia : s.n., 2003.

Clic, Aula. 2002. Aula Clic. *Aula Clic.* [En línea] 07 de 2002. [Citado el: 18 de 02 de 2014.] http://www.aulaclic.es/flash-cs5/t_1_2.htm.

CreateJS. 2014. EaselJS. *CreateJS.* [En línea] 2014. [Citado el: 17 de 11 de 2013.] http://www.createjs.com/?_escaped_fragment_=/EaselJS#!/EaselJS.

Cubadebate. 2014. Cubadebate. *Cubadebate.* [En línea] Cubadebate, 2014. [Citado el: 08 de 05 de 2014.] http://www.cubadebate.cu/serie/bloqueo-contra-cuba/.

David, Matthew. 2010. *HTML5 Designing Rich Internet Applications.* Oxford : Focal Press, 2010.

Desafíos del uso de software en la clase. **Pereira, Isidro Cornell.** Ciego de Ávila : s.n.

Desarrollos Multimedia. Desarrollos Multimedia. *Desarrollos Multimedia.* [En línea] Desarrollos Multimedia. [Citado el: 08 de 05 de 2014.] http://www.desarrollomultimedia.es/faq/que-es-la-curva-de-bezier.html.

Desarrollos Web. 2000. Login Desarrollos. *Desarrollos de Aplicaciones Web.* [En línea] 2000. [Citado el: 04 de 11 de 2013.] http://www.logindesarrollos.com/es/Servicios-aplicaciones-web-21.

Digital, Creación. 2013. Creación Digital. *IDEC-Universitat Pompeu Fabra.* [En línea] Creación Digital, 2013. [Citado el: 28 de 01 de 2013.] http://creaciodigital.upf.edu/~smiguel/b12animacion.html.

Drowell, Eric. 2013. Web Graphics Trends 2013. *HMTL5 Canvas Tutorials.* [En línea] 2013. [Citado el: 13 de 03 de 2014.] http://www.html5canvastutorials.com/articles/web-graphics-trends-in-2013/.

Flash Professional . 2002. Flash Professional CS5. *Aula Clic.* [En línea] 07 de 2002. [Citado el: 17 de 11 de 2013.] http://www.aulaclic.es/flash-cs5/t_1_2.htm.

Free Software Fundation. 2009. Free Software Fundation. *Free Software Fundation.* [En línea] Free Software Fundation, 2009. [Citado el: 08 de 05 de 2014.] http://www.fsf.org/es.

Google. 2014. Google Web Designer. *Google Web Designer.* [En línea] Google, 2014. [Citado el: 25 de 04 de 2014.] www.google.com/webdesigner.

HelpNDoc. 2014. HelpNDoc. *HelpNDoc.* [En línea] HelpNDoc, 2014. [Citado el: 30 de 04 de 2014.] http://www.helpndoc.com/feature-tour.

Hernán, Marcelo Schenone. 2004. *Tesis de Grado en Ingeniería en Informática. Facultad de Ingeniería.Universidad de Buenos Aires.* Buenos Aires : s.n., 2004.

Hickson, Ian. 2010. HTML5: A vocabulary and associated APIs for HTML and XHTML. W3C Working Draft. [En línea] 24 de 06 de 2010. [Citado el: 04 de 11 de 2013.] http://www.w3.org/TR/html5.

Highsmith, Jim. 2002. *Agile Software Development Ecosystems.* s.l. : Addison-Wesley., 2002.

HMTL5 Maker. 2013. HMTL5 Maker. *HMTL5 Maker.* [En línea] 2013. [Citado el: 19 de 12 de 2013.] html5maker.com/app.

Informaticando Tf. 2014. Informaticando Tf. *Informaticando Tf.* [En línea] Informaticando Tf, 2014. [Citado el: 08 de 05 de 2014.] http://informaticandotf.blogspot.com/2010/08/multiplataforma-definicion.html.

Jacobson, Ivar. 2004. *El Proceso Unificado de Desarrollo de Software.* s.l. : Wesley : Pearson Adisson,, 2004.

KoolMoves. 2008. KoolMoves. *KoolMoves.* [En línea] 29 de 02 de 2008. [Citado el: 08 de 11 de 2013.] http://www.koolmoves.com/index.html.

Larman, Craig. 1999. *UML y Patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* México : Prentice Hall, 1999. ISBN: 970-17-0261-1.

Llorente, César de la Torre. 2010. *Guía de Arquitectura N-Capas orientada al Dominio con .NET 4.0.* España : Microsoft Ibérica S.R.L, 2010. ISBN:978-84-936696-3-8.

Marcela, Daniele. 2007. Análisis y Diseño de Sistemas. *Departamento de Computación.* [En línea] 2007. [Citado el: 11 de 11 de 2013.] http://dc.exa.unrc.edu.ar/nuevodc/materias/sistemas/2007/TEORICOS/TEORIA_1.

Marcos Guglielmettti. 2005. Master Megazine. *Master Megazine.* [En línea] 2005. [Citado el: 5 de 05 de 2014.] http://www.mastermagazine.info/termino/3868.php..

Meza, Mirna. 2011. Herramientas Case. *Herramientas Case.* [En línea] 2011. [Citado el: 17 de 11 de 2013.] http://fds-herramientascase.blogspot.com/.

Microsoft. 2013. Microsoft. Developer Network. *Centro de desarrollo de Internet Explorer.* [En línea] Micorsoft, 2013. [Citado el: 13 de 12 de 2013.] http://msdn.microsoft.com/es-es/library/ie/gg193983%28v=vs.85%29.aspx#High_Level_Overview_of_Vector_Graphic_Scenarios.

Moonbase. 2012. Moonbase. *Moonbase.* [En línea] 19 de 12 de 2012. [Citado el: 17 de 11 de 2013.] http://www.moonbase.com/.

NetBeans. 2010. NetBeans. *NetBeans.* [En línea] NetBeans Org, 2010. [Citado el: 08 de 11 de 2013.] http://netbeans.org/community/releases/roadmap.html.

Node.js. 2014. Node.js. *Node.js.* [En línea] Joyent Inc, 2014. [Citado el: 12 de 05 de 2014.] http://nodejs.org/.

Osmani, Addy. 2012. *Learning JavaScript Design Patterns.* Londres : O'Reilly Media, 2012.

Pérez, Javier Eguíluz. 2011. *Introducción a JavaScript.* Washington : The Scholarly Publishing & Academic Resources Coalition 21, 2011.

Puckey, Jonathan. 2013. About Paper.js. *Paper.js.* [En línea] Paper.js, 2013. [Citado el: 15 de 12 de 2013.] http://paperjs.org/about/.

QUnit. 2014. QUnit. *Qunit.* [En línea] The jQuery Fundation, 2014. [Citado el: 17 de 11 de 2013.] http://qunitjs.com/.

Robert Quattlebaum. 2014. Synfig Studio About. *Synfig Studio.* [En línea] Synfig Studio Organization, 2014. [Citado el: 08 de 11 de 2013.] http://www.synfig.org/cms/.

Sanders, Adrien-Luc. 2014. About.com Computing Animation. *About.com .* [En línea] 2014. [Citado el: 10 de 02 de 2014.] http://animation.about.com/od/glossaryofterms/g/layer_def.htm.

Sencha. 2013. Sencha Animator. *Sencha Inc.* [En línea] Sencha Inc, 2013. [Citado el: 08 de 11 de 2013.] https://www.sencha.com/store/animator/.

The Eclipse Foundation. 2001. The Official Eclipse FAQs. *The Eclipse Foundation.* [En línea] The Eclipse Foundation, 11 de 2001. [Citado el: 12 de 12 de 2013.] http://wiki.eclipse.org/The_Official_Eclipse_FAQs.

Thornton, Otto. 2010. Bootstrap. *Bootstrap.* [En línea] Twitter, 2010. [Citado el: 04 de 12 de 2013.] http://getbootstrap.com/about/#history.

Vega, John Freddy. 2011. *Guía HTML5. El presente de la web. HTML5, CSS3 y JavaScript.* 2011. CC BY-NC-SA 3.0.

Visual Paradigm. 2014. Visual Paradigm for UML. *Visual Paradigm.* [En línea] Visual Paradigm, 2014. [Citado el: 27 de 01 de 2014.] www.visual-paradigm.com/product/vpuml/.

Walden, Shore. 2007. *The Art of Agile Development.* s.l. : O'Reilly, 2007.

Zeldman, Jeffery. 2004. *Diseño con estándares web.* s.l. : ANAYA MULTIMEDIA, 2004. 9788441516083.

Glosario

API: (*Application Program Interface)* es un conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación. Cuando se intenta estandarizar una plataforma, se estipulan unos APIs comunes a los que deben ajustarse todos los desarrolladores de aplicaciones. (Marcos Guglielmettti, 2005)

***Applets*:** componente de software (que suele ser pequeño) escrito en un lenguaje de programación como Java, que se ejecuta bajo el control de una aplicación más grande que lo contiene, como un navegador web. (Algesa, 2014)

Bloqueo contra Cuba: cerco comercial, económico y financiero impuesto por Estados Unidos a Cuba desde el 7 de febrero de 1962. Fue convertido en ley en 1992 y 1995. Es uno de los más duraderos de la historia, condenado quince veces por las Naciones Unidas. (Cubadebate, 2014)

Curva de Bézier: la curva Bézier es un tipo de línea curva que consigue definir las transiciones suaves de las curvaturas. Una curva de Bézier tiene cuatro puntos de control, dos de los puntos corresponden a los extremos de la línea, son denominados puntos de anclaje y los otros dos puntos son denominados puntos de control y determinan la dirección con que la curvatura ingresa a los extremos. (Desarrollos Multimedia)

Dependencia tecnológica: es un fenómeno contemporáneo muy complejo que afecta a la mayor parte de los países. Constituye uno de los obstáculos principales para el desarrollo económico, y es a la vez un reflejo de la dependencia económica, política y cultural en que están inmersos los países subdesarrollados. El software libre y los estándares abiertos son necesarios para garantizar la independencia tecnológica de una nación. (Bqto, 2009)

Imagen de mapa de bits: están compuestas por puntos individuales denominados píxeles, dispuestos y coloreados de formas diversas para formar un patrón. Si aumenta el tamaño del mapa de bits, también aumentará el número de píxeles individuales, haciendo que las líneas y las formas tengan un aspecto dentado. (Aula Clic, 2010)

Imagen vectorial: se definen matemáticamente como una serie de puntos unidos por líneas. Los elementos gráficos presentes en un archivo vectorial se denominan objetos. Cada objeto es una entidad completa con propiedades tales como color, forma, contorno, tamaño y posición en la pantalla, que están incluidas en su definición. (Aula Clic, 2010)

Multiplataforma: un programa, lenguaje de programación o elemento de hardware es multiplataforma cuando tiene la capacidad de funcionar en más de un sistema operativo con similares características y sin que su funcionalidad varíe en exceso. (Informaticando Tf, 2014)

***Plugins*:** programa que puede anexarse a otro para aumentar sus funcionalidades. No se trata de un parche ni de una actualización, es un módulo aparte que se incluye opcionalmente en una aplicación. (Algesa, 2014)

Software educativo: se caracteriza por ser altamente interactivo, a partir del empleo de recursos multimedia, como videos, sonidos, fotografías, explicaciones de profesores, ejercicios y juegos instructivos que apoyan las funciones de evaluación y diagnóstico. (Desafíos del uso de software en la clase)

Software libre: hace referencia a la libertad de los usuarios o desarrolladores de tener acceso libre al código fuente para copiarlo, leerlo, distribuirlo, cambiarlo y mejorarlo a su gusto. Respaldado legalmente por la licencia GNU/GPL (*General Public License*). Existen otras variantes de código abierto como Mozilla Public License, L-GPL, etc. (Free Software Fundation, 2009)

Software privativo: se refiere a todo aquel software o fragmento de software cuya licencia comercial, de uso o de publicación, no permite compartir el código fuente del mismo o, lo hace, pero restringe las libertades de uso del mismo. (Albuera)

Anexos

Anexo 1. Tareas de ingeniería, primera iteración

Tabla 31. Tarea 1: Establecer estándar de código y comentarios.

Tarea No.1	**HU1**
Nombre de tarea: Establecer estándar de código y comentarios.	
Tipo de tarea: Configuración	**Estimación:** 1 día
Fecha de inicio: 13 de enero del 2014	**Fecha de fin:** 13 de enero del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Redactar el estándar de código a emplear siguiendo los utilizados internacionalmente en proyectos de JavaScript. Consultar documentación en la red para la definición del mismo.	

Tabla 32. Tarea 2: Implementar las clases principales del núcleo de la aplicación

Tarea No.2	**HU1**
Nombre de tarea: Implementar las clases principales del núcleo de la aplicación.	
Tipo de tarea: Implementación	**Estimación:** 5 días
Fecha de inicio: 14 de enero del 2014	**Fecha de fin:** 18 de enero del 2014
Programador responsable: Orelbis Lago Vasallo	
Descripción: Implementar las clases *ControlTimeLine.js, ControlDraw.js, Layer.js, Photogram.js*, las cuales serán empleadas para la posterior implementación de las demás funcionalidades.	

Tabla 33. Tarea 3: Diseñar interfaz de dibujo

Tarea No.3	**HU1**
Nombre de tarea: Diseñar interfaz de dibujo.	
Tipo de tarea: Implementación	**Estimación:** 1 día
Fecha de inicio: 19 de enero del 2014	**Fecha de fin:** 19 de enero del 2014
Programador responsable: Wendy Reyes Jiménez	

Descripción: Diseñar interfaz de dibujo en el Index.html siguiendo las tendencias actuales.

Tabla 34. Tarea 4: Dibujar elementos línea, rectángulo, círculo y forma

Tarea No.4	**HU1**
Nombre de tarea: Dibujar elementos línea, rectángulo, círculo y forma.	
Tipo de tarea: Implementación	**Estimación:** 2 días
Fecha de inicio: 20 de enero del 2014	**Fecha de fin:** 21 de enero del 2014
Programador responsable: Orelbis Lago Vasallo	
Descripción: Desarrollar las funcionalidades *drawStart(event), drawDrag(event)* para el dibujo de figuras geométricas en la clase controladora *ControlDraw.js*, utilizando el *framework* Paper.js. Debe permitir el dibujo de los elementos gráficos línea, rectángulo, círculo y formas irregulares visualizándose en el escenario.	

Tabla 35. Tarea 5: Seleccionar color de borde y de relleno para el elemento a dibujar

Tarea No.5	**HU1**
Nombre de tarea: Seleccionar color de borde y de relleno para el elemento a dibujar.	
Tipo de tarea: Implementación	**Estimación:** 1 día
Fecha de inicio: 22 de enero del 2014	**Fecha de fin:** 22 de enero del 2014
Programador responsable: Orelbis Lago Vasallo	
Descripción: Utilizar las funcionalidades de la biblioteca ColorPicker para seleccionar el color de borde y de relleno para el elemento a trazar.	

Tabla 36. Tarea 6: Insertar elemento texto

Tarea No.6	**HU2**
Nombre de tarea: Insertar elemento texto.	
Tipo de tarea: Implementación	**Estimación**: 2 días
Fecha de inicio: 23 de enero del 2014	**Fecha de fin:** 24 de enero del 2014
Programador responsable: Orelbis Lago Vasallo	

Descripción: Adicionar procedimientos necesarios a la función *drawStart(event)* de la clase *ControlDraw.js* para insertar los elementos texto en el escenario. Además se crea en el *Index.html* con el uso del *framework* Bootstrap el formulario *form_insert_text*, para mostrar la información.

Tabla 37. Tarea 7: Insertar elemento imagen

Tarea No.7	**HU2**
Nombre de tarea: Insertar elemento imagen.	
Tipo de tarea: Implementación	**Estimación**: 2 días
Fecha de inicio: 25 de enero del 2014	**Fecha de fin:** 26 de enero del 2014
Programador responsable: Orelbis Lago Vasallo	
Descripción: Adicionar procedimientos necesarios a la función *drawStart(event)* de la clase *ControlDraw.js* para insertar los elementos texto en el escenario. Además se crea en el *Index.html* el visual para mostrar la información.	

Tabla 38. Tarea 8: Seleccionar, borrar, trasladar, escalar y rotar un elemento seleccionado

Tarea No.8	**HU3**
Nombre de tarea: Seleccionar, borrar, trasladar, escalar y rotar un elemento seleccionado.	
Tipo de tarea: Implementación	**Estimación:** 4 días
Fecha de inicio: 27 de enero del 2014	**Fecha de fin:** 30 de enero del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Adicionar procedimientos necesarios a la función *drawStart(event)* y *drawDrag(event)* de la clase *ControlDraw.js* para seleccionar, además de borrar, trasladar, escalar y rotar un elemento seleccionado, utilizando la biblioteca Paper.js.	

Tabla 39. Tarea 9: Editar curva de Bézier de los elementos geométricos básicos

Tarea No.9	**HU3**
Nombre de tarea: Editar curva de Bézier de los elementos geométricos básicos.	
Tipo de tarea: Implementación	**Estimación:** 1 día
Fecha de inicio: 31 de enero del 2014	**Fecha de fin:** 31 de enero del 2014

Programador responsable: Wendy Reyes Jiménez
Descripción: Adicionar procedimientos necesarios a la función *drawStart(event)* y *drawDrag(event)* de la clase *ControlDraw.js* para editar curva de Bézier de un elemento seleccionado, utilizando la biblioteca Paper.js.

Tabla 40. Tarea 10: Modificar orden de visualización de los elementos

Tarea No.10	**HU3**
Nombre de tarea: Modificar orden de visualización de los elementos	
Tipo de tarea: Implementación	**Estimación:** 2 días
Fecha de inicio: 1 de febrero del 2014	**Fecha de fin:** 2 de febrero del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Desarrollar las funcionalidades *sendToBack()* y *bringToFront()* en la clase Photogram.js y adicionar procedimientos necesarios a la función *drawStart(event)* de la clase ControlDraw.js para modificar orden de visualización del elemento seleccionado. Mover hacia delante de todos los elementos o hacia el fondo utilizando el *framework* Paper.js.	

Tabla 41. Tarea 11: Agregar, mover y eliminar puntos de los elementos geométricos básicos

Tarea No.11	**HU4**
Nombre de tarea: Agregar, mover y eliminar un punto de los elementos geométricos básicos	
Tipo de tarea: Implementación	**Estimación:** 6 días
Fecha de inicio: 3 de febrero del 2014	**Fecha de fin:** 8 de febrero del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción Adicionar procedimientos necesarios a la función *drawStart(event)* y *drawDrag(event)* de la clase *ControlDraw.js* para agregar, mover y eliminar puntos a los elementos línea, rectángulo, círculo y forma en el escenario utilizando la biblioteca Paper.js.	

Tabla 42. Tarea 12: Cambiar alto y ancho del escenario

Tarea No.12	**HU5**
Nombre de tarea: Cambiar alto y ancho del escenario.	
Tipo de tarea: Implementación	**Estimación:** 1 día

Fecha de inicio: 9 de febrero del 2014	**Fecha de fin:** 9 de febrero del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Desarrollar la funcionalidad *rezise()* en la clase *ControlDraw.js* para cambiar ancho y alto del escenario.	

Anexo 2. Tareas de ingeniería, segunda iteración

Tabla 43. Tarea 13: Diseñar interfaz del panel de propiedades

Tarea No.13	**HU6**
Nombre de tarea: Diseñar interfaz del panel de propiedades.	
Tipo de tarea: Implementación	**Estimación:** 1 día
Fecha de inicio: 10 de febrero del 2014	**Fecha de fin:** 10 de febrero del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Diseñar interfaz del panel de propiedades de los elementos en el *Index.html* siguiendo las tendencias actuales.	

Tabla 44. Tarea 14: Cambiar color de borde y relleno de los elementos

Tarea No.14	**HU6**
Nombre de tarea: Cambiar color de borde y color de relleno de los elementos geométricos básicos.	
Tipo de tarea: Implementación	**Estimación:** 3 días
Fecha de inicio: 11 de febrero del 2014	**Fecha de fin:** 13 de febrero del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Desarrollar la funcionalidad *onSpin()* de la clase *ControlDraw.js* para cambiar color de borde y color de relleno de un elemento, utilizando la biblioteca de jQuery Colorpicker.	

Tabla 45. Tarea 15: Cambiar ancho de línea de los elementos

Tarea No.15	**HU6**
Nombre de tarea: Cambiar ancho de línea de los elementos geométricos básicos..	

Tipo de tarea: Implementación	**Estimación:** 1 día
Fecha de inicio: 14 de febrero del 2014	**Fecha de fin:** 14 de febrero del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Desarrollar la funcionalidad *changeWidth()* de la clase *ControlDraw.js* para cambiar el ancho de línea de un elemento.	

Tabla 46. Tarea 16: Cambiar contenido, tipo y tamaño de fuente del elemento texto

Tarea No.16	**HU7**
Nombre de tarea: Cambiar contenido, tipo y tamaño de fuente del elemento texto.	
Tipo de tarea: Implementación	**Estimación:** 2 días
Fecha de inicio: 15 de febrero del 2014	**Fecha de fin:** 16 de febrero del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Utilizar las funcionalidades de jQuery para cambiar contenido, tipo y tamaño de la fuente del elemento texto.	

Tabla 47. Tarea 17: Diseñar interfaz de la línea de tiempo

Tarea No.17	**HU9**
Nombre de tarea: Diseñar interfaz de la línea de tiempo.	
Tipo de tarea: Implementación	**Estimación:** 1 día
Fecha de inicio: 17 de febrero del 2014	**Fecha de fin:** 17 de febrero del 2014
Programador responsable: Orelbis Lago Vasallo	
Descripción: Diseñar interfaz de la línea de tiempo en el *Index.html* siguiendo las tendencias actuales.	

Tabla 48. Tarea 18: Crear, eliminar y duplicar capa

Tarea No.18	**HU9**
Nombre de tarea: Crear, eliminar y duplicar capa.	
Tipo de tarea: Implementación	**Estimación:** 6 días

Fecha de inicio: 18 de febrero del 2014	**Fecha de fin:** 23 de febrero del 2014
Programador responsable: Orelbis Lago Vasall	
Descripción: Desarrollar las funcionalidades *createLayer(), cloneLayer(select_layer), deleteLayer(select_layer)* de la clase *ControlTimeLine.js* para crear nuevas capas. Además de permitir eliminar y duplicar capas creadas.	

Tabla 49. Tarea 19: Nombrar, visualizar, ocultar y bloquear capa

Tarea No.19	**HU10**
Nombre de tarea: Renombrar, visualizar, ocultar y bloquear capa.	
Tipo de tarea: Implementación	**Estimación:** 7 días
Fecha de inicio: 24 de febrero del 2014	**Fecha de fin:** 2 de marzo del 2014
Programador responsable: Orelbis Lago Vasallo	
Descripción: Desarrollar las funcionalidades *blockLayer(number), rename(select_layer), visualLayer(number)* de la clase *ControlTimeLine.js* para renombrar, visualizar, ocultar y bloquear capas.	

Anexo 3. Tareas de ingeniería, tercera iteración

Tabla 50. Tarea 20: Crear y eliminar fotograma

Tarea No.20	**HU12**
Nombre de tarea: Crear y eliminar fotograma.	
Tipo de tarea: Implementación	**Estimación:** 4 días
Fecha de inicio: 3 de marzo del 2014	**Fecha de fin:** 6 de marzo del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Desarrollar las funcionalidades *createPhotogram(), deletePhotogram(photogram)* en la clase *ControlTimeLine.js* y *clearPhotogram(photogram), clonePhotogram(photogram), addPhotogramPos(), deletePhotogram(photogram)* en la clase *Layer.js*, para crear y eliminar fotograma.	

Tabla 51. Tarea 21: Duplicar y limpiar fotograma

Tarea No.21	**HU12**
Nombre de tarea: Duplicar y limpiar fotograma.	
Tipo de tarea: Implementación	**Estimación:** 3 días
Fecha de inicio: 7 de marzo del 2014	**Fecha de fin:** 9 de marzo del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Adicionar procedimientos necesarios a la función *clearPhotogram(photogram)* y *clonePhotogram(photogram)* en la clase *ControlTimeLine.js* para duplicar y limpiar fotograma.	

Tabla 52. Tarea 22: Reproducir y detener animación

Tarea No.22	**HU13**
Nombre de tarea: Reproducir y detener animación.	
Tipo de tarea: Implementación	**Estimación:** 10 días
Fecha de inicio: 10 de marzo del 2014	**Fecha de fin:** 19 de marzo del 2014
Programador responsable: Orelbis Lago Vasallo	
Descripción: Desarrollar las funcionalidades *animInside(), stopAnimInside(), calcMayorPhotogram()* en la clase *ControlTimeLine.js* para reproducir y detener animación, y que se muestre una vista previa de la animación en la aplicación.	

Tabla 53. Tarea 23: Cambiar cantidad de fotogramas por segundo

Tarea No.23	**HU13**
Nombre de tarea: Cambiar cantidad de fotogramas por segundo.	
Tipo de tarea: Implementación	**Estimación:** 4 días
Fecha de inicio: 20 de marzo del 2014	**Fecha de fin:** 23 de marzo del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Utilizar las funcionalidades del *framework* jQuery en la clase *ControlTimeLine.js* para cambiar la cantidad de fotogramas por segundo en que se reproducirá la animación.	

Anexo 4. Tareas de ingeniería, cuarta iteración

Tabla 54. Tarea 24: Copiar, cortar y pegar elementos

Tarea No.24	**HU8**
Nombre de tarea: Copiar, cortar y pegar un elemento seleccionado.	
Tipo de tarea: Implementación	**Estimación:** 2 días
Fecha de inicio: 24 de marzo del 2014	**Fecha de fin:** 25 de marzo del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Desarrollar los eventos en la clase *ControlDraw.js* para copiar, cortar y pegar elementos.	

Tabla 55. Tarea 25: Generar y visualizar el código de la animación

Tarea No.25	**HU15**
Nombre de tarea: Generar y visualizar el código de la animación.	
Tipo de tarea: Implementación	**Estimación:** 11 días
Fecha de inicio: 26 de marzo del 2014	**Fecha de fin:** 5 de abril del 2014
Programador responsable: Orelbis Lago Vasallo	
Descripción: Desarrollar la funcionalidad *generateCode(fps, canvas_id, url)* en la clase *ControlDocument.js* para generar y visualizar el código en JavaScript. Desarrollar la funcionalidad uploads() en el fichero out.js del servidor.	

Tabla 56. Tarea 26: Descargar la animación

Tarea No.26	**HU15**
Nombre de tarea: Descargar la animación.	
Tipo de tarea: Implementación	**Estimación:** 8 días
Fecha de inicio: 6 de abril del 2014	**Fecha de fin:** 13 de abril del 2014
Programador responsable: Orelbis Lago Vasallo	

Descripción: Desarrollar la funcionalidad download() en el fichero out.js del servidor.

Anexo 5. Tareas de ingeniería, quinta iteración

Tabla 57. Tarea.27: Diseñar la interfaz de panel de elementos dibujados

Tarea No.27	**HU13**
Nombre de tarea: Diseñar la interfaz del panel que contiene el listado de elementos dibujados.	
Tipo de tarea: Implementación	**Estimación:** 2 días
Fecha de inicio: 14 de abril del 2014	**Fecha de fin:** 15 de abril del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Diseñar la interfaz en *Index.html* del panel que contiene el listado de elementos dibujados siguiendo las tendencias actuales.	

Tabla 58. Tarea 28: Visualizar listado de elementos dibujados y seleccionar un elemento

Tarea No.28	**HU13**
Nombre de tarea: Visualizar listado de elementos dibujados y seleccionar un elemento.	
Tipo de tarea: Implementación	**Estimación:** 6 días
Fecha de inicio: 16 de abril del 2014	**Fecha de fin:** 21 de abril del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Desarrollar la funcionalidad *updateTree()* en la clase *ControlListElements.js* para visualizar el listado de los elementos dibujados y permitir seleccionar un elemento del listado. Se utiliza el *framework* jQuery.	

Tabla 59. Tarea 29: Eliminar elemento seleccionado del listado

Tarea No.29	**HU13**
Nombre de tarea: Eliminar elemento seleccionado del listado.	
Tipo de tarea: Implementación	**Estimación:** 6 días
Fecha de inicio: 21 de abril del 2014	**Fecha de fin:** 27 de abril del 2014

Programador responsable: Wendy Reyes Jiménez
Descripción: Utilizar las funcionalidades de jQuery en la clase *ControlListElements.js* para renombrar y eliminar un elemento seleccionado del listado de elementos dibujados.

Tabla 60. Tarea 30: Crear nuevo proyecto

Tarea No.30	**HU14**
Nombre de tarea: Crear nuevo proyecto.	
Tipo de tarea: Implementación	**Estimación:** 2 días
Fecha de inicio: 28 de abril del 2014	**Fecha de fin:** 29 de abril del 2014
Programador responsable: Orelbis Lago Vasallo	
Descripción: Desarrollar la funcionalidad *newProyect()* en la clase *ControlDocument.js* para crear un proyecto nuevo.	

Tabla 61. Tarea 31: Establecimiento estándar XML del proyecto

Tarea No.31	**HU14**
Nombre de tarea: Establecimiento estándar XML del proyecto.	
Tipo de tarea: Configuración	**Estimación:** 1 días
Fecha de inicio: 30 de abril del 2014	**Fecha de fin:** 30 de abril del 2014
Programador responsable: Wendy Reyes Jiménez	
Descripción: Redactar el estándar XML del proyecto. Consultar documentación en la red para la definición del mismo.	

Tabla 62. Tarea 32: Guardar proyecto en XML.

Tarea No.32	**HU14**
Nombre de tarea: Guardar proyecto en XML.	
Tipo de tarea: Implementación	**Estimación:** 5 días
Fecha de inicio: 1 de mayo del 2014	**Fecha de fin:** 5 de mayo del 2014
Programador responsable: Orelbis Lago Vasallo	

Descripción: Desarrollar la funcionalidad *saveProyect()* en la clase *ControlDocument.js* para guardar la animación creada como un documento XML.

Tabla 63. Tarea 33: Abrir proyecto en XML

Tarea No.33	**HU14**
Nombre de tarea: Abrir proyecto en XML.	
Tipo de tarea: Implementación	**Estimación:** 6 días
Fecha de inicio: 6 de mayo del 2014	**Fecha de fin:**11 de mayo del 2014
Programador responsable: Orelbis Lago Vasallo	
Descripción: Desarrollar la funcionalidad *loadProyect()* de la clase *ControlDocument.js* para abrir el proyecto guardado como fichero XML. Desarrollar la funcionalidad uploads() del fichero tmp.js en el servidor.	

Anexo 6. Casos de prueba de aceptación, versión 0.1

Tabla 64. CPA1: Dibujar elemento línea

Caso de prueba de aceptación No.1	
Código: HU1_P1	**HU1**
Nombre: Dibujar elemento línea.	
Condiciones de ejecución: . La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta dibujar línea. 2. Se traza en el escenario manteniendo el clic desde el punto de inicio hasta el punto de fin de la línea.	
Resultado esperado: El gráfico es creado en el fotograma y capa seleccionada y graficado sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 65. CPA2: Dibujar elemento rectángulo

Caso de prueba de aceptación No.2

Código: HU1_P2	**HU1**
Nombre: Dibujar elemento rectángulo.	
Condiciones de ejecución: . La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta dibujar rectángulo. 2. Se traza en el escenario manteniendo el clic desde el punto de inicio hasta el punto de fin del rectángulo.	
Resultado esperado: El gráfico es creado en el fotograma y capa seleccionada y graficado sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 66. CPA3: Dibujar elemento círculo

Caso de prueba de aceptación No.3	
Código: HU1_P3	**HU1**
Nombre: Dibujar elemento círculo.	
Condiciones de ejecución: . La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta para dibujar círculo. 2. Se traza en el escenario dando clic en el punto central del círculo. 3. Se mueve el puntero en dirección horizontal sin soltar el clic para ajustar el tamaño.	
Resultado esperado: El gráfico es creado en el fotograma y capa seleccionada y graficado sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 67. CPA4: Dibujar elemento forma

Caso de prueba de aceptación No.4	
Código: HU1_P4	**HU1**
Nombre: Dibujar elemento forma.	

Condiciones de ejecución: . La capa tiene que estar no bloqueada y visible.
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta para dibujar forma. 2. Se traza en el escenario punto a punto. 3. Si se mueve el cursor sin soltar el clic del último punto dibujado la línea entre él y el punto anterior se convierte en una curva bezier. 4. Se pueden trasladar los puntos ya creados antes de cerrar el polígono. 5. Si el usuario quiere terminar el trazo cuando está dibujando el polígono da clic en el botón *Terminar Trazo*. 6. Si el usuario quiere un polígono cerrado da clic en el punto de inicio del polígono y se termina el trazo de forma automática.
Resultado esperado: El gráfico es creado en el fotograma y capa seleccionada y graficado sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 68. CPA5: Seleccionar color de borde para el elemento a dibujar

Caso de prueba de aceptación No.5	
Código: HU1_P5	**HU1**
Nombre: Seleccionar color de borde para el elemento a dibujar.	
Condiciones de ejecución: . La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se da clic en el desplegable *Borde* de la barra de herramientas. 2. Se puede seleccionar el color y la transparencia en el panel derecho. 3. Se puede seleccionar un color utilizado con anterioridad en el panel izquierdo. 4. Se puede seleccionar un color mediante la configuración rgb(x, y, z) o #abcdef. 5. Se puede limpiar la selección de color mediante el botón superior derecho X. 6. Se da clic en el desplegable *Borde* para que el cambio se efectúe.	
Resultado esperado: El gráfico es creado con el color de borde elegido en el fotograma y capa seleccionada y graficado sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 69. CPA6: Seleccionar color de relleno para el elemento a dibujar

Caso de prueba de aceptación No.6	
Código: HU1_P5	**HU1**
Nombre: Seleccionar color de relleno para el elemento a dibujar.	
Condiciones de ejecución: . La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se da clic en el desplegable *Relleno* de la barra de herramientas. 2. Se puede seleccionar el color y la transparencia en el panel derecho. 3. Se puede seleccionar un color utilizado con anterioridad en el panel izquierdo. 4. Se puede seleccionar un color mediante la configuración rgb(x, y, z) o #abcdef. 5. Se puede limpiar la selección de color mediante el botón superior derecho X. 6. Se da clic en el desplegable *Relleno* para que el cambio se efectúe.	
Resultado esperado: El gráfico es creado con el color de relleno elegido en el fotograma y capa seleccionada y graficado sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 70. CPA7: Insertar elemento texto

Caso de prueba de aceptación No.7	
Código: HU2_P1	**HU2**
Nombre: Insertar elemento texto.	
Condiciones de ejecución: . La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta insertar texto. 2. Se abre un formulario para escribir el texto ajustando tipo y tamaño de fuente. 3. Se da clic en el botón insertar. 4. Se da clic en la posición del escenario en la que se quiere el texto.	
Resultado esperado: El gráfico es creado en el fotograma y capa seleccionada y graficado sin errores.	

Evaluación de la prueba: Satisfactoria.

Tabla 71. CPA8: Insertar imagen

Caso de prueba de aceptación No.8	
Código: HU2_P2	**HU2**
Nombre: Insertar imagen.	
Condiciones de ejecución: Las imágenes a insertar deben estar en la carpeta *Vista/Images/Upload* ubicada en la raíz del programa. . La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta insertar imagen. 2. En el panel de propiedades se da clic en el botón *Examinar*. 3. El usuario navega a la ubicación de las imágenes. 4. Se da clic en posición del escenario donde será insertada.	
Resultado esperado: El gráfico es creado en el fotograma y capa seleccionada y graficado sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 72. CPA9: Seleccionar un elemento del escenario

Caso de prueba de aceptación No.9	
Código: HU3_P1	**HU3**
Nombre: Seleccionar un elemento del escenario.	
Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se escoge la herramienta seleccionar. 2. Se da clic en el elemento que desea seleccionar.	
Resultado esperado: El elemento se selecciona en el fotograma y capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 73. CPA10: Rotar un elemento seleccionado

Caso de prueba de aceptación No.10	
Código: HU3_P2	**HU3**
Nombre: Rotar un elemento seleccionado.	
Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 3. Se selecciona la herramienta rotar. 4. Se da clic en el elemento que desea rotar. 5. Se mueve el puntero horizontalmente, hacia la izquierda el elemento rota a favor de las manecillas del reloj y hacia la derecha en sentido contrario.	
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 74. CPA11: Trasladar el elemento seleccionado

Caso de prueba de aceptación No.11	
Código: HU3_P3	**HU3**
Nombre: Trasladar el elemento seleccionado.	
Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta trasladar. 2. Se da clic en el elemento que desea trasladar. 3. Se suelta el clic en la posición final deseada.	
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 75. CPA12: Escalar el elemento seleccionado

Caso de prueba de aceptación No.12	
Código: HU3_P4	**HU3**
Nombre: Escalar el elemento seleccionado.	
Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta escalar. 2. Se da clic en el elemento que desea escalar. 3. Se mueve el puntero en dirección horizontal, hacia la izquierda aumenta de tamaño el elemento y hacia la derecha disminuye.	
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 76. CPA13: Editar curva de Bézier

Caso de prueba de aceptación No.13	
Código: HU3_P5	**HU3**
Nombre: Editar curva de Bézier.	
Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta editar curva de Bézier. 2. Se da clic en el elemento que se desea editar. 3. Se da clic en el punto de ese elemento que se desea editar. 4. Se mueve el puntero en cualquier dirección hasta obtener la modificación deseada.	
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 77. CPA14: Borrar un elemento

Caso de prueba de aceptación No.14	
Código: HU3_P6	**HU3**
Nombre: Borrar un elemento.	
Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta borrar. 2. Se da clic en el elemento que se desea borrar.	
Resultado esperado: El elemento se elimina del fotograma y capa que lo contiene sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 78. CPA15: Enviar elemento hacia el fondo

Caso de prueba de aceptación No.15	
Código: HU3_P7	**HU3**
Nombre: Enviar elemento hacia el fondo.	
Condiciones de ejecución: Deben haber elementos insertados en el escenario. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona el elemento que se quiere mover hacia detrás de todos los demás elementos del escenario. 2. Se selecciona en la opción *Enviar atrás* en el panel de propiedades.	
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 79. CPA16: Traer elemento al frente

Caso de prueba de aceptación No.16	
Código: HU3_P8	**HU3**

Nombre: Traer elemento al frente.
Condiciones de ejecución: Deben haber elementos insertados en el escenario. La capa tiene que estar no bloqueada y visible.
Entrada/Pasos de ejecución: 1. Se selecciona el elemento que se quiere mover hacia adelante de todos los demás elementos del escenario. 2. Se selecciona en la opción *Traer adelante* en el panel de propiedades.
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 80. CPA17: Mover un punto de los elementos línea, rectángulo, círculo y forma

Caso de prueba de aceptación No.17	
Código: HU4_P1	**HU4**
Nombre: Mover un punto de los elementos línea, rectángulo, círculo y forma.	
Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta mover un punto. 2. Se da clic en el punto del elemento que se va a mover. 3. Se traslada el punto hasta la posición deseada.	
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 81. CPA18: Agregar un punto a los elementos línea, rectángulo, círculo y forma

Caso de prueba de aceptación No.18	
Código: HU4_P2	**HU4**
Nombre: Agregar un punto a los elementos línea, rectángulo, círculo y forma.	

Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta agregar un punto. 2. Se da clic en el borde del elemento seleccionado donde se quiere agregar el punto.
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 82. CPA19: Eliminar un punto a los elementos línea, rectángulo, círculo y forma

Caso de prueba de aceptación No.19	
Código: HU4_P3	**HU4**
Nombre: Eliminar un punto a los elementos línea, rectángulo, círculo y forma.	
Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona la herramienta eliminar un punto. 2. Se da clic en el punto que se desea eliminar del elemento seleccionado.	
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 83. CPA20: Cambiar ancho del escenario

Caso de prueba de aceptación No.20	
Código: HU5_P1	**HU5**
Nombre: Cambiar ancho del escenario.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: 1. Se modifica el valor en píxeles del campo de entrada *Ancho*.	

Resultado esperado: El elemento se modifica sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 84. CPA21: Cambiar alto del escenario

Caso de prueba de aceptación No.21	
Código: HU5_P2	**HU5**
Nombre: Cambiar alto del escenario.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: 1. Se modifica el valor en píxeles del campo de entrada *Alto*.	
Resultado esperado: El elemento se modifica sin errores.	
Evaluación de la prueba: Satisfactoria.	

Anexo 7. Casos de prueba de aceptación, versión 0.2

Tabla 85. CPA22: Cambiar color de borde del elemento seleccionado

Caso de prueba de aceptación No.22	
Código: HU6_P1	**HU6**
Nombre: Cambiar color de borde del elemento seleccionado.	
Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona el elemento que se quiere editar. 2. Se da clic en el desplegable *Borde*. 3. Se puede seleccionar el color y la transparencia en el panel derecho. 4. Se puede seleccionar un color utilizado con anterioridad en el panel izquierdo. 5. Se puede seleccionar un color mediante la configuración rgb(x, y, z) o #abcdef. 6. Se puede limpiar la selección de color mediante el botón superior derecho X. 7. Se da clic en el desplegable *Borde* para que el cambio se efectúe.	

Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 86. CPA23: Cambiar color de relleno del elemento seleccionado

Caso de prueba de aceptación No.23	
Código: HU6_P2	**HU6**
Nombre: Cambiar color de relleno del elemento seleccionado.	
Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo. El elemento debe tener área. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona el elemento que se quiere editar. 2. Se da clic en el desplegable *Relleno*. 3. Se puede seleccionar el color y la transparencia en el panel derecho. 4. Se puede seleccionar un color utilizado con anterioridad del panel izquierdo. 5. Se puede seleccionar un color mediante la configuración rgb(x, y, z) o #abcdef. 6. Se puede limpiar la selección de color mediante el botón superior derecho X. 7. Se da clic en el desplegable *Relleno* para que el cambio se efectúe.	
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 87. CPA24: Cambiar ancho de línea del elemento seleccionado

Caso de prueba de aceptación No.24	
Código: HU6_P3	**HU6**
Nombre: Cambiar ancho de línea del elemento seleccionado. La capa tiene que estar no bloqueada y visible.	
Condiciones de ejecución: Deben haber gráficos dibujados en el lienzo.	
Entrada/Pasos de ejecución: 1. Se selecciona el elemento que se quiere editar.	

2. Se selecciona en la opción *Ancho de línea* el valor en píxeles deseado.
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 88. CPA25: Editar contenido del elemento texto

Caso de prueba de aceptación No.25	
Código: HU7_P1	**HU7**
Nombre: Editar contenido del elemento texto.	
Condiciones de ejecución: Deben haber elementos texto insertados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona el elemento texto que se quiere editar. 2. Se selecciona en la opción *Editar* en el panel de propiedades. 3. Se abre el formulario con el texto existente. 4. Se escribe el contenido del texto en el panel central. 5. Se da clic en el botón *Aceptar* para completar la edición.	
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 89. CPA26: Editar tipo de fuente del elemento texto

Caso de prueba de aceptación No.26	
Código: HU7_P2	**HU7**
Nombre: Editar tipo de fuente del elemento texto.	
Condiciones de ejecución: Deben haber elementos texto insertados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona el elemento texto que se quiere editar. 2. Se selecciona en la opción *Editar* en el panel de propiedades.	

3. Se abre un formulario con el texto existente. 4. Se modifica en el desplegable superior izquierdo el tipo de fuente. 5. Se da clic en el botón *Aceptar* para completar la edición.
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 90. CPA27: Editar tamaño de fuente del elemento texto

Caso de prueba de aceptación No.27	
Código: HU7_P3	**HU7**
Nombre: Editar tamaño de fuente del elemento texto.	
Condiciones de ejecución: Deben haber elementos texto insertados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona el elemento texto que se quiere editar. 2. Se selecciona en la opción *Editar* en el panel de propiedades. 3. Se abre un formulario con el texto existente. 4. Se modifica en el desplegable superior derecho el tamaño de fuente. 5. Se da clic en el botón *Aceptar* para completar la edición.	
Resultado esperado: El elemento se modifica en el fotograma y capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 91. CPA28: Crear capa nueva

Caso de prueba de aceptación No.28	
Código: HU9_P1	**HU9**
Nombre: Crear capa nueva.	
Condiciones de ejecución: Existe una capa creada por defecto.	
Entrada/Pasos de ejecución:	

1. Se selecciona la opción crear nueva capa.
Resultado esperado: Se crea una nueva capa sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 92. CPA29: Eliminar capa seleccionada

Caso de prueba de aceptación No.29	
Código: HU9_P2	**HU9**
Nombre: Eliminar capa seleccionada.	
Condiciones de ejecución: Existen capas creadas.	
Entrada/Pasos de ejecución: 1. Se selecciona la capa a eliminar. 2. Se hace clic en la opción eliminar capa.	
Resultado esperado: Se elimina la capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 93. CPA30: Duplicar capa seleccionada

Caso de prueba de aceptación No.30	
Código: HU9_P3	**HU9**
Nombre: Duplicar capa seleccionada.	
Condiciones de ejecución: Existen capas creadas.	
Entrada/Pasos de ejecución: 1. Se selecciona la capa a duplicar. 2. Se hace clic en la opción duplicar capa.	
Resultado esperado: Se duplican los elementos que pertenecen a la capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 94. CPA31: Nombrar capa seleccionada

Caso de prueba de aceptación No.31	
Código: HU10_P1	**HU10**
Nombre: Renombrar capa seleccionada.	
Condiciones de ejecución: Existen capas creadas.	
Entrada/Pasos de ejecución: 1. Se selecciona la capa a nombrar. 2. Se hace clic en la opción nombrar capa o se da doble clic sobre la capa. 3. Aparece un formulario para introducir el nombre. 4. Se hace clic en el botón *Aceptar*.	
Resultado esperado: Se renombra la capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 95. CPA32: Visualizar capa seleccionada

Caso de prueba de aceptación No.32	
Código: HU10_P2	**HU10**
Nombre: Visualizar capa seleccionada.	
Condiciones de ejecución: Existen capas creadas. Debe haberse ocultado la capa.	
Entrada/Pasos de ejecución: 1. Se hace clic en la opción visualizar capa.	
Resultado esperado: Se visualiza la capa seleccionada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 96. CPA33: Ocultar capa seleccionada

Caso de prueba de aceptación No.33	
Código: HU10_P3	**HU10**
Nombre: Ocultar capa seleccionada.	
Condiciones de ejecución: Existen capas creadas.	

Entrada/Pasos de ejecución: 1. Se hace clic en la opción ocultar capa.
Resultado esperado: Se ocultan los elementos que pertenecen a la capa seleccionada sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 97. CPA34: Bloquear capa seleccionada

Caso de prueba de aceptación No.34	
Código: HU10_P4	**HU10**
Nombre: Bloquear capa seleccionada.	
Condiciones de ejecución: Existen capas creadas.	
Entrada/Pasos de ejecución: 1. Se hace clic en la opción bloquear capa.	
Resultado esperado: La capa bloqueada no puede ser editada. Le sale un mensaje al usuario si intenta editarla.	
Evaluación de la prueba: Satisfactoria.	

Tabla 98. CPA35: Desbloquear capa seleccionada

Caso de prueba de aceptación No.35	
Código: HU10_P5	**HU10**
Nombre: Desbloquear capa seleccionada.	
Condiciones de ejecución: Existen capas creadas y bloqueadas.	
Entrada/Pasos de ejecución: 1. Se hace clic en la opción desbloquear capa.	
Resultado esperado: La capa es desbloqueada sin errores.	
Evaluación de la prueba: Satisfactoria.	

Anexo 8. Casos de prueba de aceptación, versión 0.3

Tabla 99. CPA36: Crear fotograma nuevo

Caso de prueba de aceptación No.36	
Código: HU11_P1	**HU11**
Nombre: Crear fotograma nuevo.	
Condiciones de ejecución: Existe un fotograma creado por defecto.	
Entrada/Pasos de ejecución: 1. Se selecciona la opción crear fotograma.	
Resultado esperado: Se crea un fotograma nuevo al final de la línea de tiempo sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 100. CPA37: Eliminar fotograma seleccionado

Caso de prueba de aceptación No.37	
Código: HU11_P2	**HU11**
Nombre: Eliminar fotograma seleccionado.	
Condiciones de ejecución: Existen fotogramas creados.	
Entrada/Pasos de ejecución: 1. Se selecciona el fotograma a eliminar. 2. Se da clic en la opción eliminar fotograma.	
Resultado esperado: Se elimina el fotograma sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 101. CPA38: Duplicar fotograma seleccionado

Caso de prueba de aceptación No.38	
Código: HU11_P3	**HU11**
Nombre: Duplicar fotograma seleccionado.	
Condiciones de ejecución: Existen fotogramas creados.	

Entrada/Pasos de ejecución: 1. Se selecciona el fotograma a duplicar. 2. Se da clic en la opción duplicar fotograma.
Resultado esperado: El fotograma es duplicado sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 102. CPA39: Limpiar fotograma seleccionado

Caso de prueba de aceptación No.39	
Código: HU11_P4	**HU11**
Nombre: Limpiar fotograma seleccionado.	
Condiciones de ejecución: Existen fotogramas creados.	
Entrada/Pasos de ejecución: 1. Se selecciona el fotograma a limpiar. 2. Se da clic en la opción limpiar fotograma.	
Resultado esperado: Se eliminan todos los elementos del fotograma limpiado sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 103. CPA40: Reproducir animación

Caso de prueba de aceptación No.40	
Código: HU12_P1	**HU12**
Nombre: Reproducir animación.	
Condiciones de ejecución: Existen capas y fotogramas creados.	
Entrada/Pasos de ejecución: 1. Se hace clic en el botón *Reproducir*.	
Resultado esperado: Se visualiza la animación sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 104. CPA41: Detener animación

Caso de prueba de aceptación No.41	
Código: HU12_P2	**HU12**
Nombre: Detener animación.	
Condiciones de ejecución: Existen capas y fotogramas creados.	
Entrada/Pasos de ejecución: 1. Se hace clic en el botón *Detener*.	
Resultado esperado: Se detiene la animación sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 105. CPA42: Cambiar cantidad de fotogramas por segundo de la animación

Caso de prueba de aceptación No.42	
Código: HU12_P3	**HU12**
Nombre: Cambiar cantidad de fotogramas por segundo de la animación.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: 1. Se modifica el valor del campo de entrada *FPS,* donde se introduce la cantidad de fotogramas por segundos a la que se visualizará la animación.	
Resultado esperado: Se cambian la cantidad de fotogramas por segundo sin errores.	
Evaluación de la prueba: Satisfactoria.	

Anexo 9. Casos de prueba de aceptación, versión 0.4

Tabla 106. CPA43: Copiar un elemento

Caso de prueba de aceptación No.43	
Código: HU8_P1	**HU8**
Nombre: Copiar un elemento.	
Condiciones de ejecución: Deben haber elementos texto insertados en el lienzo. La capa tiene que estar no bloqueada y visible.	

Entrada/Pasos de ejecución: 1. Se selecciona el elemento a copiar. 2. Se da clic en la opción copiar o con la combinación de teclas Ctrl+C.
Resultado esperado: Se copia el elemento sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 107. CPA44: Cortar un elemento

Caso de prueba de aceptación No.44	
Código: HU8_P2	**HU8**
Nombre: Cortar un elemento.	
Condiciones de ejecución: Deben haber elementos texto insertados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona el elemento a cortar. 2. Se da clic en la opción cortar o con la combinación de teclas Ctrl+X.	
Resultado esperado: Se corta el elemento de la capa y fotograma sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 108. CPA45: Pegar un elemento

Caso de prueba de aceptación No.45	
Código: HU8_P3	**HU8**
Nombre: Pegar un elemento.	
Condiciones de ejecución: Tienen que haber elementos copiados o cortados. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona el fotograma y capa donde se desea pegar el elemento. 2. Se da clic en la opción pegar o con la combinación de teclas Ctrl+V.	
Resultado esperado: Se pega el elemento en la capa y fotograma seleccionados sin errores.	

Evaluación de la prueba: Satisfactoria.

Tabla 109. CPA46: Generar código JavaScript

Caso de prueba de aceptación No.46	
Código: HU15_P1	**HU15**
Nombre: Generar código JavaScript.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: 1. Se selecciona la opción *Exportar*. 2. Se introduce el id del canvas del proyecto donde será utilizada la animación. 3. Se introduce la ruta de la carpeta donde están las imágenes si tiene. 4. Se selecciona la opción *Repetir animación* si se quiere que la animación al concluir vuelva a iniciar. 5. Se selecciona el botón *Generar animación* para mostrar el código y generar la animación. 6. Se selecciona el botón *Descargar* y se ubica la carpeta del ordenador donde se va a guardar la animación. 7. Se selecciona el botón *Vista previa* para ver el código en el formulario. 8. Se selecciona el botón *Cancelar* para salir de la ventana.	
Resultado esperado: El código es generado sin errores.	
Evaluación de la prueba: Satisfactoria.	

Anexo 10. Casos de prueba de aceptación, versión 1.0

Tabla 110. CPA47: Visualizar listado de elementos dibujados por fotograma

Caso de prueba de aceptación No.47	
Código: HU13_P1	**HU13**
Nombre: Visualizar listado de elementos dibujados en el fotograma seleccionado.	
Condiciones de ejecución: Deben existir elementos dibujados en el lienzo.	
Entrada/Pasos de ejecución:	

Resultado esperado: Se visualiza el listado de los elementos del fotograma seleccionado sin errores.
Evaluación de la prueba: Satisfactoria.

Tabla 111. CPA48: Seleccionar elemento dibujado del listado

Caso de prueba de aceptación No.48	
Código: HU13_P2	**HU13**
Nombre: Seleccionar elemento dibujado del listado.	
Condiciones de ejecución: Deben existir elementos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona un elemento del listado de elementos dibujados por fotogramas.	
Resultado esperado: El elemento se muestra seleccionado en el escenario y en el listado sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 112. CPA49: Eliminar elemento seleccionado del listado

Caso de prueba de aceptación No.49	
Código: HU13_P3	**HU13**
Nombre: Eliminar elemento seleccionado del listado.	
Condiciones de ejecución: Deben existir elementos dibujados en el lienzo. La capa tiene que estar no bloqueada y visible.	
Entrada/Pasos de ejecución: 1. Se selecciona un elemento del listado de elementos dibujados por fotogramas. 2. Se hace clic en el botón *Eliminar*.	
Resultado esperado: El elemento se elimina del escenario y del listado sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 113. CPA50: Crear nuevo documento

Caso de prueba de aceptación No.50	
Código: HU14_P1	**HU14**
Nombre: Crear nuevo documento.	
Condiciones de ejecución:	
Entrada/Pasos de ejecución: 1. Se selecciona la opción *Nuevo*. 2. Sale en la pantalla una ventana de confirmación con la opción de guardar el proyecto actual. 3. Si se hace clic en *Aceptar* se crea un nuevo proyecto sin guardar el proyecto anterior. 4. Se hace clic en el botón *Cancelar* se vuelve al proyecto actual.	
Resultado esperado: El proyecto es creado sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 114. CPA51: Abrir proyecto existente en XML

Caso de prueba de aceptación No.51	
Código: HU14_P2	**HU14**
Nombre: Abrir proyecto existente en XML.	
Condiciones de ejecución: Debe existir un proyecto guardado.	
Entrada/Pasos de ejecución: 1. Se selecciona la opción *Cargar proyecto*. 2. Se navega hasta la localización del proyecto guardado. 3. Se da clic en la opción *Aceptar*.	
Resultado esperado: El proyecto es cargado y visualizado en la aplicación sin errores.	
Evaluación de la prueba: Satisfactoria.	

Tabla 115. CPA52. Salvar documento como XML

Caso de prueba de aceptación No.52	
Código: HU14_P3	**HU14**

Nombre: Salvar documento como XML.
Condiciones de ejecución:
Entrada/Pasos de ejecución: 1. Se selecciona la opción *Guardar*. 2. Se selecciona el nombre para el documento. 3. Se da clic en la opción *Guardar*. 4. Se navega hasta la localización a guardar. 5. Se fa clic en la opción *Cancelar* para eliminar la acción.
Resultado esperado: El documento es guardado sin errores.
Evaluación de la prueba: Satisfactoria.

Anexo 11. Casos de pruebas unitarias

Tabla 116. Caso de prueba unitaria No.1

Caso de prueba unitaria No.1					
Iteración: 2					
Descripción: Se prueba la obtención de un fotograma de una capa específica					
Responsable: Orelbis Lago Vasallo					
Funcionalidad	**Método utilizado**	**Recibe**	**Respuesta esperada del método**	**Respuesta real del método**	**Observaciones**
getPhotogramPosition()	getPhotogramPosition()	new Photogram(0, 0, 'Fotograma', '', [], '#671a1a', 0, 0, 0, 0)	Se obtiene un fotograma	1 assertions of 1 passed, 0 failed	Método de la clase Layer.

Tabla 117. Caso de prueba unitaria No.2

Caso de prueba unitaria No.2

Iteración: 2					
Descripción: Se prueba la obtención de un fotograma nuevo de una capa específica					
Responsable: Orelbis Lago Vasallo					
Funcionalidad	**Método utilizado**	**Recibe**	**Respuesta esperada del método**	**Respuesta real del método**	**Observaciones**
getPhotogramPositionNew()	getPhotogramPositionNew()	new Photogram(0, 0, 'Fotograma', '', [], '#671a1a', 0, 0, 0, 0)	Se obtiene un nuevo fotograma	1 assertions of 1 passed, 0 failed	Método de la clase Layer.

Tabla 118. Caso de prueba unitaria No.3

Caso de prueba unitaria No.3					
Iteración: 2					
Descripción: Se prueba la obtención de un fotograma de una capa específica					
Responsable: Orelbis Lago Vasallo					
Funcionalidad	**Método utilizado**	**Recibe**	**Respuesta esperada del método**	**Respuesta real del método**	**Observaciones**
deletePhotogram()	deletePhotogram()	0	undefined	1 assertions of 1 passed, 0 failed	Método de la clase Layer.
deletePhotogram()	deletePhotogram()	1	1	1 assertions of 1 passed, 0 failed	Método de la clase Layer.

Tabla 119. Caso de prueba unitaria No.4

Caso de prueba unitaria No.4
Iteración: 4

Descripción: Se prueba la obtención de un fotograma de una capa específica					
Responsable: Wendy Reyes Jiménez					
Funcionalidad	**Método utilizado**	**Recibe**	**Respuesta esperada del método**	**Respuesta real del método**	**Observaciones**
sumPoint(p1,p2)	sumPoint(p1,p2)	p1={1,2} p2={4,4}	{5,6}	1 assertions of 1 passed, 0 failed.	Método de la clase ControlDocument.

Anexo 12. Resumen de entrevistas con personal del Centro FORTES

Preguntas realizadas en la entrevista con desarrolladores de animaciones web de varios proyectos del Centro FORTES.

- Qué rol desempeña en el proyecto.
- Le gusta desarrollar animaciones web.
- Qué herramientas utiliza para crear las animaciones web.
- Si no usan ninguna, por qué y con qué tecnología las desarrollan.
- Qué tiempo promedio se demora en crear una animación web.

Resumen de las respuestas

Todos los miembros de los equipos que crean las animaciones web son desarrolladores. No utilizan ninguna herramienta especializada para la creación de estos recursos porque las que tienen las prestaciones necesarias son de licencia privativa y no siguen la línea de desarrollo libre del Centro FORTES. Por lo que hay que tener dominio avanzado de los lenguajes CSS, JavaScript y HTML5, además de los frameworks EaselJS, Paper.js u otro, que son utilizados para el trabajo en la etiqueta <canvas> de HTML5. Por estas razones el tiempo promedio del desarrollo de las animaciones depende del dominio por parte de los desarrolladores de todas estas tecnologías. El proceso de crear estos componentes visuales consume muchos recursos del ordenador, porque es necesario tener varias herramientas ejecutándose como el IDE y el navegador.