INSTITUTO POLITÉCNICO NACIONAL
ESCUELA SUPERIOR DE CÓMPUTO

ESCOM

Trabajo Terminal

"Herramienta Web para la creación de páginas web animadas con HTML 5 y CSS 3" 2012 – A028

Que para cumplir con la opción de titulación curricular en la carrera de:

"Ingeniería en Sistemas Computacionales"

Presentan

López Rivas Carlos Esteban
Hernández Sánchez Christian Adán
Ramírez Ibarra Jair

Director

Dr. *Miguel Santiago Suárez Castañón*

México DF, Mayo 2013

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

SUBDIRECCIÓN ACADÉMICA

No. de Registro: 2012-A028 Serie: Amarilla 16 de Mayo de 2013

Documento técnico

Herramienta Web para la creación de páginas web animadas con HTML 5 y CSS 3

Presentan

López Rivas Carlos Esteban[1]
Hernández Sánchez Christian Adán[2]
Ramírez Ibarra Jair[3]

Director

Dr. Miguel Santiago Suárez Castañón

RESUMEN

El trabajo terminal presentado consiste en el desarrollo de una aplicación Web que permita a los usuarios de la misma crear animaciones que puedan ser visualizadas en los navegadores más populares. Dicha herramienta Web permite a usuarios que no poseen conocimientos de programación generar animaciones. Las animaciones son generadas con código HTML CSS y JavaScript.

PALABRAS CLAVE: CSS, HTML, JavaScript, Navegador Web, Servicio Web, TypeScript, W3C, WYSIWYG.

[1] E-mail: carlos@blewblew.com
[2] E-mail: chadancito@gmail.com
[3] E-mail: jairamirez91@gmail.com

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

SUBDIRECCIÓN ACADÉMICA

ESCOM

DEPARTAMENTO DE FORMACIÓN INTEGRAL E INSTITICIONAL

COMISIÓN ACADÉMICA DE TRABAJO TERMINAL

ING APOLINAR FCO. CRUZ LÁZARO
PRESIDENTE DE LA COMISIÓN ACADÉMICA DE TRABAJO TERMINAL
PRESENTE

Por medio del presente, informo que los alumnos que integran el **TRABAJO TERMINAL 2012-A028** titulado **"Herramienta Web para la creación de páginas web animadas con HTML 5 y CSS 3",** concluyeron satisfactoriamente su trabajo.

El empastado del Reporte Técnico Final y el Disco Compacto (CD) fueron revisados ampliamente por su servidor y corregidos, cubriendo el alcance y el objetivo planeados en el protocolo original y de acuerdo a los requisitos establecidos por la Comisión que Usted preside.

ATENTAMENTE

Dr. Miguel Santiago Suárez Castañón

Advertencia

"Este documento contiene información desarrollada por la Escuela Superior de Cómputo del Instituto Politécnico Nacional, a partir de datos y documentos con derecho de propiedad y por lo tanto, su uso quedará restringido a las aplicaciones que explícitamente se convengan."

La aplicación no convenida exime a la escuela su responsabilidad técnica y da lugar a las consecuencias legales que para tal efecto se determinen.

Información adicional sobre este reporte técnico podrá obtenerse en:

La Subdirección Académica de la Escuela Superior de Cómputo del Instituto Politécnico Nacional, situada en Av. Juan de Dios Bátiz s/n Teléfono: 57296000 Extensión 52000

ÍNDICE

1 ÍNDICE DE TABLAS

2 ÍNDICE DE FIGURAS

3 INTRODUCCIÓN

3.1 ANTECEDENTES

El desarrollo de aplicaciones Web ha cambiado constantemente desde su aparición. Dichas aplicaciones han pasado de ser páginas sencillas con grandes cantidades de texto a herramientas completamente animadas, coloridas y que presentan una gran variedad de interacción con el usuario, haciendo de la Web un lugar más cómodo y atractivo al pasar el tiempo. Las tres tecnologías principales en las que se basa toda aplicación Web son: JavaScript, HTML y CSS. Dichas tecnologías también han incorporado soporte para nuevas funcionalidades. Así mismo, los navegadores Web también han sufrido transformaciones al ser modificadas las tecnologías subyacentes de toda aplicación Web. Un elemento que ha sido fundamental en las nuevas aplicaciones, es la generación de contenido animado. Sin embargo las versiones anteriores de estas tres tecnologías (JavaScript, HTML y CSS) no incorporaban soporte específico para crear animaciones, haciendo de esta tarea, un trabajo difícil y no estandarizado. Así, surgieron productos como Adobe Flash y Microsoft Silverlight que permiten la creación de contenido animado pero que no están basados en las tres tecnologías que hemos mencionado, haciendo la interoperabilidad una tarea dependiente de la tecnología empleada para crear animaciones y no del apego al estándar.

3.2 DEFINICIÓN DEL PROBLEMA

Actualmente la W3C[4] está trabajando en la generación de nuevas versiones de JavaScript, HTML y CSS. Estas nuevas versiones incluyen soporte, entre otras cosas, a la creación de contenido animado. Sin embargo al ser estándares en desarrollo, como lo especifica la W3C, aún es difícil trabajar basándose completamente en ellos. Además, al ser estándares relativamente nuevos, no existen herramientas que faciliten el desarrollo de aplicaciones o que oculten detalles de implementación, limitando el rango de desarrolladores solo a aquellos que cuentan con suficientes conocimientos de las tecnologías y dejando de lado a usuarios comunes que no cuenten con conocimientos de programación, pero que necesiten o deseen crear animaciones Web.

Otro problema, es que la velocidad de actualización de los navegadores es diferente en cada uno de ellos, por lo que algunos pueden estar apegados a los nuevos estándares antes que otros, creando conflictos de interoperabilidad y por consiguiente, tomando más tiempo a los desarrolladores construir sitios Web atractivos.

3.3 SOLUCIÓN PROPUESTA

Nuestra propuesta de solución consiste en facilitar el desarrollo de sitios Web animados utilizando tecnologías nativas al navegador, como HTML, CSS y JavaScript, aprovechando las nuevas versiones de éstas tecnologías. Para esto, se desarrolló una herramienta tipo WSYIWYG[5] que genera código específico para cada navegador. Esto permite que el sitio desarrollado por el usuario corra de manera idéntica en los navegadores que la solución soporta.

[4] La World Wide Web Consortium es una comunidad internacional en la que las organizaciones miembro, el personal o staff de tiempo completo y el público trabajan juntos para desarrollar los estándares de la Web.
[5] What You See Is What You Get

La herramienta en cuestión es 100% Web, esto permite que la solución pueda ejecutarse en diversos navegadores.

La aplicación también estará compuesta por una parte que correrá en un servidor Web para proporcionar la persistencia del código generado.

3.4 JUSTIFICACIÓN

En la actualidad, la tendencia de crear sitios atractivos a los usuarios a través de contenido enriquecido como videos, imágenes y animaciones se ha extendido en prácticamente todos los desarrollos de aplicaciones en el mundo. El desarrollo de animaciones atractivas ha sido un éxito entre los interesados en desarrollo de aplicaciones Web. A partir de este gran éxito, con el tiempo fue necesario crear herramientas de desarrollo que hicieran posible la creación de animaciones creativas en la menor cantidad de tiempo. De tal modo que se han creado herramientas intuitivas y fáciles de usar para el desarrollo de animaciones. Actualmente se cuenta con soluciones completas, que hasta cierto punto han satisfecho las necesidades que hasta hace unos años existían.

Sin embargo, con la proliferación de Internet en prácticamente todos los ambientes de nuestra vida, comenzaron a surgir diversas corrientes tecnológicas, como Adobe Flash, que a la larga han provocado que las soluciones desarrolladas anteriormente para la creación de animaciones para la Web, no sean desplegables en todos los navegadores, dispositivos y tecnologías disponibles actualmente. Esto hace obligatorio el uso de aplicaciones externas, como Adobe Flash Player, para la visualización de dichos contenidos. Esto, finalmente, ha dado como resultado que toda la industria del desarrollo Web dependa de dichas tecnologías para poder crear sitios animados. El depender de estas tecnologías crea problemas de compatibilidad, debido a que no están disponibles para todos los dispositivos que se utilizan hoy en día para acceder a la Web, especialmente los móviles. Esto se debe a la necesidad de re-implementar el software de Adobe Flash Player para cada arquitectura.

Ahora bien, con los nuevos avances en las tecnologías Web es posible crear soluciones generales, apegadas al estándar, que resuelvan los problemas de compatibilidad que tenemos actualmente con diversos dispositivos y navegadores. Es posible dejar de crear aplicaciones basadas en tecnologías externas a los estándares del desarrollo Web, sin dejar de crear sitios atractivos, vistosos e interesantes a los usuarios.

HTML 5 y CSS 3, al ser estándares en desarrollo, tienen el inconveniente de no contar aún con herramientas que hagan intuitiva, fácil y rápida la generación de animaciones con dichas tecnologías.

Así, nuestra solución consiste en la creación de una herramienta que haga posible el desarrollo de animaciones a través de la generación de código JavaScript y CSS. De este modo, es posible dejar de lado todas las tecnologías externas actualmente usadas que no forman parte del estándar para el desarrollo Web, y suplirlas con tecnologías nativas a los navegadores, que estén sujetas completamente al estándar actual, haciendo posible, no solamente la compatibilidad con todos los navegadores que lo sigan, sino con todos los dispositivos que soporten la visualización de código HTML 5, CSS 3 y JavaScript; incluyendo a los dispositivos móviles.

El proyecto se desarrolló haciendo uso principalmente de CSS 3 y ECMAScript 5.

Para el back-end (administración de proyectos) se utilizará el patrón de diseño MVC[6]. El patrón MVC es utilizado en la mayoría de las Aplicaciones Web, debido a la naturaleza de la Web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página. El modelo es el Sistema de Gestión de Base de Datos y la Lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

El patrón de diseño modelo-vista-controlador consiste en separar la aplicación en 3 tipos de componentes: Modelos, encargados de representar la lógica del negocio, generalmente consiste en las clases de almacenamiento en el sistema o entidades dentro de una base de datos; Vistas, encargadas de mostrar la información al usuario por medio de algún formato, en nuestro caso HTML; y Controladores, encargados de recibir eventos de entrada, así como pasar parámetros a la vista que fueron obtenidos de cierto modelo.

3.5 OBJETIVO

Crear una herramienta 100% Web que facilite a desarrolladores experimentados y a usuarios que no poseen conocimientos en programación, la creación de animaciones Web con tecnologías estándar (JavaScript, CSS, HTML) ocultando detalles de implementación y reduciendo el problema de interoperabilidad.

3.5.1 Objetivos específicos

Objetivos específicos
La aplicación es fácil e intuitiva para un usuario que no tiene experiencia de programación
La aplicación es usable en tablets
El código resultante se genera instantáneamente, actualizando la aplicación en tiempo real
El usuario puede insertar - Texto - Figuras geométricas - Imágenes - Videos - Tablas - Listas
El código resultante puede ejecutarse con un alto grado de similitud en Microsoft Internet Explorer, Mozilla Firefox, Google Chrome y Safari, así como sus versiones móviles.
Múltiples elementos pueden seleccionarse en el canvas
El usuario puede animar elementos.
El usuario puede modificar algunas de las propiedades de los elementos desde la aplicación sin necesidad de escribir o modificar código CSS directamente.
La aplicación puede ejecutarse con un alto grado de similitud en las versiones de escritorio de Internet Explorer, Chrome, Firefox y Safari.
La aplicación soportará la persistencia del código generado mediante una aplicación Web.

Tabla 1: Objetivos específicos

[6] Patrón de diseño de software Modelo-Vista-Controlador

3.6 ESTADO DEL ARTE

Hoy en día el desarrollo de sitios Web depende de muchas tecnologías que ofrecen diversas funcionalidades. Una de ellas, Adobe Flash, ha sido fundamental en el desarrollo de animaciones y ha generado una gran dependencia a dicha tecnología, sin embargo, al ser una tecnología independiente de los estándares para el desarrollo Web, muchos navegadores y dispositivos no son capaces de visualizar muchos contenidos creados a partir de esta tecnología.

Actualmente no existen muchas alternativas a este problema, sin embargo, es importante considerar la adopción de HTML 5 y CSS 3 que hacen posible implementar funcionalidad similar a la de Adobe Flash. La mayoría de los navegadores más usados en las computadoras convencionales ya soportan estas tecnologías, incluyendo Internet Explorer, Firefox, Chrome y Safari; también algunos navegadores móviles, como el de Android y iOS.

Hasta donde nosotros sabemos no ha sido desarrollado en la ESCOM un trabajo terminal similar al que proponemos. Sin embargo, existen actualmente varias alternativas. En la tabla siguiente se muestran las características y desventajas, en comparación con la solución propuesta, de estas herramientas.

Herramienta	**Descripción**
Adobe Edge Animate	Animación completamente en Javascript, HTML 5, y CSS 3. Hace uso del canvas para prácticamente todas las aplicaciones
Sencha	Sencha Complete ofrece a los desarrolladores acceso a la mejor combinación de tecnología, las herramientas y el apoyo para el desarrollo de plataformas múltiples aplicaciones Web basadas en HTML5. Tiene como desventaja que sólo está disponible en MAC OS X.
Purple	Purple es una herramienta de animación. Está diseñado para crear cuenta cuentos visuales en el iPad o en el iPhone. Con Purple, se pueden crear animaciones en 2D de mapa de bits utilizando una línea de tiempo-editor. Purple soporta múltiples escenas y transiciones entre ellos. Funcionalidad básica de interacción permite definir saltar a las escenas provocadas por botones.
Tumult Hype	Herramienta que permite el desarrollo de contenido Web en HTML5 de manera rápida y sin necesidad de escribir código. Las animaciones y el contenido interactivo creado con Tumult Hype es posible ser visualizado en aplicaciones de escritorio,

	smartphones, iPads.
Radi	Herramienta que permite la creación de imágenes vectoriales, animaciones, videos, efectos, gráficos en tiempo real. Genera código HTML y por consiguiente es compatible con smartphones, iPads, eBooks. No requiere del plug-in de Flash.

Tabla 2: Comparación de herramientas existentes

4 MARCO TEÓRICO

4.1 WORLD WIDE WEB

La *World Wide Web* (abreviado como WWW o W3, comúnmente conocida como la Web) es un sistema de documentos de hipertexto interrelacionados accedidos vía Internet. Con un navegador Web, es posible visualizar páginas Web que puedan contener texto, imágenes, video, y otros elementos multimedia, y navegar entre ellos por medo de hyperlinks.

Usando conceptos de su sistema de hipertexto predecesor como ENQUIRE, el ingeniero inglés, científico y en ese tiempo empleado del CERN[7], Sir Tim Berners-Lee, ahora director de la World Wide Web Consortium, escribió una propuesta en marzo de 1989 para la cual eventualmente vendría a convertirse la World Wide Web. En el CERN,, Berners-Lee y el científico Robert Cailliau propusieron en 1990 el uso de hipertexto para "ligar y acceder información de varios tipos como una red de nodos en la cual el usuario puede buscar a su voluntad" e introdujeron públicamente el proyecto en Diciembre del mismo año[8].

4.2 NAVEGADOR WEB

Un navegador Web es una aplicación para obtener, presentar y enviar fuentes de información en la World Wide Web. Una fuente de información puede ser una página Web, una imagen, un video o alguna otra pieza de información[9]. Dicha fuente de información está identificada por un URI[10]. Un navegador Web también puede ser definido como un programa diseñado para permitir a los usuarios acceder, obtener y visualizar documentos y otras fuentes de información en Internet. Aunque los navegadores Web están primordialmente enfocados a usar la World Wide Web, también pueden ser usados para acceder a información proporcionada por servidores Web en redes

[7] El CERN es el Conseil Européen pour la Recherche Nucléaire (Centro Europeo de Investigacion Nuclear). El CERN es una organización nuclear europea cerca de Geneva, ubicada en el borde entre Francia y Suiza.
[8]Berners-Lee, Tim "Pre-W3C Web and Internet Background". World Wide Web Consortium
[9] Jacobs, Ian; Walsh, Norman (15 December 2004). "URI/Resource Relationships". *Architecture of the World Wide Web, Volume One*.World Wide Web Consortium.

[10] Uniform Resource Identifier.

privadas o archivos almacenados en un sistema de archivos. Entre los navegadores Web más importantes se encuentran Google Chrome, Mozilla Firefox, Internet Explorer, Opera y Safari[11].

4.3 HTML

HyperText Markup Language es el principal lenguaje de marcado para crear páginas Web y otra información que pueda ser mostrada en un navegador Web.

HTML está escrito en la forma de elementos HTML formados de etiquetas encerradas en los símbolos (<>). Como ejemplo tenemos el elemento HTML el cual se representa como (<html>). Las etiquetas HTML generalmente vienen en pares como <h1> y </h1> aunque algunas etiquetas conocidas como elementos vacíos, no tienen par, como ejemplo <img>. La primera etiqueta en un par es la etiqueta de inicio, la segunda etiqueta es la etiqueta final (también pueden llamarse como etiqueta que abre y etiqueta que cierra) En medio de estas etiquetas los diseñadores Web pueden agregar texto, etiquetas, comentarios y otros tipos de contenido basados en texto.

El propósito de un navegador Web es leer documentos HTML y transformarlos en visibles o audibles páginas Web. El navegador no muestra las etiquetas, pero dichas etiquetas sirven para que el navegador interprete cómo debe de mostrar el contenido de la página.

Los elementos HTML forman los bloques que construyen todos los sitios Web. HTML permite agregar imágenes y objetos dentro de una página y pueden ser usados para crear formularios interactivos. También permite agregar pequeños fragmentos de código llamados scripts escritos en lenguajes como JavaScript, los cuales afectan el comportamiento de las páginas Web.

Los navegadores Web también pueden referenciar “Hojas de estilo” (CSS en inglés, Cascading Style Sheet) para definir la apariencia y diseño del texto y otro material. La W3C, la organización que mantiene los dos estándares, alienta a los desarrolladores a usar CSS en lugar de HTML para este propósito[12].

4.4 CSS

Cascading Style Sheets es un lenguaje de hoja de estilos usado para describir la presentación semántica (la apariencia y el formato) de un documento escrito en un lenguaje de marcado. Su más común aplicación es aplicar estilos a páginas Web escritas en HTML, pero el lenguaje puede ser aplicado a cualquier tipo de documento XML[13].

CSS está diseñado primordialmente para permitir a los usuarios la separación del contenido de un documento (escrito en HTML o en un lenguaje de marcado parecido) de la presentación del mismo incluyendo elementos como diseño, colores y fuentes[14]. Esta separación puede mejorar el acceso al contenido, proveer mejor flexibilidad y control en la especificación de las características de la presentación, permite que muchas páginas sean formateadas con el mismo estilo y reduce la

[11] "Tim Berners-Lee: WorldWideWeb, the first Web client". W3.org.

[12] HTML 4 --- Conformance: Requirements and Recomendations. W3.org.

[13] eXtended Markup Language

[14] What is CSS? World Wide Web Consortium W3C.org

complejidad y la repetición del contenido estructural (como es el caso del diseño no basado en tablas). CSS también permite que una página sea presentada en diferentes estilos para diferentes métodos de presentación. También puede ser usado para permitir que la página sea presentada diferente dependiendo del tamaño de la pantalla o dispositivo en que se esté visualizando. Mientras que el autor de una página Web, típicamente liga una página Web a una hoja de estilo CSS, el lector puede usar una hoja diferente, quizá en su propia computadora, sobrescribiendo la que el autor ha especificado.

CSS especifica un esquema de prioridad para determinar cuáles reglas de estilo se aplicarán si más de una regla aplica a un elemento en particular. En esta llamada *cascada,* las prioridades o *pesos* son asignadas a reglas de tal modo que los resultados son predecibles.

Las especificaciones de CSS son mantenidas por la W3C.

4.5 JAVASCRIPT

JavaScript es un lenguaje de programación interpretado. Fue originalmente implementado como parte de los navegadores Web de tal modo que fragmentos de código pudieran interactuar con el usuario, controlar el navegador, comunicarse asíncronamente, y alterar el contenido del documento que está siendo presentado.

JavaScript es un lenguaje de scripting, prototipado, dinámico, de tipado débil, y tiene funciones de primera clase. Su sintaxis fue fuertemente influenciada por el lenguaje C. JavaScript copió muchos nombres y convenciones de Java, pero estos dos lenguajes no están relacionados y tienen semánticas muy diferentes. Es un lenguaje multiparadigma, soportando la orientación a objetos, y los estilos imperativo y funcional.

El uso de JavaScript en aplicaciones fuera de las páginas Web (e.g. en documentos PDF, navegadores de sitios específicos, y controles de aplicaciones de escritorio) es también significativo. Recientes y más veloces máquinas virtuales de JavaScript y frameworks construidos sobre ellas (Notablemente Node.js) también han incrementado la popularidad de JavaScript para aplicaciones que corren en servidores Web.

JavaScript fue formalizado en el estándar del lenguaje ECMAScript y primordialmente usado como parte del navegador Web. Esto permite acceso por medio de programación a objetos computacionales dentro de un ambiente local[15].

4.6 TYPESCRIPT

TypeScript es un lenguaje de programación libre y de código abierto desarrollado por Microsoft. TypeScript es un estricto superconjunto de JavaScript, y esencialmente agrega tipado estático opcional y orientación a objetos basado en clases al lenguaje.

TypeScript extiende la sintaxis de JavaScript, de tal modo que cualquier programa funcionando, escrito en JavaScript trabaja con TypeScript sin ningún cambio. TypeScript está diseñado para el desarrollo de grandes aplicaciones.

[15] Proposed ECMAScript 4th Edition – Language Overview

TypeScript soporta cabeceras de archivos los cuales agregan tipos de información a librerías existentes escritas en JavaScript, extendiendo los beneficios de librerías populares como JQuery, MongoDB o Node.js.

TypeScript se origina de la necesidad de desarrollar aplicaciones JavaScript de gran escala.[16].

4.7 FRAMEWORK

En programación un framework o software framework es una abstracción en la que software que provee alguna funcionalidad general puede ser selectivamente cambiado por código escrito por usuarios adicionales, proveyendo de esta manera software de aplicación específica. Un framework es una universal, reusable plataforma de software usada para desarrollar aplicaciones, productos y soluciones. Frameworks incluyen programas de soporte, compiladores, bibliotecas de código y un conjunto de herramientas que proporcionan componentes que permiten el desarrollo de un producto o solución.

4.8 MVC

Model-View-Controller es un patrón de arquitectura de software que separa la representación de la información, de la interacción del usuario con él. El modelo consiste de datos de aplicación, reglas, lógica y funciones de negocio típicamente interpretadas en tres entidades: modelos, vistas y controladores. Una vista puede ser cualquier salida con la representación de los datos, como una gráfica o un diagrama. Múltiples vistas de los mismos datos son posibles, como una gráfica de barras para la administración y una vista tabular para los contadores. El controlador trabaja con la entrada, convirtiéndola en comandos para el modelo o la vista. Las ideas centrales de MVC son reusabilidad de código, y separación de los problemas.

Además de dividir la aplicación en tres tipos de componentes, el diseño MVC define la interacción entre ellos.

Un controlador puede mandar comandos a su vista asociada para cambiar la presentación de la vista de un modelo. (e.g. al utilizar la barra de desplazamiento a través de un documento) También puede mandar comandos al modelo para cambiar el estado del modelo. (e.g. ejemplo en la edición de un documento)[17].

Un modelo notifica a sus vistas y controladores asociados cuando ha habido un cambio en su estado. Esta notificación permite a las vistas ofrecer información actualizada y a los controladores cambiar el conjunto de comandos disponibles. Una implementación pasiva de MVC omite esas notificaciones, porque la aplicación no las requiere o la plataforma de software no la implementa o soporta.

Una vista solicita del modelo la información que necesita para generar una representación de los datos de salida.

---

[16] http://blogs.msdn.com/b/somasegar/archive/2012/10/01/typescript-javascript-development-at-application-scale.aspx

[17] Buschmann, Frank (1996) Pattern-Oriented Software Architecture.

Aunque originalmente fue desarrollada para la computación personal, MVC ha sido ampliamente adaptado como arquitectura para aplicaciones en la World Wide Web en los más importantes lenguajes de programación. Varios frameworks comerciales y no comerciales han sido creados para reforzar el patrón. Esos frameworks varían en sus interpretaciones, principalmente en la manera en que las responsabilidades MVC son divididas entre el cliente y el servidor[18].

4.9 FRAMEWORK ASP .NET MVC

El framework ASP MVC .NET es un framework de código abierto que implementa el patrón de diseño MVC.

Basado en ASP .NET, éste permite a los desarrolladores construir aplicaciones Web como composición de tres roles: modelo, vista y controlador[19].

5 ANÁLISIS Y DISEÑO DEL SISTEMA

5.1 REQUERIMIENTOS FUNCIONALES

Los requerimientos funcionales definen el comportamiento interno de nuestra aplicación. Establecen los comportamientos del sistema.

En esta sección se identifican los requerimientos funcionales para poder evaluar las acciones realizadas por el sistema. Estos requerimientos se describen en las tablas de la 3 a la 26.

Identificador	RF1
Nombre	Seleccionar herramienta
Descripción	Establecer qué herramienta se desea utilizar para dibujar o insertar en el área de trabajo.
Entradas	Click en herramienta específica
Origen de entradas	Mouse, touch o navegación con tecla tabulador
Salida	Confirmación de que la herramienta ha sido seleccionada
Actores	Usuario
Precondición	Ninguna
Postcondición	Existe una herramienta seleccionada
Errores	No se puede cargar la herramienta seleccionada.

Tabla 3: Requerimiento funcional RF1 - Seleccionar herramienta

[18] Leff, Avraham; James T. Rayfield (September 2001). "Web-Application Development Using the Model/View/Controller Design Pattern". IEEE Enterprise Distributed Object Computing Conference. pp. 118–127.

[19] Microsoft ASP.NET MVC home page http://www.asp.net/mvc

Identificador	RF2
Nombre	Insertar objeto en área de trabajo
Descripción	Insertar un objeto, cuyo tipo depende de la herramienta seleccionada, en el área de trabajo.
Entradas	Click en área de trabajo en la posición específica donde se desea insertar el objeto.
Origen de entradas	Mouse o touch
Salida	El objeto mostrado en el área de trabajo
Actores	Usuario
Precondición	Existe una herramienta seleccionada
Postcondición	Ninguna
Errores	No se puede insertar objeto en área de trabajo

Tabla 4: Requerimiento funcional RF2 - Insertar objeto en el área de trabajo

Identificador	RF3
Nombre	Seleccionar un objeto
Descripción	Seleccionar un solo objeto en el área de trabajo
Entradas	Click en el objeto que se desea seleccionar
Origen de entradas	Mouse o touch
Salida	El objeto muestra un marco de selección.
Actores	Usuario
Precondición	Ninguna
Postcondición	Existe una selección en el área de trabajo
Errores	No se puede seleccionar el objeto deseado

Tabla 5: Requerimiento funcional RF3 - Seleccionar un objeto

Identificador	RF4
Nombre	Seleccionar múltiples objetos
Descripción	Seleccionar múltiples objetos en el área de trabajo
Entradas	Click en área de trabajo en una posición arbitraria. Formar un rectángulo con el click, haciendo que los objetos que se desean seleccionar queden dentro del rectángulo.
Origen de entradas	Mouse o touch
Salida	Los objetos muestran un marco de selección
Actores	Usuario
Precondición	Ninguna
Postcondición	Existe una selección en el área de trabajo
Errores	No se puede seleccionar alguno de los objetos deseados

Tabla 6: Requerimiento funcional RF4 - Seleccionar múltiples objetos

Identificador	RF5

Nombre	Mostrar propiedades de un objeto
Descripción	Mostrar las propiedades editables de un objeto presente en el área de trabajo.
Entradas	Click en un solo objeto o dibujar rectángulo para seleccionar múltiples objetos.
Origen de entradas	Mouse o touch
Salida	Se muestran las propiedades del objeto en una sección específica de la interfaz gráfica, conocida como Editor de Propiedades.
Actores	Usuario
Precondición	Ninguna
Postcondición	Ninguna
Errores	El objeto no tiene propiedades editables

Tabla 7: Requerimiento funcional RF5 - Mostrar propiedades de un objeto

Identificador	RF6
Nombre	Editar propiedad de objeto
Descripción	Modificar el valor de una propiedad perteneciente a un objeto. Esta propiedad se muestra en la interfaz gráfica del editor de propiedades.
Entradas	Nuevo valor de la propiedad
Origen de entradas	Teclado físico o virtual
Salida	-La propiedad muestra el nuevo valor en el editor de propiedades -El objeto en el área de trabajo refleja el cambio realizado a la propiedad. Es decir, se actualiza para proveer una representación gráfica de su nuevo estado.
Actores	Usuario
Precondición	Existe una selección en el área de trabajo. Se muestra el editor de propiedades para cierto objeto.
Postcondición	Ninguna
Errores	No se puede modificar el valor de esta propiedad.

Tabla 8: Requerimiento funcional RF6 - Editar propiedad de un objeto

Identificador	RF7
Nombre	Agregar nueva capa
Descripción	Agregar una nueva capa por medio de la interfaz gráfica.
Entradas	Click en botón de agregar nueva capa.
Origen de entradas	Mouse o touch
Salida	La nueva capa se muestra en la lista de capas
Actores	Usuario

Precondición	Ninguna
Postcondición	Ninguna
Errores	No se puede agregar una nueva capa.

Tabla 9: Requerimiento funcional RF7 - Agregar nueva capa

Identificador	RF8
Nombre	Seleccionar un fotograma
Descripción	Seleccionar un fotograma en la línea de tiempo.
Entradas	Click en el fotograma que se desea seleccionar.
Origen de entradas	Mouse o touch
Salida	El área de trabajo se limpia y muestra los objetos pertenecientes a ese fotograma.
Actores	Usuario
Precondición	Ninguna
Postcondición	Existe un fotograma seleccionado
Errores	No se puede seleccionar el fotograma

Tabla 10: Requerimiento funcional RF8 - Seleccionar un fotograma

Identificador	RF9
Nombre	Mostrar opciones de fotograma
Descripción	Mostrar menú de opciones para un fotograma seleccionado.
Entradas	Click en el fotograma.
Origen de entradas	Mouse o touch
Salida	Se muestra un menú contextual para el fotograma seleccionado.
Actores	Usuario
Precondición	Ninguna
Postcondición	Ninguna
Errores	No se puede mostrar menú contextual.

Tabla 11: Requerimiento funcional RF9 - Mostrar opciones de fotograma

Identificador	RF10
Nombre	Insertar fotograma clave
Descripción	Insertar un fotograma clave en el fotograma actualmente seleccionado.
Entradas	Seleccionar opción de insertar fotograma clave en menú de opciones de fotograma. (RF9)
Origen de entradas	Mouse o touch
Salida	El fotograma se muestra como fotograma clave en la interfaz gráfica de la línea de tiempo.

Actores	Usuario
Precondición	-Existe un fotograma seleccionado. -Se muestra el menú de opciones de fotograma.
Postcondición	Ninguna
Errores	No se puede insertar fotograma clave en este fotograma.

Tabla 12: Requerimientos funcionales RF10 - Insertar fotograma clave

Identificador	RF11
Nombre	Eliminar fotograma clave
Descripción	Eliminar fotograma clave del fotograma actualmente seleccionado
Entradas	Seleccionar opción de eliminar fotograma clave en menú de opciones de fotograma.
Origen de entradas	Mouse o touch
Salida	El fotograma se muestra como fotograma normal en la interfaz gráfica de la línea de tiempo.
Actores	Usuario
Precondición	-Existe un fotograma seleccionado. -El fotograma seleccionado es fotograma clave. -Se muestra el menú de opciones de fotograma.
Postcondición	Ninguna
Errores	No se puede eliminar el fotograma clave del fotograma seleccionado.

Tabla 13: Requerimientos funcionales RF11 - Eliminar fotograma clave

Identificador	RF12
Nombre	Vaciar fotograma
Descripción	Eliminar todos los objetos pertenecientes al fotograma actualmente seleccionado.
Entradas	Seleccionar opción de vaciar fotograma en menú de opciones de fotograma.
Origen de entradas	Mouse o touch
Salida	El área de trabajo se vacía.
Actores	Usuario
Precondición	-Existe un fotograma seleccionado. -Se muestra el menú de opciones de fotograma.
Postcondición	Ninguna
Errores	No se puede vaciar el fotograma seleccionado.

Tabla 14: Requerimientos funcionales RF12 - Vaciar fotograma

Identificador	RF13
Nombre	Animar a fotograma
Descripción	Generar fotogramas intermedios entre el último

	fotograma clave y el fotograma seleccionado.
Entradas	Seleccionar opción de animar a fotograma clave en menú de opciones de fotograma al que se desea animar.
Origen de entradas	Mouse o touch
Salida	Se muestra flecha de animación en fotogramas intermedios en la interfaz gráfica de la línea de tiempo.
Actores	Usuario
Precondición	-Existe un fotograma seleccionado. -Existe un fotograma clave antes del fotograma seleccionado. -Se muestra el menú de opciones de fotograma.
Postcondición	Ninguna
Errores	No se puede animar al fotograma seleccionado.

Tabla 15: Requerimientos funcionales RF13 - Animar a fotograma

Identificador	RF14
Nombre	Mover objeto
Descripción	Mover de posición un objeto actualmente presente en el área de trabajo.
Entradas	Seleccionar el objeto que se desea mover sin soltar el botón del mouse o touchscreen. Arrastrar el objeto a la posición a la que se desea mover. Soltar el mouse o touchscreen.
Origen de entradas	Mouse o touch
Salida	El objeto se encuentra ahora en la posición deseada.
Actores	Usuario
Precondición	Existe un objeto en el área de trabajo.
Postcondición	Ninguna
Errores	No se puede mover el objeto seleccionado.

Tabla 16: Requerimiento funcional RF14 - Mover objeto

Identificador	RF15
Nombre	Seleccionar capa
Descripción	Seleccionar una capa previamente creada por medio de la interfaz gráfica de la línea de tiempo.
Entradas	Click en capa que se desea seleccionar.
Origen de entradas	Mouse o touch
Salida	La capa seleccionada se muestra resaltada en la interfaz gráfica de la línea de tiempo.
Actores	Usuario
Precondición	-Existe una capa que seleccionar

Postcondición	-Existe una capa seleccionada
Errores	No se puede seleccionar la capa.

Tabla 17: Requerimiento funcional RF15 - Seleccionar capa

Identificador	RF16
Nombre	Reproducir vista previa de animación
Descripción	Reproducir una vista previa de la animación a generar.
Entradas	Click en botón de vista previa.
Origen de entradas	Mouse o touch
Salida	Se muestra la animación en el área de trabajo. Se bloquea el área de trabajo para que el usuario no pueda insertar objetos mientras se reproduce la animación.
Actores	Usuario
Precondición	Ninguna
Postcondición	-Existe una animación reproduciéndose en el área de trabajo
Errores	No se puede reproducir la animación.

Tabla 18: Requerimiento funcional RF16 – Reproducir vista previa de animación

Identificador	RF17
Nombre	Detener vista previa de la animación
Descripción	Detener una animación que actualmente se esté visualizando en el área de trabajo.
Entradas	Click en botón de detener vista previa.
Origen de entradas	Mouse o touch
Salida	Se deja de mostrar la animación en el área de trabajo. Se desbloquea el área de trabajo.
Actores	Usuario
Precondición	-Existe una animación reproduciéndose en el área de trabajo.
Postcondición	-Ya no existe una animación reproduciéndose en el área de trabajo.
Errores	No se puede detener la animación.

Tabla 19: Requerimiento funcional RF17 - Detener vista previa de la animación

Identificador	RF18
Nombre	Guardar
Descripción	Sincronizar el documento actual con el servidor.
Entradas	Click en botón de guardar.
Origen de entradas	Mouse o touch
Salida	Se muestra confirmación de que se ha guardado

	satisfactoriamente.
Actores	Usuario
Precondición	-Existen cambios no guardados
Postcondición	Ninguna
Errores	No se puede contactar al servidor.

Tabla 20: Requerimiento funcional RF18 - Guardar

Identificador	RF18
Nombre	Autoguardar
Descripción	Automáticamente sincronizar el documento actual con el servidor.
Entradas	Documento actual con cambios.
Origen de entradas	Mouse o touch
Salida	Se actualiza la barra de status con la última fecha en que se autoguardó.
Actores	Ninguno
Precondición	-Existen cambios no guardados -Ha transcurrido el tiempo necesario para autoguardar
Postcondición	Ninguna
Errores	No se puede contactar al servidor.

Tabla 21: Requerimiento funcional RF19 - Autoguardar

5.2 REQUERIMIENTOS NO FUNCIONALES

Son aquellos requerimientos en los que se especifican criterios que podemos usar para juzgar la operación de nuestra aplicación en lugar de sus comportamientos específicos. No describen información a guardar ni funciones a realizar.

Identificador	RNF1
Nombre	Tiempo de carga
Descripción	La aplicación debe inicializarse en un tiempo menor a 2 segundos. Este tiempo mide sólo inicialización, es evidente que el tiempo de carga varía dependiendo de la conexión a Internet del usuario.

Tabla 22: Requerimiento no funcional RNF1 - Tiempo de carga

Identificador	RNF2
Nombre	Estabilidad
Descripción	La aplicación no debe hacer que el navegador se vuelva no responsivo en ninguno de los casos de uso descritos.

Tabla 23: Requerimiento no funcional RFN2 - Estabilidad

Identificador	RNF3
Nombre	Congruencia de datos locales
Descripción	La aplicación no debe corromper los datos del usuario en ningún momento.

Tabla 24: Requerimiento no funcional RFN3 - Congruencia de datos locales

Identificador	RNF4
Nombre	Congruencia entre datos locales y remotos
Descripción	La aplicación debe asegurar que los datos generados remotamente (código resultante) correspondan con los datos que ha ingresado el usuario.

Tabla 25: Requerimiento no funcional RNF4 - Congruencia entre datos locales y remotos

Identificador	RNF5
Nombre	Seguridad
Descripción	La aplicación debe proveer un mecanismo de autentificación para que los usuarios inicien sesión. Este mecanismo debe asociar los archivos de proyecto sólo con el usuario que los creó para garantizar seguridad.

Tabla 26: Requerimiento no funcional RNF5 - Seguridad

5.3 DIAGRAMAS DE CASOS DE USO

En esta sección se presentan los diferentes diagramas de casos de uso que implementan los requerimientos establecidos en la sección anterior.

El siguiente diagrama muestra la funcionalidad básica para crear proyectos y administrarlos, por ello, incluye la descripción de la funcionalidad de la administración de páginas.

Figura 1: Diagrama de caso de uso de administración de proyectos

El siguiente diagrama muestra la interacción entre el usuario y el sistema para administrar la cuenta de cada usuario.

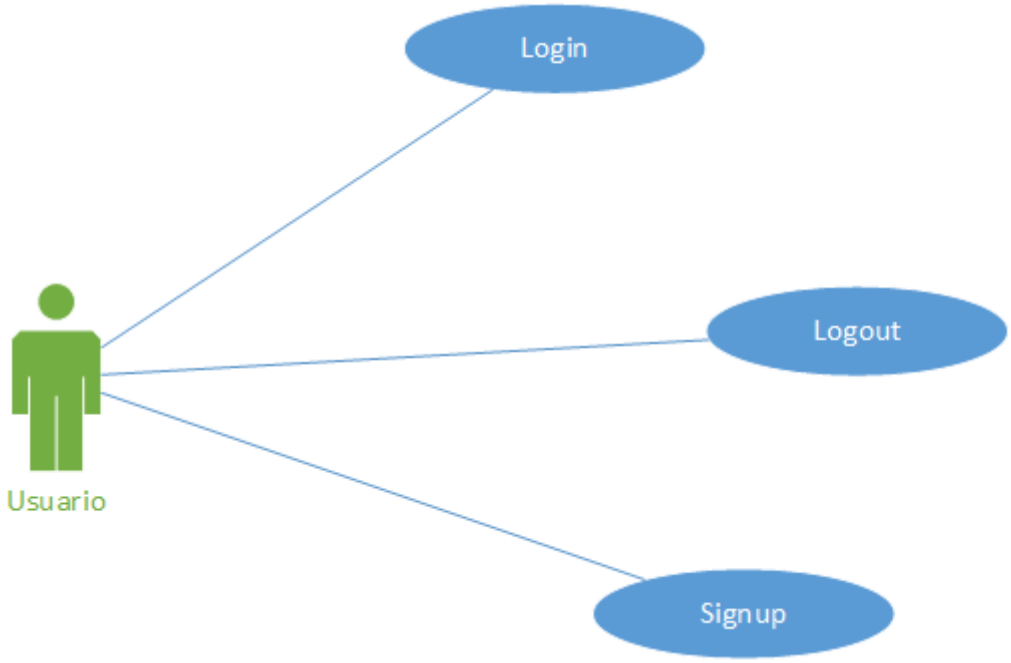


Figura 2: Diagrama de caso de uso de administración de cuenta de usuario

El siguiente diagrama muestra la interacción entre el usuario y el sistema para crear animaciones. Muestra los procesos de creación de fotogramas, capas y animaciones.

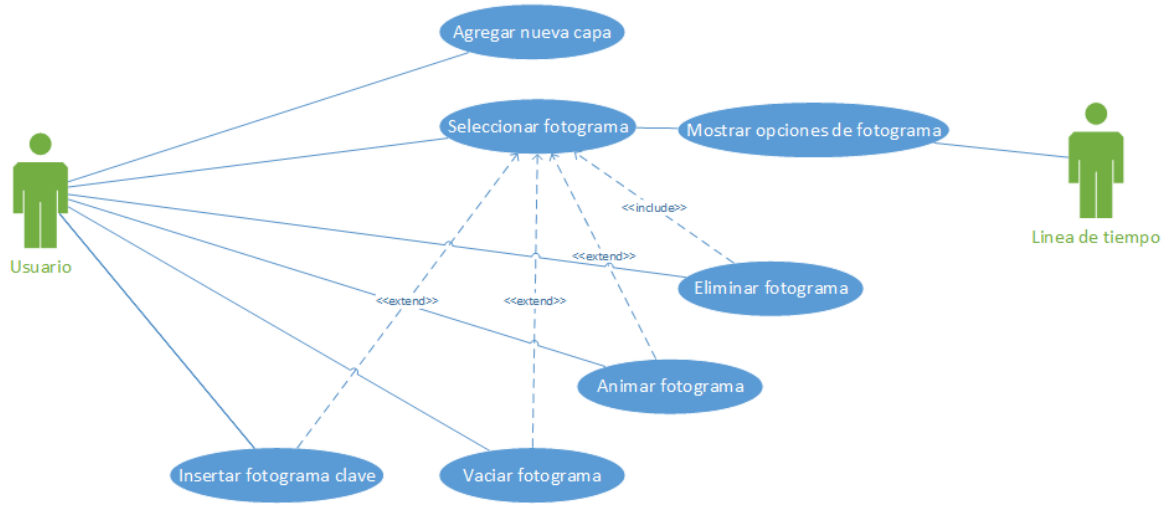


Figura 3: Diagrama de caso de uso de creación de animaciones

El siguiente diagrama muestra los procesos necesarios para insertar elementos en el canvas y editar sus propiedades que soporta

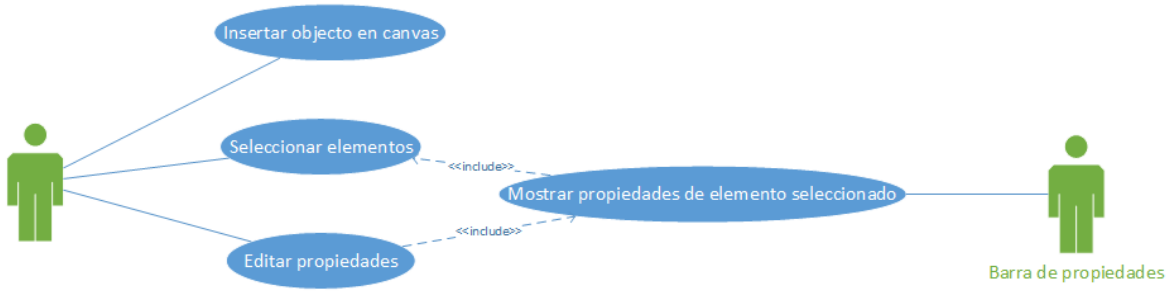


Figura 4: Diagrama de caso de uso de edición de propiedades

El siguiente diagrama muestra los procesos para poder insertar objetos en el canvas, seleccionarlos y moverlos dentro del canvas.

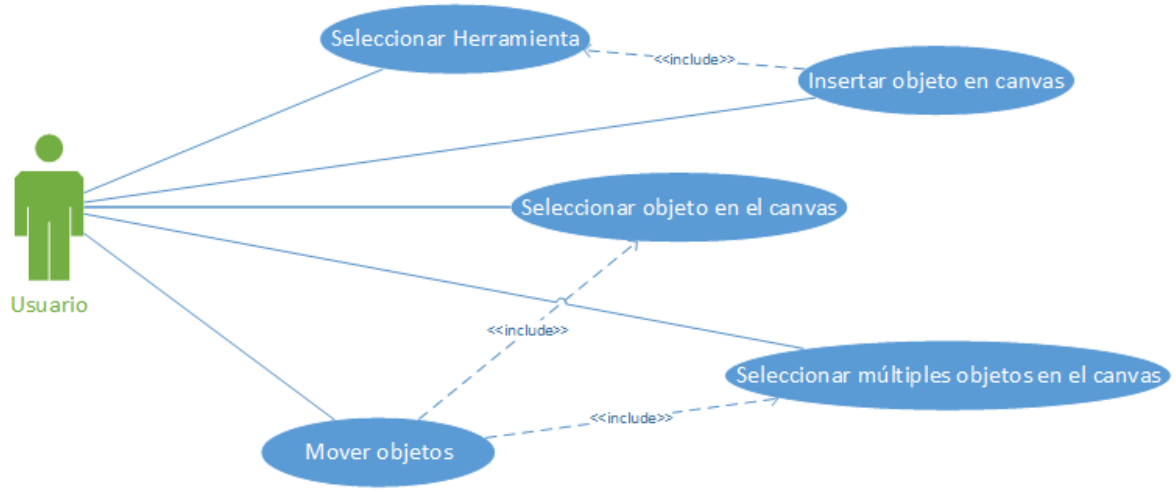


Figura 5: Diagrama de caso de uso de inserción de elementos en el canvas

El siguiente diagrama muestra los procesos para visualizar una animación dentro de la aplicación.

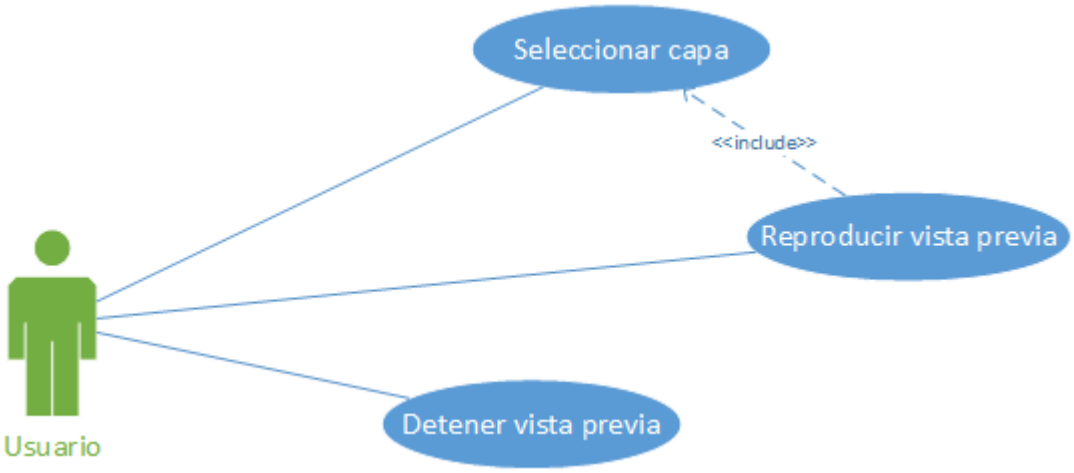


Figura 6: Diagrama de caso de uso de previsualización de animación

5.4 ESCENARIOS DE USO

La aplicación se enfoca en proyectos casuales o semi-profesionales, debido a que el tipo de usuarios que desarrollan este tipo de proyectos tienden a no tener experiencia utilizando herramientas profesionales, que son más complicadas y tienen una curva de aprendizaje mayor. Además, este tipo de usuarios no tienen conocimientos de programación, así que la aplicación les da acceso a tecnologías que normalmente no podrían utilizar, ya que tendrían que escribir el código ellos mismos.

5.4.1 Escenario de Uso: Marco el diseñador

Marco es un diseñador al cual le han dado la tarea de diseñar la página de una empresa encargada de vender balones: Él tiene la idea de hacer una página principal donde salgan algunos balones botando. Lo que él hace para lograrlo es:

1. Marco crea un nuevo documento en su proyecto
2. Marco genera el contenido no animado (texto e imágenes).
3. Agrega las nuevas imágenes de los balones
4. Selecciona un balón y lo coloca en la posición y con el tamaño que él desea
5. Marco crea un nuevo Keyframe, modifica el tamaño y posición del balón
6. Marco repite el paso 7 para tantos Keyframes la animación que tiene en mente necesita.
7. Marco repite los pasos 4-6 para cada uno de los balones que ha agregado
8. Marco prueba su página y hace los ajustes a las animaciones que considere necesarios.

5.4.2 Escenario de Uso: Gabriel el estudiante

Gabriel está estudiando Comunicación, por lo que para practicar lo que ha aprendido quiere hacer su propia página personal, pero no tiene mucha experiencia en páginas Web. Lo que él hace para empezar su página Web es:

1. Gabriel crea un nuevo Proyecto.
2. Gabriel agrega un nuevo documento a su proyecto.
3. Gabriel genera bocetos a mano de lo que quiere mostrar en su página.
4. Gabriel arrastra los controles de la barra de herramientas que él cree podrían serle de ayuda para crear un diseño similar al que hay en sus bocetos a mano. Toma la decisión de esto, basado en las descripciones de los controles.
5. Gabriel modifica algunas de las propiedades de los elementos, con la finalidad de obtener un resultado similar al de su boceto.
6. Finalmente él queda contento con su resultado, pero le gustaría agregar más contenido después, por lo que guarda su documento y proyecto para abrirlos después.

5.4.3 Límites de la aplicación

El manejo de propiedades y atributos de las etiquetas HTML y de los estilos CSS, está limitado a los que están listados posteriormente en este mismo documento. Por lo que puede que sea necesario tener que editar el código de manera manual para lograr ciertos resultados.

Las animaciones son basadas en keyframes, y son activadas según sucedan ciertos eventos (carga de página, respuesta a petición AJAX, etc.), por lo que es posible que algunas animaciones necesiten de añadir código en JavaScript para lograr ciertos comportamientos.

Debido a la lenta transición que se está teniendo hacia HTML5 y CSS3, puede que algunas animaciones o layouts creados con la aplicación se muestren diferente en diferentes navegadores. La aplicación tratará de hacer código a la medida para cada uno de los navegadores más usados para mantener las cosas tan fieles al original como sea posible, pero no se garantiza que la fidelidad se pueda conseguir en todas la funcionalidades.

5.4.4 Ejemplos reales de resultados que podrían ser obtenidos por la aplicación

5.4.4.1 Animación de hombre caminando

La página muestra un hombre caminando, totalmente animado con CSS3. Este tipo de animaciones podrán ser hechas en su totalidad por la aplicación.

bit.ly/WalkingCSS3

5.4.4.2 Animación de elementos ordenándose en una tabla periódica

La animación y funcionalidad está realizada con CSS3D, lo cual queda fuera de los alcances de esta aplicación, pero se podría generar una animación similar a la inicial (sin 3D) sin problemas en la aplicación.

bit.ly/PeriodicTableCSS3

5.4.4.3 Animación de lluvia de caracteres

Este es un ejemplo de animación que podría ser conseguida con la aplicación con necesidad de un poco de código JavaScript extra, para darle funcionalidad y aleatoriedad.

bit.ly/TypographicRainCSS3

5.4.4.4 Animación de cohete espacial

Este ejemplo de animación hecha en CSS3 que podría ser considerada como objetivo para nuestra aplicación.

bit.ly/RocketCSS3h

5.4.4.5 Animación de máquina caminando

Este es un ejemplo de animación creada completamente con CSS 3

bit.ly/WalkingRobotCSS3

5.4.4.6 Animaciones de diferentes clases

En este ejemplo, se tienen muchos tipos de animaciones, en este todas se tratan de gifs animados, en nuestro proyecto se debería de poder tener resultados similares sin necesidad de gifs.

bit.ly/SoleilNoirAnimation

5.5 DISEÑO PROPUESTO

5.5.1 Definiciones

A continuación se presentan las definiciones de diferentes conceptos que forman parte de la aplicación.

Término	Descripción
Página	Un archivo que representa a una página Web. Puede ser editado con la aplicación. Se almacena en el servidor.
Proyecto	Un conjunto de páginas que pertenecen a un usuario que puede editarlos por medio de la aplicación.
Página actual	El página que actualmente está siendo editado con la aplicación.
Código resultante	El código que la aplicación genera a partir de las interacciones del usuario con la interfaz gráfica.
Canvas	Región principal de la interfaz gráfica. Muestra el estado actual de la página. Contiene elementos que conforman la página. El usuario interactúa con el canvas para editar el documento.
Barra de herramientas	Es componente de la interfaz gráfica que permite al usuario seleccionar una herramienta que insertará en la página
Herramienta	Es un componente de la aplicación. Se muestra en la barra de herramientas y pueden ser insertados en el canvas.
Elemento	Es un objeto de tipo específico que es parte de la página. Este elemento también tiene una representación de código en el código resultante. Se inserta por medio de la barra de herramientas y el canvas.
Propiedad	Es una característica asociada con cierto tipo de elementos. Por ejemplo, un div puede tener color de fondo.
Atributo	Es una propiedad a la que se le ha asignado un valor
Línea de tiempo	Es un componente de la interfaz gráfica que el usuario utiliza para modificar los fotogramas que conforman su animación.
Posición en la línea de tiempo	Es un estado de la página en un tiempo específico.
Punto clave/KeyFrame	Es un punto de tiempo en el que el valor de los atributos de un elemento cambia, generando

	una transición.
Transición	Es un cambio en los atributos de uno o más elementos.
Layout	La forma en que las partes de algo están dispuestos o la disposición.

Tabla 27: Glosario de términos generales

5.5.1.1 PRESENTACIÓN GENERAL

La aplicación está compuesta por diferentes elementos de la interfaz gráfica con los que el usuario interactúa. Se presentan a continuación.

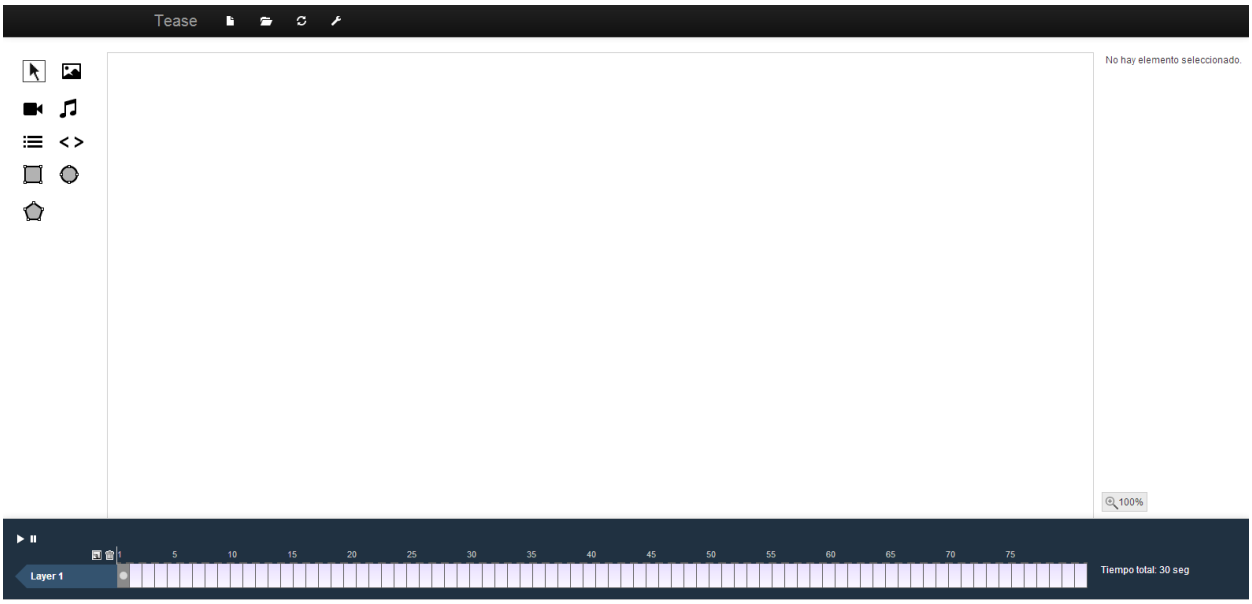


Figura 7: Elementos de la interfaz gráfica

Elemento de la interfaz gráfica	**Descripción**
Canvas	Es el principal medio de interacción del usuario. Muestra una vista previa del sitio en un punto de tiempo especificado por la línea de tiempo. El usuario lo utiliza frecuentemente para insertar, mover, transformar o seleccionar elementos gráficos que forman parte de su sitio.
Barra de comandos básicos	Contiene los comandos más generales de la aplicación, como Abrir, Guardar.
Línea de tiempo	Proporciona una escala que representa tiempo en la cual se insertan fotogramas clave en tiempos específicos. El usuario puede transportarse a cualquier fotograma en un tiempo dado. También proporciona la interfaz para manejar capas.
Editor de propiedades	Proporciona un diálogo por medio del cual el

	usuario puede editar las propiedades del elemento que actualmente tiene seleccionado en el canvas.
Capas	El usuario puede agregar capas a su sitio para agrupar elementos lógicamente.

Tabla 28: Descripción de elementos de la interfaz gráfica

En términos generales, la barra de herramientas contiene una lista de herramientas que el usuario puede utilizar para dibujar en el canvas. La línea de tiempo sirve para tener una idea de animaciones con respecto de su duración y orden. El editor de propiedades es el encargado de proporcionar al usuario las herramientas necesarias para editar las propiedades fácilmente de los elementos insertados en el canvas.

El canvas es el elemento de visualización del trabajo realizado. Es el área de trabajo en la cual se reflejan todos los cambios y alteraciones al diseño creado con la aplicación.

Es necesario implementar un módulo de generación de código que permita traducir todos los cambios efectuados con la interfaz gráfica a un producto final que pueda ser utilizado en cualquier aplicación Web. Dicho generador de código estará actualizando el estado del diseño creado y guardando copias en un servidor Web que servirá para la persistencia de los datos y el diseño creado.

5.5.1.2 Descripción de características

5.5.1.2.1 Canvas

El canvas es el área donde el usuario podrá visualizar y editar el diseño de su animación. En ella se podrán insertar y eliminar las herramientas que estén presentes en la barra de herramientas. El canvas soporta la selección de elementos para poder manipular sus propiedades mediante el editor de propiedades. Es posible también editar propiedades como tamaño, posición y rotación mediante las herramientas de edición de elementos que se construirán. Esto permite que el usuario edite dichas propiedades de manera más rápida sin necesidad de interactuar con los valores mostrados en el editor de propiedades o con el código generado. El canvas soporta la selección de múltiples elementos para poder cambiar su ubicación sin necesidad de editar la posición de cada uno de los elementos seleccionados. En el canvas es posible ordenar los elementos en relación a su profundidad, es decir, el canvas proporciona la herramienta para poner un elemento enfrente de otro, ubicarlo hasta atrás o ubicarlo como el elemento que está encima de todos los demás elementos.

El canvas es un área a la que se le pueden editar sus dimensiones. Dichas dimensiones pueden ser editadas introduciendo los valores correspondientes en el editor de propiedades.

5.5.1.2.2 Barra de comandos básicos

La barra de comandos básicos provee un conjunto de acciones que el usuario puede realizar al documento actual.

Comando	**Descripción**
Nuevo	Crea un nuevo documento en el servidor.
Abrir	Muestra la interfaz para abrir un documento que se encuentra en el servidor

Guardar	Guarda el documento actual en el servidor
Opciones	Muestra un diálogo de opciones generales para el usuario
Cerrar sesión	Cierra la sesión actual de la aplicación

Tabla 29: Comandos básicos

5.5.1.2.3 Opciones por elemento

También será posible acceder a opciones específicas de cada elemento seleccionado o que se esté editando. Dichas opciones serán agregadas conforme lo permita el desarrollo del proyecto y los tiempos propuestos.

5.5.1.2.4 Barra de herramientas

En la barra de herramientas se ubican las herramientas que el usuario puede insertar en el canvas. Las herramientas mostradas en el canvas pueden ser seleccionadas e insertadas en el canvas.

5.5.1.2.5 Herramientas

Las herramientas son elementos que pueden ser agregados, editados y eliminados del área del canvas. Las herramientas están divididas en 4 grandes grupos:

- Herramientas de selección
- Herramientas de listas
- Herramientas de medios
- Herramientas geométricas

Las herramientas han sido clasificadas de esa manera para ayudar al usuario a encontrarlas de manera más sencilla y rápida. Las propiedades de las herramientas se modificarán a través del editor de propiedades. Las herramientas serán construidas de manera que soporten un conjunto de propiedades que puedan ser modificadas por el usuario sin necesidad de conocer su representación HTML o CSS.

5.5.1.2.6 Línea de tiempo

La línea de tiempo es el área en donde se muestra el comportamiento de las herramientas en relación con el tiempo. Es decir, en la línea de tiempo es posible observar cómo cambian las propiedades de un elemento.

Definimos una posición en el tiempo como un número natural comenzando en 1 que representa un momento en el tiempo. Estas posiciones se muestran en la parte superior de la línea de tiempo.

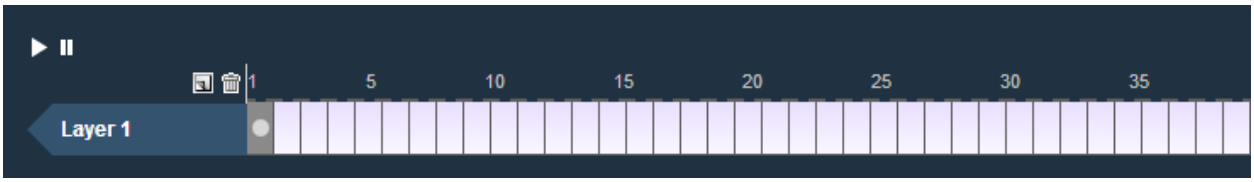


Figura 8: Línea de tiempo

En la figura 8 se muestra la línea de tiempo. Las posiciones de tiempo se muestran arriba de los fotogramas, empezando con 1.

5.5.1.2.7 Animación

Por medio de la línea de tiempo se pueden agregar animaciones a la página. Estas animaciones se logran por medio de transiciones, las cuales definimos como cambios de los valores en atributos de uno o más elementos.

En la línea de tiempo es posible especificar una transición al insertar fotogramas clave, que son los momentos en el tiempo en que la herramienta ha terminado de cambiar sus propiedades o se ha transformado.

En la línea de tiempo se podrá visualizar el tiempo total de las animaciones generadas, así como los fotogramas que contienen transiciones.
El usuario genera animaciones de la siguiente forma.

1. Inserta un fotograma clave o utiliza el primero que se genera automáticamente
2. Inserta elementos por medio de las barras de herramientas y el canvas
3. Posiciona los elementos con el canvas dependiendo de dónde los quiera inicialmente para la animación
4. Selecciona un fotograma en el futuro haciendo click en él.
5. Selecciona la opción Animar en el menú de opciones del fotograma
6. Modifica las propiedades de los elementos dependiendo de cómo las quiera para la animación

Figura 9: Composición gráfico de la línea de tiempo

La aplicación se encarga de generar todos los fotogramas intermedios entre el fotograma clave inicial y el fotograma en el que el usuario seleccionó la animación.

5.5.1.2.8 Editor de propiedades

El editor de propiedades es el área donde el usuario puede visualizar las propiedades que cada una de las herramientas soporta. Al ser seleccionado un elemento en el canvas, el editor de propiedades la identifica y muestra al usuario qué propiedades presenta y los valores de las mismas. Así mismo, el usuario puede editar los valores de las propiedades y visualizar instantáneamente en el canvas el efecto que tiene su modificación.

5.5.1.2.9 Persistencia

La aplicación es capaz de mantener un registro seguro del contenido generado en la aplicación. Los cambios realizados cada cierto periodo de tiempo son guardados en un servidor Web para proteger el trabajo realizado y visualizarlo en posteriores ocasiones. La aplicación cuenta para ello con interacción con el servidor y se utiliza el patrón de desarrollo descrito en "Backend" para la creación de esta característica.

Para hacer uso de la aplicación el usuario deberá estar registrado en el servidor Web. Los documentos se almacenan en el servidor Web y se asocian a un usuario.

La aplicación debe guardar automáticamente el documento actual cada 10 minutos.

5.5.1.2.10 Generación de código

La aplicación es capaz de generar código que puede ser ejecutado en los navegadores soportados. El código generado representa el producto final de la aplicación ya que en ella estarán contenidos los elementos y animaciones que el usuario haya especificado.

Cabe mencionar que es posible no poder visualizar diferentes efectos y animaciones dependiendo del navegador que se esté utilizando, sin embargo, si este es el caso, el motivo será la falta de soporte del navegador y no la incapacidad de la aplicación de generar contenido específico de acuerdo a los estándares en desarrollo.

5.5.1.2.11 Herramientas

5.5.1.2.11.1 Herramientas de selección

Herramienta	**Descripción**
Selección	Permite seleccionar uno o múltiples objetos en el canvas, así como moverlos alrededor del mismo.

Tabla 30: Herramientas de selección

5.5.1.2.11.2 Herramientas de listas

Herramienta	**Descripción**
Tabla	Inserta una tabla al canvas
Lísta ordenada	Inserta una lista ol al canvas
Lista no ordenada	Inserta una lista ul al canvas

Tabla 31: Herramientas de listas

5.5.1.2.11.3 Herramientas de medios

Herramienta	**Descripción**
Imagen	Inserta una imagen al canvas. El formato de las imágenes son de tipo jpg y png.
Video	Inserta un video al canvas
Canvas	Inserta una etiqueta canvas al canvas
Audio	Inserta audio al canvas

Tabla 32: Herramientas de medios

6 METODOLOGÍA

Scrum es una metodología de desarrollo de software ágil que nos brinda un marco de proceso para la creación de su propio proceso ágil, cumpliendo con las metas de nuestro proyecto. A continuación se presentan algunos aspectos por los que se eligió SCRUM.

6.1 GESTIÓN DEL RIESGO Y EL CAMBIO CON EFICACIA

Mediante el uso de pequeños pasos y retroalimentación rápida (pruebas, clientes) errores de malentendidos son rápidamente abordados.

Centrándonos en el aspecto más valioso y arriesgado mayor parte del proyecto desde el principio reducir el costo del fracaso y por lo tanto dar una mayor comprensión de los riesgos del proyecto.

Dejando opciones abiertas hasta el último momento reduce el riesgo de trabajo perdido y ayuda a facilitar el cambio en línea con el objetivo del proyecto. La arquitectura evoluciona en lugar de ser fija.

6.2 LIBERACIÓN MÁS RÁPIDA

Por la liberación de la funcionalidad temprano y con frecuencia, la retroalimentación es mucho mayor y la comprensión real de lo que se necesita es más rápido para llegar.

Cuando el trabajo se realiza en piezas de auto pequeños contenidos que pueden ser creados y entregados rápidamente.

6.3 MEJORA DE LA CALIDAD

Los defectos se descubren y se abordan temprano al incluir aspectos de pruebas y el solo el producto necesario para el cliente es desarrollado.

6.4 MEJORA DE LA SATISFACCIÓN DE LOS INTERESADOS

Las partes interesadas (clientes, testers, desarrolladores, BA, etc) tienen una mayor participación e influencia en el desarrollo de productos y por lo tanto tienen más afinidad con el proyecto.

6.5 UN MAYOR COMPROMISO DE LOS MIEMBROS DEL EQUIPO

Participa en el proceso de toma de decisiones y actividades que les dan una mayor comprensión del valor de su trabajo.

6.6 UNA MAYOR PRODUCTIVIDAD Y MENORES COSTOS

Sólo la construcción de lo que necesita de inmediato reduce la pérdida de desarrollar cosas que no se desean.

Con pequeñas porciones de código deben tener en promedio menos errores, por lo que escribir menos código es más productivo.

6.7 ESPECIFICACIONES PROPUESTAS PARA LA METODOLOGÍA

Dadas las necesidades de nuestro proyecto hemos definido las siguientes especificaciones:

- La duración de cada Sprint será de 30 días
- Se realizaran reuniones diarias
- Se realizaran reuniones de Revisión de resultados del Sprint
- Se realizaran reuniones de Restrospectiva

7 DESARROLLO

A continuación se detalla el plan de desarrollo para la aplicación.

7.1 ARQUITECTURA GENERAL

La aplicación consta de dos partes principales: el cliente y el servidor. El cliente se ejecuta como JavaScript/HTML/CSS en el navegador Web del usuario. El servidor almacena el contenido creado por el usuario, así como su información de sesión. El cliente se comunica con el servidor por medio de peticiones HTTP asíncronas, comúnmente llamado AJAX.

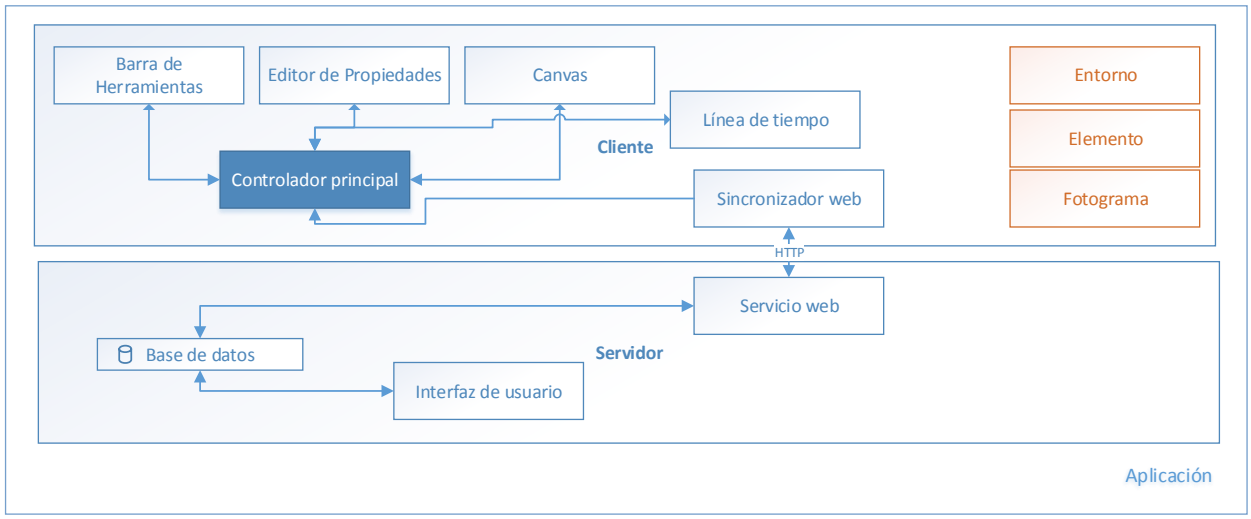


Figura 10: Arquitectura general

Por el lado del cliente, la aplicación se compone de diversos controladores, representados con borde azul en el diagrama anterior, que se encargan de recibir eventos de la interfaz gráfica. Dichos controladores trabajan de manera independiente y se comunican por medio de un controlador principal. El controlador principal se suscribe a eventos provistos por los demás controladores y actualiza el estado interno del sistema, de esta forma se evita la comunicación directa entre controladores, evitando la dependencia entre ellos y facilitando su integración.

También se cuenta con bastantes modelos en el lado del cliente, se muestran algunos con borde naranja en el diagrama anterior. Estos representan diferentes componentes abstractos en la aplicación, tales como el entorno general de la aplicación, los elementos dentro del canvas y los fotogramas que componen la línea de tiempo. Estas clases son utilizadas por todos los controladores.

En la sección de diseño detallado se profundiza en la implementación de todos estos componentes y las relaciones que existen entre ellos.

7.2 JUSTIFICACIÓN

La aplicación se ejecutará en navegadores Web, por lo que es necesario establecer las tecnologías a utilizar tanto en el cliente como en el servidor.

7.2.1 Tecnologías utilizadas en cliente

El cliente deberá estar construido en JavaScript, HTML y CSS para que pueda ejecutarse en todos los navegadores Web soportados.

Considerando que nuestra aplicación contiene bastantes componentes y puede modelarse usando un paradigma orientado a objetos, se decidió utilizar TypeScript.

TypeScript es un lenguaje *superset* de JavaScript que añade soporte apropiado de orientación a objetos, así como tipos estáticos. Estas características facilitan el desarrollo de aplicaciones complejas, ya que JavaScript por sí mismo no soporta tipos estrictos y su orientación a objetos es limitada debido a que es prototipal.

Al ser un superset, también soporta todas las librerías para JavaScript actualmente disponibles, como jQuery. El desarrollo de aplicaciones en TypeScript se realiza por medio de un compilador, integrado con Visual Studio, que se encarga de traducir el código de TypeScript a JavaScript para poder ejecutarse en cualquier navegador Web.

7.2.1.1 Lenguajes considerados

A continuación se muestra una tabla con los lenguajes de programación considerados para el cliente.

Lenguaje	**Desarrollador**	**Tipado**	**Compatibilidad con librerías existentes**	**Licencia**
JavaScript	ISO	Dinámico débil	Sí	Standard abierto
TypeScript	Microsoft	Estático fuerte	Sí, al ser superset de JavaScript	Apache
Dart	Google	Opcional	No	BSD
CoffeeScript	Jeremy Ashkenas	Dinámico débil	No	MIT

Tabla 33: Lenguajes considerados

Se llegó a la conclusión de utilizar TypeScript, debido a que provee un ambiente estáticamente tipado, lo cual facilita la implementación de nuestro diseño. También tiene la ventaja de que soporta todas las librerías existentes de JavaScript.

7.2.2 Librerías y frameworks

La aplicación utiliza las siguientes librerías y frameworks.

Librería	**Descripción**	**Justificación**
Bootstrap	Es un framework desarrollado por Twitter que facilita el desarrollo de hojas de estilo CSS.	El diseño multi-columna de nuestra aplicación puede ser fácilmente implementado en bootstrap. Minimizando el tiempo de desarrollo y facilitando su manutención.
JQuery	Es una librería de JavaScript que facilita la manipulación de DOM,	Debido a que nuestra aplicación debe manipular

	peticiones HTTP remotas (AJAX) y animaciones.	el DOM constantemente para mostrar los elementos en el canvas es preferible contar con una herramienta para facilitarlo. También provee una forma de realizar animaciones en navegadores que no soportan CSS 3.

Tabla 34: Librerías utilizadas

7.2.3 Servidor

Para el lado del servidor se utiliza ASP.NET utilizando C# y MVC 4. Esto debido a la familiaridad de los desarrolladores con dichas tecnologías, así como el alto nivel de madurez y soporte de éstas.

7.3 DISEÑO DETALLADO DEL CLIENTE

7.3.1 Tipos de archivo

En esta sección se hace referencia a archivos .ts. Estos archivos representan código en TypeScript. El lector puede asumir que a cada archivo .ts corresponde un archivo de JavaScript .js que representa el mismo código pero compilado. Para evitar confusiones nos referiremos a todos los archivos con la extensión .ts, pero en realidad nos estamos refiriendo tanto al código fuente en TypeScript, como al código generado por el compilador.

7.3.2 Arquitectura MVC

Adoptamos una arquitectura MVC (Model View Controller) para el cliente. Debido al costo de rendimiento que tendría dividir el cliente en múltiples vistas y cargarlas asíncronamente, se tiene una sola vista, desde la cual se carga el controlador principal, que se encarga de instanciar los demás controladores y comunicarlos entre sí.

7.3.3 Eventos

Otro de los patrones de diseño utilizados alrededor del sistema es el uso de eventos. La manera básica en que los componentes del sistema interactúan es por medio de eventos.

Un controlador principal registra diferentes eventos y realiza las llamadas correspondientes a los demás controladores. Así podemos adoptar un entorno orientado a eventos sin sacrificar los beneficios del patrón MVC.

Todos los eventos pasan información a sus manejadores por medio del atributo `detail` del objeto `Event` que regresan. Esta es una práctica standard en JavaScript.

Figura 11: Diagrama de clase Eventable

La clase `Eventable` provee los métodos necesarios para agregar y quitar eventos. Los siguientes controladores extienden esta clase para tener soporte de eventos

7.3.4 Inicialización

Cuando un usuario abre la aplicación desde su navegador Web se carga el archivo index.html. Este archivo incluye todos los archivos de JavaScript necesarios para correr la aplicación. Entre ellos se encuentra el archivo Main.ts. En este archivo se define e instancia el controlador principal (`MainController`).

Durante la inicialización del controlador principal se instancian los siguientes controladores: `Toolbar`, `Canvas`, `PropertyEditor`, `Timeline`. También se añaden manejadores de eventos para los eventos provistos por estos controladores.

Estos eventos son los siguientes.

Controlador	Evento	Descripción	Detail
Canvas	`canvasinsert`	Este evento ocurre inmediatamente después de que algo se inserta al canvas.	Un objeto `Element` que representa el elemento que se acaba de insertar en el canvas
Canvas	`canvasselect`	Ocurre cuando el usuario selecciona un elemento en el canvas.	Un objeto `Element` que representa el elemento que se seleccionó.
Toolbar	`toolselect`	Ocurre cuando el usuario selecciona una herramienta en la barra de herramientas.	Un objeto `Tool` que representa la herramienta que el usuario seleccionó
Timeline	`layerselect`	Ocurre cuando el usuario cambia de capa mediante la línea de tiempo.	Un `number` que corresponde al índice de la capa seleccionada por el usuario. Iniciando en 0.

Timeline	`playbuttonclick`	Ocurre cuando el usuario hace click en el botón de reproducir vista previa en la línea de tiempo.	-
Timeline	`frameselect`	Ocurre cuando el usuario hace click en un fotograma en la línea de tiempo.	Un `number` que representa la posición en el tiempo del frame que seleccionó el usuario. Iniciando en 1.

Tabla 35: Eventos de inicialización

De esta manera, el controlador principal implementa los manejadores para los eventos lanzados por los demás controladores. Esto permite, por ejemplo, redibujar el canvas de acuerdo con el fotograma que el usuario ha seleccionado, proporcionando una forma de comunicación indirecta y concisa entre los demás controladores y manteniendo su independencia y modularidad.

7.3.5 Controlador principal

Es el primer controlador que se ejecuta. Se encarga de inicializar la aplicación, así como instanciar los componentes necesarios.

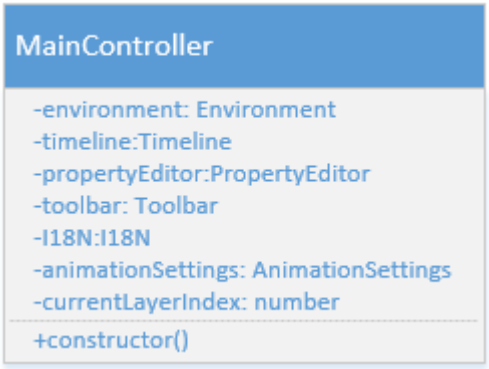


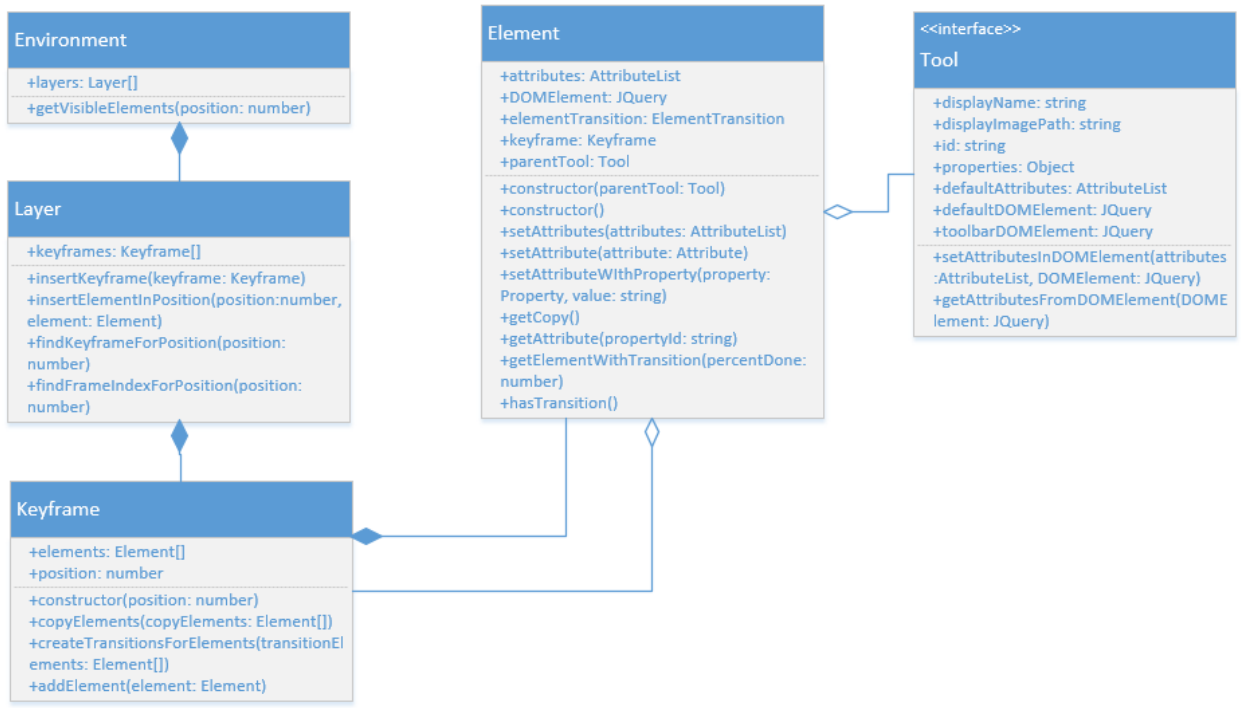
Figura 12: Diagrama de clase MainController

7.3.6 Estructuras básicas

Existen diferentes estructuras básicas en el sistema. La clase que representa el estado total del sistema es `Environment`, esta contiene un arreglo de `Layer` que representa las diferentes capas que el usuario ha creado, en orden ascendente.

Cada `Layer` contiene un arreglo de `Keyframe`. Un keyframe es un fotograma clave que el usuario ha agregado o que se ha generado por medio de animación. Cada keyframe contiene un arreglo de `Element` que representa los elementos contenidos en ese fotograma. Debido a que sólo se almacenan los fotogramas clave, los elementos de cada keyframe son diferentes.

Esta es la jerarquía básica que representa el status del sistema. Sin embargo, existen relaciones complejas entre los participantes de esta jerarquía y los componentes de la aplicación. Dichas relaciones se describen en los siguientes diagramas de clases.



7.3.6.1

Figura 13: Diagrama de clases Environment, Layer, Keyframe, Element, Tool

7.3.6.2 Environment

Representa el estado total del sistema. Contiene un arreglo de `Layer`.

Método	Argumentos	Descripción
`getVisibleElements`	`position: number`	Regresa una lista de `Element` que contiene los elementos visibles en una posición de tiempo especificada por el argumento `position`.

Tabla 36: Métodos de clase Environment

7.3.6.3 Layer

La clase `Layer` representa una capa en el documento actual. El usuario puede crear múltiples capas para organizar su trabajo.

Método	Argumentos	Descripción
`insertKeyframe`	`keyframe: Keyframe`	Inserta un keyframe en la capa.
`insertElementInPosition`	`Position: number` `Element: Element`	Inserta un elemento especificado en la posición especificada. Es decir, en el keyframe que corresponde a la posición especificada.
`addElement`	`Element: Element`	Inserta un elemento en el keyframe.

Tabla 37: Métodos de clase Layer

7.3.6.4 Keyframe

Un objeto `Keyframe` representa un fotograma en el documento actual. Cada fotograma tiene una posición en el tiempo, así como la lista de elementos que se encuentran insertados en ese fotograma.

Método	Argumentos	Descripción
`Constructor`	`Position: number`	Crea un `Keyframe`
`copyElements`	`Elements: Element[]`	Inserta una copia de cada uno de los elementes presentes en una lista especificada.
`createTransitionForElements`	`Elements: Element[]`	Crea elementos de transición dentro de los elementos especificados
`findFrameIndexForPosition`	`Position: number`	Regresa el índice de la lista de keyframes correspondiente al keyframe que corresponde a una posición especificada.

Tabla 38: Métodos de clase Keyframe

7.3.6.5 Element

Un objeto Element representa un elemento que está presente en el canvas. Dicho elemento puede ser una representación directa de elementos standard HTML, como una imagen; o una representación de un objeto más abstracto, como una figura geométrica.

Cada `Element` se crea a partir de una herramienta, representada por las clases que implementan la interfaz `Tool`. Estas herramientas definen el comportamiento por default que los elementos creados a partir de las mismas muestran. Tales como el objeto DOM que se inserta por default y métodos para traducir entre dominios de propiedades.

Método	Argumentos	Descripción
`constructor`	`parentTool: Tool`	Crea un `Element` usando el argumento `parentTool` como herramienta base.
`setAttributes`	`Attributes: AttributeList`	Agrega un conjunto de atributos al elemento.
`setAttribute`	`Attribute:Attribute`	Agrega un atributo al elemento.
`setAttributeWithProperty`	`Property: Property Value: string`	Establece con el valor especificado el atributo que pertenece a la propiedad especificada.
`getCopy`		Regresa una copia profunda del elemento.
`getAttribute`	`propertyId: String`	Regresa el atributo que pertenece a la propiedad con la id especificada.
`getElementWithTransition`	`percentDone: number`	Regresa un elemento intermedio en una transición. Se especifica el

		porcentaje realizado de dicha transición.

Tabla 39: Métodos de clase Element

7.3.6.6 Tool

La interfaz Tool representa una herramienta que puede ser usada en la barra de herramientas. Esta provee un prototipo a partir del cual se crearán los Elements de dicha herramienta.

Método	Argumentos	Descripción
`setAttributesInDOMElement`	`Attributes: AttributeList` `DOMElement: JQuery`	Aplica una lista de atributos a un elemento en el DOM. Es decir, traduce del dominio de propiedades abstractas al del DOM.
`getAttributesFromDOMElement`	`DOMElement: JQuery`	Regresa una lista de atributos correspondientes a las propiedades del DOM del elemento. Es decir, traduce del dominio de propiedades del DOM al dominio de propiedades abstractas.

Tabla 40: Métodos de clase Tool

7.3.7 Dominios de propiedades

Existen dos dominios de propiedades que son manejados por la aplicación. Uno de ellos representa las propiedades que el usuario modifica fácilmente mediante la interfaz gráfica, estas no son propiedades reales de un elemento de HTML o estilo CSS, sólo se proveen para que sea fácil editar elementos. Por ejemplo, el radio de un círculo.

El otro dominio de propiedades representa las verdaderas propiedades de un elemento de HTML o estilo CSS.

Debe existir una forma de asociar un dominio de propiedades con el otro. Cada herramienta tiene una manera diferente de traducir sus diferentes propiedades entre estos dominios, la implementación de esto reside en el método `setAttributesInDOMElement` de cualquier clase que implemente la interfaz `Tool`, es decir, cada una de las herramientas.

Por ejemplo, una propiedad modificable por el usuario en un elemento de audio sería `source`, que indica la ruta del archivo de audio a reproducir. Sus propiedades verdaderas correspondientes sería crear un elemento HTML `<source>` dentro de la etiqueta `<audio>` y modificar sus propiedades `src` y `type`. De esta manera, el usuario solo tiene que especificar la ruta del archivo, sin conocimiento de que la aplicación en realidad está agregando una nueva etiqueta con varias propiedades HTML.

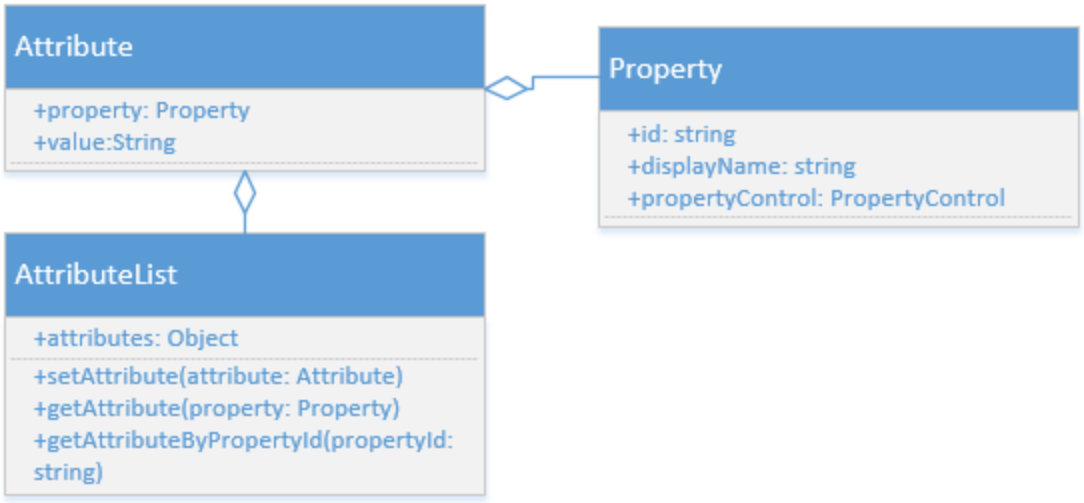


Figura 14: Diagrama de clases Attribute, Property y AttributeList

7.3.8 Property

La clase `Property` representa una propiedad en el sistema. Esta propiedad se caracteriza por un identificador único `id`, un nombre con el que se le muestra al usuario `displayName`. También puede contener un campo `propertyControl`, que se explica posteriormente. El campo `reverseProperty` se utiliza para asignar a la propiedad una propiedad CSS a la cual corresponde, esto es útil para tener más información sobre las propiedades de bajo nivel que una `Property` representa cuando se crea una animación.

7.3.9 Attribute

La clase Attribute representa un valor asignado a una propiedad. Cuenta con una referencia a la `Property` y el valor se representa como un `string`.

7.3.10 AttributeList

La clase AttributeList representa una lista de atributos. Se utiliza para acceder eficientemente a los atributos de un elemento, ya que almacena estos atributos en un objeto de JavaScript. La mayoría de los navegadores implementan los objetos de JavaScript con árboles, esto hace eficiente la búsqueda de un atributo específico en la lista.

7.3.11 Barra de herramientas

La clase `Toolbar` representa un controlador de la barra de herramientas.

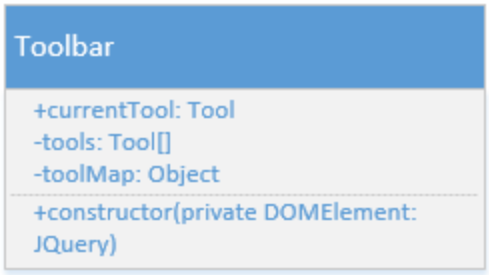


Figura 15: Clase Toolbar

La clase `Toolbar` se encarga de cargar las herramientas soportadas en la interfaz gráfica, así como gestionar su uso y lanzar eventos cuando el usuario cambia de herramienta.

7.3.11.1 Eventos soportados

Evento	Descripción
`toolselect`	El usuario ha seleccionado una herramienta

Tabla 41: Eventos soportados

El siguiente diagrama de secuencia muestra la interacción entre Canvas y Toolbar cuando el usuario selecciona una herramienta.

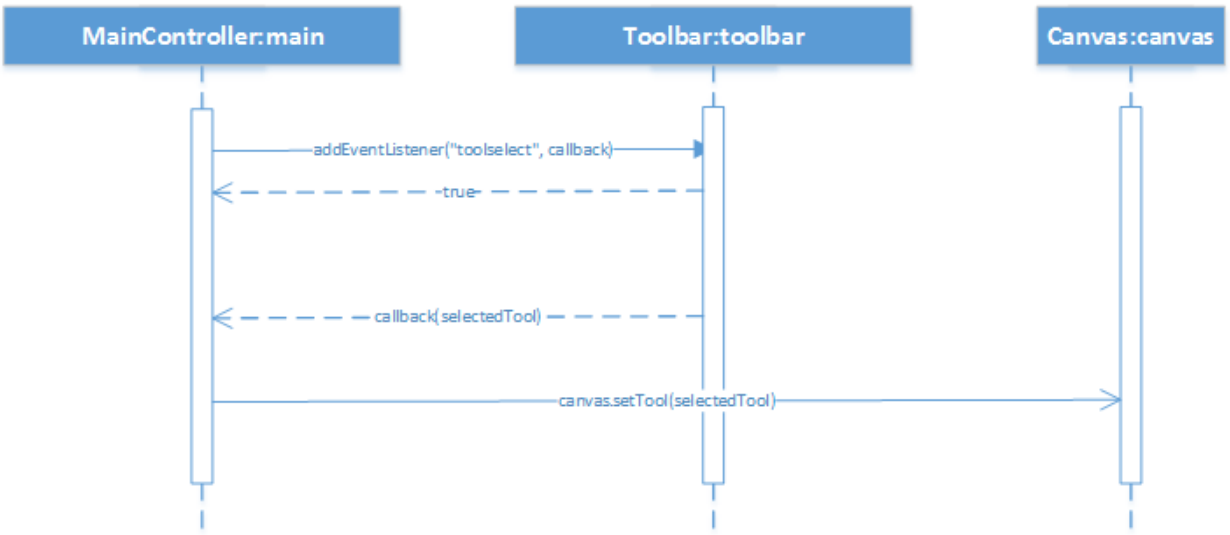


Figura 16: Diagrama de secuencia de canvas y toolbar

Este diagrama de secuencia ejemplifica la manera en que diversos componentes de la aplicación interactúan entre sí. Esto es, registrando eventos y utilizando una función de callback que se ejecuta cuando el evento sucede.

7.3.12 Editor de propiedades

Figura 17: Editor de propiedades

El editor de propiedades se representa mediante la clase PropertyEditor. Esta se encarga de manejar los eventos relacionados con el editor de propiedades, como cambiar el valor de un atributo.

7.3.13 Eventos soportados

Evento	Descripción
`propertychange`	El usuario ha cambiado las propiedades de un elemento por medio del editor de propiedades

Tabla 42: Tabla de eventos soportados del editor de propiedades

El siguiente diagrama de secuencia muestra la interacción entre el editor de propiedades y el canvas cuando el usuario modifica las propiedades de un elemento.

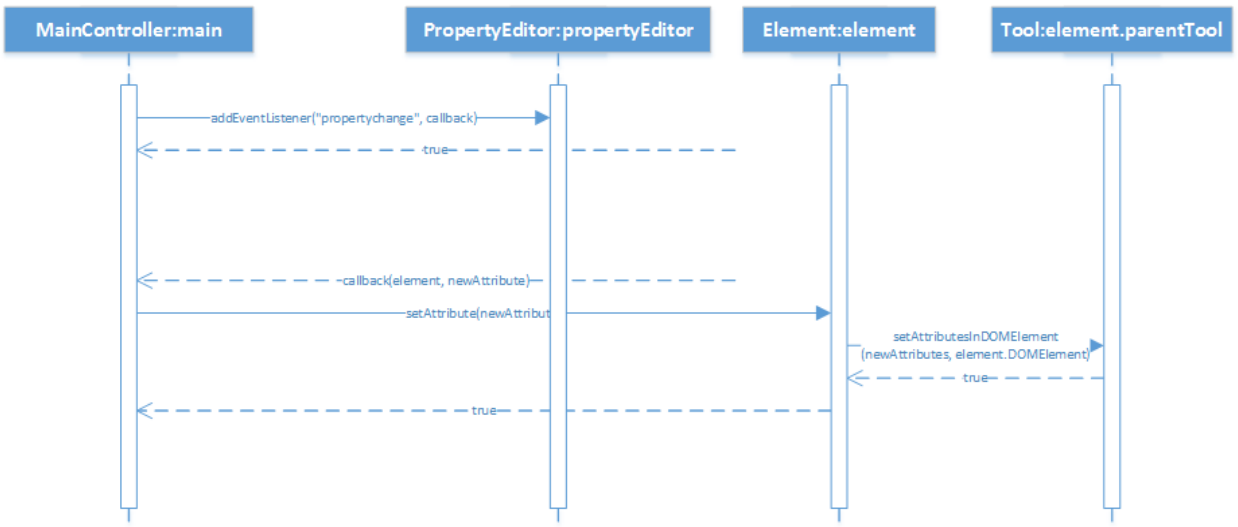


Tabla 43: Diagrama de secuencia de propertyEditor, element y parentTool

Aunque `Canvas` no se muestra en este diagrama de secuencia, posee una referencia al `DOMElement` del elemento afectado, por lo que el elemento que se muestra en el canvas está actualizado.

7.3.14 Línea de tiempo

La línea de tiempo provee al usuario con una interfaz para animar los diferentes elementos en el canvas.

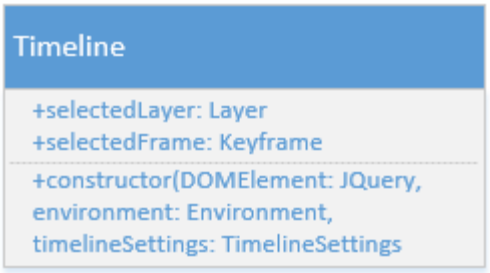


Figura 18: Clase Timeline

La clase Timeline representa la línea de tiempo, contiene los elementos que tienen puntos clave (transiciones), así como un árbol de puntos.

Método	Argumentos	Descripción

`constructor`	`DOMElement: JQuery`
	`Environment: Environment`
	`timelineSettings: TimelineSettings`
		Crea un Timeline dentro del DOMElement especificado con los ajustes especificados.

Tabla 44: Tabla de métodos de clase Timeline

7.3.15 Eventos soportados

Evento	Descripción
`playbuttonclick`	Ocurre cuando el usuario hace click en el botón de reproducir vista previa en la línea de tiempo.
`layerselect`	Un `number` que corresponde al índice de la capa seleccionada por el usuario. Iniciando en 0.
`frameselect`	

Tabla 45: Tabla de eventos soportados por la clase TimeLine

7.3.16 Canvas

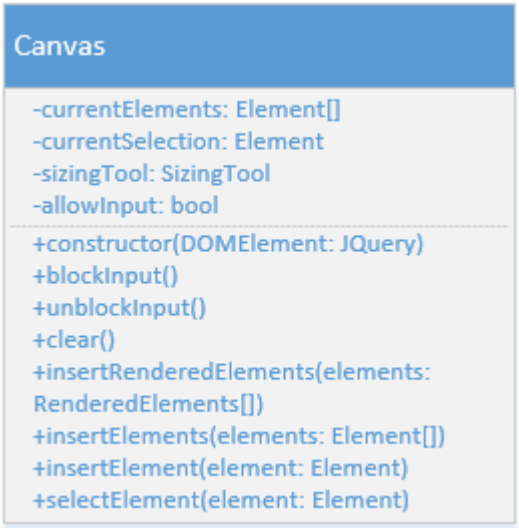


Figura 19: Clase Canvas

La clase Canvas controla la interfaz gráfica del canvas. Se encarga de recibir los eventos de inserción de elementos. Mantiene una lista de elementos mostrados actualmente y la selección actual del usuario.

Método	**Argumentos**	**Descripción**
`constructor`	`DOMElement: JQuery currentTool: Tool environment: Environment`	Crea un nuevo canvas dentro del DOMElement especificado.
`blockInput`		Hace que el Canvas deje de aceptar entradas. Es decir, no reaccione al click del usuario. Esto se utiliza para impedir que el usuario utilice el canvas cuando está en proceso una vista previa de la animación.
`unblockInput`		Hace que el Canvas vuelva a aceptar entradas del usuario.
`clear`		Regresa el Canvas a su estado vacío original.
`insertRenderedElements`	`Elements: RenderedElement[]`	Inserta un conjunto de `RenderedElements` al canvas para mostrar una animación que ya se ha generado.
`insertElements`	`Elements: Element[]`	Inserta un conjunto de `Elements` al canvas
`insertElement`	`Element: Element`	Inserta un `Element` al canvas

`selectElement`	`Element: Element`	Selecciona un `Element` que ya está dentro del canvas.

Tabla 46: Tabla de métodos de clase Timeline

7.3.17 Animación

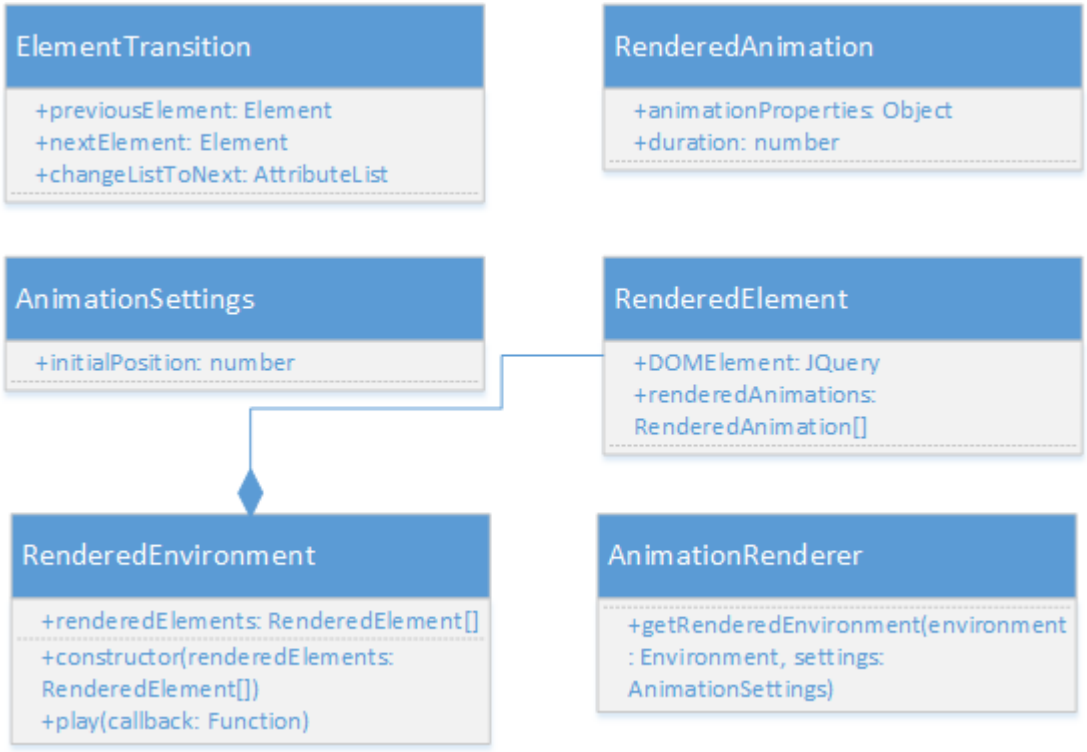


Figura 20: Clases ElementTransition, RenderedAnimation, RenderedElement, AnimationSettings. RenderedEnvironment y AnimationRenderer

Las animaciones se representan por medio de `ElementTransition`. Esta es básicamente una lista doblemente enlazada para asociar un elemento con uno previo y uno siguiente.

Los `Element` que contienen una animación tienen definido un `ElementTransition`, el cual especifica el siguiente `Element` al cual animar.

Una `RenderedAnimation` representa un conjunto de propiedades CSS a las cuales se les ha asignado un valor. La clase `AnimationSettings` sirve para especificar los ajustes de la animación, como un offset para la posición inicial y los cuadros por segundo.

La clase `RenderedElement` representa un elemento cuyas animaciones ya han sido generadas. Un `RenderedEnvironment` representa todo el entorno cuyas animaciones ya han sido generadas y contiene una lista de `RenderedElements.`

Método	**Argumentos**	**Descripción**
`constructor`	`renderedElements: RenderedElement[]`	Crea un nuevo RenderedEnvironment
`play`	`Callback: Function`	Ejecuta las animaciones generadas.

Tabla 47: Métodos de RenderedEnvironment

7.3.18 Generación de código

La generación de código se realiza traduciendo el contenido de un `Environment` a un objecto del DOM, representado dentro de un `RenderedEnvironment`.

Figura 21: Proceso de generación de código

A grandes rasgos la generación de código en la aplicación se realiza de la siguiente manera.

1. El usuario interactúa con la aplicación utilizando la interfaz gráfica
2. Los controladores mantienen un objeto `Environment` en memoria que representa el documento en el que el usuario está trabajando.
3. La clase `AnimationRenderer` genera un `RenderedEnvironment` a partir de este Environment.
4. La clase `CodeGenerator` genera el código resultante en HTML, JavaScript y CSS a partir del `RenderedEnvironment`. Lo almacena en memoria en un objeto `EnvironmentSyncer`.
5. El objeto `EnvironmentSyncer` se sincroniza con el servidor. En el servidor se escriben los archivos resultantes.

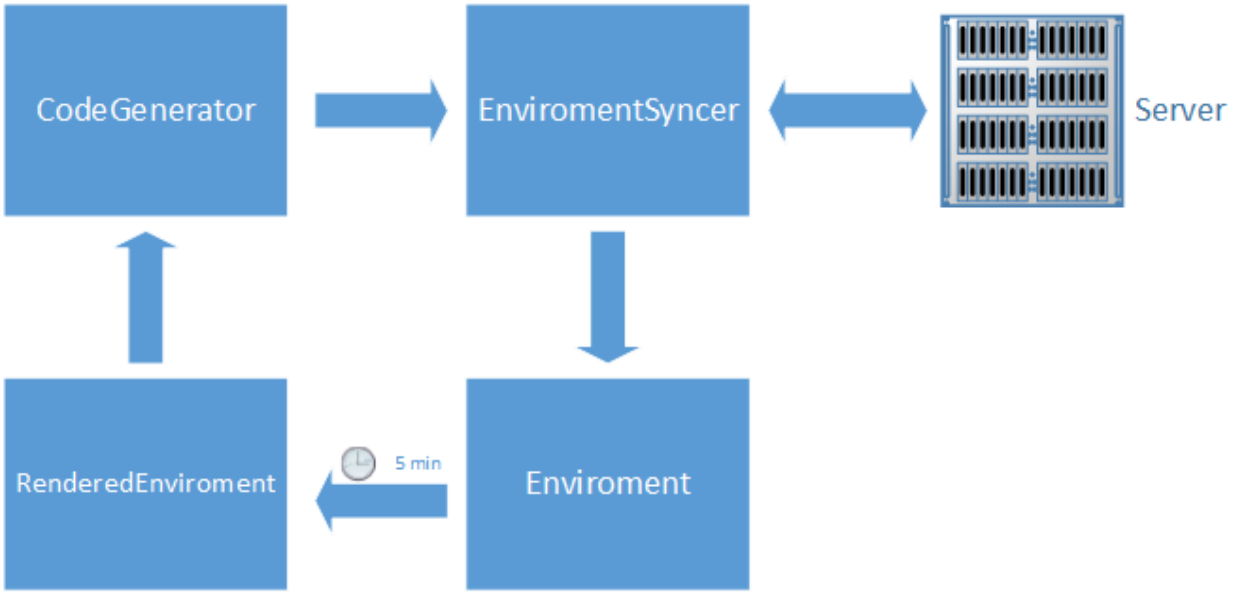


Tabla 48: Flujo de código generado

El cliente se comunica con el servidor por medio de peticiones HTTP asíncronas, también conocido como AJAX. El servidor Web expone un servicio Web en cual acepta las peticiones del cliente y responde con objetos JSON.

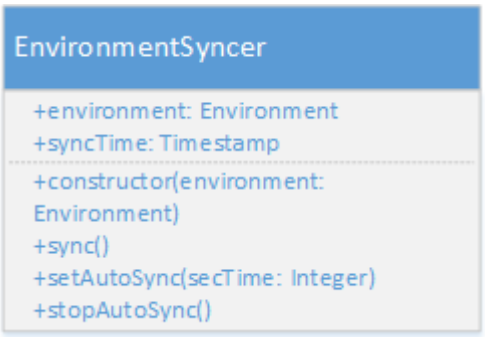


Figura 22: Clase EnvironmentSyncer

Para facilitar la sincronización periódica, `EnvironmentSyncer` soporta sincronización automática cada cierto tiempo mediante el método `setAutoSync`.

7.3.18.1.1 Eventos soportados

Evento	**Descripción**
load	Se ha terminado de cargar el documento por primera vez
loadfail	Ha ocurrido un error al intentar cargar el documento
sync	Se ha terminado de sincronizar el documento
Syncfail	Ha ocurrido un error al intentar sincronizar el documento

Tabla 49 Eventos soportados por clase RemoteEnvironment:

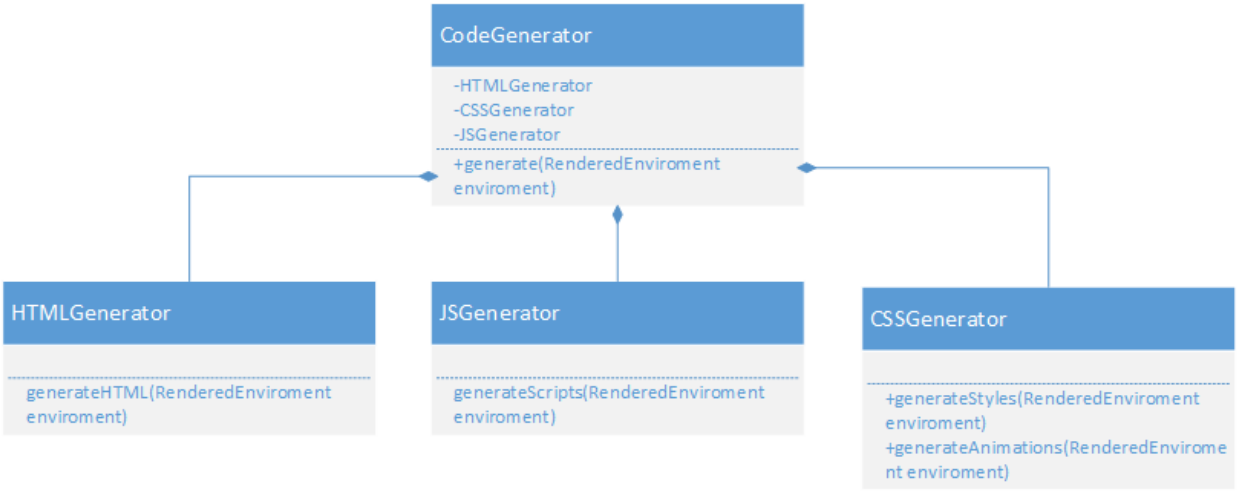


Tabla 50: Diagramas de clases de generación de código

7.3.18.2 CodeGenerator

Se encarga de generar el código multi-navegador de un Enviroment dado. Se compone de objetos HTMLGenerator, CSSGenerator y JSGenerator

Método	**Argumentos**	**Descripción**

`generate`	`enviroment: RenderedEnviroment`	Genera los tres archivos correspondientes al código del Enviroment dado.

Tabla 51: Métodos de clase CodeGenerator

7.3.18.3 HTMLGenerator

Se encarga de generar el código HTML multi-navegador de un Enviroment dado.

Método	**Argumentos**	**Descripción**
`generate`	`enviroment: RenderedEnviroment`	Genera el archivo HTML correspondiente al código del Enviroment dado.

Tabla 52: Métodos de clase HTMLGenerator

7.3.18.4 JSGenerator

Se encarga de generar el código JavaScript multi-navegador de un Enviroment dado.

Método	**Argumentos**	**Descripción**
`generateScripts`	`enviroment: RenderedEnviroment`	Genera el archivo HTML correspondiente al código del Enviroment dado.

Tabla 53: Métodos de clase HTMLGenerator

7.3.18.5 CSSGenerator

Se encarga de generar el código CSS multi-navegador de un Enviroment dado.

Método	**Argumentos**	**Descripción**
`generateStyles`	`enviroment: RenderedEnviroment`	Genera el archivo CSS correspondiente a los estilos usados en el Enviroment dado.
`generateAnimations`	`enviroment: RenderedEnviroment`	Genera el archivo CSS correspondiente a las animaciones usadas en el Enviroment dado.

Tabla 54: Métodos de clase CSSGenerator

7.3.19 Sincronización con servidor

7.3.19.1.1 Autosync

Para facilitar la sincronización periódica, `RemoteEnvironment` soporta sincronización automática cada cierto tiempo mediante el método setAutoSync.

7.3.19.1.2 Eventos soportados

Evento	**Descripción**
load	Se ha terminado de cargar el documento por primera vez
loadfail	Ha ocurrido un error al intentar cargar el documento
sync	Se ha terminado de sincronizar el documento
syncfail	Ha ocurrido un error al intentar sincronizar el documento

Tabla 55: Tabla de eventos soportados por RemoteEnvironment

7.4 DISEÑO DETALLADO DEL SERVIDOR

Es importante mencionar que la gran parte del proyecto está enfocada a el desarrollo de una aplicación que en su mayoría se ejecuta en el cliente. Sin embargo, al considerar la posibilidad de la generación de "proyectos", es inevitable pensar en una aplicación que corra en un servidor Web y que en general se encargue de la persistencia de los archivos que conforman el proyecto (código HTML, CSS y JavaScript).

Por tanto la aplicación en el servidor solo atacará el problema de la persistencia de datos así como el control de usuarios, esto para ofrecer al usuario un espacio personalizado y confidencial donde pueda organizar sus diseños de manera segura. Dicho objetivo se cumplirá a través de la construcción de un servicio Web que permita las operaciones básicas sobre los proyectos generados por el usuario (CRUD).

Se mantendrá una base de datos, donde se almacenarán sólo los datos estrictamente necesarios para nuestro propósito, y la construcción del servicio Web se implementará a través del modelo REST para la creación de servicios Web, siendo JSON el formato para la transmisión de contenido entre el cliente y el servidor, haciendo de nuestra aplicación únicamente dependiente del protocolo HTTP.

El servidor será construido siguiendo el patrón de desarrollo MVC.

7.4.1 Tecnologías consideradas

Para el desarrollo de la aplicación consideramos diferentes alternativas de tecnologías. En general, las tecnologías ofrecen diversas ventajas competitivas y atractivas para el desarrollo de la aplicación. Es difícil elegir alguna de las tecnologías y especificar cuáles son las ventajas una sobre otra que nos atrajeron para elegir dicha tecnología, ya que nuestra aplicación en el servidor es muy limitada y solo nos servirá para cumplir los requerimientos mínimos de control de usuarios y relación con los proyectos creados, así como la persistencia de datos de cada uno de los proyectos.

Así llegamos a la conclusión de que para el desarrollo de dicha aplicación utilizaremos .NET MVC ya que los integrantes del equipo tenemos mayor experiencia en desarrollo con dicha tecnología y nos es rápida la administración en ella.

7.5 MODELOS

Las siguientes entidades representan los datos que en general construyen la aplicación en el servidor Web.

7.5.1 Usuarios

Figura 23: Clase User

7.5.2 Página

Una página se compone de tres archivos: código HTML, CSS y JavaScript. Una página además, es el resultado del proceso de generación de código de nuestra aplicación. Así, cada diseño que cree el usuario será el equivalente a una página.

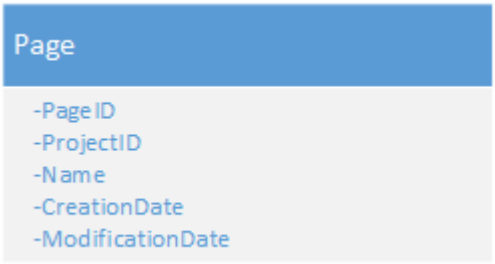


Figura 24: Clase Page

7.5.3 Proyectos

Un proyecto es un conjunto de páginas. Cada proyecto tiene propiedades como fecha de creación, modificación, nombre, etcétera. Un usuario puede tener muchos proyectos, y los proyectos solo pueden ser accedidos por un usuario único registrado en la aplicación.

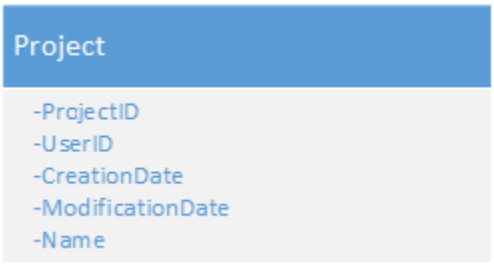


Figura 25: Clase Project

7.5.4 Recursos

Los recursos son los archivos que el usuario necesita para crear su animación.

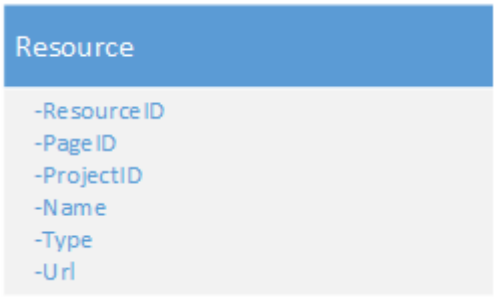


Figura 26 Clase Resource

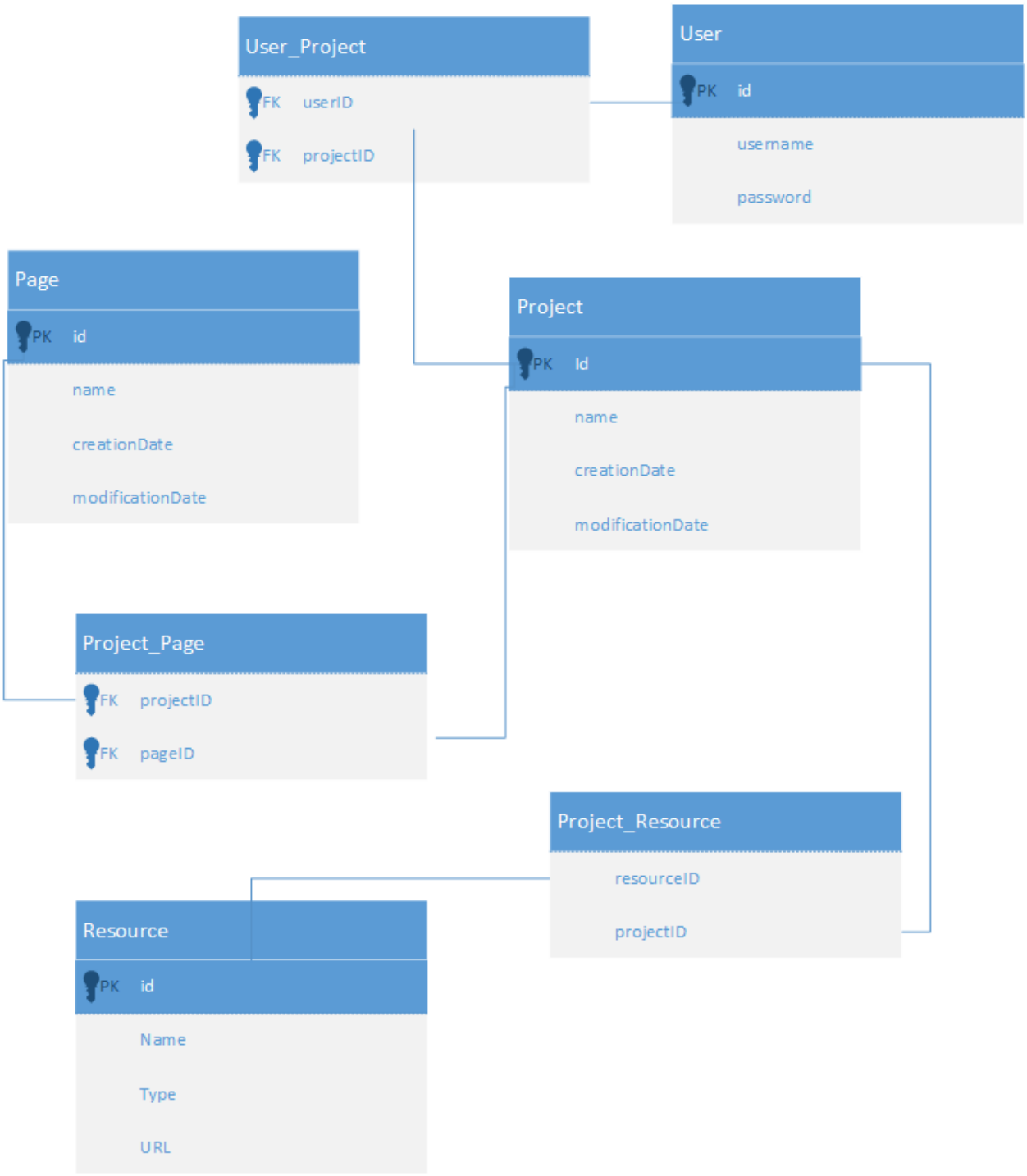


Figura 27: Diagrama Entidad Relación de la base de datos

7.6 CONTROLADORES

La aplicación cuenta con tres controladores: Usuario, proyecto y página

7.6.1 Usuario

El controlador Usuario soporta la creación de usuarios así como iniciar y cerrar sesión

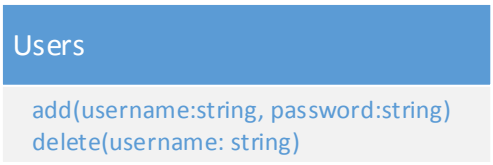


Figura 28: Clase Users

Método	Argumentos	Descripción
Register	username: string password: string	Registra un nuevo usuario en la base de datos
Register		Proporciona la vista para registrar usuario
Login	username: string password: string	Inicia sesión de un usuario
Login		Proporciona la vista para iniciar sesión
Logoff		Cierra la sesión del usuario

Tabla 56: Tabla de métodos de clase Users

7.6.2 Proyecto

El controlador Proyecto administra la creación, edición y eliminación de proyectos.

Figura 29: Clase Projects

Método	Argumentos	Descripción
Create	project: Project	Crea un nuevo proyecto en la base de datos
Index		Muestra la lista de proyectos del usuario
Details	projectID : int	Muestra los detalles del proyecto con el id especificado
Create		Muestra la vista para poder registrar un nuevo proyecto

Edit	projectID : int	Muestra la vista para editar un proyecto
Edit	project : Project	Edita el proyecto especificado
Delete	projectID : int	Muestra la vista para eliminar el proyecto con id especificado
DeleteConfirm	projectID : int	Elimina el proyecto especificado
Zip	projectID: int	Descarga un archivo comprimido con los archivos del proyecto

Tabla 57: Tabla de métodos de clase Projects:

7.6.3 Página

Es el controlador encargado de recibir los eventos relacionados con las páginas (eliminar, editar, visualizar, crear).

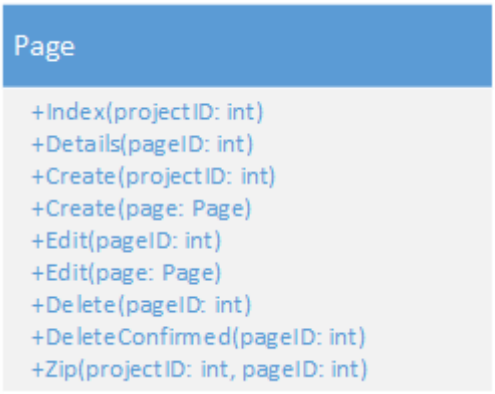


Figura 30: Clase Page

Método	Argumentos	Descripción
Create	projectID: int	Muestra la vista para crear una nueva página en el proyecto especificado
Index	projectID: int	Muestra la lista de páginas del proyecto especificado
Details	pageID : int	Muestra los detalles de la página con el id especificado
Create	page: Page	Crea una página
Edit	pageID : int	Muestra la vista para editar una página que contenga el id especificado
Edit	page : Page	Edita la página con los datos especificados
Delete	pageID : int	Muestra la vista para eliminar la página con id especificado
DeleteConfirm	projectID : int	Elimina la página con ID especificado
Zip	projectID: int pageID: int	Descarga un archivo comprimido con los archivos de la página especificada

Tabla 58: Métodos de clase Page

7.6.4 JSON

Es el controlador encargado de recibir y enviar toda la información que se manipulará en formato JSON a través de servicios web.

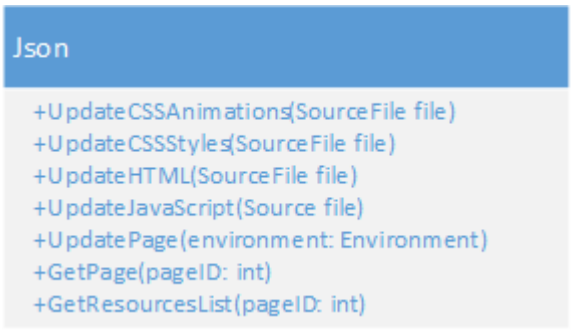


Figura 31Clase Json

Método	Argumentos	Descripción
UpdateCSSAnimations	file: SourceFile	Recibe un objeto JSON con el contenido e información del archivo CSS a guardar
UpdateCSSStyles	file: SourceFile	Recibe un objeto JSON con el contenido e información del archivo CSS a guardar
UpdateHTML	file: SourceFile	Recibe un objeto JSON con el contenido e información del archivo HTML a guardar
UpdateJavaScript	file: SourceFile	Recibe un objeto JSON con el contenido e información del archivo JavaScript a guardar
UpdatePage	environment:Environment	Recibe un objeto JSON con el estado de la aplicación en el cliente.
GetPage	pageID : int	Regresa el objeto JSON que indica el último estado en el que se quedó la aplicación cuando se editaba la página con id especificado
GetResourcesList	pageID : int	Muestra la lista de recursos que se pueden usar en la página con id especificado

Tabla 59 Métodos de clase JSON

7.6.5 Resource

Es el controlador encargado de recibir los archivos que el usuario necesite para crear su página.

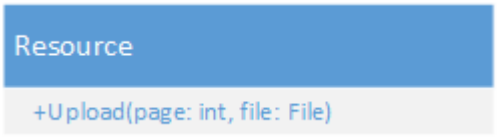


Figura 32 Clase Resource

Método	Argumentos	Descripción
Upload	page: int file: File	Recibe un archivo que pertenece a la página con el id especificado

Tabla 60 Métodos de clase Resource

7.7 VISTAS

Las vistas que integran la interfaz gráfica del sitio Web son las siguientes:

- Login
- Registrar
- Lista de proyectos
- Crear proyectos
- Detalles de proyecto
- Eliminar proyecto
- Lista de páginas de un proyecto
- Crear página
- Detalles de página
- Eliminar página
- Main
- Index

Index es la página que se muestra a usuarios no registrados. Login es es la vista para iniciar sesión. Registrar es la vista para crear una nueva cuenta. Lista de proyectos y de páginas de proyecto muesran las listas de los elementos mencionados. También es posible ver los detalles de páginas y de proyecto individualmente en una vista, así como crear y eliminar páginas y proyectos enteros. Main es la vista principal en donde se pueden crear y editar animaciones.

8 RESULTADOS

En esta sección se muestra la utilización básica de la aplicación desarrollada. Cabe destacar que los resultados y pruebas documentados en esta sección se basan en la primera versión de la aplicación.

8.1 INICIO DE SESIÓN

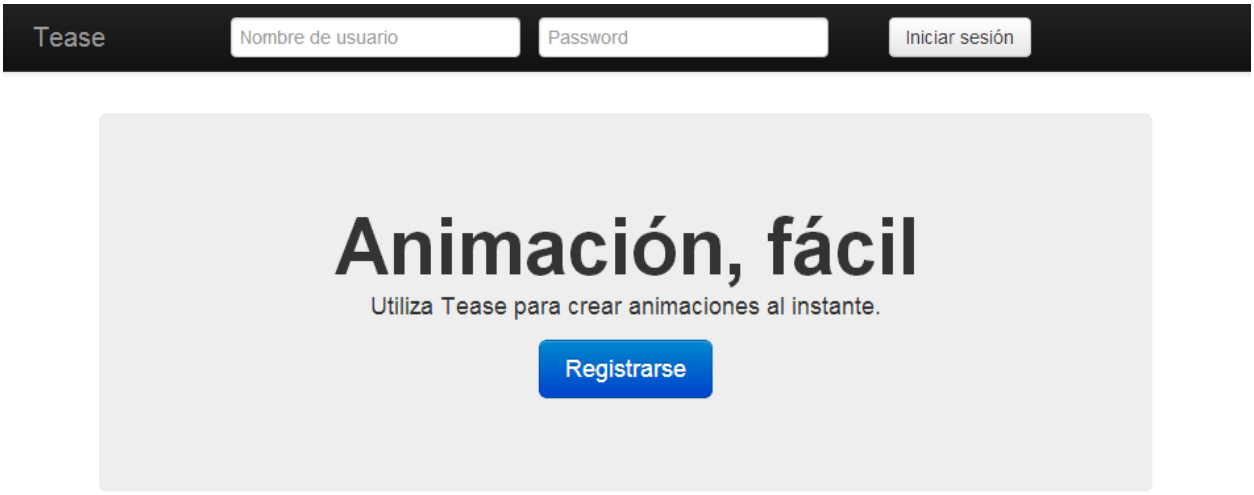


Figura 33: Pantalla de inicio de sesión

Para utilizar la aplicación el usuario debe ingresar sus credenciales en la pantalla de inicio de sesión. Posteriormente se muestran los proyectos correspondientes al usuario.

El usuario puede seleccionar el proyecto en el que desea trabajar o crear uno nuevo. Posteriormente se muestran la lista de páginas correspondiente al proyecto seleccionado.

Al seleccionar una página la aplicación la carga en la pantalla principal, detallada en la siguiente sección.

8.2 PANTALLA PRINCIPAL

A continuación se muestra la pantalla principal de la aplicación. En esta se aprecian los principales elementos de la interfaz gráfica, descritos en las subsecciones siguientes.

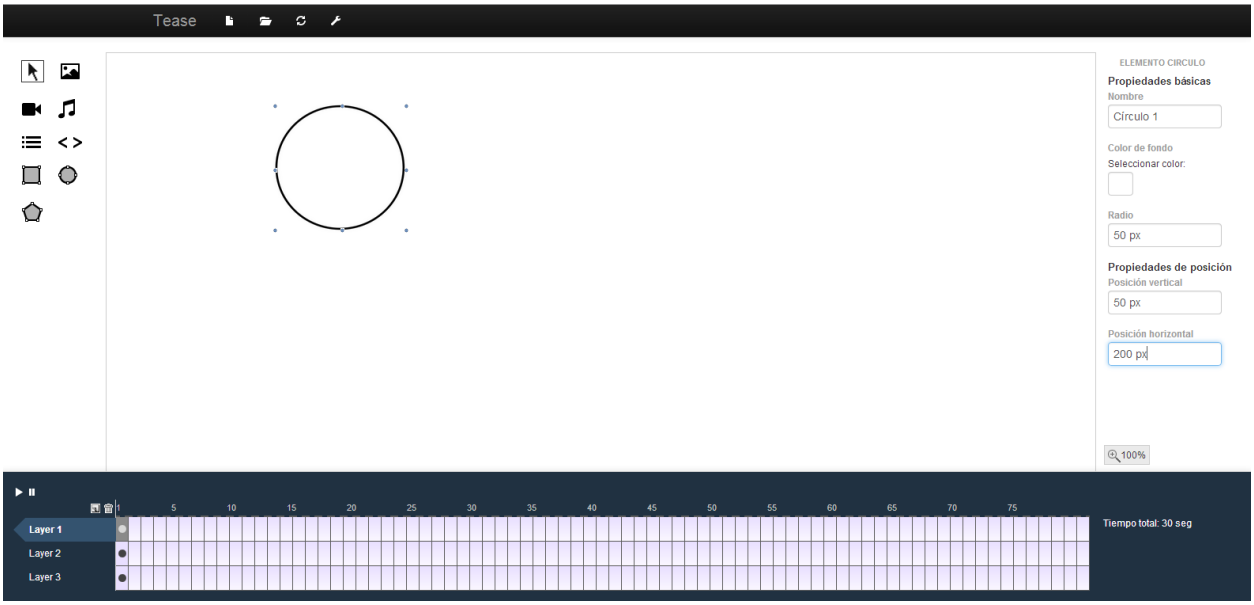


Figura 34: Pantalla principal de la aplicación

8.2.1 Barra de herramientas básicas

Contiene los comandos básicos de Nuevo, Abrir, Guardar y Configuración. Los comandos se muestran con íconos y muestran una descripción con texto cuando el usuario posiciona el mouse sobre estos.

8.2.2 Barra de herramientas

Contiene las herramientas soportadas por la aplicación. Estas se muestran con una imagen y se muestra una descripción con texto cuando el usuario posiciona el mouse sobre ellos. Se muestran 2 herramientas por fila, o una sola en caso de que la resolución horizontal sea menor a 1235 pixeles.

8.2.3 Línea de tiempo

Contiene la lista de capas. Que permite al usuario agregar capas nuevas para organizar lógicamente su trabajo, así como proveer una jerarquía vertical de elementos. También permite eliminar la capa actualmente seleccionada. El usuario puede seleccionar una capa haciendo click o tocándola en una touchscreen. Al seleccionar una capa se seleccionan los elementos correspondientes a la misma en el área de trabajo. También contiene una lista de los fotogramas correspondientes a cada capa. La lista de fotogramas permite al usuario seleccionar un fotograma para mostrar su estado en el área de trabajo. Al hacer click derecho con el mouse o tocar un fotograma con la touchscreen se muestra el menú de opciones de fotograma. Con este menú el usuario puede Insertar un fotograma clave, Vaciar el fotograma, Animar al fotograma y Eliminar un fotograma clave en caso de que se haya seleccionado un fotograma clave. También contiene un editor de las propiedades de animación del documento, en el cual el usuario puede modificar los fotogramas por segundo y la duración total de la animación. En la parte superior se encuentran los botones de vista previa, con las que el usuario puede iniciar la vista previa o pausarla en caso de que esté reproduciéndose.

8.2.4 Área de trabajo

Muestra los objetos actualmente presentes en el documento. El usuario puede interactuar con ellos seleccionándolos, moviéndolos o insertando nuevos objetos en el documento.

8.2.5 Editor de propiedades

Muestra las propiedades de un objeto seleccionado. El usuario puede asignar valores a estas propiedades utilizando diferentes controles de propiedades.

8.2.6 Barra de estado

Muestra el nombre de la página en la que actualmente se está trabajando y la última vez que se sincronizó el documento con el servidor.

8.3 INSERCIÓN DE ELEMENTOS

El usuario puede seleccionar una herramienta en la barra de herramientas. Posteriormente, la herramienta mostrará un marco de selección, que nos indica la herramienta actualmente seleccionada.

Ahora el usuario puede utilizar la herramienta para interactuar con el área de trabajo. Insertando elementos independientes.

El usuario puede seleccionar uno o varios elementos utilizando la herramienta de puntero. Con esta herramienta también puede mover los elementos seleccionados.

8.4 MODIFICACIÓN DE PROPIEDADES

El editor de propiedades provee una interfaz con la que el usuario puede modificar las propiedades de un objeto fácilmente. Dependiendo del tipo de objeto que estemos seleccionando, se muestran diferentes propiedades y controles para modificarlas.

8.5 GENERACIÓN DE CÓDIGO

El usuario puede utilizar el comando guardar para generar el código y sincronizarlo con el servidor. La aplicación se auto-sincroniza con el servidor cada 10 minutos, por lo que el usuario no tiene que preocuparse por perder su trabajo

A continuación se presenta un ejemplo que describe una animación en un proyecto y se muestra el código que la aplicación genera para éste. Se trata de una imagen que se escala a 1.5X inmediatamente después de la carga de la página.

```
<!DOCTYPE html>
<html>
<head>
    <title>Página 1</title>
    <link rel="stylesheet" type="text/css" href="ejemplo-styles.css">
    <link rel="stylesheet" type="text/css" href="ejemplo-animations.css">
    <script type="javascript/text" src="jquery.js"> </script>
    <script type="javascript/text" src="ejemplo.js"> </script>
</head>

<body onload="runInitialAnimations();">
    <div>
        <img src="images/carlos.png" class="image1"> </img>
    </div>
</body>

</html>
```

Código 1: Generador de código - ejemplo.html

```
runInitialAnimations() {
    $(".image1").toggleClass("image1animation1");
}
```

Código 2: Generador de código - ejemplo.js

```
@keyframes animation1 {
    from {
        transform: none;
    }

    to {
        transform: scale(1.5);
    }
}

@-webkit-keyframes animation1 {
    from {
        -webkit-transform: none;
    }

    to {
        -webkit-transform: scale(1.5);
    }
}

@-moz-keyframes animation1 {
    from {
```

```
        -moz-transform: none;
    }

    to {
        -moz-transform: scale(1.5);
    }
}

@-ms-keyframes animation1 {
    from {
        -ms-transform: none;
    }

    to {
        -ms-transform: scale(1.5);
    }
}

.image1animation1 {
    animation: animation1 .5s;
    -webkit-animation: animation1 .5s;
    -moz-animation: animation1 .5s;
    -ms-animation: animation1 .5s;
}
```

Código 3: Generador de código - ejemplo-animations.css

```
.image1 {
    border: 0px;
}
```

Código 4: Generador de código - ejemplo-style.css

9 PRUEBAS

En esta sección se detallan las pruebas que en general se realizaron a la aplicación para verificar su correcto funcionamiento, tiempos de respuesta y manejo de errores. Las características del equipo de cómputo en el que se realizaron las pruebas son las siguientes:

- Procesador: Intel Core i3 370M a 2.40 GHz
- Memoria RAM: 4:00 GB
- Sistema: Windows 8 Pro de 64 bits

También es importante mencionar que las pruebas se realizaron en los navegadores:

- Microsoft Internet Explorer 10
- Mozilla Firefox 20.0.1
- Google Chrome 26.0.1410.64
- Safari 4

Caso de prueba: Carga de vista principal de la aplicación	
Descripción	Cargar de aplicación principal (pantalla de edición de animaciones)
Resultado	Carga todos los elementos iniciales (scripts, imágenes y otros archivos pertenecientes a los frameworks utilizados)
Tiempo de respuesta	• 233 ms en Google Chrome • 240 ms Internet Explorer • 233 Mozilla Firefox • Safari 5.1.7

Tabla 61: Resultados de caso de prueba Carga de vista principal a la aplicación

Caso de prueba: Insertar herramienta en el canvas	
Descripción	Seleccionar varias herramientas arbitrariamente e insertar dicha herramienta en el canvas
Resultado	Se inserta el elemento en la posición en la que se soltó la herramienta
Tiempo de respuesta	Los tiempos de respuesta no variaron demasiado entre cada una de las herramientas ya que en todos los casos fue de 1ms a 2ms. Solo las herramientas que requieren cargar fuentes de información como imágenes tardaron más en ejecutarse dependiendo del tamaño del archivo. Ejemplos de dichas herramientas: imágenes, audio.

Tabla 62: Resultados de caso de prueba Insertar herramienta en el canvas

Caso de prueba: Marcar selección de herramienta individual en el canvas	
Descripción	Seleccionar una herramienta individualmente y esperar a que se muestra la herramienta para cambiar de tamaño un elemento o se muestre la indicación de selección.
Resultado	Se muestra alguna de las dos siguientes herramientas: la herramienta para cambiar de posición un elemento o se selecciona el elemento con 4 puntos
Tiempo de respuesta	En un promedio de 2ms a 3 ms en todos los navegadores.

Tabla 63: Resultados de caso de prueba Marcar selección de herramienta individual en el canvas

Caso de prueba: Marcar selección de múltiples herramienta en el canvas	
Descripción	Seleccionar un conjunto de herramientas

	mediante la herramienta de selección y esperar a ver los elementos marcados.
Resultado	Se muestra alguna de las dos siguientes herramientas: la herramienta para cambiar de posición un elemento o se selecciona el elemento con 4 puntos
Tiempo de respuesta	En un promedio de 2ms a 3 ms en todos los navegadores.

Tabla 64: Resultados de caso de prueba Mostrar selección de múltiples herramientas en el canvas

Caso de prueba: Mover elementos dentro del canvas	
Descripción	Seleccionar un conjunto de elementos mediante la herramienta de selección y cambiar su posición
Resultado	Los elementos que hayan sido seleccionados son arrastrados hacia la posición que indique el puntero. Cuando son soltados los elementos, la herramienta coloca los elementos en la nueva posición y marca los elementos seleccionados.
Tiempo de respuesta	El tiempo varía dependiendo de la cantidad de elementos que se estén moviendo. Se registraron tiempos de 1ms a 2ms para grupos de 1 a 5 elementos. El máximo tiempo registrado es de 14 ms para un total de 35 elementos.

Tabla 65: Resultados de caso de prueba Mover elementos dentro del canvas

Caso de prueba: Cargar herramienta para cambiar tamaño de elemento	
Descripción	Seleccionar un solo elemento insertado previamente en el canvas
Resultado	El elemento al que se quiera seleccionar, si es un elemento que soporta redimensionamiento entonces aparece sobre él la herramienta para cambiar tamaño (8 puntos)
Tiempo de respuesta	Para cualquier elemento seleccionado, no importa que tipo, el tiempo de respuesta máximo es de 2ms.

Tabla 66: Resultados de caso de prueba Mover elementos dentro del canvas

Caso de prueba: Cambiar el tamaño de un elemento con la herramienta de redimensionamiento	
Descripción	Seleccionar un solo elemento insertado previamente en el canvas y esperar a que aparezca la herramienta de redimensionamiento. Cambiar su tamaño

	mediante esta opción.
Resultado	El elemento cambia sus dimensiones y en ciertos puntos, su posición.
Tiempo de respuesta	Para cualquier elemento seleccionado, no importa que tipo, el tiempo de respuesta máximo es de 3ms.

Tabla 67: Resultados de caso de prueba Cambiar el tamaño de un elemento con la herramienta de redimensionamiento

Caso de prueba: Cambiar profundidad de un elemento en el canvas	
Descripción	Seleccionar un solo elemento insertado previamente en el canvas y esperar a que aparezca el menú para poder cambiar su posición en el índice z. Seleccionar cualquiera de las opciones.
Resultado	El elemento cambia su profundidad en el canvas, es decir, se posiciona atrás o enfrente de algún elemento específico, o se posiciona hasta el fondo o al frente de todos los elementos que en ese momento están en el canvas
Tiempo de respuesta	El tiempo de respuestas varía dependiendo de la cantidad de elementos que se encuentre y de la opción seleccionada. En el caso de enviar atrás o delante de un solo elemento, es decir cambiar su posición en una sola unidad, el tiempo de respuesta máximo es de 2ms. Este tiempo fue calculado con un máximo de 35 elementos. En el caso de enviar al fondo o al frente de todos los elementos, el tiempo máximo de respuesta es de 8ms. Este tiempo fue calculado con un máximo de 8 ms.

Tabla 68: Resultados de caso de prueba Cambiar profundidad de un elemento en el canvas

Caso de prueba: Rotar un elemento en el canvas	
Descripción	Seleccionar un solo elemento insertado previamente en el canvas y seleccionar la herramienta para cambiar su inclinación.
Resultado	El elemento cambia su inclinación dependiendo de la posición en la que esté el puntero en relación al elemento. El elemento se inclina un determinado número de grados.
Tiempo de respuesta	El tiempo de respuesta máximo es de 3ms para cualquier elemento en todos los navegadores.

Tabla 69: Resultados de caso de prueba Rotar un elemento en el canvas

Caso de prueba: Ampliación de canvas

Descripción	Seleccionar en un punto arbitrario del canvas y esperar a que se dibujen sus propiedades en el editor de propiedades, editar dichas propiedades y esperar a ver los cambios realizados en las dimensiones del canvas.
Resultado	El canvas aumenta sus dimensiones y los valores son los que el usuario introdujo en el editor de propiedades.
Tiempo de respuesta	El tiempo de respuesta máximo es de 3ms en todos los navegadores.

Tabla 70: Resultados de caso de prueba Ampliación de canvas

Caso de prueba: Mostrar propiedades de elemento seleccionado	
Descripción	Seleccionar un solo elemento insertado previamente en el canvas y esperar a que se muestren las propiedades que soporta en la barra de propiedades.
Resultado	El elemento se marca como seleccionado y se muestran todas las propiedades que soporta
Tiempo de respuesta	El tiempo de respuesta máximo es de 4 ms. El tiempo varía dependiendo del elemento, ya que algunos elementos tienen más propiedades que otros.

Tabla 71: Resultados de caso de prueba Mostrar propiedades de un elemento seleccionado

Caso de prueba: Editar propiedades en el editor de propiedades	
Descripción	Seleccionar un solo elemento insertado previamente en el canvas y esperar a que se muestren las propiedades que soporta en la barra de propiedades, editar propiedades arbitrariamente.
Resultado	El elemento cambia dependiendo de la propiedad que se esté modificando de dicho elemento. El cambio se muestra en el canvas.
Tiempo de respuesta	El tiempo de respuesta máximo es de 2 ms para cualquier elemento. El tiempo es completamente arbitrario y no se pudo determinar el factor del cual depende dicho tiempo de respuesta.

Tabla 72: Resultados de caso de prueba Editar propiedades en el editor de propiedades

Caso de prueba: Insertar fotograma	
Descripción	Seleccionar alguno de los fotogramas en la línea del tiempo y seleccionar la opción “insertar keyframe”
Resultado	Se muestra un punto que indica que se ha insertado un fotograma en ese punto de la

	línea del tiempo
Tiempo de respuesta	El tiempo de respuesta máximo es de 1 ms para todos los navegadores.

Tabla 73: Resultados de caso de prueba Insertar fotograma

Caso de prueba: Eliminar frame	
Descripción	Seleccionar alguno de los fotogramas insertados en la línea del tiempo en cualquiera de las capas y dar click en la opción eliminar frame
Resultado	Se elimina el punto que marcaba la existencia de un fotograma en esa posición y se elimina toda la información relacionada con ese fotograma (elementos, animaciones etcétera)
Tiempo de respuesta	El tiempo de respuesta máximo es de 2 ms para todos los navegadores. Este tiempo no depende del número de elementos o animaciones presentes en el fotograma previamente insertado

Tabla 74: Resultados de caso de prueba Eliminar frame

Caso de prueba: Animar frame	
Descripción	Seleccionar alguno de los fotogramas insertados en la línea del tiempo en cualquiera de las capas y dar click en la opción animar
Resultado	Se crean todos los frames intermedios entre el último fotograma presente en la línea del tiempo y el fotograma animado. En dichos frames intermedios se muestra como los objectos se van modificando dependiendo del instante de tiempo en el que se encuentren.
Tiempo de respuesta	El tiempo de respuesta para este caso es muy variable. Depende del número de elementos que hayan sido modificados desde el fotograma inicial hasta el final y del número de propiedades que cambiaron entre dichos fotogramas. También presenta variaciones dependiendo del número de fotogramas intermedios que existen entre los fotogramas finales e inicial. El máximo tiempo de respuesta calculado es de 70ms, para un total de 20 elementos con un número de 5 a 10 propiedades por elemento.

Tabla 75: Resultados de caso de prueba Animar frame

Caso de prueba: Insertar capa en línea de tiempo	
Descripción	En la parte izquierda de la línea del tiempo dar click en el símbolo de marco y esperar a que se cree la interfaz gráfica de la nueva capa creada.
Resultado	Se crea una nueva capa con la misma cantidad de frames que la anterior y con las mismas opciones de inserción, eliminación y animación de fotogramas,.
Tiempo de respuesta	El tiempo de respuesta máximo es de 4 ms para todos los navegadores

Tabla 76: Resultados de caso de prueba Insertar capa en línea del tiempo

Caso de prueba: Eliminar capa en línea del tiempo	
Descripción	Seleccionar alguna de las capas en la línea del tiempo. En la parte izquierda de la línea del tiempo dar click en el símbolo de basura y esperar a que se elimine la capa seleccionada.
Resultado	Se elimina por completo toda la capa con sus elementos, animaciones y fotogramas en su totalidad.
Tiempo de respuesta	El tiempo máximo de respuesta es de 4 ms para todos los navegadores.

Tabla 77: Resultados de caso de prueba Eliminar capa en línea del tiempo

Caso de prueba: Reproducir animación de capa en línea de tiempo	
Descripción	Seleccionar alguna de las capas y dar click en el botón de “play” en la parte izquierda de la línea de tiempo y observar la animación.
Resultado	Se inicia la animación en el canvas.
Tiempo de respuesta	El tiempo de respuesta desde que se da click hasta que es totalmente iniciada la animación es de 4 ms máximo. Este tiempo no depende de ningún factor.

Tabla 78: Resultados de caso de prueba Reproducir animación de capa en la línea del tiempo

Caso de prueba: Pausar animación de capa en línea de tiempo	
Descripción	Seleccionar alguna de las capas y dar click en el botón de “play” en la parte izquierda de la línea de tiempo. Cuando la animación esté en curso, dar click en el botón “pausa” para detener la animación.
Resultado	Es pausada la animación y es posible reiniciar la animación desde el punto en el que se quedó.
Tiempo de respuesta	El tiempo de respuesta desde que se da click hasta que es totalmente iniciada la animación es de 4 ms máximo. Este tiempo no depende de

	ningún factor.

Tabla 79: Resultados de caso de prueba Pausar animación de capa en línea de tiempo

10 CONCLUSIONES Y TRABAJO A FUTURO

10.1 CONCLUSIONES

Se desarrolló una aplicación en la que se pueden crear animaciones que corren en los navegadores Web más populares, Dichas animaciones pueden ser realizadas sin la necesidad de escribir una sola línea de HTML, CSS o JavaScript. Fue posible abstraer la complejidad de crear animaciones Web tradicionalmente y crear una herramienta en la que se pueden desarrollar animaciones bajo el sencillo esquema de insertar elementos en un lienzo o canvas y editar sus propiedades en algún punto de tiempo durante la animación.

El código finalmente generado es resultado de procesar los cambios que el usuario de la aplicación ha hecho, y dicho producto es código HTML, CSS y JavaScript. Al ser código basado en tecnologías completamente estándar y entendibles por los navegadores Web más populares, es posible ocultar los problemas de interoperabilidad. También podemos concluir que es posible crear herramientas que exploten las posibilidades de los módulos de CSS relacionados con transformaciones y animaciones en la nueva versión de dicho estándar. Otro factor importante es que es posible ofrecer a los usuarios herramientas que no impliquen de conocimientos profundos de las tecnologías Web, haciendo más grande el rango de personas que pueden acceder a este tipo de tecnologías o desarrollos.

También se desarrolló un pequeño sitio web para la administración de proyectos o diseños y la asociación de dichos proyectos con usuarios individuales. El sitio web desarrollado es muy sencillo ya que la prioridad de este proyecto es la aplicación en el cliente y el desarrollo de las herramientas necesarias para hacer fácil crear animaciones, pero existen muchas posibilidades para ofrecer diversos servicios y hacer más cómoda la aplicación.

A continuación se presentan varias ideas que se podrían desarrollar en el futuro a partir de la aplicación.

Debido a que la aplicación puede tener muchos usos y existen funcionalidades que pueden hacer mucho más cómodo el uso de nuestra aplicación presentamos una descripción de algunas de las características propuestas para mejorar la aplicación en iteraciones futuras.

Actualmente la interfaz para descargar los archivos de un proyecto específico o una página es hasta cierto punto incómoda y lenta, por lo que la transferencia de esos archivos, se vería sumamente mejorada si se soportara sincronización de archivos haciendo uso del protocolo FTP y se implementara un cliente con dichas funcionalidades de transmisión de archivos.

Al considerar la creación de proyectos, es posible insertar el concepto de colaboración en equipo o de versión de un proyecto específico. Por lo que la implementación de un control de versiones ayudaría a soportar múltiples usuarios dentro de un solo proyecto y la incorporación de trabajo colaborativo, así como la recuperación de versiones anteriores de un cierto diseño o página.

Otra funcionalidad interesante es la inserción de elementos 3D y la animación de los mismos. La creación de elementos en 3D puede ser lograda mediante la utilización de WebGL o del elemento llamado canvas de HTML5.

Otra característica interesante es hacer de la aplicación una herramienta profesional para usuarios que conozcan de desarrollo Web y aprovechen en su totalidad el uso de las tecnologías que soporta el navegador. Para ello hay que implementar herramientas que permitan la edición directa del código generado y puedan agilizar el proceso de desarrollo de dichos desarrolladores que actualmente siguen.

11 GLOSARIO

Término	Descripción
DOM	Document Object Model. Es el modelo que rige la estructura de un documento HTML mostrado en un navegador Web.
MVC	Modelo, Vista, Controlador. Es un patrón de diseño que separa la lógica de la interfaz de usuario y las interacciones entre ambas.
Estructura	Se refiere a cualquier estructura de datos, clase, objeto o struct.
Modelo	Se refiere a clases bien definidas que contienen datos u operaciones en datos. Fungen como actores principales en el sistema.
Vista	Se refiere a una interfaz gráfica que se muestra al usuario. En este caso en HTML y CSS.
Controlador	Se refiere a lógica específica que sucede cuando el usuario interactúa con una vista. Efectivamente coordinando la interacción entre vistas y modelos.

Tabla 80: Glosario de términos de descripción detallada del servidor

12 BIBLIOGRAFÍA

[1] Crockford Douglas, JavaScript: The good parts. Sebastopol, 1st ed, CA: O'Reilly Media, 2008, 155p.

[2] Cederholm Dan. CSS 3 for Web designers. 1st ed, New York, New York: A Book Apart, 2010, 125p.

[3] Flanagan David, JavaScript The definitive Guide, 6th ed. Sebastopol, CA: O'Reilly Media. 2011, 1079p.

[4] Gasston Peter. The book of CSS 3. A developer's guide to the future of Web design. 1st ed, San Francisco, CA: No Starch Press, 2011, 290p.

[5] Keith Jeremy. HTML 5 for Web designers. 1st ed, New York, New York: A Book Apart, 2010, 89p.

[6] Lawson Bruce, Sharp Remy. Introducing HTML 5, 1st ed, Berkeley, CA: New Riders. 2011, 233p.