Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Centro de Investigación en Comunicaciones y Redes

Generador de Cursos Multimedia basado en ActionScript 3 con Soporte para Texto, Imágenes, Audio, Video y Autoevaluaciones

Trabajo Especial de Grado
presentado ante la Ilustre
Universidad Central de Venezuela
por la Bachiller:

Fiorella Carnero
C.I.: 82.124.418
E-mail: fiorecar@gmail.com

para optar al título de Licenciado en Computación

Tutor:
Prof. Eric Gamess

Caracas, Octubre 2015

Universidad Central de Venezuela

Facultad de Ciencias
Escuela de Computación
Centro de Investigación en Comunicaciones y Redes

ACTA DEL VEREDICTO

Quienes suscriben, Miembros del Jurado designados por el Consejo de la Escuela de Computación, para examinar el Trabajo Especial de Grado, presentado por la Bachiller Fiorella Patricia Carnero Ramírez, C.I.:82.124.418, con el título **"Generador de Cursos Multimedia basado en ActionScript 3 con soporte para Texto, Imágenes, Audio, Video y Autoevaluaciones"**, a los fines de cumplir con el requisito legal para optar al título de Licenciado en Computación, dejan constancia de lo siguiente:

Leído el trabajo por cada uno de los Miembros del Jurado, se fijó el día 14 de octubre del 2015, para que su autor lo defienda en forma pública, en el laboratorio de Internet II de la Escuela de Computación, mediante una exposición oral de su contenido, y luego respondieron satisfactoriamente a las preguntas que le fueron formuladas por el Jurado, todo ello conforme a lo dispuesto en la Ley de Universidades y demás normativas vigentes de la Universidad Central de Venezuela.

Finalizada la defensa pública del Trabajo Especial de Grado, el jurado decidió aprobarlo.

En fe de lo cual se levanta la presente Acta, en Caracas a los catorce días del mes de octubre del año dos mil quince, dejando constancia que el Profesor Eric Gamess actuó como Coordinador del Jurado.

_____

Prof. Eric Gamess

(Tutor)

_____	_____
Prof. Roger Bello	Prof. Miguel Astor
(Jurado Principal)	(Jurado Principal)

RESUMEN

Título:
Generador de Cursos Multimedia basado en ActionScript 3 con soporte para Texto, Imágenes, Audio, Video y Autoevaluaciones.

Autor:
Fiorella Carnero

Tutor:
Prof. Eric Gamess

El presente Trabajo Especial de Grado consistió en el diseño e implementación de un generador de cursos multimedia basado en ActionScript 3, los cuales no dependen de una conexión a Internet activa, e integran diversos recursos multimedia, tales como texto, imágenes, audio y video, así como autoevaluaciones con el fin de verificar que el estudiante haya cumplido los objetivos de cada capítulo.

La necesidad de este generador surge debido a que no existe una solución que permita proveer cursos gratuitos a estudiantes de bajos recursos que no tengan fácil acceso a una conexión a Internet estable y segura. Estos cursos permitirán a los estudiantes complementar las actividades en el salón o estudiar un tema por su propia cuenta, sin necesidad de la supervisión activa de un profesor o instructor.

El generador propuesto se diseñó de forma tal que pueda ser utilizado para crear cursos de un amplio rango de tópicos, debido a que el contenido del curso está almacenado en un lenguaje de etiquetas personalizado basado en la especificación XML (Extensible Markup Language). De esta forma, se pueden crear múltiples cursos sobre diversos temas escribiendo el contenido de los mismos en el lenguaje de etiquetas definido para el generador. De forma similar, este lenguaje de etiquetas no está acoplado al generador, lo cual permitiría utilizar otras tecnologías para crear otro generador que opere sobre el contenido de cursos preexistentes, sin necesidad de modificar los documentos XML de éstos.

Para validar el generador desarrollado, se instanció un curso para la enseñanza del lenguaje de programación C.

Palabras Claves:
Generador, XML, Aprendizaje a Distancia, Educación Multimedia, ActionScript.

Tabla de Contenido

Índice de Figuras

Índice de Tablas

1. Introducción

No cabe duda que las técnicas utilizadas para la enseñanza están evolucionando de una manera vertiginosa. Hace algunos años, la enseñanza era dictada únicamente a través de clases presenciales utilizando el clásico pizarrón. En la actualidad, los profesores se apoyan en una diversidad de herramientas tecnológicas, tales como: transparencias o diapositivas, software para realizar simulaciones, aplicaciones multimedia interactivas, entre otros.

En los últimos 3 años se han desarrollado múltiples plataformas educativas en línea que hospedan cursos que permiten a los estudiantes reforzar los contenidos expuestos en clase y estudiar diversos tópicos por su propia cuenta. Estas plataformas tienen la ventaja de ser en su mayoría gratuitas y estar disponibles en varios idiomas; sin embargo, todas las plataformas disponibles en la actualidad requieren acceso estable a Internet, lo cual dificulta su uso para un gran porcentaje de los estudiantes en el mundo (61% a nivel mundial, 45% en Venezuela).

Debido a estas razones, un gran porcentaje de los estudiantes a nivel mundial no tienen acceso confiable a cursos multimedia que le permitan aprender nuevos tópicos por su cuenta o reforzar los contenidos expuestos en el salón de clases. En este Trabajo Especial de Grado se presenta el desarrollo de un generador de cursos multimedia independientes del acceso a Internet que pretende solventar estas carencias.

1.1. Planteamiento del Problema

En la actualidad, existe una gran variedad de plataformas educativas para desarrollar cursos en línea tanto gratuitos como comerciales, pero ninguna de estas plataformas permite generar un curso independiente del acceso a Internet. Estas plataformas no pueden ser utilizadas fácilmente por los estudiantes de bajos recursos sin fácil acceso a una conexión estable a Internet, lo cual los coloca en una amplia desventaja con respecto al resto de los estudiantes con mayores recursos.

Debido a esto, se plantea el diseño y la creación de una herramienta que permita generar cursos independientes del acceso a Internet que sea de fácil uso por parte de los profesores e instructores. Estos cursos serán distribuidos en DVDs (Imágenes ISO) e incluirán diversos recursos multimedia para facilitar la comprensión del contenido, y proveerán autoevaluaciones con el fin de verificar que el estudiante haya cumplido los objetivos del mismo.

1.2. Objetivo General

Desarrollar una herramienta que permita generar cursos multimedia de diversas disciplinas que puedan ser utilizados sin acceso a Internet.

1.3. Objetivos Específicos

- Definir un lenguaje de etiquetas personalizado basado en la especificación XML para describir cursos sobre cualquier tópico.
- Proveer acceso a recursos multimedia que permitan un fácil aprendizaje de cualquier tópico, como por ejemplo archivos de audio y video con ejemplos que permitan ilustrar el contenido del curso generado.
- Proveer acceso a herramientas de evaluación fuera de línea para que el estudiante pueda determinar si ha comprendido el contenido del curso generado.
- Implementar una aplicación que dado un curso descrito en el lenguaje de etiquetas definido anteriormente genere una imagen ISO que pueda ser distribuida fácilmente.
- Permitir la generación de diversos cursos sin necesidad de modificar la aplicación desarrollada, encapsulando el contenido del curso en archivos de dato separados del resto de la herramienta.
- Desarrollar un curso prototipo que muestre las capacidades del lenguaje de etiquetas diseñado y el generador implementado.

1.4. Justificación de la Propuesta

Actualmente todas las herramientas de aprendizaje disponibles requieren acceso estable a Internet para ser utilizadas, lo cual es problemático para estudiantes con pocos recursos que no dispongan de este acceso en sus hogares. Una herramienta para generar cursos multimedia fuera de línea agilizará la creación de éstos por parte de los profesionales docentes, facilitando el acceso al conocimiento a todos los estudiantes, incluyendo a aquellos con pocos recursos.

1.5. Alcance

Los cursos generados deben ser autocontenidos de forma que no sea necesario el acceso a Internet para poder utilizarlos.

El generador de cursos debe leer el contenido de los cursos a generar, así como los recursos multimedia para soportarlos sin necesidad de modificar el código fuente del mismo. De esta forma los profesores e instructores no necesitarán aprender ActionScript 3 para generar nuevos cursos; será suficiente que aprendan el lenguaje de etiquetas esperado por el generador para desarrollar cursos sobre otros tópicos.

El lenguaje de etiquetas para describir los cursos debe ser independiente del generador desarrollado en este Trabajo Especial de Grado; de esta forma será posible implementar otro generador en un lenguaje de programación diferente sin necesidad de cambiar todos los cursos desarrollados anteriormente.

El resto del presente documento está estructurado por capítulos, cuyos contenidos se especifican a continuación:

- En el Capítulo 2 se presentan los trabajos relacionados, en el cual se describe el estado actual de las plataformas que ofrecen cursos en línea abiertos masivos (MOOC), incluyendo los principales proveedores a nivel mundial.
- En el Capítulo 3 se presentan las tecnologías y herramientas que se utilizaron para diseñar el lenguaje de etiquetas y desarrollar el generador de cursos.
- En el Capítulo 4 se presenta el marco metodológico, el cual corresponde al proceso de desarrollo empleado para cumplir a cabalidad los objetivos planteados.
- En el Capítulo 5 se presenta el marco aplicativo, el cual corresponde a las etapas del proceso de desarrollo utilizado.
- En el Capítulo 6 se presenta el contenido teórico del curso instanciado.
- En el Capítulo 7 se presenta la especificación del lenguaje de etiquetado XML para la creación de cursos, utilizando como ejemplo el curso prototipo instanciado.
- En el Capítulo 8 se presentan las conclusiones y se listan los posibles trabajos futuros.

2. Trabajos Relacionados

En este capítulo se define el término MOOC, se presentan sus características más relevantes, se define qué es una plataforma MOOC así como ciertos lineamientos a seguir a la hora de desarrollar el contenido de un MOOC. Finalmente se exponen algunas de las plataformas MOOC más conocidas a nivel mundial.

2.1. Cursos Online Abiertos Masivos (MOOC)

MOOC es el acrónimo en inglés de *Massive Open Online Course*, pero más allá de ser un curso en línea, abierto y masivo, un MOOC es una innovación educativa que combina y amplía las capacidades de las tecnologías existentes, basada en una serie de enfoques de *e-learning* para ofrecer un nuevo producto educativo en una forma innovadora hacia mercados que no han sido abarcados previamente. Estos cursos en línea en su mayoría son ofrecidos sin costo a cualquier participante que desee inscribirse [6].

Massive se refiere a la capacidad que tienen los MOOCs de acomodar grandes cantidades de estudiantes, mucho más allá de los que pueden ser acomodados en un salón de clases. No hay un número asociado con masivo. El término MOOC fue acuñado en 2008 por Dave Cormier[1] en respuesta a un curso llamado “Connectivism and Connective Knowledge”, el cual tuvo una duración de 12 semanas, donde se registraron más de 2.200 participantes de todo el mundo. Los MOOCs más recientes han atraído más de 150.000 estudiantes [8].

La capacidad de los MOOCs de ser masivos refleja desarrollos en las Tecnologías de Información y Comunicación (TIC) así como en la pedagogía en línea y aprendizaje a distancia. Los avances más relevantes en las TICs incluyen:

- Infraestructura y servicios para almacenar, indexar y acceder remotamente grandes cantidades de contenido digital, como por ejemplo: YouTube, Google Books, bibliotecas digitales, procesamiento en la nube.
- Registro, identificación y autenticación segura de un amplio número de usuarios de forma simultánea.
- Software y servicios robustos, confiables y seguros que sean capaces de servir múltiples usuarios de forma simultánea a gran escala.

La capacidad de servir una gran cantidad de estudiantes de forma simultánea tiene implicaciones significativas para la pedagogía en los MOOCs, los cuales serán discutidos en la Sección 2.3.

Open significa abierto, pero tiene diferentes interpretaciones en el ámbito de los MOOCs. Las interpretaciones más comunes se definen a continuación:

- Acceso abierto: El curso está disponible a estudiantes de todas las edades, sin importar cualquier diferencia intelectual o física entre ellos, o su inscripción en alguna institución específica. Algunos cursos sugieren que los estudiantes deben

[1] http://davecormier.com/edblog/2008/10/02/the-cck08-mooc-connectivism-course-14-way

tener ciertos conocimientos previos, pero este requisito no es estrictamente verificado.

Otra interpretación de acceso abierto se refiere a que no se cobra por ningún aspecto del curso, incluyendo el reconocimiento por haberlo culminado satisfactoriamente. Algunos cursos ofrecen certificados de culminación calificados por un costo adicional, los cuales usualmente mencionan el profesor y la universidad de origen.

- Educación abierta: El curso aplica políticas y prácticas para incrementar el acceso a la educación formal a estudiantes que enfrenten barreras físicas, cognitivas, geográficas o temporales para participar, las cuales son implementadas por universidades abiertas u otras instituciones educativas.
- Filosóficamente abierto: El curso adopta principios similares a los del código abierto (*Open Source*) con respecto a los derechos de reproducción y reutilización del material educativo y al código fuente para los sistemas de información que dan soporte al curso. Este tipo de apertura se puede ver en el uso de estándares y formatos abiertos para programar, almacenar y compartir recursos y datos de aprendizaje, los cuales son generalmente conocidos como Recursos Educacionales Abiertos (u OER, por sus siglas en inglés, *Open Educational Resources*).
- Itinerario abierto: Los estudiantes pueden tomar el curso en cualquier momento, en cualquier fecha por el período de su elección. Sin embargo, no todos los MOOCs proveen esta flexibilidad; los cursos ofrecidos por universidades por lo general restringen la disponibilidad de sus MOOCs al tiempo que el curso es dictado para los estudiantes que asisten a clases en persona, con el objetivo de reducir la carga sobre el profesor.

Online se refiere a que el curso es a distancia, utilizando una conexión a Internet como principal medio de comunicación. El primer curso considerado MOOC tuvo 2200 participantes, de los cuales solamente 24 tuvieron acceso presencial a las clases, mientras que el resto únicamente tuvo acceso al material educativo a través de Internet. Muchos MOOCs provistos por universidades todavía tienen un curso equivalente en el salón, pero sólo algunos de estos cursos son ofrecidos simultáneamente a participantes registrados y no registrados en la universidad.

Sin importar si el MOOC es provisto simultáneamente con un equivalente en el salón o no, la mayoría de la participación en línea es asíncrona: los estudiantes eligen el momento apropiado para acceder al contenido y seguir las actividades del curso, en lugar de atender a lecturas y reuniones en vivo de forma remota. Los lugares y momentos en los cuales partes del MOOC se llevan a cabo afectan los tipos de actividades de aprendizaje pueden ser utilizadas. Por ejemplo, una sesión de preguntas y respuestas con el profesor sólo se puede realizar de forma síncrona, ya que es necesaria la atención del mismo durante la actividad; pero observar una demostración de un procedimiento se puede realizar de forma asíncrona, ya que los estudiantes pueden acceder a esta sin necesidad de la presencia activa del profesor.

Debido a que los MOOCs son en línea y abiertos, estos pueden ser utilizados como recursos educacionales abiertos para cursos presenciales como complemento. Incluso, es posible utilizar un MOOC para convertir un curso presencial en un curso de aprendizaje mixto, donde parte del aprendizaje ocurre en línea, en el MOOC, y otra parte ocurre cara a cara en el salón. Una adaptación específica de este modelo es el "salón invertido" (o *flipped classroom* en inglés), dónde los participantes estudian el MOOC en casa y las clases presenciales son utilizadas para aclarar y resolver dudas y dificultades que surgen de las asignaciones en el MOOC.

Un informe del Departamento de Educación de los Estados Unidos halló que los cursos que incluyen educación en línea (ya sean completamente virtuales o semipresenciales) producen, en promedio, resultados mucho más sólidos de aprendizaje en los estudiantes que los cursos brindados exclusivamente de manera presencial [9].

Course significa curso, y se refiere a una secuencia de actividades orientadas al aprendizaje, que suele conllevar una serie de ejercicios y evaluaciones para acreditar el conocimiento adquirido. El curso debe ser diseñado, desarrollado, ejecutado, evaluado y revisado; el esfuerzo implicado en estas tareas es considerable, particularmente cuando ellas son abiertas en línea a un número masivo de participantes de diversas habilidades y orígenes. El lanzamiento de un MOOC implica un costo sustancial y un gran equipo altamente calificado [3].

2.2. Plataformas MOOC

El término plataforma MOOC es utilizado para describir el ambiente de software que asiste a los diseñadores, desarrolladores e instructores en proveer el contenido del curso, y que permite a los estudiantes participar en él. En un sentido más simple, una plataforma MOOC es el software que permite desarrollar un curso y participar en él.

Existen plataformas MOOC tanto propietarias como código abierto. Las plataformas propietarias más conocidas en la actualidad son aquellas desarrolladas por Coursera y Udacity. Plataformas abiertas incluyen Course Builder de Google, el cual está incorporado en Open EdX.

Las plataformas MOOC por lo general ofrecen las siguientes funcionalidades:

- Administración de identidad de acceso o una interfaz transparente para utilizar un sistema de administración de identidad ya existente a nivel institucional, para el control de acceso al MOOC.
- Sistema de administración de contenidos o un conjunto de servicios que identifique y administre o referencie recursos de aprendizaje en diferentes ubicaciones en diversos formatos.
- Un generador de quices o una interfaz que permita utilizar quices o actividades de evaluación externas.
- Un foro o blog para intercambiar ideas o una interfaz hacia un sistema equivalente externo.
- Servicios que provean enlaces a recursos externos en el Internet.

- Un ambiente de desarrollo de cursos o una interfaz a una herramienta externa para realizar esta actividad.
- Una interfaz para el curso a ser utilizada por el profesor.
- Una interfaz para el curso a ser utilizada por los estudiantes.
- Servicios que identifiquen y entreguen recursos de aprendizaje a los estudiantes cuando sea requerido.
- Herramientas administrativas para reportar y habilitar métricas de análisis sobre el acceso por parte de los estudiantes así como el progreso actual por parte de ellos.

Los recursos de aprendizaje de los MOOCs por lo general, son desarrollados utilizando herramientas externas a la plataforma MOOC. Una variedad de herramientas externas para producir grabaciones de audio, video, captura de pantalla y otros elementos multimedia son utilizadas para crear el contenido del curso.

2.3. Lineamientos para Desarrollar un MOOC

El carácter masivo, abierto, y en línea de los MOOCs presenta múltiples desafíos y limitaciones a los diseñadores del curso a la hora de preparar el contenido del mismo. Los MOOCs son cursos en línea influidos por la tecnología de la información y comunicaciones. Los tipos de materiales y actividades de aprendizaje incluidos en un curso están limitadas por las capacidades de la tecnología y las acciones que los educadores que pueden implementar utilizando la misma. La interfaz del MOOC para el profesor establece las capacidades de la plataforma, independientemente de la intención que tuvo el desarrollador de la misma durante la implementación. Debido a que el MOOC es abierto a cualquier estudiante, en cualquier lugar, en casi cualquier momento, se requiere una consideración cuidadosa de los objetivos de aprendizaje que se desean alcanzar, así como de la secuencia y ritmo adecuado a seguir, entre otros aspectos de diseño del curso.

Entre las decisiones pedagógicas que un profesor y un diseñador de cursos MOOC deben tomar son:

- Propósito y Audiencia:
  - La meta del curso.
  - Objetivos de aprendizaje.
  - Nivel del curso (teniendo en cuenta el carácter abierto de la matrícula).
- Horario, ritmo y esfuerzo requerido por el curso:
  - Fechas de inicio y culminación, o si es abierto o fijo relativo a semestre en un instituto u universidad de acuerdo a la disponibilidad del profesorado.
  - Duración del curso: Actualmente, el rango de los MOOCs va desde 2 o 3 semanas hasta 15 o 26 semanas de duración.

- Ritmo: Las opciones generalmente incluyen dejarlo a elección del estudiante, establecer actividades semanales con o sin fechas tope fijas, y una combinación de estas opciones. Muchos cursos ofrecidos por universidades involucran actividades semanales con fechas tope, particularmente si el curso puede ser evaluado.
- Esfuerzo esperado por parte del estudiante: Se debe estimar el número de horas que el estudiante debe invertir en cada etapa o en cada actividad. Esto es importante para la planificación y preparación del estudiante, así como para la acreditación de los MOOCs donde créditos y transferencia de créditos son concedidas.

- Estructura del Curso: Debido a que cada componente de aprendizaje necesita ser pre-diseñado y, usualmente, pre-empacado, la tendencia es utilizar diseños basados en objetivos e lugar a diseños basados en tópicos. Cada sesión se define con base en los conocimientos que los estudiantes deben adquirir, y el contenido y las actividades de aprendizaje son dirigidas hacia los objetivos de aprendizaje de cada sesión. Estos objetivos por lo general son reforzados mediante el uso de quices y asignaciones.
- Contenido del Curso:
  - Multimedia: El contenido del MOOC generalmente está basado en videos cortos de 5 a 10 minutos de duración. Estos videos usualmente incluyen una combinación de diagramas, presentaciones y la persona que dicta el curso, lo cual hace que sean más atractivos para los estudiantes.
  - Propiedad Intelectual: Garantizar que se tienen los derechos del material desarrollado por otras entidades, y decidir sobre la propiedad intelectual y protección de derechos para el material nuevo creado por el desarrollador del MOOC.
  - Ejercicios y otras actividades de aprendizaje activas.
  - Quices y asignaciones: ¿Hasta qué punto serán integrados con videos o se mantendrán separadas de los mismos?.
  - Secuencia: Puede ser lineal, o se puede permitir a los estudiantes ramificarse a diversos contenidos durante todo el curso, o en diferentes etapas. La mayoría de los MOOCs actualmente ofrecen solo secuencia lineal.
- Interacción del Diseño: La interacción efectiva en foros y en discusiones requiere un diseño de preguntas y una estructura enfocada a fomentar la discusión, en lugar de un enfoque más liberal.
- Evaluaciones: A la hora de diseñar las evaluaciones el desarrollador del MOOC debe responder las siguientes preguntas:
  - ¿Qué va a ser evaluado y cuándo?.
  - ¿Se le permitirá a un estudiante repetir una evaluación?.

- ¿Qué combinación de evaluaciones será utilizada: evaluación basada en computador, evaluaciones por pares o autoevaluaciones?.
- ¿Los estándares de evaluación serán moderados externamente?.
- ¿Será verificada la identidad de los estudiantes para acreditación?.
- ¿La verificación será en línea o requerirá presencia física en algún centro de evaluaciones?.

Dado el auge que ha experimentado este tipo de cursos, aparecieron distintas iniciativas privadas, siendo los principales proveedores Coursera, Udacity, EdX, entre otros. A continuación se presenta una breve descripción de cada uno de ellos.

2.4. Coursera

Coursera[2] es una plataforma educativa asociada con las universidades y organizaciones más renombradas de todo el mundo. Por medio de esta colaboración, ofrece cursos gratuitos en línea que cualquier estudiante en el mundo puede tomar.

Esta plataforma fue fundada por profesores de Ciencias de la Computación (Daphne Koller y Andrew Ng) de la Universidad de Stanford. En otoño del 2011 lanzan sus dos primeros MOOCs, “Introduction to DataBases” y “Machine Learning”. Este último tuvo una alta receptividad con más de 100 mil estudiantes inscritos.

Coursera actualmente ofrece más de 1016 cursos de 117 instituciones, de los cuales 839 cursos son impartidos en idioma inglés, 65 cursos son impartidos en chino, 55 cursos son impartidos en español, entre otros.

Muchas de las instituciones asociadas hacen uso de la plataforma en línea para maximizar la experiencia de aprendizaje de sus estudiantes presenciales. Este modelo mixto de aprendizaje ha arrojado, en estudios realizados sobre el tema, un aumento en el compromiso de los estudiantes, así como en su continuidad y desempeño.

2.5. Udacity

Udacity[3] es una organización educacional con fines de lucro fundada por Sebastian Thrun, David Stavens, y Mike Sokolsky que ofrece Cursos Online Abiertos Masivos (MOOC).

Udacity nació de un experimento de la Universidad de Stanford en la que Sebastian Thrun y Peter Norvig ofrecieron su curso en línea “Introducción a la Inteligencia Artificial”, de forma gratuita. Se registraron más de 160 mil estudiantes en más de 190 países.

[2] https://www.coursera.org
[3] https://www.udacity.com/nanodegree

2.6. EdX

EdX[4] es una organización educativa abierta sin fines de lucro fundada en el 2012 por la Universidad de Harvard y el Massachusetts Institute of Technology (MIT) que ofrece clases online interactivas y MOOCs de las mejores instituciones, universidades y organizaciones del mundo. Los temas incluyen biología, negocios, química, ciencias de la computación, economía, finanzas, electrónica, ingeniería, alimentación y nutrición, historia, humanidades, derecho, literatura, matemáticas, música, filosofía, física, estadística, entre otros.

Su curso prototipo, Circuitos y Electrónica, enseñado por Anant Agarwal, se lanzó en diciembre 2011 a través de MITx, el programa online y masivo del MIT, el cual atrajo a 155.000 estudiantes de 162 países. Actualmente, Anant Agarwal es el Director Ejecutivo de EdX.

EdX difiere de otras plataformas MOOC, tales como Coursera y Udacity, en que ésta es una organización sin fines de lucro y corre en una plataforma de código abierto llamada Open EdX.

EdX hospeda cursos online de nivel universitario de un amplio rango de disciplinas para todo el mundo, sin costo, para propiciar la investigación y el aprendizaje.

2.7. Moodle

Moodle[5] es una plataforma de aprendizaje diseñada para proporcionar a educadores, administradores y estudiantes un sistema único, robusto y seguro para crear entornos de aprendizaje personalizados. Moodle es un sistema para el manejo del aprendizaje en línea, de distribución libre, que les permite a los educadores la creación de sus propios sitios web privados, llenos de cursos dinámicos que extienden el aprendizaje, en cualquier momento, en cualquier sitio.

Moodle proporciona un conjunto de herramientas flexibles para soportar tanto el aprendizaje mixto como el aprendizaje en línea. Debido a su flexibilidad, Moodle ha sido adoptado para usarse en distintos ámbitos incluyendo: educación, negocios, organizaciones no lucrativas, contextos comunitarios, entre otros.

Actualmente, Moodle se utiliza en más de 230 países con más de 71 millones de usuarios en todo el mundo, lo que lo convierte en una de las plataformas de aprendizaje más ampliamente utilizadas a nivel mundial.

En septiembre del 2013, Moodle lanza su primer MOOC oficial “Teaching with Moodle: An Introduction”, el cual introdujo a más de 9000 participantes a las características básicas de Moodle. El MOOC se repitió en enero del 2015 y se continuará realizando cada seis meses. Learn Moodle[6] es el sitio web oficial de Moodle para la entrega de sus MOOCs.

[4] https://www.edx.org
[5] https://moodle.org
[6] https://learn.moodle.net

2.8. CourseSites

CourseSites[7] es una plataforma de aprendizaje en línea gratuita e interactiva que permite crear cursos en línea y ofrece a profesores de primaria, secundaria, universitarios y educadores la posibilidad de añadir un componente basado en web a sus cursos o incluso hospedar un curso completo en Internet. Puede elegir incluso su propia URL, para que así los estudiantes puedan encontrar su página con facilidad.

Blackboard Learn es un entorno de aprendizaje y sistema de gestión de cursos desarrollado por Blackboard Inc. CourseSites está diseñado específicamente para el profesor individual. CourseSites incorpora la tecnología más reciente de Blackboard Learn, junto con elementos de Blackboard Mobile, Blackboard Collaborate y Blackboard Connect. La mayoría de las características principales de estas tecnologías están disponibles de forma gratuita para su uso por parte de los profesores y de los estudiantes de todos los niveles. En la actualidad, CourseSites utiliza la versión 9.1 de Blackboard Learn.

2.9. Lynda

Lynda.com[8] es una empresa líder en el aprendizaje en línea que ayuda a cualquier persona a aprender de negocios, software, tecnología y habilidades creativas para alcanzar metas personales y profesionales. A través de las suscripciones individuales, corporativas, académicas y gubernamentales, los miembros tienen acceso a la extensa biblioteca de vídeo tutoriales de alta calidad, que abarca cientos de miles de vídeos impartidos por reconocidos expertos en la industria. La compañía también ofrece un amplio contenido de video tutoriales en alemán, francés y español.

Lynda.com, Inc. fue fundada por Lynda Weinman en 1995 y tiene su sede en Carpintería, California, con oficinas en San Francisco, Londres, Sydney y Graz. Actualmente cuenta con más de 500 empleados a tiempo completo y más de 140 instructores. El 7 de Abril del 2015, Lynda.com fue adquirida por Linkedin por $ 1.5 mil millones.

Tanto Lynda.com como LinkedIn son utilizados por organizaciones pequeñas y grandes que hacen un gran énfasis en el talento humano, ya sea adquiriéndolo, entrenándolo o reteniéndolo. El contenido de los video tutoriales de lynda.com se utilizan a través de instituciones académicas y empresas por igual para ayudar a entrenar a sus empleados y para educar a sus estudiantes.

[7] https://www.coursesites.com
[8] http://www.lynda.com

2.10. CBT Nuggets

CBT Nuggets[9] ofrece videos de entrenamiento bajo demanda para profesionales de las Tecnologías de la Información. CBT Nuggets ofrece formación innovadora en Tecnologías de la Información a través de vídeos de alta calidad en línea bajo demanda.

Los suscriptores pueden entrenar 24 horas del día, 7 días a la semana desde la comodidad de un computador o dispositivo móvil. Sus aplicaciones móviles están disponibles en iTunes App Store, Kindle App Store y Google Store. Cuentas Premium pueden descargar y reproducir videos sin requerir acceso a Internet.

Entre los cursos en línea para exámenes de certificación que ofrece CBT Nuggets se encuentran los siguientes: Cisco, CompTIA, Microsoft, Linux, Amazon Web Services, VMware y Juniper.

2.11. TrainSignal

TrainSignal[10] fue fundada por Scott Skinger en el 2002. TrainSignal provee cursos de entrenamiento basados en computadores llamados "Total Experience", los cuales brindan a los estudiantes las herramientas y la confianza necesaria para afrontar los desafíos del mundo real, aprobar su examen de certificación y tener éxito en el competitivo mercado global de Tecnologías de Información.

TrainSignal ofrece una amplia gama de paquetes de capacitación en informática, entre ellos: Microsoft, Cisco, CompTIA, VMware y Microsoft Office. El 6 de agosto del 2013, TrainSignal fue adquirido por Pluralsight por $23,6 millones.

2.12. Pluralsight

Pluralsight[11] es una empresa de educación en línea privada que ofrece una variedad de cursos de entrenamiento para los desarrolladores de software, los administradores de Tecnologías de la Información y los profesionales creativos a través de su sitio web.

Pluralsight fue fundada en 2004 por Aaron Skonnard (actual Director Ejecutivo), Keith Brown, Fritz Cebolla y Bill Williams, la empresa tiene su sede en Farmington, Utah. En septiembre de 2014, declaró que utiliza más de 600 expertos en la materia como autores del curso, ofrece más de 3,700 cursos en su catálogo, en más de 150 países alrededor del mundo.

[9] https://www.cbtnuggets.com
[10] http://www.pluralsight.com/tag/trainsignal?pageSize=48&sort=new
[11] http://www.pluralsight.com

3. Tecnologías y Herramientas Utilizadas

En este capítulo se presentan las tecnologías y herramientas utilizadas para el desarrollo del generador y el diseño del lenguaje de etiquetas para describir cursos a ser utilizados por el mismo.

3.1. Extensible Markup Language (XML)

Lenguaje de Etiquetas Extensible, o XML por sus siglas en inglés, es una especificación para almacenar datos y para describir la estructura de la misma, derivado del *Standard Generalized Markup Language* (SGML). XML fue desarrollado por el *World Wide Web Consortium* (W3C), debido a que la cantidad de datos disponibles a través de Internet ha pasado a ser prácticamente incontable, los cuales están, en su mayoría, escritos en *HyperText Markup Language* (HTML), una forma sencilla pero elegante de mostrar datos en un navegador Web. La simplicidad de HTML ha contribuido a impulsar la popularidad de la Web, pero se han presentado limitaciones reales a la hora de representar grandes volúmenes de información disponible en Internet [5].

XML ha tomado su lugar junto a HTML como lenguaje fundacional en Internet, el cual es frecuentemente utilizado para almacenar datos y para transmitirlos entre diversas clases de sistemas y aplicaciones. Esta popularidad se debe a que XML fue diseñado para almacenar, manejar y transportar datos, mientras que HTML fue diseñado principalmente para mostrar información al usuario.

A diferencia de HTML, XML no es un lenguaje por sí mismo, sino un conjunto de reglas para definir lenguajes de etiquetas personalizados. Esta especificación permite a un grupo de desarrolladores definir su propio lenguaje de etiquetas a ser utilizado por una aplicación o sistema. Luego, otros desarrolladores pueden definir datos para esta aplicación o interpretar datos obtenidos de la misma utilizando el lenguaje de etiquetas personalizado definido por ésta. Por ejemplo, se podrían especificar lenguajes de etiquetas para describir datos genealógicos, químicos, o de negocios, los cuales se utilizarían para escribir documentos XML en cada uno.

Al no ser un lenguaje específico, XML no define etiquetas o atributos propios como HTML. En lugar de esto, XML permite al desarrollador definir cualquier etiqueta o atributo que necesite para representar los datos manejados por su aplicación o sistema. La única condición es que estas etiquetas y atributos se adhieran a las reglas de la especificación XML.

Por ejemplo, se puede definir un lenguaje de etiquetas para describir los componentes que tiene una bicicleta (ver Figura 3.1).

bicicleta.xml

```xml
<?xml version='1.0' encoding='utf-8'?>

<bicicleta>
    <marco>
        <material>Aluminio</material>
        <tamaño unidad="cm">56</tamaño>
        <color>Azul</color>
    </marco>
    <rueda eje="delantero">
        <material>Acero</material>
        <diámetro unidad="mm">700</diámetro>
        <ancho unidad="mm">23</ancho>
    </rueda>
    <rueda eje="trasero">
        <material>Acero</material>
        <diámetro unidad="mm">700</diámetro>
        <ancho unidad="mm">25</ancho>
    </rueda>
    <asiento>
        <acolchado>Suave</acolchado>
        <ancho unidad="mm">155</ancho>
    </asiento>
    <pedales>
        <color>Blanco</color>
    </pedales>
</bicicleta>
```

Figura 3.1: Archivo XML que describe los componentes de una bicicleta

En este caso, este lenguaje de etiquetas permite describir una bicicleta con sus componentes, los cuales se deben definir de la siguiente manera:

- La etiqueta “bicicleta” se utiliza para describir una bicicleta en específico. Este elemento debe tener exactamente dos ruedas (una en el eje delantero y una en el eje trasero), un marco, un manubrio, un asiento y los pedales.
- La etiqueta “marco” se utiliza para describir el marco de la bicicleta. Este elemento debe tener un material y un tamaño.
- La etiqueta “rueda” se utiliza para describir una rueda en la bicicleta. Este elemento debe tener un material, un diámetro y un ancho.
- Las etiquetas “diámetro”, “ancho” y “tamaño” se utilizan para describir una medida, las cuales deben tener una unidad y un valor entero.

En este documento en particular, se describe una bicicleta de aluminio de 56cm con dos ruedas de acero, donde la rueda trasera es 2mm más ancha que la rueda delantera.

A continuación se procederá a explicar la estructura de un documento XML así como las partes que lo componen.

3.1.1. Estructura Básica de un Archivo XML

De forma similar a los documentos HTML, los documentos XML están compuestos por etiquetas y datos. La mayor diferencia entre ambos es que las etiquetas utilizadas en un documento XML no están predefinidas por la especificación XML, sino que están especificadas por el lenguaje de etiquetas personalizado definido por la aplicación o sistema que lo va a interpretar. Por lo general los documentos XML son autoexplicativos, es decir, las etiquetas describen el tipo de datos que contienen, y un usuario o desarrollador debería ser capaz de entender el contenido del mismo sin tener la especificación completa del lenguaje a la mano.

Todo documento XML debe comenzar con la línea `<?xml version=”1.0”?>`, lo cual indica cuál es la versión XML utilizada en el documento. Actualmente existen únicamente dos versiones, 1.0 y 1.1; sin embargo, la versión 1.1 no está ampliamente soportada y solamente es recomendada para aquellos desarrolladores que necesiten sus funcionalidades adicionales, como el soporte de caracteres no incluidos en Unicode 3.2.

Luego, el documento debe contener un único elemento raíz, el cual a su vez puede tener cualquier número de elementos anidados, los cuales son llamados hijos. Estos elementos hijos a su vez pueden tener cualquier número de elementos anidados, lo cual implica que un documento XML tiene una estructura de árbol con una sola raíz, donde cada elemento es un nodo del árbol.

3.1.2. Elementos, Atributos y Valores

Un elemento es un nodo en el documento XML, el cual contiene:

- Una etiqueta. Las etiquetas XML deben comenzar con una letra, un carácter de subrayado o dos puntos. Las etiquetas que comienzan con el prefijo “xml” (en mayúscula, minúscula o cualquier combinación) están reservados y no pueden ser utilizados. Los demás caracteres pueden ser letras, dígitos o caracteres de subrayado.
- Opcionalmente uno o más atributos. Los nombres de los atributos deben seguir las mismas restricciones que aplican a las etiquetas. Los valores de los atributos deben estar entre comillas simples o dobles, siempre y cuando sea consistente.
- Contenido. Cada elemento puede contener otros elementos y/o cualquier cantidad de texto. Sin embargo, no se recomienda que los elementos contengan texto y otros elementos de forma simultánea.

En la **Figura 3.1**, el elemento raíz tiene la etiqueta “bicicleta”, el cual contiene 5 elementos hijos: un elemento con la etiqueta “marco”, dos elementos con la etiqueta “rueda”, un elemento con la etiqueta “asiento” y un elemento con la etiqueta “pedales”. El elemento “marco” contiene 3 elementos hijos para definir el material, tamaño y color, donde el elemento que describe el tamaño contiene un atributo “unidad”, el cual tiene por valor “cm”.

3.2. Adobe Flash CS4 Professional

Adobe Flash CS4 Professional[12] es una herramienta que permite generar documentos interactivos de manera atractiva. Al igual que en las películas, los documentos de Flash dividen el tiempo en fotogramas, los cuales se organizan en una la línea de tiempo para controlar el contenido de los documentos, colocándolos en el orden en que se desea que éstos aparezcan en el contenido final [1].

Adobe Flash utiliza gráficos vectoriales e imágenes rasterizadas, sonido, código de programa, flujo de vídeo y audio bidireccional. El lenguaje de programación embebido en Adobe Flash es ActionScript, el cual es un lenguaje de programación orientado a objetos que trata de ver el entorno de programación como el mundo real, donde cada objeto tiene propiedades y métodos.

Cuando se crean y guardan documentos en el entorno de edición de Adobe Flash, el formato de archivo generado es `FLA`, el cual contiene la información básica de objetos multimedia, línea de tiempo y scripts de un documento de Flash. Se puede agregar programas escritos en ActionScript a documentos de Flash para mejorar su comportamiento y conseguir que respondan a las acciones del usuario.

Para mostrar un documento en Adobe Flash Player se debe exportar o publicar el documento como archivo `SWF`. Adobe Flash permite exportar contenido e imágenes en los siguientes formatos: `.ai`, `.gif`, `.bmp`, `.swf`, `.flv`, `.jpg`, `.png`, `.mov`, `.wav`, `.avi`, entre otros. Adobe Flash permite publicar el archivo `FLA` en formatos de archivo interactivos (`.gif`, `.jpg`, `.png` y `.mov`) con el código `HTML` necesario para mostrarlos en la ventana del navegador.

El contenido de Adobe Flash se puede ver en varios navegadores, plataformas y teléfonos móviles. Al ser Adobe Flash una herramienta multiplataforma que provee gran interactividad entre el usuario y la aplicación a desarrollar, éste se convirtió en el candidato ideal para desarrollar la aplicación didáctica.

3.3. Tabla Comparativa de las Tecnologías

La Tabla 3.1 muestra una tabla comparativa de las distintas tecnologías investigadas.

	Authorware 7	Director 11	Lectora	Toolbook 9.5	Flash CS4
Empresa	Adobe Systems	Adobe Systems	Trivantis Corporation	SumTotal Systems	Adobe Systems
Licencia	Comercial	Comercial	Comercial	Comercial	Comercial
Costo	$2.999	$999	Desde $1.790 hasta $3.185	$2.795	$699

[12] http://www.adobe.com/es/products/flash

Idioma	Inglés	Alemán Francés Inglés	Inglés	Inglés	Alemán Checo Coreano Español Francés Holandés Inglés Italiano Japonés Polaco Portugués Ruso Suizo
Plataforma	Windows Mac	Windows Mac	Windows	Windows	Windows Mac
Lenguaje de programación embebido	No	Lingo	No	No	AS3
Soporte de media	Sí	Sí	Sí	Sí	Sí
Soporte de evaluaciones	Sí	Sí	Sí	Sí	No
Extensibilidad	No	Sí	No	No	Sí

Tabla 3.1: Tabla Comparativa de las Tecnologías

Una vez evaluadas las distintas tecnologías, se decidió utilizar Flash CS4 por cuatro razones principales:

1. Es extensible, permitiendo ampliar las funcionalidades de la aplicación.
2. Tiene un lenguaje de programación embebido, AS3.
3. Está disponible en idioma español.
4. Tiene menor costo.

4. Marco Metodológico

En este capítulo se explica la técnica de desarrollo de software elegida para llevar a cabo el desarrollo del presente Trabajo Especial de Grado.

4.1. ¿Qué es un Proceso del Software?

Un proceso del software es un conjunto de actividades que conducen a la creación de un software. Existen cuatro actividades fundamentales de procesos que son comunes para todos los procesos del software [10]. Estas actividades son:

- *Especificación del software:* Donde los clientes e ingenieros definen la funcionalidad del software a producir y las restricciones.
- *Diseño e Implementación del software:* Donde el software se diseña y programa.
- *Validación del software:* Donde el software se valida para asegurar que hace lo que el cliente desea.
- *Evolución del software:* Donde el software se modifica para adaptarlo a los cambios requeridos por el cliente.

4.2. ¿Qué es un Modelo de Procesos del Software?

Un modelo de procesos del software es una representación abstracta de un proceso del software. Estos modelos pueden incluir actividades que son parte de los procesos y productos de software y el papel de las personas involucradas en la ingeniería del software.

4.3. ¿Cuáles son los Modelos Generales del Proceso del Software?

La mayor parte de los modelos de procesos del software se basan en uno de los tres modelos generales de desarrollo de software:

- **El enfoque en cascada:** Este enfoque considera las actividades fundamentales del proceso como fases separadas del proceso, tales como: la especificación de requerimientos, el diseño del software, la implementación, las pruebas, entre otros. Después que cada etapa queda definida, el desarrollo continúa con la siguiente etapa.
- **El enfoque iterativo:** Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones muy abstractas. Éste se refina basándose en las peticiones del cliente para producir un sistema que satisfaga las necesidades de dicho cliente. El sistema puede entonces ser entregado. De forma alternativa, se puede reimplementar utilizando un enfoque más estructurado para producir un sistema más sólido y mantenible.

- **El enfoque basado en componentes:** Este enfoque se basa en la existencia de un número significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca en integrar estos componentes en el sistema más que desarrollarlos desde cero.

4.4. ¿Cuáles son los Costos de los Modelos del Proceso del Software?

La distribución de costos a través de las diferentes actividades en el proceso del software depende del proceso utilizado y del tipo de software que se vaya a desarrollar. Si se considera que el costo total del desarrollo de un sistema de software es de 100 unidades de costo, la Figura 4.1 muestra cómo se gastan estas unidades en las diferentes actividades del proceso.

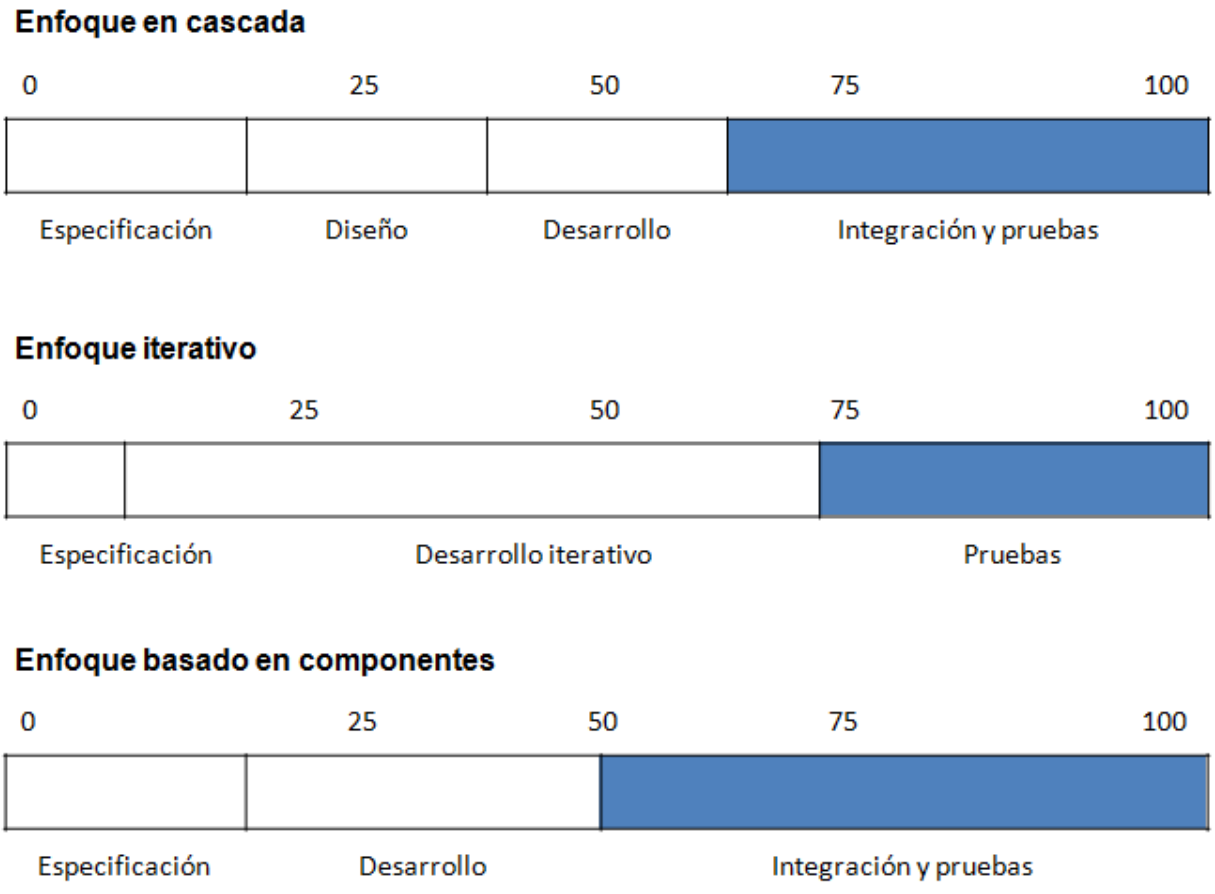

Figura 4.1: Costos de las Diferentes Actividades del Proceso

- En el enfoque en cascada, los costos de especificación, diseño, implementación e integración se miden de forma separada. Se puede observar que la integración y pruebas del sistema son las actividades de desarrollo más costosas.
- En el enfoque iterativo no existe división entre la especificación, el diseño y el desarrollo. En este enfoque, los costos de la especificación se reducen debido a

que sólo se produce la especificación de alto nivel antes que el desarrollo. La especificación, el diseño, la implementación, la integración, y las pruebas se llevan a cabo en paralelo dentro de una actividad de desarrollo. Sin embargo, aún se necesita una actividad independiente de pruebas del sistema una vez que la implementación inicial esté completa.

- El enfoque basado en componentes sólo ha sido ampliamente utilizado durante un corto período de tiempo. En este enfoque, no se tiene una representación exacta para los costos de las diferentes actividades. Sin embargo, se sabe que los costos de desarrollo se reducen en relación a los costos de integración y pruebas.

4.5. Modelo de Proceso del Software Utilizado

El modelo de proceso de desarrollo de software utilizado para el desarrollo del Trabajo Especial de Grado se basa en el enfoque incremental.

El enfoque en cascada requiere que los clientes de un sistema cumplan un conjunto de requerimientos antes de que se inicie el diseño y requiere que el diseñador cumpla estrategias particulares de diseño antes de la implementación. Los cambios de requerimientos implican rehacer el trabajo de captura de éstos, de diseño e implementación. Sin embargo, la separación en el diseño y la implementación deben dar lugar a sistemas bien documentados susceptibles de cambios.

En contraste, un enfoque iterativo se basa en la idea de desarrollar una implementación inicial, exponiéndola a los comentarios del usuario y refinándola a través de las diferentes versiones hasta que se desarrolla un sistema adecuado. Las actividades de especificación, desarrollo y validación se entrelazan en vez de separarse, con una rápida retroalimentación entre éstas.

En la producción de sistemas, un enfoque iterativo para el desarrollo de software suele ser más efectivo que el enfoque en cascada, ya que satisface las necesidades inmediatas de los clientes. La ventaja de un proceso de desarrollo de software que basa en un enfoque iterativo es que la especificación se puede desarrollar de forma creciente. Tan pronto como los usuarios desarrollen un mejor entendimiento de su problema, éste se puede reflejar en el sistema.

El enfoque incremental es un enfoque intermedio que combina las ventajas de ambos enfoques. En un proceso de desarrollo incremental, los clientes identifican, a grandes rasgos, los servicios que proporcionará el sistema. Identifican qué servicios son más importantes y cuáles menos importantes. Entonces, se definen varios incrementos en donde cada uno proporciona un subconjunto de la funcionalidad del sistema.

Una vez que los incrementos del sistema se han identificado, los requerimientos que se van a entregar en el primer incremento, se definen en detalle, y se desarrolla. Durante el desarrollo, se puede llevar a cabo un análisis adicional de requerimientos para los requerimientos posteriores, pero no se aceptan cambios en los requerimientos para el incremento actual.

Una vez que un incremento se completa y entrega, los clientes pueden ponerlo en servicio. Esto significa que tienen una entrega temprana de parte de la funcionalidad del sistema. Pueden experimentar con el sistema, lo cual les ayuda a clarificar sus requerimientos para los incrementos posteriores y para las últimas versiones del incremento actual. Tan pronto como se completan los nuevos incrementos, se integran en los existentes de tal forma que la funcionalidad del sistema mejora con cada incremento entregado. Los servicios comunes se pueden implementar al inicio del proceso o de forma incremental tan pronto como sean requeridos por un incremento.

Este proceso de desarrollo incremental tiene varias ventajas:

- Los clientes no tienen que esperar hasta que el sistema completo se entregue para sacar provecho de él. El primer incremento satisface los requerimientos más críticos de forma que puedan utilizar el software inmediatamente.
- Los clientes pueden utilizar los incrementos iniciales como prototipos y obtener experiencia sobre los requerimientos de los incrementos posteriores del sistema.
- Existe un bajo riesgo de un fallo total del proyecto. Aunque se pueden encontrar problemas en algunos incrementos, lo normal es que el sistema se entregue de forma satisfactoria al cliente.
- Puesto que los servicios de más alta prioridad se entregan primero, y los incrementos posteriores se integran en ellos, es inevitable que los servicios más importantes del sistema sean a los que se les hagan más pruebas. Esto significa que es menos probable que los clientes encuentren fallos de funcionamiento del software en las partes más importantes del sistema.

4.6. Actividades del Proceso Incremental

Las actividades que constituyen el proceso incremental son:

- Especificación del software
- Diseño e implementación del software
- Validación del software
- Evaluación del software

A continuación, se describen cada una de estas actividades.

4.6.1. Especificación del Software

La etapa de especificación del software es el proceso de comprensión y definición de qué servicios se requieren del sistema y de identificación de las restricciones de su funcionamiento y desarrollo del mismo.

Las actividades del proceso de especificación son:

- **Estudio de viabilidad:** Se estima si las necesidades del usuario se pueden satisfacer con las tecnologías actuales de software y hardware.
- **Obtención y análisis de requerimientos:** Se obtienen los requerimientos del sistema por medio de la observación de otros sistemas existentes, discusiones con usuarios potenciales, análisis de tareas, entre otros. Esto puede implicar el desarrollo de uno o más modelos y prototipos del sistema que ayudan al analista a comprender el sistema a desarrollar.
- **Especificación de requerimientos:** Se traduce la información recopilada durante la actividad de análisis en un documento que define un conjunto de requerimientos.
- **Validación de requerimientos:** Se comprueba la veracidad, consistencia y completitud de los requerimientos. Durante este proceso, inevitablemente se descubren errores en el documento de requerimientos. Se debe modificar el documento para corregir estos problemas.

4.6.2. Diseño e Implementación del Software

La etapa de implementación es el proceso de convertir una especificación del sistema en un sistema ejecutable. Siempre implica los procesos de diseño y programación de software, pero, como se utiliza un enfoque evolutivo, también puede implicar un refinamiento de la especificación del software.

Las actividades del proceso de diseño son:

- **Diseño arquitectónico:** Los subsistemas que forman el sistema y sus relaciones se identifican y documentan.
- **Especificación abstracta:** Para cada subsistema se produce una especificación abstracta de sus servicios y las restricciones bajo las cuales deben funcionar.
- **Diseño de la interfaz:** Para cada subsistema se diseña y documenta su interfaz con otros subsistemas.
- **Diseño de componentes:** Se asignan servicios a los componentes y se diseñan sus interfaces.
- **Diseño de la estructura de datos:** Se diseña en detalle y especifica la estructura de datos utilizada en la implementación del sistema.
- **Diseño de algoritmo:** Se diseñan en detalle y especifican los algoritmos utilizados para proporcionar los servicios.

4.6.3. Validación del Software

La etapa de validación de software se utiliza para mostrar que el sistema se ajusta a su especificación y cumple las expectativas del usuario. Implica procesos de comprobación,

como las inspecciones y revisiones en cada etapa del proceso desde la definición de requerimientos hasta el desarrollo del programa. Los errores en los componentes del programa pueden descubrirse durante las pruebas del sistema. Por lo tanto, el proceso es iterativo y se retroalimenta tanto de las últimas etapas como de la primera parte del proceso.

Las actividades del proceso de validación son:

- **Prueba de componentes:** Se prueban los componentes individuales para asegurar que funcionan correctamente. Cada uno se prueba de forma independiente, sin los otros componentes del sistema. Los componentes pueden ser entidades simples como funciones o clases de objetos.
- **Prueba del sistema:** Los componentes se integran para formar el sistema. Este proceso comprende encontrar errores que son el resultado de interacciones no previstas entre los componentes y su interfaz.
- **Prueba de aceptación:** Es la etapa final en el proceso de pruebas antes de que se acepte que el sistema se ponga en funcionamiento. Éste se prueba con los datos proporcionados por el cliente más que con datos de prueba simulados.

Si se utiliza un enfoque incremental, cada incremento debe ser probado cuando se desarrolla, con estas pruebas basadas en los requerimientos de ese incremento.

4.6.4. Evolución del Software

La etapa de evolución del software corresponde a los cambios que se le pueden hacer en cualquier momento, durante o después del desarrollo del sistema. Considerado como el mantenimiento del software.

5. Marco Aplicativo

En el Capítulo 4 se expuso la metodología utilizada que se llevó a cabo para el diseño e implementación de la aplicación; el proceso está basado en el enfoque incremental, el cual combina las ventajas del enfoque en cascada y del enfoque iterativo.

A continuación, se definen y explican los incrementos que se tuvieron que hacer para cumplir a cabalidad los objetivos de la aplicación en donde cada incremento proporciona una funcionalidad o un subconjunto de funcionalidades implementadas.

5.1. Incremento 1: Crear Prototipo Inicial de la Aplicación

5.1.1. Especificación del Software

Se requirió un prototipo inicial que incluya soporte básico para la elaboración de quices con preguntas de selección simple y múltiple, carga de videos y una sección llamada "Ayuda", la cual debía mostrar la información de contacto en un texto animado.

5.1.2. Diseño e Implementación del Software

Se diseñó la primera versión de la Interfaz Gráfica de Usuario (ver Figura 5.1). Se diseñaron e implementaron las clases `MainClip`, `BackgroundClip`, `ImageClip`, `QuestionClip`, `QuizClip`, `CreditsClip` y `VideoClip`. También se definió el formato del archivo XML de un quiz.

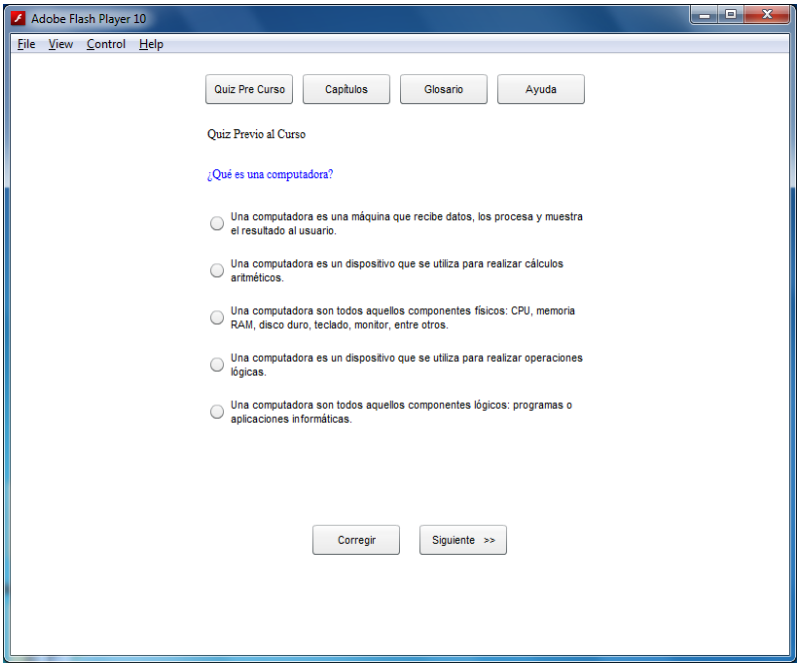

Figura 5.1: Interfaz Gráfica de Usuario del Módulo de Quices v.1

5.1.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.2. Incremento 2: Implementar el Módulo de Capítulos y Lecciones

5.2.1. Especificación del Software

Se requirió un prototipo que despliegue un índice de los capítulos a impartir y que una vez que se seleccione alguno se despliegue una lista de las lecciones de ese capítulo.

5.2.2. Diseño e Implementación del Software

Se definió el formato del archivo `course.xml`, que contiene la estructura del curso, el cual se divide en capítulos y lecciones. También se crearon las clases `CourseClip` y `ChapterClip` para mostrar los capítulos del curso y sus lecciones.

5.2.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.3. Incremento 3: Implementar el Despliegue de las Lecciones del Curso

5.3.1. Especificación del Software

Se requirió un prototipo que despliegue el contenido de una lección integrando diversos recursos multimedia, tales como: texto, imágenes, video o archivos SWF.

5.3.2. Diseño e Implementación del Software

Se diseñó la segunda versión de la Interfaz Gráfica de Usuario (ver Figura 5.2). Se definió el formato de un archivo XML de una lección, la cual se divide en una o más páginas que pueden incluir texto, imágenes, video o SWFs. En este incremento se implementaron las clases `LessonClip` y `PageClip` para implementar la navegación dentro de una lección y el despliegue de su contenido. También se creó la clase `ExternalSWFClip` para mostrar el contenido de archivos SWF.

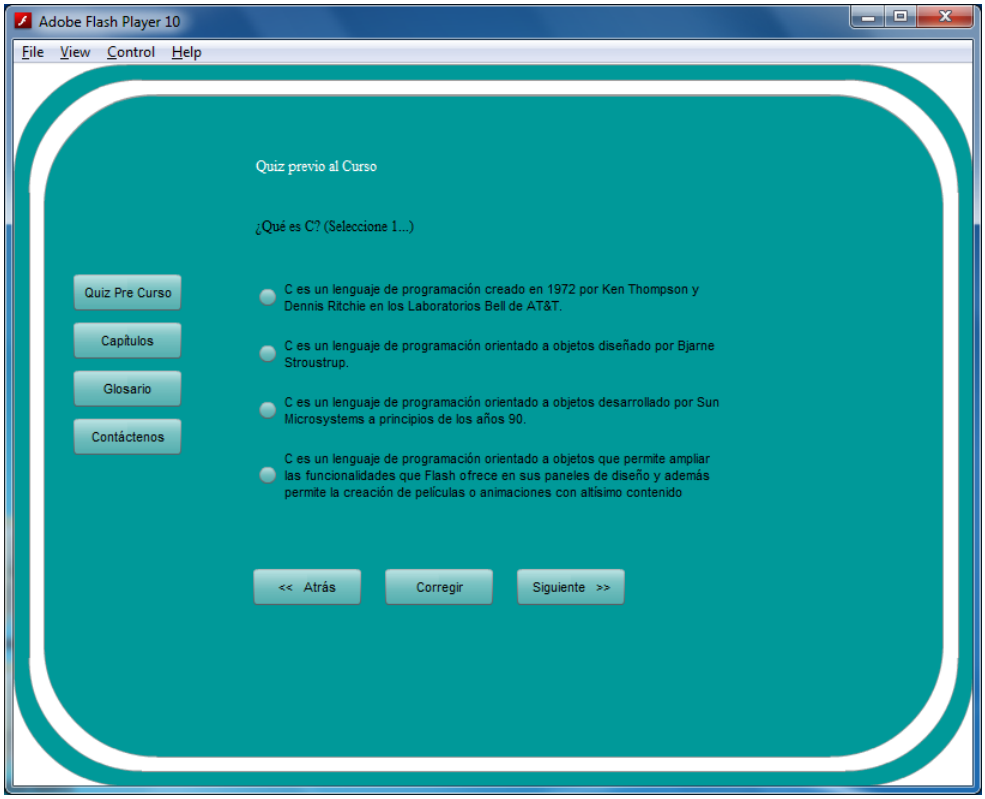

Figura 5.2: Interfaz Gráfica de Usuario del Módulo de Quices v.2

5.3.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.4. Incremento 4: Añadir Soporte para Sonidos en las Lecciones del Curso

5.4.1. Especificación del Software

Se requirió un prototipo que incluya soporte básico para la reproducción de audio en las lecciones, el cual debía mostrar un botón de iniciar y detener reproducción.

5.4.2. Diseño e Implementación del Software

Se diseñó la tercera versión de la Interfaz Gráfica de Usuario (ver Figura 5.3). Se extendió el formato de los XML para las lecciones, permitiendo el uso de la etiqueta `sound` para cargar un único archivo de sonido por lección. Se creó la clase `SoundClip`, la cual carga un archivo MP3 y crea botones para iniciar y detener la reproducción del archivo.

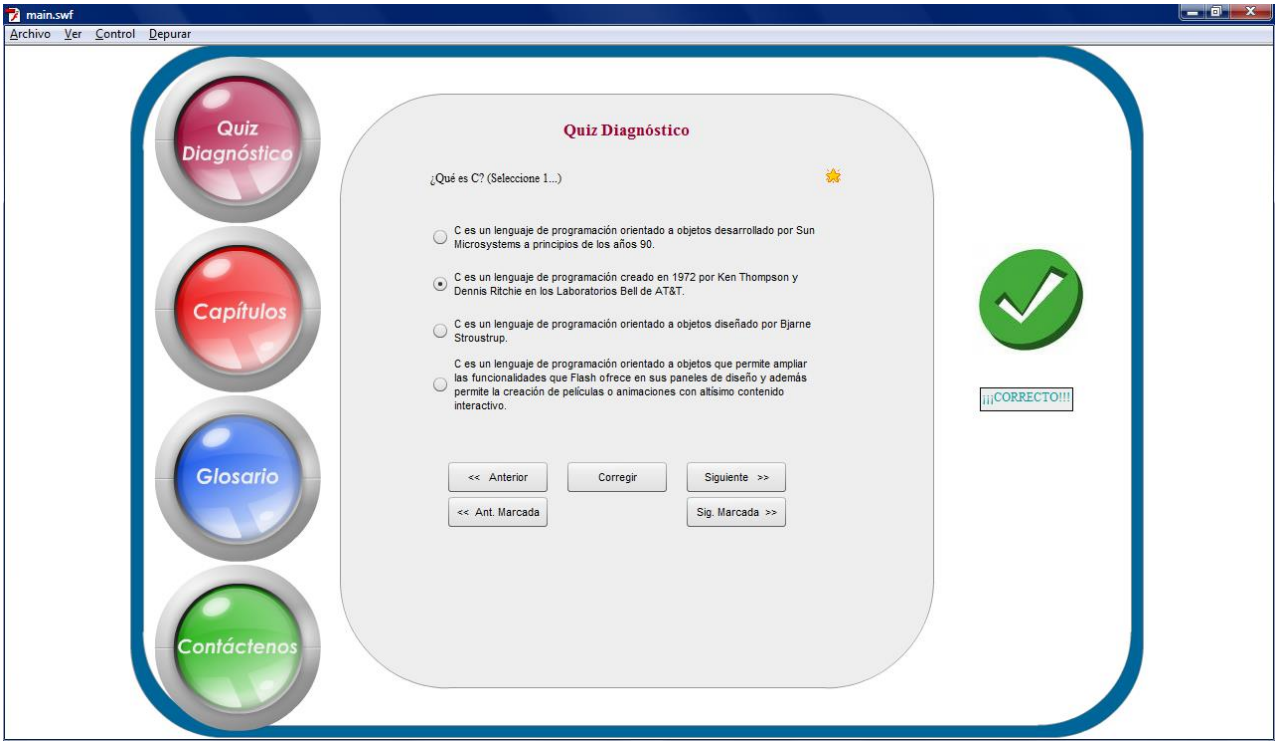

Figura 5.3: Interfaz Gráfica de Usuario del Módulo de Quices v.3

5.4.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.5. Incremento 5: Integrar Glosario Existente en la Aplicación

5.5.1. Especificación del Software

Se requirió un prototipo que incluya un módulo llamado "Glosario", el cual ya estaba implementado en otro sistema.

5.5.2. Diseño e Implementación del Software

Se diseñó la cuarta versión de la Interfaz Gráfica de Usuario (ver Figura 5.4). Se creó la clase `GlossaryClip` para cargar un glosario existente de otro programa, el cual estaba contenido en un archivo SWF. Como la interfaz del glosario existente no tenía armonía con la interfaz principal y además, no era flexible ni configurable se decidió implementar un nuevo glosario.

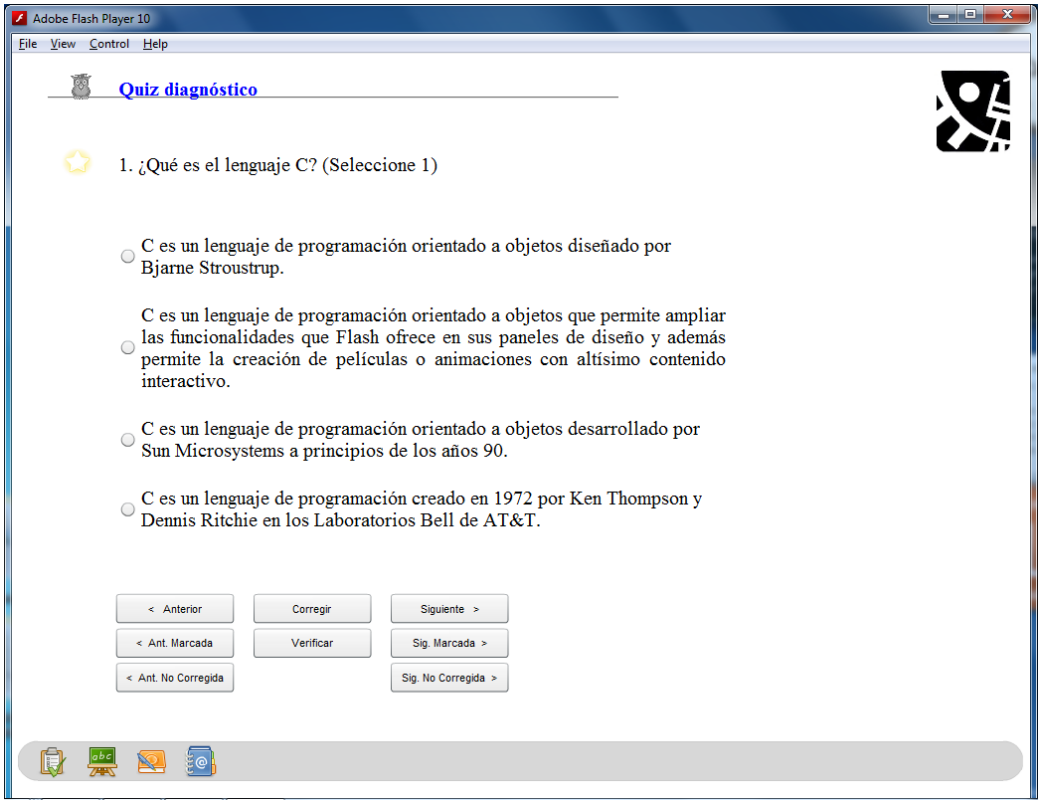

Figura 5.4: Interfaz Gráfica de Usuario del Módulo de Quices v.4

5.5.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.6. Incremento 6: Implementar Funcionalidad Propia para el Glosario

5.6.1. Especificación del Software

Se requirió un prototipo que despliegue un glosario nuevo, el cual debía mostrar en orden alfabético una lista de letras horizontalmente y al seleccionar alguna letra se debía mostrar en orden alfabético una lista de los términos asociados verticalmente.

5.6.2. Diseño e Implementación del Software

Se diseñó la última versión de la Interfaz Gráfica de Usuario (ver Figura 5.5). Se definió el formato del archivo glossary.xml, el cual contiene los términos del glosario. También se modificó la clase `GlossaryClip` para leer y mostrar el contenido de dicho archivo XML, sin utilizar la funcionalidad de glosario ya existente.

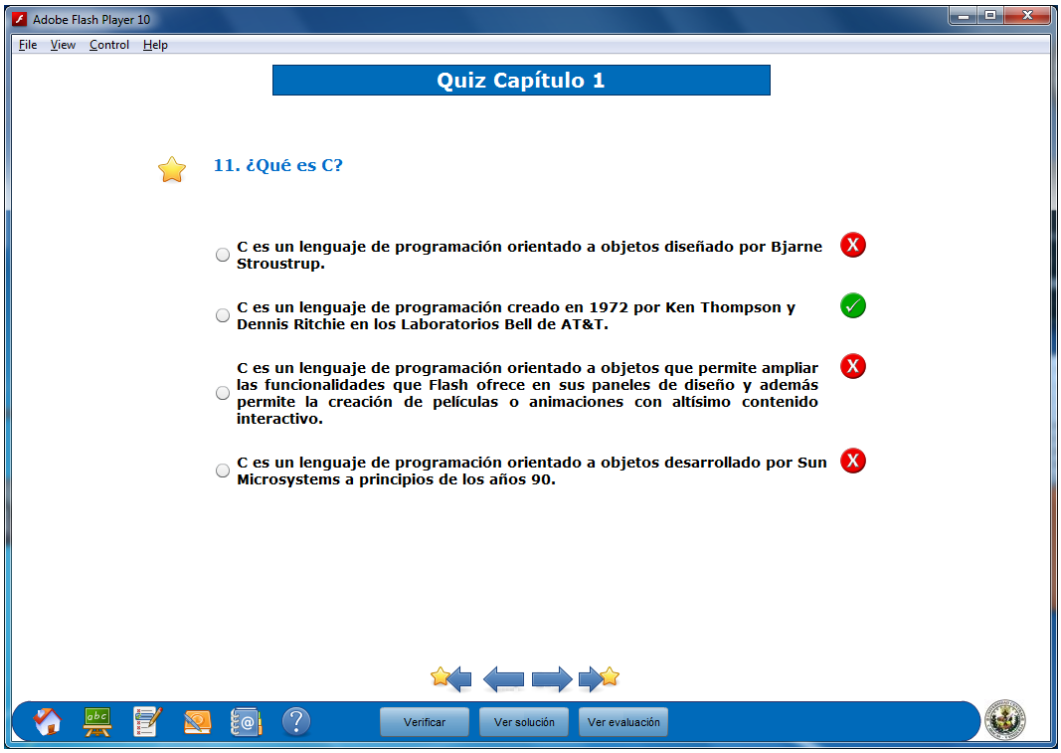

Figura 5.5: Interfaz Gráfica de Usuario del Módulo de Quices v.5

5.6.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.7. Incremento 7: Añadir Controles para la Reproducción de Videos

5.7.1. Especificación del Software

Se requirió un prototipo que incluya la barra de reproducción y la barra de volumen en los videos de las lecciones.

5.7.2. Diseño e Implementación del Software

Se modificó la clase `VideoClip` añadiendo controles para su reproducción, los cuales incluyen un botón para iniciar o detener la reproducción, una barra para mover el cabezal de reproducción y una barra para modificar el volumen del video.

5.7.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.8. Incremento 8: Extender Controles para la Reproducción de Audio

5.8.1. Especificación del Software

Se requirió un prototipo que incluya soporte completo para la reproducción de audio en las lecciones, el cual debía mostrar un botón de iniciar, pausar y detener. Además, debía mostrar una barra de reproducción y una barra de volumen.

5.8.2. Diseño e Implementación del Software

Se extendió la clase `SoundClip` para mostrar controles que permitieran mover el punto de reproducción del sonido y el volumen del mismo.

5.8.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.9. Incremento 9: Añadir Soporte de Impresión para las Lecciones

5.9.1. Especificación del Software

Se requirió un prototipo que imprima el contenido de la página actual de las lecciones.

5.9.2. Diseño e Implementación del Software

Cuando se muestran las lecciones se observa un botón de impresora, el cual permite imprimir la página actual del curso. Para esto, se implementó la clase `PrintJob` utilizando funcionalidades de impresión incluidas en AS3.

5.9.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas. Los resultados no fueron satisfactorios debido a que Flash CS4 sólo soporta imágenes de 72 DPI[13], lo que causa que las imágenes salgan desproporcionadas cuando se solicita una impresión directa.

5.10. Incremento 10: Modificar soporte de Impresión para Generar Archivos PDF

5.10.1. Especificación del Software

Se requirió un prototipo nuevo que permita imprimir el contenido de la página actual de las lecciones sin perder resolución en las imágenes.

5.10.2. Diseño e Implementación del Software

Se modificó la funcionalidad de impresión para generar archivos PDF con la página actual, lo cual se implementó en la clase `PDFPrinter`. Para la generación del archivo, se utilizó una biblioteca de uso libre llamada `AlivePDF`.

5.10.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.11. Incremento 11: Añadir Página de Inicio

5.11.1. Especificación del Software

Se requirió un prototipo que muestre la página inicial del curso.

5.11.2. Diseño e Implementación del Software

Se creó la clase `HomeClip` para mostrar un texto estático con la bienvenida al curso. También se añadió un botón en la barra inferior para ir a la página principal, y se modificó la aplicación para mostrar esta página al inicio de la aplicación.

[13] DPI: Dots per Inch (puntos por pulgada)

5.11.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas. Los resultados fueron satisfactorios pero se sugirió cambiar el texto estático por una animación contenida en un archivo SWF.

5.12. Incremento 12: Modificar Página de Inicio para Mostrar Contenido Animado

5.12.1. Especificación del Software

Se requirió un prototipo que muestre una animación para el módulo de inicio.

5.12.2. Diseño e Implementación del Software

Se modificó la clase `HomeClip` para mostrar un archivo SWF con la bienvenida del curso, utilizando la clase `ExternalSWFClip` existente. Este archivo SWF se creó directamente en Flash, y contiene la información de bienvenida así como fotos de la Facultad de Ciencias.

5.12.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.13. Incremento 13: Navegabilidad entre Lecciones

5.13.1. Especificación del Software

Se requirió un prototipo que permita navegar entre las lecciones del curso.

5.13.2. Diseño e Implementación del Software

Se modificó el formato de los archivos XML de las lecciones de forma que contenga enlaces que permitan ir a otra lección diferente o mostrar el contenido de otro capítulo. Se añadió soporte para esta funcionalidad en la clase `MainClip` y se modificó la clase `LessonClip` para utilizarla.

5.13.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.14. Incremento 14: Simplificar la Sección de Contacto

5.14.1. Especificación del Software

Se requirió un prototipo que despliegue un texto estático en el módulo de contáctenos.

5.14.2. Diseño e Implementación del Software

Se modificó la clase `CreditsClip` para mostrar el texto de forma estática sin animaciones, debido a problemas durante la animación.

5.14.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.15. Incremento 15: Añadir Soporte para Imprimir un Tópico Completo

5.15.1. Especificación del Software

Se requirió un prototipo que imprima el contenido de todo un capítulo.

5.15.2. Diseño e Implementación del Software

Se modificó la clase `PDFPrinter` para permitir la generación de un archivo PDF con el contenido de un capítulo completo. También se modificó la clase `CourseClip` para que mostrara un ícono pequeño de impresora al lado izquierdo de cada capítulo, el cual se utilizaría para imprimir el capítulo asociado.

5.15.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.16. Incremento 16: Implementar Preguntas Fill in Blank y Drag and Drop

5.16.1. Especificación del Software

Se requirió un prototipo que agregue soporte para preguntas donde se debe arrastrar y soltar objetos (*drag and drop*) y llenado de espacios en blanco (*fill in blank*).

5.16.2. Diseño e Implementación del Software

Se extendió la funcionalidad de los quices para incluir preguntas de llenado de espacios en blanco y ordenamiento de elementos. Aquí se crearon las clases `SimpleQuestionClip`, `MultipleQuestionClip`, `FillQuestionClip` y `DragQuestionClip`. También se extendió el formato de los archivos XML para que soporte los diversos tipos de quices.

5.16.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.17. Incremento 17: Añadir Soporte para Corregir y Evaluar un Quiz

5.17.1. Especificación del Software

Se requirió un prototipo que permita corregir la respuesta de una pregunta y evaluar la misma.

5.17.2. Diseño e Implementación del Software

Se modificaron las clases `QuizClip`, `QuestionClip`, `SimpleQuestionClip`, `MultipleQuestionClip`, `FillQuestionClip` y `DragQuestionClip` para añadir soporte de puntuación a los quices. También se creó la clase `QuizStatusClip` para mostrar las preguntas respondidas y la puntuación obtenida hasta el momento.

5.17.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.18. Incremento 18: Crear Módulo de Quices

5.18.1. Especificación del Software

Se requirió un prototipo que englobe en un módulo todos los quices del curso.

5.18.2. Diseño e Implementación del Software

Se definió el formato del archivo quices.xml, el cual contiene una lista con todos los quices disponibles en el curso. También se creó la clase `QuicesClip`, la cual se encarga de mostrar una lista con todos los quices y le permite al alumno seleccionar uno.

5.18.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.19. Incremento 19: Definir el Contenido del Curso

5.19.1. Especificación del Software

Se requirió esbozar todo el contenido del curso, incluyendo lecciones y quices.

5.19.2. Diseño e Implementación del Software

Se creó el archivo course.xml para el curso de C y se generaron archivos vacíos para cada una de las lecciones del curso. También se creó el archivo quices.xml para el curso de C y se generaron archivos vacíos para cada uno de los quices del curso.

5.19.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.20. Incremento 20: Preparar el Contenido del Curso

5.20.1. Especificación del Software

Se requirió redactar todo el contenido del curso, incluyendo lecciones y quices. También se realizaron las imágenes, videos y audios a integrar con el contenido.

5.20.2. Diseño e Implementación del Software

Se escribió el contenido de cada una de las lecciones del curso. También se crearon los recursos multimedia asociados a cada lección, como imágenes, videos y archivos de audio. Finalmente, se crearon las preguntas y las respuestas para cada uno de los quices del curso.

5.20.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.21. Incremento 21: Reestructurar el Contenido de las Lecciones

5.21.1. Especificación del Software

Se requirió una revisión de todas las lecciones.

5.21.2. Diseño e Implementación del Software

Se cambió del orden de algunas lecciones para facilitar la comprensión del contenido.

5.21.3. Validación del Software

Se revisaron y probaron las funcionalidades implementadas en este incremento para asegurar que funcionan correctamente.

5.22. Manejo General de los Botones

En toda la aplicación los botones se manejan utilizando la clase `ImageClip`, modificándola para que proporcione la funcionalidad que debe tener un botón.

Para crear un botón se necesita crear una instancia de la clase `ImageClip`, a la cual se le asigna una posición `(X, Y)`, se le añaden tres eventos para responder a los movimientos y clics del ratón, y por último se agrega al escenario. Este proceso se repite por cada botón que se desea agregar.

Por lo general, las clases que necesitan crear botones tienen un método llamado `setupButtons()`, el cual se encarga de realizar el procedimiento antes descrito por cada botón que desee desplegar.

5.23. Manejo General de las Descripciones Emergentes

Las descripciones emergentes (llamados en inglés *tooltips*) se manejan creando campos de texto inicialmente invisibles arriba del elemento que describen. Luego, se añade un evento para detectar cuando el ratón se coloca sobre el elemento descrito, de forma tal que la descripción se haga visible. También se añade un evento para detectar cuando el ratón deja de estar sobre el elemento para que la descripción se vuelva a hacer invisible.

Por lo general, las clases que necesitan crear descripciones emergentes tienen un método llamado `setupTooltips()`, el cual se encarga de realizar el procedimiento antes descrito por cada descripción que desee colocar.

5.24. Manejo de los Elementos Multimedia

La aplicación didáctica desarrollada utiliza múltiples elementos multimedia que facilitan la compresión del contenido. Entre estos elementos multimedia se encuentra texto, imágenes, audios, videos y animaciones externas en Flash. Cada uno de estos elementos se maneja de forma diferente, como se indica a continuación:

- **Texto:** El texto se maneja directamente añadiendo campos de texto `TextField` al escenario.
- **Imágenes:** Las imágenes son cargadas instanciando un objeto de la clase `ImageClip`, a la cual se le especifica la ruta de la imagen a cargar.
- **Audios:** Los audios son cargados instanciando un objeto de la clase `SoundClip`, la cual carga el archivo de audio de la ruta especificada y crea controles para iniciar o detener el audio así como para modificar el volumen y la posición actual de reproducción.
- **Videos:** Los videos son cargados instanciando un objeto de la clase `VideoClip`, la cual carga un archivo de video con el tamaño especificado, incluyendo los controles de reproducción incluidos por defecto en Flash.
- **Animaciones externas en Flash:** La aplicación provee soporte para añadir otros archivos externos en Flash con la extensión SWF. Para cargar uno de estos elementos se debe instanciar un objeto de la clase `ExternalSWFClip`, la cual recibe la ruta del archivo a cargar en el constructor. El objeto de Flash cargado mantendrá todas sus capacidades interactivas como si hubiera sido cargado directamente en un navegador.

Es importante que tenga en cuenta que Flash carga los elementos de forma asíncrona, es decir, el constructor finaliza su ejecución antes de que el elemento multimedia haya sido cargado completamente. Esto implica que algunas características del elemento no están disponibles después que se ha instanciado el objeto, como por ejemplo, el tamaño de una imagen o la longitud de un video.

5.25. Descripción de las Clases

A continuación, se describen cada una de las clases que componen a la aplicación, incluyendo sus funcionalidades y una breve descripción indicando desde dónde son invocadas (ver Figura 5.6).

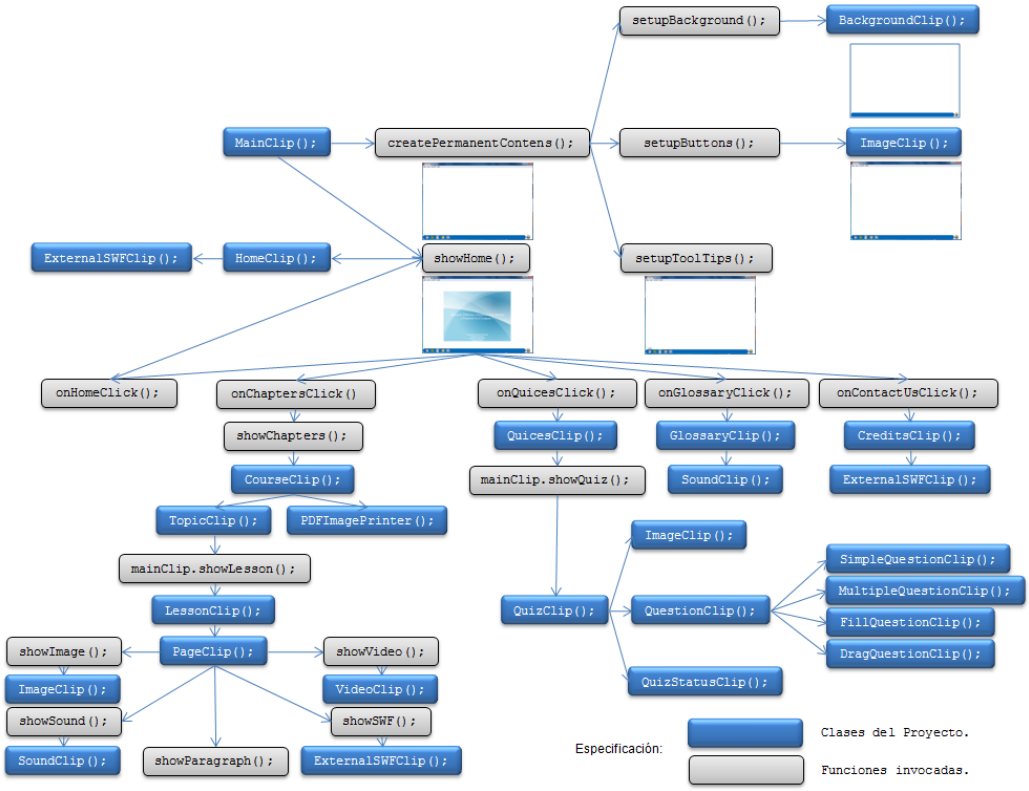

Figura 5.6: Interacción de Clases

5.25.1. MainClip

La clase `MainClip` es la clase principal de la aplicación y su única instancia es creada en el primer y único cuadro de la línea de tiempo.

Esta clase se encarga de cargar el fondo utilizando la clase `BackgroundClip` y los botones principales de la aplicación, los cuales incluyen:

- El botón de **Inicio** coloca como contenido actual el archivo `home.swf` utilizando la clase `ExternalSWFClip` en el método `showHome`.
- El botón de **Índice** coloca como contenido actual una lista de todos los capítulos del curso mediante el uso de la clase `CourseClip`.
- El botón de **Quices** coloca como contenido una lista de todos los quices del curso mediante el uso de la clase `QuicesClip`.

- El botón de **Glosario** coloca como contenido actual el glosario del curso mediante el uso de la clase `GlossaryClip`.
- El botón de **Contáctenos** coloca como contenido actual la información relacionada a los créditos utilizando la clase `CreditsClip`.

Finalmente, coloca como contenido actual la página de inicio.

5.25.2. BackgroundClip

La clase `BackgroundClip` se encarga de cargar el contenido del fondo de la aplicación. Esta clase es instanciada desde `MainClip` cuando la aplicación está iniciando. Actualmente carga una imagen estática como fondo, pero se puede modificar esta clase si se desea cargar algún contenido dinámico.

5.25.3. ImageClip

La clase `ImageClip` recibe como parámetro a la imagen a mostrar. Esta clase carga la imagen y la agrega en el escenario.

5.25.4. HomeClip

Esta clase se encarga de mostrar la pantalla de inicio de la aplicación. Actualmente carga un archivo SWF dando la bienvenida al curso mediante el uso de la clase `ExternalSWFClip`.

5.25.5. ExternalSWFClip

Esta clase se encarga de cargar un archivo externo en formato SWF, el cual mantendrá todas sus funcionalidades originales como si fuese cargado en un navegador.

5.25.6. CourseClip

Esta clase se encarga de mostrar una lista de todos los capítulos del curso, los cuales se obtienen del archivo `course.xml`. El nombre de cada capítulo se carga y se muestra utilizando un campo de texto, y se añaden manejadores de eventos de forma tal que se muestre el contenido de un capítulo cuando se hace clic sobre éste.

Esta clase es instanciada desde `MainClip` cuando se hace clic al botón correspondiente a los capítulos.

5.25.7. ChapterClip

Esta clase se encarga de mostrar todas las lecciones asociadas a un capítulo cuando se hace clic sobre éste en `CourseClip`. El nombre de cada lección se carga y se muestra utilizando un campo de texto, y se añaden manejadores de eventos de forma tal que se muestre la primera página de una lección cuando se hace clic sobre ésta. Para mostrar la lección se invoca a la función `showLesson` de `MainClip`, especificando el archivo que contiene la lección.

Esta clase es instanciada desde `CourseClip` cuando se hace clic a un enlace de capítulo en la lista creada por `CourseClip`.

5.25.8. PDFImagePrinter

Esta clase se encarga de generar archivos en formato PDF a partir de una o varias páginas de las lecciones. Esta clase es invocada cuando se desea imprimir un capítulo completo desde `CourseClip` o una página desde `PageClip`.

5.25.9. LessonClip

Esta clase se encarga de mostrar el contenido de una lección del curso, la cual se encuentra divida en varias páginas. Esta clase también se encarga de moverse a través de las páginas de todo el curso utilizando los botones de **página previa** y **siguiente página**. Las páginas se muestran utilizando clases de la instancia `PageClip`.

5.25.10. PageClip

Esta clase se encarga de mostrar el contenido de una página de una lección del curso, la cual está compuesta por texto, imágenes, audio, video y animaciones en Flash. Para mostrar estos componentes se utilizan las clases descritas en la Sección 5.24.

5.25.11. SoundClip

Esta clase se encarga de cargar un archivo de audio en formato MP3 y crear controles para manejarlo. Entre estos controles se incluyen botones para iniciar, pausar o detener la reproducción, así como dos barras de desplazamiento que permiten colocar la posición actual del cabezal y el volumen de la reproducción.

5.25.12. VideoClip

Esta clase se encarga de cargar un archivo de video en formato FLV o MP4 y crear controles para manejarlo. Estos controles vienen incluidos en la biblioteca de Flash CS4 e incluyen botones para iniciar, pausar o detener la reproducción, como dos barras de

desplazamiento que permiten colocar la posición actual del cabezal y el volumen de la reproducción, y un botón que permite silenciar la reproducción.

5.25.13. QuicesClip

Esta clase se encarga de mostrar una lista de todos los quices del curso, los cuales se obtienen del archivo `quices.xml`. El nombre de cada quiz se carga y se muestra utilizando un campo de texto, y se añaden manejadores de eventos de forma tal que se muestre el contenido de un quiz cuando se hace clic sobre éste.

Esta clase es instanciada desde `MainClip` cuando se hace clic al botón correspondiente a los quices.

5.25.14. QuizClip

Esta clase se encarga de mostrar un quiz en la aplicación, la cual carga todas las preguntas del quiz durante la construcción, También se crean botones para navegar entre las preguntas, ver la solución o verificar la pregunta actual, o mostrar la puntuación obtenida hasta el momento.

Para mostrar las preguntas se utilizan instancias de la clase `QuestionClip`, y para mostrar la puntuación se utiliza una instancia de la clase `QuizStatusClip`.

5.25.15. QuestionClip

Esta clase se encarga de mostrar una pregunta de cualquier tipo. Para realizar esto se verifica el tipo de pregunta durante la construcción y se añade una instancia del tipo específico correspondiente al escenario, por ejemplo, si la pregunta es de selección múltiple se añade una instancia de la clase `MultipleQuestionClip`.

5.25.16. SimpleQuestionClip

Esta clase se encarga de mostrar una pregunta de selección simple, la cual utiliza instancias de la clase `RadioButton` para mostrar cada una de las posibles respuestas.

5.25.17. MultipleQuestionClip

Esta clase se encarga de mostrar una pregunta de selección múltiple, la cual utiliza una instancia de la clase `CheckBox` para mostrar cada una de las posibles respuestas.

5.25.18. FillQuestionClip

Esta clase se encarga de mostrar una pregunta de completación, la cual utiliza una instancia de la clase `TextField` para recibir la respuesta del usuario. Para verificar si la respuesta es correcta se utilizan expresiones regulares.

5.25.19. DragQuestionClip

Esta clase se encarga de mostrar una pregunta de ordenamiento, la cual consiste en arrastrar líneas de código de forma tal que se genere un programa que cumpla con ciertas especificaciones. Para cargar las líneas de texto se utilizan imágenes estáticas cargadas mediante la clase `ImageClip`.

5.25.20. QuizStatusClip

Esta clase se encarga de mostrar el resultado de todas las preguntas de un quiz, incluyendo referencias a las lecciones en aquellas que hayan sido contestadas de forma errónea. Además, se muestra información sobre la cantidad de preguntas respondidas correctamente y la puntuación total obtenida hasta el momento. Si se hace clic sobre el número de alguna pregunta, ésta será mostrada.

5.25.21. GlossaryClip

Esta clase se encarga de mostrar el glosario de la aplicación mediante el uso de campos de texto. El glosario está clasificado por la letra inicial de cada uno de los términos descritos, y soporta archivos de audio para cada uno de los términos. Esta clase es instanciada únicamente desde `MainClip` cuando se presiona el botón correspondiente al glosario.

El contenido del glosario se extrae del archivo `glossary.xml`, que contiene todos los términos y sus definiciones clasificadas por la letra inicial.

5.25.22. CreditsClip

Esta clase se encarga de mostrar los créditos de la aplicación mediante el uso de una imagen estática. Esta clase es instanciada únicamente desde `MainClip` cuando se presiona el botón correspondiente a los créditos.

6. Caso de Estudio: Introducción al Lenguaje C

Para mostrar las capacidades del generador de cursos multimedia desarrollado en este Trabajo Especial de Grado se instanció un curso para enseñar a programar en el Lenguaje C, el cual está dividido en nueve (9) capítulos y provee nueve (9) quices para que el alumno pueda verificar que cumplió con los objetivos cada capítulo.

En las siguientes secciones se presenta la teoría base del curso instanciado, las cuales consisten en una introducción general al Lenguaje C para que un nuevo usuario de este lenguaje conozca los conceptos básicos de programación. Esto incluye su historia, las palabras reservadas, las estructuras de datos y las estructuras de control disponibles, así como los pasos llevados a cabo para ejecutar un programa escrito en este lenguaje [2].

6.1. Historia

C es un lenguaje de programación creado en 1972 por Ken Thompson y Dennis Ritchie en los Laboratorios Bell de AT&T. El lenguaje C evolucionó a partir de dos lenguajes de programación anteriores, BCPL (Basic Combined Programming Language) y B. El lenguaje BCPL fue diseñado por Martin Richards en 1966 debido a las dificultades experimentadas con el lenguaje de programación CPL (Combined Programming Language) durante los años 60. Años después, Dennis Ritchie utilizó BCPL como base para desarrollar el lenguaje B, que a su vez más adelante daría lugar al lenguaje de programación C. Se le dio el nombre “C” porque muchas de sus características fueron tomadas del lenguaje “B” [7].

6.2. Características

Algunas características del lenguaje C son:

1. Es un lenguaje estructurado de nivel medio, es decir, ni de bajo nivel como ensamblador, ni de alto nivel como Haskell[14].
2. Es un lenguaje fuertemente tipeado, ya que se debe especificar el tipo de datos de cada variable, parámetro y retorno de función.
3. No lleva a cabo comprobación de errores en tiempo de ejecución.
4. La primera estandarización del lenguaje C fue hecha por el ANSI (American National Standards Institute) en diciembre de 1989, con el estándar X3.159-1989. El lenguaje definido en este estándar fue conocido como ANSI C. Más adelante, en diciembre de 1990, fue aprobado como estándar por el ISO (International Organization for Standardization), con el ISO/IEC 9899:1990. La estandarización actual es el ISO/IEC 9899:1999, también denominado C99.

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

5. Dispone de una biblioteca estándar que contiene numerosas funciones que siempre se encuentran disponibles.
6. Tiene un número reducido de palabras reservadas, 32 palabras reservadas en C89 y 37 en C99.

6.3. Palabras Reservadas

Las palabras reservadas son utilizadas por el lenguaje C para implementar diversas características, tales como las estructuras de control de C. Las palabras reservadas no pueden utilizarse como identificadores. Todas las palabras reservadas de C contienen solamente letras minúsculas. La Tabla 6.1 muestra una lista completa de las palabras reservadas de C.

Palabras Reservadas				
`auto`	`break`	`case`	`char`	`const`
`continue`	`default`	`do`	`double`	`else`
`enum`	`extern`	`float`	`for`	`goto`
`if`	`int`	`long`	`register`	`return`
`short`	`signed`	`sizeof`	`static`	`struct`
`switch`	`typedef`	`union`	`unsigned`	`void`
`volatile`	`while`	–	–	–

Tabla 6.1: Palabras Reservadas definidas en C89

6.4. Fases para Ejecutar un Programa

Generalmente, los programas escritos en el lenguaje C pasan a través de cinco fases para llevar a cabo su ejecución, las cuales son: *Edición*, *Preprocesamiento*, *Compilación*, *Enlace* y *Carga*.

Edición: Esta primera fase consiste en crear y editar un archivo por parte del programador mediante un *programa de edición*. El programador escribe el código fuente del programa mediante el editor, el cual es almacenado posteriormente en un dispositivo de almacenamiento secundario. Los archivos con código fuente escrito en lenguaje C terminan con la extensión `.c`, como: `example.c`.

Preprocesamiento: En esta segunda fase se utiliza un preprocesador, el cual consiste en un programa encargado de realizar ciertas transformaciones al código fuente. Algunas acciones que realiza el preprocesador son la inclusión de otros archivos de cabecera en el código fuente, la definición de constantes simbólicas y macros, la compilación condicional del código fuente y la ejecución condicional de las directivas del preprocesador. Todas las directivas del preprocesador comienzan con el carácter numeral (`#`) y, al no ser instrucciones de C, no deben terminar con punto y coma (`;`). Las directivas del preprocesador se procesan completamente antes de que empiece la compilación.

Compilación: Una vez que el código fuente ha sido preprocesado, el compilador traduce el código fuente a código en lenguaje de máquina, también llamado código objeto,

siempre y cuando el compilador no detecte ningún error. Como resultado de esta fase, el compilador guarda el código objeto generado en un archivo con otra extensión, la cual depende del sistema operativo.

Enlace: Por lo general, los programas escritos en el lenguaje C contienen referencias a funciones y datos definidos en la biblioteca estándar u otras bibliotecas específicas. El código objeto producido por el compilador de C, por lo general, contiene "huecos" debido a estas partes faltantes. El *enlazador* resuelve las referencias externas faltantes del código objeto para producir un archivo ejecutable sin piezas faltantes. Si el programa se compila y se enlaza correctamente se produce un archivo ejecutable.

Carga: Antes de que el programa se pueda ejecutar, éste debe cargarse en memoria. Esto se lleva a cabo mediante el *cargador*, el cual toma el archivo ejecutable del disco y lo transfiere a la memoria. Finalmente, el computador, bajo el control del CPU, ejecuta el programa instrucción por instrucción.

6.5. Entornos de Desarrollo

Para facilitar el desarrollo de programas escritos en el lenguaje C existen varios entornos de desarrollo. Algunos programadores prefieren utilizar los ambientes de desarrollo no integrados, donde el editor, el compilador y el depurador se invocan en forma independiente desde la línea de comandos, como GCC (GNU C Compiler)[15]. Otros programadores prefieren utilizar un ambiente de desarrollo integrado orientado a ventanas, como Microsoft Visual C++ 2013 Express Edition[16].

6.6. Estructuras de Control

Una estructura de control es una estructura de programación que permite tomar decisiones basándose en la evaluación de una condición.

Por lo general, las instrucciones dentro de un programa se ejecutan secuencialmente, es decir, se ejecutan en el orden en el cual se escribieron. Esto se conoce como ejecución secuencial. Varias de las estructuras de control en C permiten al programador especificar que la instrucción a ejecutar no es la siguiente sino otra. A esto se le denomina *transferencia de control*.

El lenguaje C proporciona tres tipos de estructuras de selección: **if**, **if/else** y **switch**. La estructura de selección **if** realiza una o varias acciones si alguna condición especificada por el programador es verdadera o las ignora si la condición es falsa. La estructura de selección **if/else** realiza una o varias acciones si la condición es verdadera y realiza otras acciones diferentes si la condición es falsa. La estructura de selección **switch** realiza una o varias acciones dependiendo del valor de una expresión entera.

El lenguaje C proporciona tres tipos de estructuras de repetición: **while**, **do/while** y **for**. La estructura de repetición **while** le permite al programador repetir una acción o varias

[15] http://www.gcc.gnu.org
[16] http://www.microsoft.com/express/vc

acciones mientras alguna condición sea verdadera. La estructura de repetición **do/while** evalúa la condición después de que el cuerpo del ciclo se ejecuta, es decir, el cuerpo del ciclo se ejecuta al menos una vez. La estructura de repetición **for** maneja todos los detalles de una repetición controlada por un contador.

6.6.1. Estructura de Selección if

La estructura de selección **if** es una estructura de selección simple. El **if** evalúa una condición (`condition`). Si la condición es verdadera se ejecuta `sentenceT` y el flujo sigue a la siguiente sentencia después del **if**. En caso contrario, no se ejecuta `sentenceT` y el flujo sigue a la siguiente sentencia después del **if** [4].

La forma general de un **if** es:

```
if (condition)
  sentenceT;
```

La Figura 6.1 muestra el diagrama de flujo del **if**.

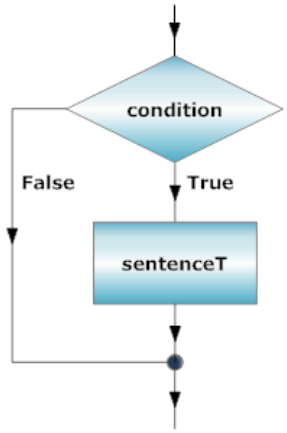

Figura 6.1: Diagrama de Flujo del if

Se pudiera reemplazar `sentenceT` por un conjunto de sentencias `sentenceT1`, `sentenceT2`, `...`, `sentenceTN`, todas estas sentencias encerradas entre llaves, como se puede observar a continuación:

```
if (condition) {
  sentenceT1;
  sentenceT2;
  ...

  sentenceTN;
}
```

La Figura 6.2 muestra el diagrama de flujo del **if** con varias sentencias.

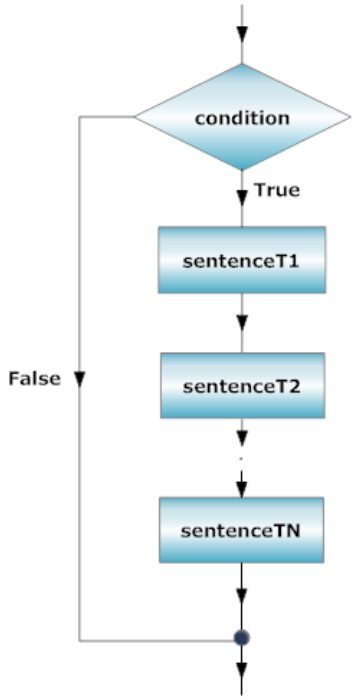

Figura 6.2: Diagrama de Flujo del if con varias sentencias

6.6.2. Estructura de Selección if/else

La estructura de selección **if/else** es una estructura de selección doble. El **if/else** evalúa una condición proporcionada por el programador (`condition`). Si la condición es verdadera se ejecuta una sentencia `sentenceT`, y en caso contrario se ejecuta otra sentencia `sentenceF`. Finalmente, sea cual sea el resultado de evaluar la condición, después de ejecutar la sentencia o el conjunto de sentencias correspondiente el flujo sigue a la siguiente sentencia después del **if/else**.

La forma general de un **if/else** es:

```
if (condition)
  sentenceT;
else
  sentenceF;
```

La Figura 6.3 muestra el diagrama de flujo del **if/else**.

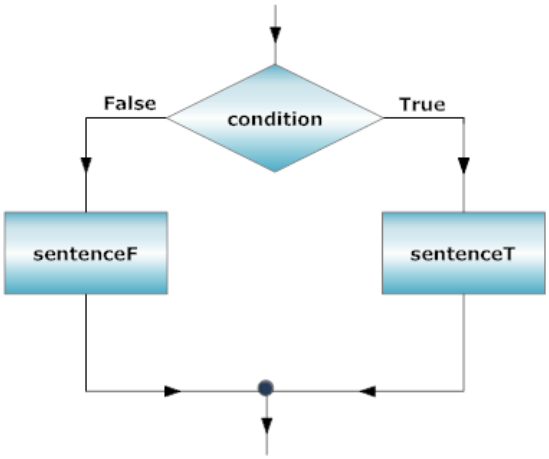

Figura 6.3: Diagrama de Flujo del if/else

Se pudiera reemplazar `sentenceT` y/o `sentenceF` por un conjunto de sentencias encerradas entre llaves como se puede observar a continuación:

Caso 1:
Reemplazando `sentenceF` con varias sentencias:

```
if (condition)
  sentenceT;
else {
  sentenceF1;
  sentenceF2;
  ...
  sentenceFN;
}
```

Caso 2:
Reemplazando `sentenceT` con varias sentencias:

```
if (condition) {
  sentenceT1;
  sentenceT2;
  ...
  sentenceTN;
} else
  sentenceF;
```

Caso 3:
Reemplazando `sentenceT` y `sentenceF` con varias sentencias:

```
if (condition) {
  sentenceT1;
  sentenceT2;
  ...
  sentenceTN;
} else {
  sentenceF1;
  sentenceF2;
  ...
  sentenceFN;
}
```

6.6.3. Estructura de Selección Múltiple switch

La estructura de selección **switch** es una *estructura de selección múltiple*; esta estructura selecciona una acción o un conjunto de acciones a realizar a partir de muchos **case** diferentes. La forma general de un **switch** es:

```
switch (expression) {
  case C1:
    sentenceT1;
  case C2:
    sentenceT2;
  case C3:
    sentenceT3;
  ...
  case CN:
    sentenceTN;
  default:
    sentenceDefault;
}
```

La Figura 6.4 muestra el diagrama de flujo correspondiente.

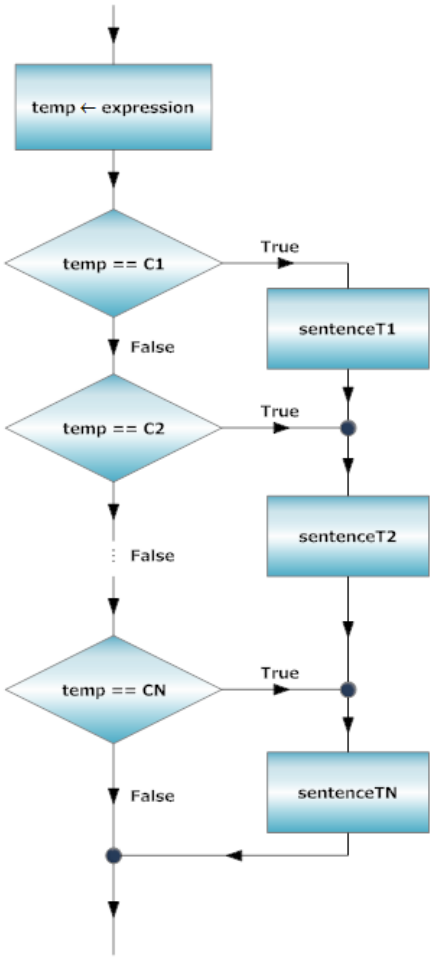

Figura 6.4: Diagrama de Flujo del switch

6.6.4. Estructura de Repetición while

Una estructura de repetición **while** permite que el programador especifique que se debe repetir una o varias acciones mientras alguna condición se cumpla. La forma general de un **while** es:

```
while (condition)
  sentenceT;
```

La Figura 6.5 muestra el diagrama de flujo del **while**.

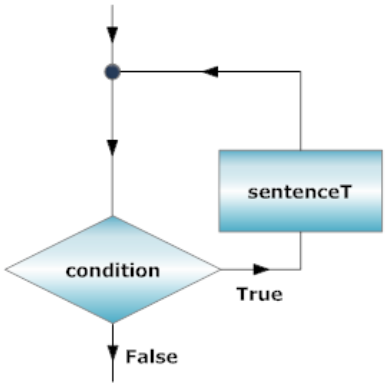

Figura 6.5: Diagrama de Flujo del while

6.6.5. Estructura de Repetición do/while

La forma general de un **do/while** es:

```
do
  sentenceT;
while (condition);
```

La estructura de repetición **do/while** evalúa la condición (`condition`) después de la primera ejecución de su cuerpo, por lo tanto, el cuerpo del ciclo (`sentenceT`) se ejecuta al menos una vez. Cuando un **do/while** termina el flujo sigue a la sentencia que viene inmediatamente después del ciclo. Los ciclos **do/while** se pueden anidar con otras estructuras de control. La Figura 6.6 muestra el diagrama de flujo correspondiente.

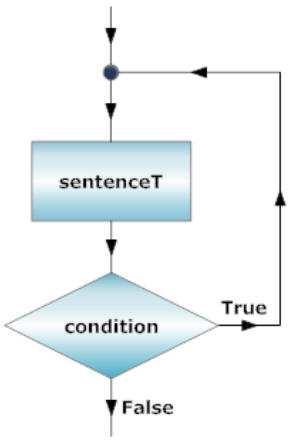

Figura 6.6: Diagrama de Flujo del do/while

6.6.6. Estructura de Repetición for

Una estructura de repetición **for** permite repetir una o varias acciones mientras alguna condición se cumpla, permitiendo añadir una sentencia de inicialización y una sentencia de incremento. La forma general de un **for** es:

```
for ([initialization]; [condition]; [increment])
  sentenceT;
```

Una estructura de repetición **for** está compuesta por cuatro partes:

Initialization

Se ejecuta una sola vez al principio y permite inicializar las variables del ciclo.

Condition

Se evalúa la condición antes de ejecutar el cuerpo del ciclo **for**. Si la condición es verdadera entonces se ejecuta el cuerpo del ciclo **for**, si no el flujo sigue a la sentencia que viene inmediatamente después del ciclo. Esta condición es opcional; si esta parte está vacía, entonces se asume que la condición es siempre verdadera.

Increment

Se especifica cómo tienen que modificarse las variables antes de ir a la siguiente iteración, es decir, se especifica el incremento o decremento de las variables.

Cuerpo del ciclo for

El cuerpo del ciclo **for** en este caso es `sentence`. Se ejecuta este cuerpo si y sólo si la condición es verdadera.

Los ciclos **for** se pueden anidar con otras estructuras de control. La Figura 6.7 muestra el diagrama de flujo del **for**.

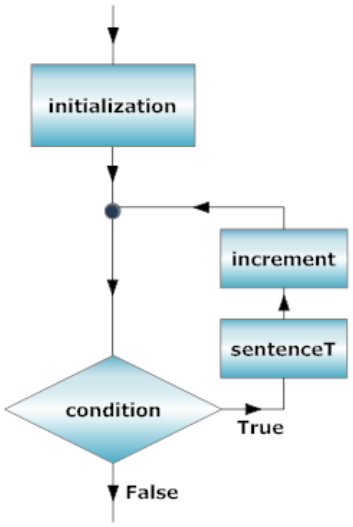

Figura 6.7: Diagrama de Flujo del for

6.6.7. Instrucciones break y continue

Las instrucciones **break** y **continue** alteran el flujo del control.

Cuando la instrucción **break** se ejecuta en una estructura **while**, **for**, **do/while** o **switch**, provoca la salida inmediata de dicha estructura. La ejecución del programa continúa con la primera instrucción después de la estructura. Los usos más comunes de la instrucción **break** son salir de un ciclo de manera anticipada o salir del resto de una estructura **switch**.

Cuando se ejecuta la instrucción **continue** en una estructura **while**, **for** o **do/while**, ignora el resto de las instrucciones en el cuerpo de esa estructura y procede con la siguiente iteración del ciclo. En las estructuras **while** y **do/while** la condición del ciclo se evalúa después de que se ejecuta la instrucción **continue**. En la estructura **for** se ejecuta la expresión del incremento y, a continuación, se evalúa la condición del ciclo.

6.7. Arreglos

Un arreglo es una colección de elementos del mismo tipo de dato. Todos los elementos de un arreglo ocupan posiciones consecutivas en la memoria, es decir, el primer elemento de un arreglo se encuentra en la dirección más baja de la memoria y el último elemento de un arreglo se encuentra en la dirección más alta de la memoria. Para acceder a un elemento del arreglo es necesario el uso de índices.

La forma general para declarar un arreglo unidimensional es:

```
tipo nombreArreglo[cantidad];
```

La forma general para declarar un arreglo bidimensional es:

```
tipo nombreArreglo[cantidadFilas][cantidadColumnas];
```

La forma general para declarar un arreglo multidimensional es:

```
tipo nombreArreglo[cantidad1][cantidad2]…[cantidadN];
```

Donde:

`tipo`: Tipo de dato de cada elemento del arreglo.
`nombreArreglo`: Nombre del arreglo.
`cantidad`: Cantidad de elementos que tiene el arreglo.
`cantidadFilas`: Cantidad de elementos de la dimensión1 (filas).
`cantidadColumnas`: Cantidad de elementos de la dimensión2 (columnas).
`cantidad1`: Cantidad de elementos de la primera dimensión del arreglo.
`cantidad2`: Cantidad de elementos de la segunda dimensión del arreglo.
`cantidadN`: Cantidad de elementos de la enésima dimensión del arreglo.

6.8. Apuntadores

Además de acceder a un elemento del arreglo utilizando índices, se puede acceder al mismo elemento utilizando apuntadores. Un apuntador es una variable que contiene una dirección de memoria. Por ser una variable es necesario declararla e inicializarla. La forma general para declarar un apuntador es:

```
tipo *nombreApuntador;
```

Donde:

`tipo`: Tipo de dato del elemento a ser apuntado por el apuntador.
`nombreApuntador`: Nombre del apuntador.

6.9. Funciones

La mayoría de los programas que resuelven problemas reales son mucho más grandes que los ejemplos presentados hasta el momento. La mejor manera de desarrollar y mantener un programa grande es dividiendo el problema en subproblemas de complejidad menor. Cada subproblema puede ser resuelto a través de funciones, las cuales consisten en bloques de instrucciones que resuelven subproblemas específicos.

Las funciones en el lenguaje C están compuestas por el prototipo y el cuerpo, las cuales se explican a continuación.

6.9.1. El Prototipo de una Función

El prototipo de la función le indica al compilador el nombre de la función, el tipo de dato del valor devuelto por esa función, el número de argumentos que la función espera recibir, el tipo de dato de cada argumento y el orden en que se esperan esos argumentos. El compilador utiliza los prototipos de funciones para verificar las llamadas de función. El identificador de una función debe seguir las reglas de creación de identificadores en C.

La lista de los argumentos, llamados argumentos formales, puede estar vacía si no hay argumentos. Los argumentos formales están separados por comas. Para especificar un argumento formal primero se escribe el tipo de dato del argumento, seguido por el identificador que se utilizará en la función para referenciarlo. Los arreglos también pueden ser pasados como argumentos formales, pero una función no puede devolver un arreglo.

6.9.2. El Cuerpo de una Función

El cuerpo de una función es un bloque de instrucciones que comienza con una llave abierta (`{`) y termina con una llave cerrada (`}`). El cuerpo de una función está compuesto por: la declaración de las variables locales, las cuales tienen un alcance hasta el final del bloque donde están declaradas, y las instrucciones a ejecutar.

La instrucción **return** permite salir de una función, opcionalmente devolviendo un valor. Al salir de una función, la ejecución del programa continúa a la instrucción siguiente a la llamada de la función.

La forma general para declarar una función es:

```
tipo nombreFuncion(listaArgumentos) {
  /* Declaración de las variables */
  /* Instrucciones por ejecutar */
}
```

Donde:

`tipo`: Tipo de dato que devuelve la función. Una función puede devolver cualquier tipo de dato excepto un arreglo. Si la función no devuelve ningún valor, el tipo de retorno debe ser `void`.

`nombreFuncion`: Nombre de la función, debe cumplir las reglas de los identificadores del lenguaje C.

`listaArgumentos`: Lista de nombres de variables separados por coma con sus respectivos tipos de datos.

El lenguaje de programación C, a diferencia de otros lenguajes, sólo permite el pase de parámetros por valor, el cual copia el valor del argumento en el parámetro formal de la

subrutina. Para simular un pase de parámetros por referencia se pasa como parámetro formal un apuntador a la variable que contiene el argumento.

6.10. Caracteres y Cadenas

Un carácter es cualquier signo tipográfico, puede ser una letra, un número, un signo de puntuación o un espacio. Una cadena es un arreglo de caracteres (`char`) finalizado con el carácter nulo (`'\0'`), es decir, una cadena contiene caracteres y el último elemento de la cadena es un cero.

La forma general para declarar una cadena es:

```
char nombreCadena[cantidad];
```

Donde:

`char`: Indica que el tipo de dato de cada elemento de la cadena es un `char`.
`nombreCadena`: Nombre del arreglo.
`cantidad`: La cantidad de elementos que tiene el arreglo.

6.11. Archivos

En el lenguaje C existen dos tipos de archivos: archivos binarios y archivos de texto. Un archivo binario es una colección de bytes, y un archivo de texto es una colección de caracteres organizados en líneas terminadas con el carácter salto de línea (`'\n'`). Los archivos se utilizan como medio de almacenamiento permanente puesto que el almacenamiento de datos en variables, arreglos y cualquier otra estructura es temporal.

Para leer o escribir sobre archivos los programas tienen que abrir el archivo (`fopen`), realizar el procedimiento de lectura (`getc/fgetc`) y/o escritura (`putc/fputc`) y cerrar (`fclose`) el archivo. Para disponer de una variable apuntador a archivo se utiliza la siguiente instrucción:

```
FILE *fp;
```

El prototipo de la función `fopen` es:

```
FILE *fopen(const char *nombreArchivo, const char *modo);
```

Donde:

`nombreArchivo`: Apuntador a la cadena que contiene el nombre del archivo.
`modo`: Apuntador a la cadena que determina cómo se abre el archivo.

Antes de intentar cualquier operación sobre un archivo se debe confirmar que `fopen` haya tenido éxito.

El prototipo de la función `fclose` es:

```
int fclose(FILE *stream);
```

Donde:

`stream`: Es el apuntador al archivo devuelto por la llamada `fopen`. Si el valor de retorno es igual a cero (0) indica que el cierre del archivo ha sido exitoso. En caso contrario, el valor de retorno será `EOF`[17].

El prototipo de la función `putc` es:

```
int putc(int c, FILE *stream);
```

Donde:

`c`: Es el carácter a escribir.

`stream`: Es el apuntador al archivo devuelto por la llamada `fopen`. Si la operación `putc` tiene éxito, ésta retorna el carácter escrito. En caso contrario, el valor de retorno será `EOF`.

El prototipo de la función `getc` es:

```
int getc(FILE *stream);
```

Donde:

`stream`: Es el apuntador al archivo devuelto por la llamada `fopen`. La función `getc` devuelve un entero, estando el carácter en el byte menos significativo. A menos que se produzca un error, el byte más significativo es cero (0).

La función `getc` devuelve `EOF` cuando se ha alcanzado el final del archivo o si se produce algún error.

[17] `EOF`, definida en `<stdio.h>` y siendo normalmente –1, es el valor de retorno cuando una función de entrada intenta leer más allá del final del archivo.

7. Creación de Cursos

En este capítulo se describen los distintos archivos que debe crear el instructor para generar un curso utilizando el generador implementado en este Trabajo Especial de Grado, junto con el lenguaje de etiquetado XML que se debe utilizar para cada uno de estos archivos.

Para ejemplificar este proceso se utilizará como caso de estudio un curso prototipo con una Introducción a la Programación en el Lenguaje C, el cual contiene lecciones y quices ilustrando los conceptos descritos en el Capítulo 6.

En las siguientes secciones se describe la estructura general del curso, así como los diversos módulos que lo componen, incluyendo los elementos que el instructor debe incluir para instanciar un nuevo curso.

7.1. Estructura General del Curso Prototipo

El curso está compuesto por tres módulos principales: lecciones, quices y glosario. Cada uno de ellos posee un punto de entrada constante descrito por un archivo XML que contiene los elementos relacionados a dicho módulo.

La interfaz gráfica permite navegar a través de estos módulos utilizando botones ubicados en la esquina inferior izquierda y permite navegar entre distintos elementos del módulo actual utilizando botones de navegación en la esquina inferior derecha.

Cada página tiene la opción de incluir un archivo de audio asociado con el contenido narrado de la misma, lo cual permitirá a los usuarios escuchar el contenido en lugar de leerlo; para interactuar con este archivo de audio se pueden utilizar controles en la barra inferior, los cuales aparecerán únicamente si hay una grabación disponible (ver Figura 7.1).

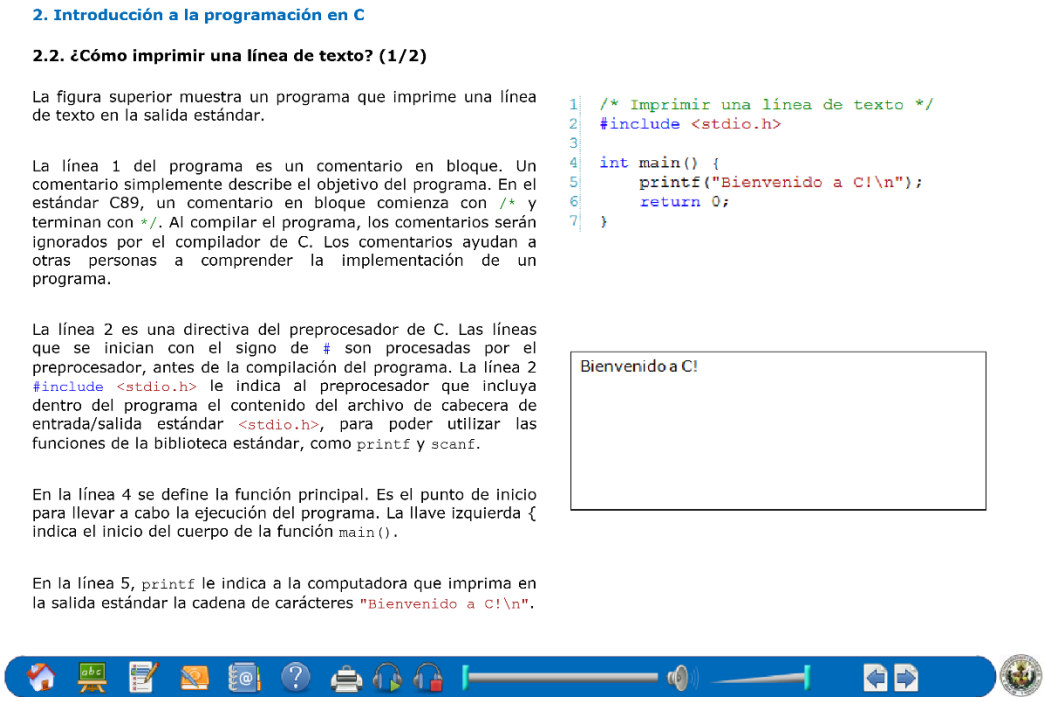

Figura 7.1: Botones de Navegación

7.2. Módulo de Lecciones

El archivo XML principal correspondiente al módulo de lecciones se debe llamar “course.xml”, el cual contiene una lista de todos los capítulos del curso, donde a su vez cada capítulo se subdivide en una o más lecciones. Cada una de estas lecciones está definida en su propio archivo XML, el cual contiene una o más páginas a mostrar al alumno. A continuación se describe la estructura de estos archivos.

7.2.1. Definición de los Capítulos disponibles en el Curso

Los quices disponibles en el curso se especifican mediante un archivo XML con un único nodo de tipo “course”, el cual contiene una lista de nodos del tipo “chapter”, los cuales a su vez contienen una lista de nodos del tipo “lesson”, que referencian a los archivos con el contenido de cada lección.

En la siguiente sección se describen los diferentes nodos que componen este archivo.

Nodo curso

Etiqueta: “course”

Atributos:

Nombre	Requerido?	Descripción
title	Sí	Título del curso a mostrar al estudiante en el índice de lecciones.

Tabla 7.1: Atributos de los nodos del tipo curso

Nodo capítulo

Etiqueta: “chapter”

Atributos:

Nombre	Requerido?	Descripción
name	Sí	Nombre del capítulo.

Tabla 7.2: Atributos de los nodos del tipo capítulo

Nodo lección

Etiqueta: “lesson”

Atributos:

Nombre	Requerido?	Descripción
name	Sí	Nombre de la lección.
file	Sí	Ruta al archivo con el contenido de la lección.

Tabla 7.3: Atributos de los nodos para hacer referencia a lecciones

En la Figura 7.2 se visualiza el archivo XML con la lista de capítulos y lecciones sobre el Lenguaje C disponibles en el curso prototipo.

course.xml

```xml
<course title="Capítulos">
    <chapter name="1. Conceptos de computación">
        <lesson name="1.1. Objetivos" file="chapter1/1_1_objetivos.xml"/>
        <lesson name="1.2. ¿Qué es una computadora?" file="chapter1/1_2_computadora.xml"/>
        <lesson name="1.3. Organización de la computadora" file="chapter1/1_3_organizacion.xml"/>
        <lesson name="1.4. La historia de C" file="chapter1/1_4_historia.xml"/>
        <lesson name="1.5. Las caracteristicas de C" file="chapter1/1_5_caracteristicas.xml"/>
        <lesson name="1.6. Fases para ejecutar un programa escrito en C" file="chapter1/1_6_fases.xml"/>
        <lesson name="1.7. Entornos de desarrollo para C" file="chapter1/1_7_entorno-desarrollo.xml"/>
        <lesson name="1.8. Resumen" file="chapter1/1_8_resumen.xml"/>
    </chapter>
    <chapter name="2. Introducción a la programación en C">
        <lesson name="2.1. Objetivos" file="chapter2/2_1_objetivos.xml"/>
        <lesson name="2.2. ¿Cómo imprimir una línea de texto?" file="chapter2/2_2_primer-programa.xml"/>
        <lesson name="2.3. ¿Cómo sumar dos enteros?" file="chapter2/2_3_segundo-programa.xml"/>
        <lesson name="2.4. Variables e Identificadores" file ="chapter2/2_4_variablesIdentificadores.xml"/>
        <lesson name="2.5. Memoria" file="chapter2/2_5_memoria.xml"/>
        <lesson name="2.6. Aritmética" file="chapter2/2_6_aritmetica.xml"/>
        <lesson name="2.7. Toma de decisiones" file="chapter2/2_7_decisiones.xml"/>
        <lesson name="2.8. Operadores de asignación aritméticos" file="chapter2/2_8_operadores-asignacion-aritmeticos.xml"/>
        <lesson name="2.9. Operadores de incremento y decremento" file="chapter2/2_9_operadores-incremento-decremento.xml"/>
        <lesson name="2.10. Resumen" file="chapter2/2_10_resumen.xml"/>
        <lesson name="2.11. Videos" file="chapter2/2_11_videos.xml"/>
    </chapter>
    <chapter name="3. Estructuras de control de selección">
        <lesson name="3.1. Objetivos" file="chapter3/3_1_objetivos.xml"/>
        <lesson name="3.2. Algoritmos" file="chapter3/3_2_algoritmos.xml"/>
        <lesson name="3.3. Estructuras de control" file="chapter3/3_3_estructuras-control.xml"/>
        <lesson name="3.4. La estructura de selección if" file="chapter3/3_4_if.xml"/>
        <lesson name="3.5. La estructura de selección if/else" file="chapter3/3_5_if-else.xml"/>
        <lesson name="3.6. La estructura de selección multiple switch" file="chapter3/3_6_switch.xml"/>
        <lesson name="3.7. Resumen" file="chapter3/3_7_resumen.xml"/>
        <lesson name="3.8. Videos" file="chapter3/3_8_videos.xml"/>
    </chapter>
```

Figura 7.2: Archivo XML correspondiente a un Curso

En la Figura 7.3 se muestra cómo se despliega la lista de lecciones cuando el estudiante ingresa al módulo de lecciones.

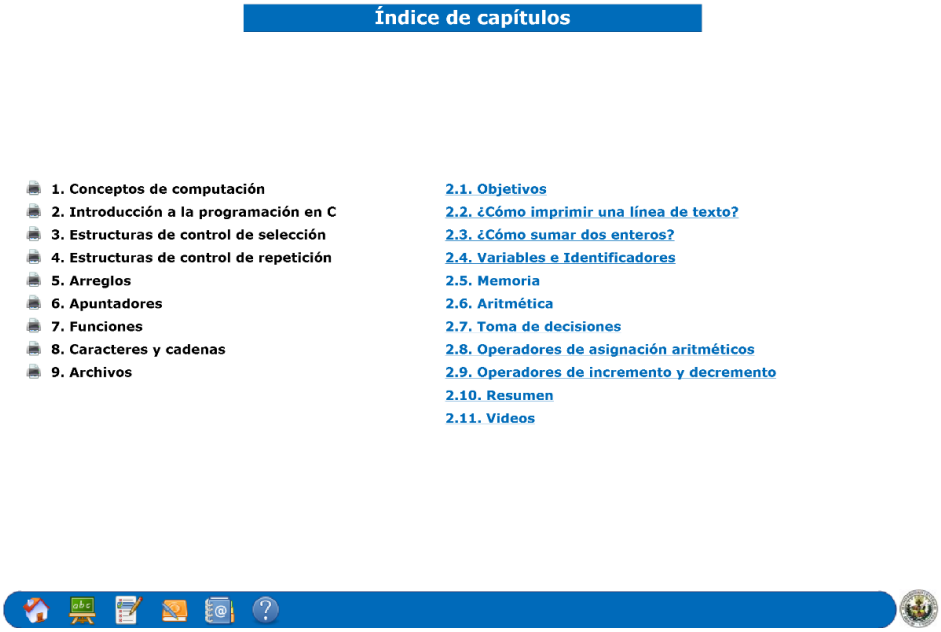

Figura 7.3: Lista de Lecciones del Curso Prototipo

7.2.2. Definición del Contenido de una Lección

El contenido de una lección se específica mediante un archivo XML con un único nodo de tipo “lesson”, el cual contiene una lista de nodos del tipo “page”, donde cada uno de estos nodos tiene el contenido a mostrar en cada página de la lección.

Las páginas de las lecciones soportan diversos elementos multimedia, los cuales se despliegan en la aplicación de diversas maneras, dependiendo del tipo de elemento:

1. **Texto:** El instructor debe especificar el ancho de todos los párrafos de la página como un atributo de la misma. Luego, cada párrafo es desplegado sucesivamente uno debajo de otro, a menos que el instructor utilice un elemento para mover párrafo, el cual indica la posición donde el siguiente párrafo debe ser desplegado.

2. **Imágenes:** El instructor debe proveer las coordenadas donde se debe desplegar la imagen, así como la ruta de la imagen a mostrar.

3. **Video:** El instructor debe proveer las coordenadas donde se debe desplegar el video, así como las dimensiones del mismo y la ruta del video a reproducir.

4. **Sonido:** El instructor debe proveer la ruta del audio a reproducir, el cual se puede manipular a través de los controles en la parte inferior de la página. Se puede especificar máximo un archivo de audio por página.

En la siguiente sección se describen los diferentes nodos que componen este archivo.

Nodo lección

Etiqueta: “lesson”

Atributos:

Nombre	Requerido?	Descripción
name	Sí	Texto a mostrar en el encabezado de todas las páginas de la lección.

Tabla 7.4: Atributos de los nodos para definir lecciones

Nodo página

Etiqueta: “page”

Atributos:

Nombre	Requerido?	Descripción
name	Sí	Texto a mostrar como título de esta página.

Tabla 7.5: Atributos de los nodos del tipo página

Nodo párrafo

Etiqueta: “paragraph”

Atributos:

Nombre	Requerido?	Descripción
text	Sí	Texto a mostrar en la página. Puede contener elementos HTML, los cuales serán formateados correctamente en la página.

Tabla 7.6: Atributos de los nodos del tipo párrafo

Nodo mover párrafo

Etiqueta: “move_paragraph”

Atributos:

Nombre	Requerido?	Descripción
x	Sí	Coordenada x donde se debe mostrar el siguiente párrafo en la página.
y	Sí	Coordenada y donde se debe mostrar el siguiente párrafo en la página.

Tabla 7.7: Atributos de los nodos del tipo mover párrafo

Nodo imagen

Etiqueta: “image”

Atributos:

Nombre	Requerido?	Descripción
file	Sí	Ruta del archivo con la imagen a mostrar.
x	Sí	Coordenada x donde se debe mostrar la imagen.
y	Sí	Coordenada y donde se debe mostrar la imagen.

Tabla 7.8: Atributos de los nodos del tipo imagen

Nodo video

Etiqueta: “video”

Atributos:

Nombre	Requerido?	Descripción
file	Sí	Ruta del archivo con el video a mostrar.
x	Sí	Coordenada x donde se debe mostrar el video.
y	Sí	Coordenada y donde se debe mostrar el video.
width	Sí	Ancho del video a mostrar en píxeles.
height	Sí	Altura del video a mostrar en píxeles.

Tabla 7.9: Atributos de los nodos del tipo video

Nodo audio

Etiqueta: “sound”

Atributos:

Nombre	Requerido?	Descripción
file	Sí	Ruta del archivo de audio a reproducir.

Tabla 7.10: Atributos de los nodos del tipo audio

La **Figura 7.4** muestra el ejemplo de un archivo XML correspondiente a una lección, la cual contiene texto, imágenes y sonido.

1_2_computadora.xml

```
<?xml version='1.0' encoding='utf-8'?>

<lesson name="1. Conceptos de computación">
    <page name="1.2. ¿Qué es una computadora?">
        <paragraph text="Una computadora es una máquina que recibe datos, los procesa y genera resultados. El procesamiento realizado consiste en cálculos matemáticos y
        la toma de decisiones lógicas."/>
        <paragraph text="Las computadoras procesan los datos bajo el control de un conjunto de instrucciones llamado &lt;i&gt;Programa&lt;/i&gt;. Los programas guían a la
        computadora a través de instrucciones especificadas por humanos, quienes son llamados &lt;i&gt;Programadores&lt;/i&gt;."/>
        <paragraph text="Una computadora está compuesta por varios dispositivos, tales como: (1) monitor, (2) tarjeta madre, (3) procesador, (4) memoria RAM, (5) tarjetas
        de expansión, (6) fuente de poder, (7) unidad de CD/DVD, (8) disco duro, (9) teclado y (10) ratón. Este conjunto de dispositivos son conocidos como
        &lt;i&gt;Hardware&lt;/i&gt;. A los programas que se ejecutan dentro de una computadora se les denomina &lt;i&gt;Software&lt;/i&gt;."/>
        <image file="chapter1/images/1_2_computadora.png" x="600" y="125"/>
        <sound file="chapter1/sounds/1_2.mp3"/>
    </page>
</lesson>
```

Figura 7.4: Archivo XML correspondiente a una Lección

En la **Figura 7.5** se muestra cómo se despliega el contenido de una lección cuando el estudiante ingresa al módulo de lecciones.

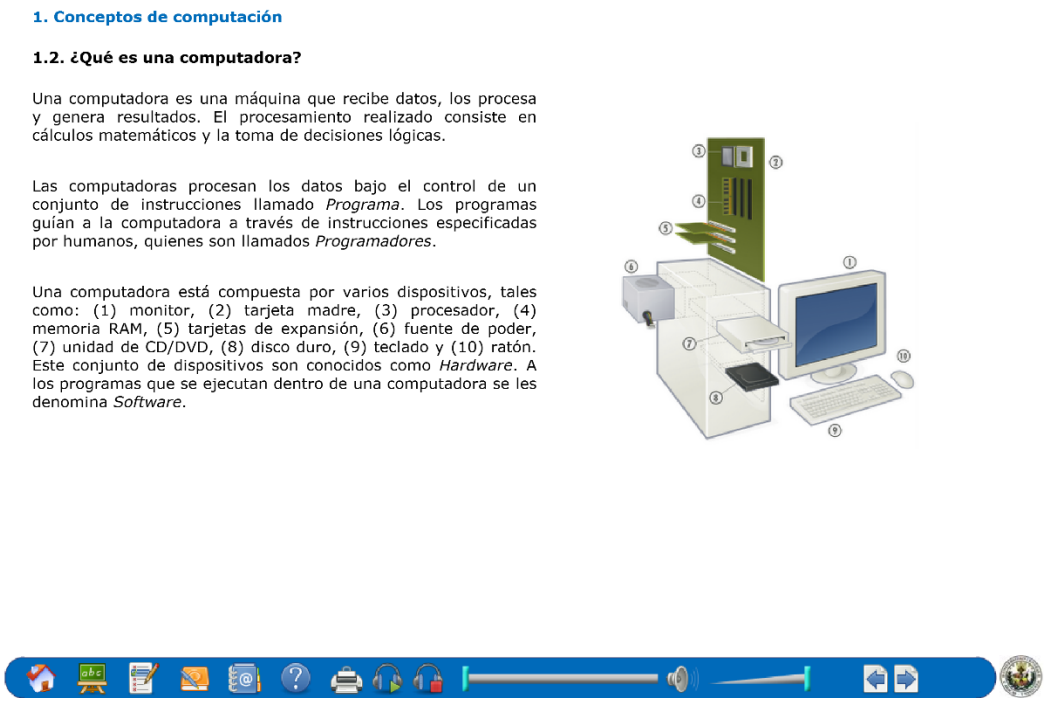

Figura 7.5: Interfaz del Estudiante visualizando una Lección

7.3. Módulo de Quices

El archivo XML principal correspondiente al módulo de quices se debe llamar “quices.xml”, el cual contiene una lista con referencias a los quices disponibles en el curso. Cada uno de estos quices está descrito en un archivo diferente, el cual contiene todas las preguntas a presentarse al estudiante. A continuación se describe la estructura de estos archivos.

7.3.1. Definición de los Quices disponibles en el Curso

Los quices disponibles en el curso se especifican mediante un archivo XML con un único nodo de tipo “quices”, el cual contiene una lista de nodos del tipo “quiz”, los cuales referencian los archivos con el contenido de cada quiz.

En la siguiente sección se describen los diferentes nodos que componen este archivo.

Nodo quices

Etiqueta: “quices”

Atributos:

Nombre	Requerido?	Descripción
title	Sí	Contiene el título a mostrar al usuario cuando se esté visualizando la lista de quices disponibles.

Tabla 7.11: Atributos de los nodos del tipo quices

Nodo quiz

Etiqueta: “quiz”

Atributos:

Nombre	Requerido?	Descripción
name	Sí	Nombre del quiz a mostrar en la lista de quices disponibles.
file	Sí	Ruta al archivo con el contenido del quiz. Esta es relativa al archivo SWF del curso.

Tabla 7.12: Atributos de los nodos para hacer referencia a quices

En la **Figura 7.6** se visualiza un ejemplo de este tipo de archivo, el cual muestra una lista de quices disponibles al estudiante evaluando cada capítulo del curso.

quices.xml

```
<?xml version='1.0' encoding='utf-8'?>
<quices title="Índice de quices">
    <quiz name="1. Quiz sobre Conceptos de computación" file="quiz1/quiz.xml"/>
    <quiz name="2. Quiz sobre Introducción a la programación en C" file="quiz2/quiz.xml"/>
    <quiz name="3. Quiz sobre Estructuras de control de selección" file="quiz3/quiz.xml"/>
    <quiz name="4. Quiz sobre Estructuras de control de repetición" file="quiz4/quiz.xml"/>
    <quiz name="5. Quiz sobre Arreglos" file="quiz5/quiz.xml"/>
    <quiz name="6. Quiz sobre Apuntadores" file="quiz6/quiz.xml"/>
    <quiz name="7. Quiz sobre Funciones" file="quiz7/quiz.xml"/>
    <quiz name="8. Quiz sobre Caracteres y cadenas" file="quiz8/quiz.xml"/>
    <quiz name="9. Quiz sobre Archivos" file="quiz9/quiz.xml"/>
</quices>
```

Figura 7.6: Archivo XML correspondiente a la Lista de Quices

En la **Figura 7.7** se muestra cómo se despliega la lista de quices cuando el estudiante ingresa al módulo de quices.

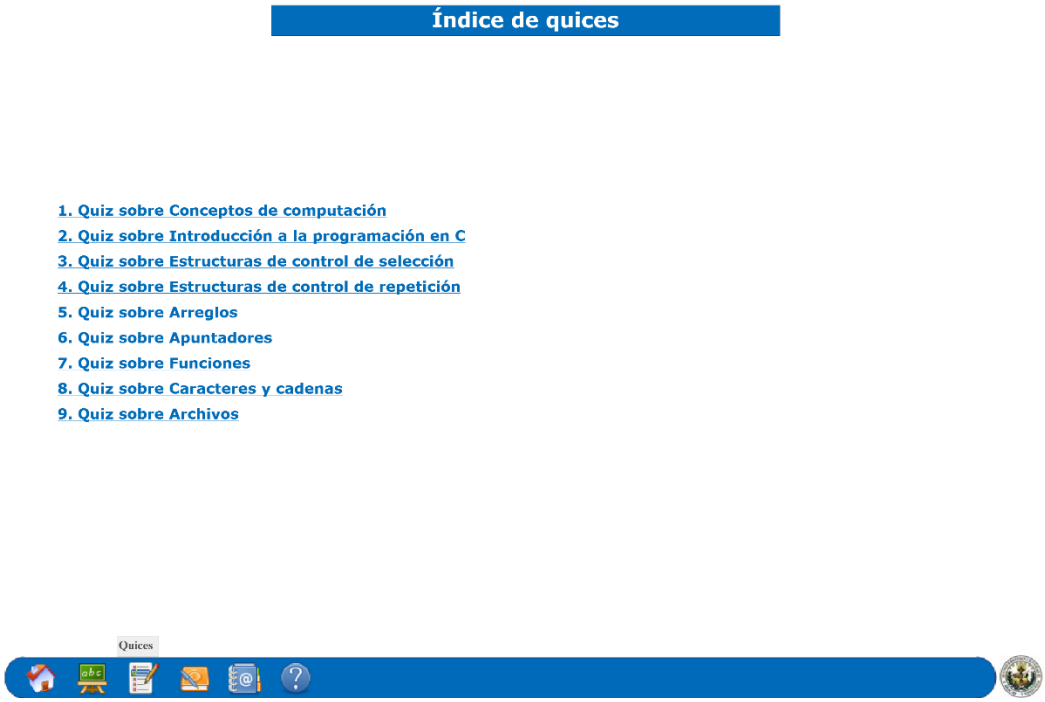

Figura 7.7: Lista de Quices del Curso Prototipo

7.3.2. Definición del Contenido de un Quiz

El contenido de un quiz se específica mediante un archivo XML con un único nodo de tipo “quiz”, el cual contiene todas las preguntas a realizarse al estudiante, junto con todas las respuestas posibles para cada pregunta.

Todos los tipos de preguntas se especifican utilizando el mismo tipo de nodo y se diferencian mediante el uso del atributo “type”. Existen cuatro tipos de preguntas, y los nodos hijos de la pregunta se interpretan dependiendo de éste tipo:

1. **Selección simple** (type=”simple”): Cada hijo debe ser un nodo respuesta, el cual indica si la respuesta es correcta o no. Sólo uno de estos hijos debe estar marcado como correcto.

2. **Selección múltiple** (type=”multiple”): Cada hijo debe ser un nodo respuesta, el cual indica si la respuesta está dentro del conjunto de respuestas correctas. Al menos uno de estos hijos debe ser marcado como correcto.

3. **Arrastrar y soltar** (type=”drag”): Cada hijo debe ser un nodo *token*, donde se asume que los nodos son especificados en el orden correcto en el que el estudiante los debe colocar.

4. **Llenado de espacios en blanco** (type=”fill”): Cada hijo debe ser un nodo respuesta, donde cada uno indica una posible respuesta correcta a la pregunta.

En la siguiente sección se describen los diferentes nodos que componen este archivo.

Nodo quiz

Etiqueta: “quiz”

Atributos:

Nombre	Requerido?	Descripción
title	Sí	Contiene el título a mostrar al usuario mientras resuelve el quiz.

Tabla 7.13: Atributos de los nodos para definir quices

Nodo pregunta

El nodo pregunta se utiliza para especificar una sola pregunta de cualquier tipo dentro de un quiz. El tipo de la pregunta se determina a partir del atributo type, el cual indica cómo se deben interpretar el resto de los atributos e hijos de este nodo.

Etiqueta: “question”

Atributos:

Nombre	Requerido?	Descripción
text	Sí	Texto con la pregunta a formular al estudiante.
type	Sí	Tipo de la pregunta. Debe ser uno de “simple”, “multiple”, “drag” o “fill”.
ref	Sí	Texto con una referencia a la lección donde se explica la respuesta correcta.
correct	Sólo para preguntas de tipo “drag” o “fill”.	Texto con la respuesta correcta para la pregunta.
correct_count	No.	Número de elementos en la respuesta correcta. De lo contrario se asume que

		es igual al número de hijos.
slot	Sólo para preguntas de arrastrar y soltar.	Imagen a utilizar como destino donde las respuestas pueden ser arrastradas.

Tabla 7.14: Atributos de los nodos del tipo pregunta

Nodo respuesta

Etiqueta: “answer”

Atributos:

Nombre	Requerido?	Descripción
text	Sólo para preguntas de selección simple o múltiple.	Texto a mostrar como posible respuesta para una pregunta de selección simple o múltiple.
correct	No	Debe ser uno entre “yes” o “no”, indicando si la respuesta es correcta. Si no es especificado se asume que la respuesta es incorrecta.
pattern	Sólo para preguntas de llenado de espacios en blanco.	Expresión regular para verificar si la respuesta ingresada por el usuario es correcta o no.
options	No	Opciones a utilizar para ejecutar la expresión regular. Si no es especificada no se especifica ninguna opción a la hora de ejecutar la expresión regular.

Tabla 7.15: Atributos de los nodos del tipo respuesta

Nodo token

Etiqueta: “token”

Atributos:

Nombre	Requerido?	Descripción

File	Sí	Ruta al archivo con la imagen a utilizar como objeto a arrastrar por el usuario.

Tabla 7.16: Atributos de los nodos del tipo token

En la **Figura 7.8** se puede observar el archivo XML correspondiente al quiz de un capítulo, el cual contiene diversos ejemplos de preguntas, entre ellas, selección simple, múltiple y arrastrar y soltar objetos (*drag and drop*).

quiz.xml

```
<?xml version='1.0' encoding='utf-8'?>
<quiz title="Quiz Capitulo 1">
    <question text="Ordene las fases para ejecutar un programa." type="drag" slot="quiz1/drag1/slot.png"
              correct=" 1. Edición\n 2. Preprocesamiento\n 3. Compilación\n 4. Enlace\n 5. Carga\n" ref="Sección 1.6">
        <token file="quiz1/drag1/1-5.png"/>
        <token file="quiz1/drag1/2-5.png"/>
        <token file="quiz1/drag1/3-5.png"/>
        <token file="quiz1/drag1/4-5.png"/>
        <token file="quiz1/drag1/5-5.png"/>
    </question>
    <question text="¿Cuáles son características de C? (Escoja 4 opciones)" type="multiple" ref="Sección 1.11">
        <answer text="C es un lenguaje fuertemente tipeado ya que se debe especificar el tipo de datos de cada variable, parámetro y retorno de función." correct="yes"/>
        <answer text="C es un lenguaje estructurado de nivel medio, es decir, ni de bajo nivel como ensamblador, ni de alto nivel como Haskell." correct="yes"/>
        <answer text="C no lleva a cabo comprobación de errores en tiempo de ejecución." correct="yes"/>
        <answer text="C tiene un número reducido de palabras reservadas, 32 palabras reservadas en C89, y 37 en C99." correct="yes"/>
        <answer text="No dispone de una biblioteca estándar que contiene numerosas funciones que siempre se encuentran disponibles." correct="no"/>
    </question>
    <question text="Programa utilizado para escribir código fuente:" type="simple" ref="Sección 1.6">
        <answer text="Editor." correct="yes"/>
        <answer text="Preprocesador." correct="no"/>
        <answer text="Compilador." correct="no"/>
        <answer text="Enlazador." correct="no"/>
        <answer text="Cargador." correct="no"/>
    </question>
```

Figura 7.8: Archivo XML correspondiente a un Quiz

En la **Figura 7.9** se puede visualizar cómo se despliega los elementos correspondientes a la pregunta de arrastrar y soltar (drag and drop) mientras el estudiante resuelve el quiz.

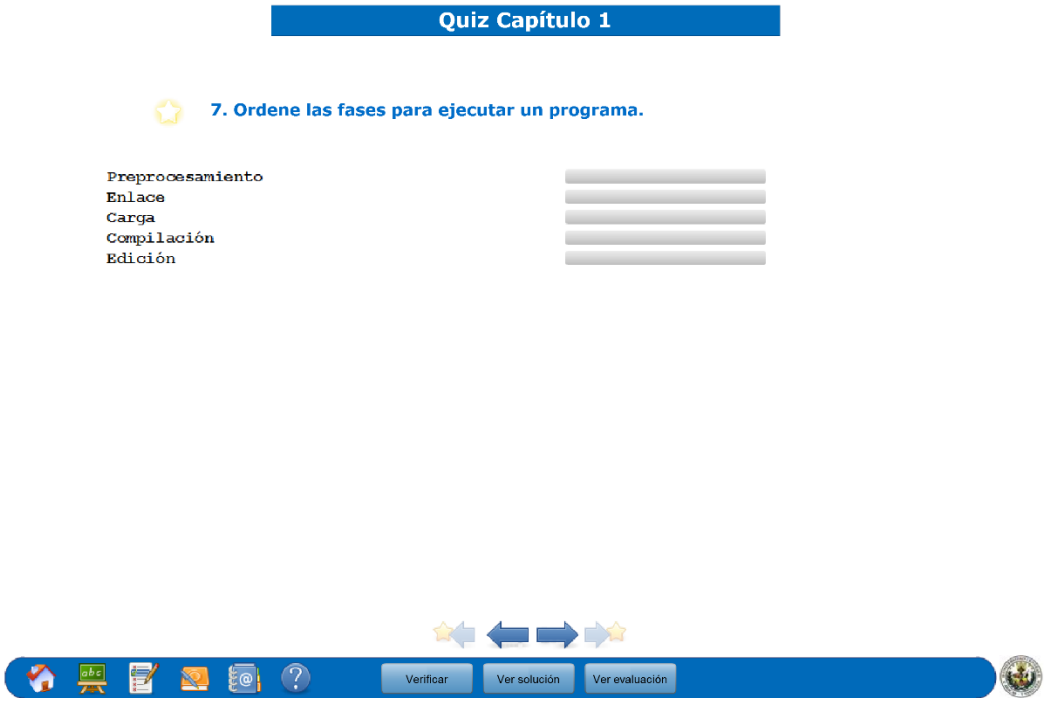

Figura 7.9: Interfaz del Estudiante visualizando la pregunta de Arrastrar y Soltar

7.4. Módulo del Glosario

El archivo XML principal correspondiente al módulo del glosario se debe llamar "glossary.xml", el cual contiene una lista por cada letra del alfabeto, donde cada lista contiene una lista de términos que comiencen con dicha letra, con su respectiva definición.

En la siguiente sección se describen los diferentes nodos que componen este archivo.

Nodo glosario

Etiqueta: "glossary"

Atributos: Ninguno.

Nodo letra

Etiqueta: "letter"

Atributos:

Nombre	Requerido?	Descripción
name	Sí	Letra por la cual comienzan todos los términos en este grupo.

Tabla 7.17: Atributos de los nodos del tipo glosario

Nodo término

Etiqueta: "term"

Atributos:

Nombre	Requerido?	Descripción
name	Sí	Nombre del término a ser definido.
desc	Sí	Definición del término a mostrar al estudiante cuando haga clic sobre éste en el glosario.
sound	No	Ruta a un archivo de audio con una narración de la descripción del término.

Tabla 7.18: Atributos de los nodos del tipo término

En la **Figura 7.10** se puede observar un ejemplo de los elementos XML que describe los términos de una letra.

glossary.xml

```xml
<?xml version='1.0' encoding='utf-8'?>
<glossary>
    <letter name="A">
        <term name="Abstracción" desc="La abstracción es una estrategia de resolución de problemas en la cual el programador se concentra en resolver una parte del problema ignorando completamente los detalles sobre cómo se resuelven el resto de las partes." sound="glossary/A/abstraccion.mp3"/>
        <term name="Alcance de una variable" desc="El alcance de una variable determina desde dónde dicha variable es accesible. Una variable definida fuera de una función tiene un alcance global, y puede ser accedida por el programa durante cualquier momento de su ejecución. Una variable dentro de una función no está disponible al resto del programa, y por ello tiene un alcance local; de la misma manera, las variables definidas fuera de una función tampoco pueden ser accedidas desde dentro de la función."/>
        <term name="Algoritmo" desc="Un algoritmo es una lista de instrucciones bien definida y finita que permite hallar la solución de un problema. Es importante especificar el orden en el que se van a ejecutar las instrucciones."/>
        <term name="ALU" desc="ALU (Arithmetic Logic Unit) es la unidad de la computadora en la cual se llevan a cabo las operaciones aritméticas y lógicas sobre los datos. Los resultados de las operaciones realizadas en la ALU se pueden transferir a la unidad de memoria o a la unidad de salida."/>
        <term name="ANSI C" desc="La primera estandarización del lenguaje C fue hecha por el ANSI (American National Standards Institute) en diciembre de 1989, con el estándar X3.159-1989. El lenguaje definido en este estándar fue conocido como ANSI C."/>
        <term name="Apuntador" desc="Un apuntador es una variable que contiene una dirección de memoria. Por ser una variable es necesario declararla e inicializarla."/>
        <term name="Archivo" desc="Un archivo es un conjunto de datos estructurados. Los archivos se utilizan como medio de almacenamiento permanente puesto que el almacenamiento de datos en variables, arreglos y cualquier otra estructura es temporal."/>
        <term name="Archivo binario" desc="Un archivo binario es una colección de bytes. Es un archivo que puede contener texto, imágenes, peliculas, archivos de sonido y gráficos."/>
        <term name="Archivo de texto" desc="Un archivo de texto es una colección de caracteres, organizadas en líneas terminadas con el carácter salto de línea."/>
        <term name="Arreglo" desc="Un arreglo es una colección de elementos del mismo tipo de dato. Todos los elementos de un arreglo ocupan posiciones consecutivas en la memoria, es decir, el primer elemento de un arreglo se encuentra en la dirección más baja de la memoria y, el último elemento de un arreglo, se encuentra en la dirección más alta de la memoria. Para acceder a un elemento del arreglo es necesario el uso de índices."/>
    </letter>
    <letter name="B">
        <term name="B" desc="El lenguaje B..."/>
        <term name="Biblioteca Estándar de C" desc="Los programas escritos en C consisten en módulos que se denominan funciones. Usted puede programar todas las funciones que necesita para formar un programa desde cero o aprovechar una gran recopilación de funciones existentes conocidas como la Biblioteca Estándar de C."/>
```

Figura 7.10: Archivo XML correspondiente al Glosario

En la **Figura 7.11** se muestra cómo el estudiante visualiza la información contenida en el archivo glossary.xml.

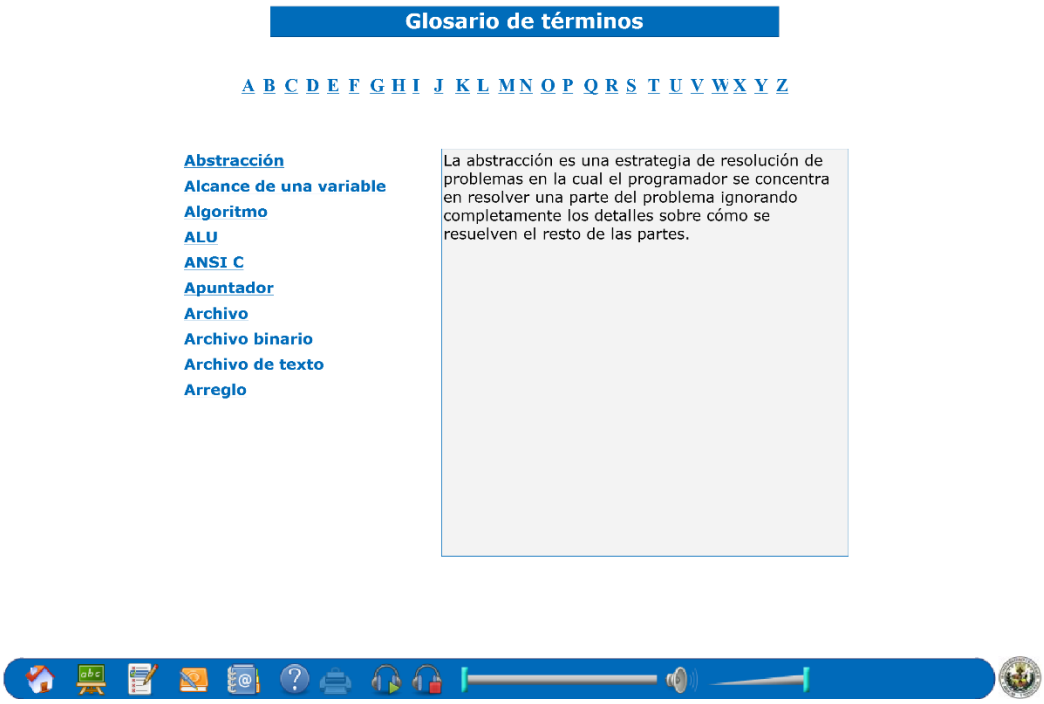

Figura 7.11: Interfaz del Estudiante visualizando el Glosario

8. Conclusiones

En el presente Trabajo Especial de Grado se propuso la implementación de un Generador de Cursos Multimedia con soporte para texto, imágenes, audio, video y autoevaluaciones, el cual pueda ser utilizado para la creación de cursos sobre diversos tópicos sin necesidad de modificar el código fuente del generador desarrollado. Esto permite que los cursos generados sean diseñados por instructores sin conocimientos previos en ActionScript 3, lo cual amplía la población capaz de utilizar la herramienta provista en este Trabajo Especial de Grado.

Como parte de la herramienta diseñada e implementada en este Trabajo Especial de Grado se provee la especificación de un lenguaje basado en XML para especificar el contenido y las evaluaciones de los cursos generados. Este lenguaje XML se diseñó de forma tal que permita desarrollar cursos de forma sencilla y flexible, con una sintaxis que permita el uso de una gran parte de la población docente. Este lenguaje basado en XML está débilmente acoplado a la aplicación desarrollada en ActionScript 3; esto permite que en un futuro sea posible desarrollar un generador en otro lenguaje de programación de ser necesario, sin la obligación de reescribir los cursos previamente desarrollados.

Para ilustrar la flexibilidad del generador desarrollado se diseñó e implementó una Aplicación Didáctica e Interactiva para Enseñar a Programar en el Lenguaje C, utilizando diversos recursos multimedia para facilitar el aprendizaje por parte de los estudiantes. En cada uno de los capítulos de esta aplicación se incluyeron: los objetivos que se pretenden alcanzar, el contenido de las lecciones, un resumen, unos videos y una evaluación por capítulo. Estas evaluaciones tienen 4 tipos de preguntas: selección simple, selección múltiple, arrastrar y soltar objetos (*drag and drop*) y llenado de espacios en blanco (*fill in blank*).

Se espera que la flexibilidad provista por la herramienta desarrollada en este Trabajo Especial de Grado facilite la creación de múltiples cursos a nivel global, los cuales tengan un impacto significativo en la educación a nivel mundial.

9. Referencias Bibliográficas

[1] R. Chun, “Adobe Flash Professional CS6, Classroom in a Book”, Primera Edición, Adobe Press, 2012.

[2] H. Deitel y P. Deitel, "C How to Program", Prentice Hall, Cuarta edición, Agosto 2002.

[3] Edinburgh University MOOC Teams, “MOOCs @ Edinburgh 2013 Report #1”, Mayo 2013.

[4] E. Gamess, "Los Secretos del Lenguaje C", Departamento de Ciencias de la Computación, Universidad del Valle, Cali, Colombia, Primera Edición, Enero 1997.

[5] K. Howard, “XML: Visual QuickStart Guide”, Peachpit Press, Segunda Edición, 2009.

[6] P. Kim, “Massive Open Online Courses, The MOOC Revolution”, Routledge, Primera Edición, 2015.

[7] B. Kernighan y D. Ritchie, "El Lenguaje de Programación C", Prentice Hall, Primera Edición, Abril 1985.

[8] K. Jordan, “Initial Trends in Enrollment and Completion of Massive Open Online Courses”, The International Review of Research in Open and Distance Learning, 15(1), www.irrodl.org/index.php/irrodl/article/view/1651, 2014.

[9] B. Means, Y. Toyama, R. Murphy, M. Bakia, K. Jones, “Evaluation of Evidence-Based Practices in Online Learning: A Meta-Analysis and Review of Online Learning Studies”, Septiembre 2010.

[10] I. Sommerville, “Ingeniería del Software”, Addison - Wesley, Séptima Edición, Enero 2005.