

DESARROLLO DE COMPONENTES DE INTERFAZ DE USUARIO PARA SISTEMAS BASADOS EN DISTRIBUCIÓN DE CONTENIDO DE VIDEO EN UN FRAMEWORK BASADO EN MODELOS



Universidad
del Cauca

Trabajo de grado

LUIS ALEJANDRO CRUZ ORDÓÑEZ
LUIS FELIPE HOLGUÍN LÓPEZ

Director: Mag. Ing. Alexandra Ruiz Gaona
Co-Director: PhD. José Luis Arciniegas

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Programa de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Popayán, 2017

Contenido

Capítulo 1	1
1.1. Problema y motivación.....	2
1.2. Objetivos	4
1.2.1. Objetivo general	4
1.2.2. Objetivos específicos	4
1.3. Hipótesis.....	4
1.4. Experimentación	5
1.5. Conclusiones y Divulgación	5
1.6. Metodología de Trabajo	5
1.7. Estructura del Documento	6
Capítulo 2	7
2.1. Estado del arte	7
2.1.1. Experiencias previas de herramientas para el diseño de interfaz de usuario basado en modelos MBUID	8
2.1.2. Aproximaciones metodológicas para la creación de editores para lenguajes específicos de dominio (DSL).....	14
2.2. Marco teórico.....	16
2.2.1. Conceptos generales involucrados con interfaces de usuario y sus componentes..	16
2.2.2. Conceptos relacionados a la ingeniería dirigida por modelos.....	25
3.1. Selección de contexto de investigación.....	31
3.2. Exploración de componentes de sistemas interactivos en el contexto educativo.....	33
3.3. Componentes seleccionados.....	37
Capítulo 4	47
4.1. Elección de tecnología.....	47
4.2. Sintaxis abstracta.....	48
4.2.1. Creación de Metamodelo a través de ECORE.....	48
4.2.2. Descripción de elementos	51
4.2.3. Relaciones identificadas entre clases	54

4.2.4. Presentación de modelo final.....	54
4.3. Sintaxis concreta	55
4.3.1. Bosquejos para el modelo de definición gráfica.....	55
4.3.2. Bosquejos para la definición de la herramienta	56
5.1. Desarrollo elemental de componentes	59
5.1.1. Definición de la metodología SCRUM.....	60
5.1.2. Aplicación de la Metodología SCRUM	63
5.1.3. Proceso de desarrollo.....	69
5.1.4. Prototipo final de desarrollo de componentes	72
5.2. Implementación del Framework basado en modelos	75
5.2.1. Requisitos previos	76
5.2.2. Creación de un proyecto GMF.....	79
5.2.3. Generación de código a partir de un modelo.....	80
5.2.4. Creación del editor gráfico	83
5.2.5. Generación de código.....	92
6.1. Validación	95
6.1.1. Validación de los componentes.....	96
6.1.2. Validación del editor gráfico.....	96
6.2. Evaluación	101
6.2.1. Prueba de evaluación aplicada.....	102
6.2.2. Resultados de la evaluación	103
6.2.3. Caso de estudio en el contexto educativo.....	106
7.1. Conclusiones generales	109
7.2. Lecciones aprendidas y recomendaciones	110
7.3. Trabajos futuros	111
Bibliografía	113

Índice de figuras

Figura 1. Overview de MetaEdit+, la tecnología que permite elaborar herramientas de modelado en un entorno de Metacase.	14
Figura 2. Roles definidos en DSL (Domain Specific Language)	15
Figura 3. Entorno para el diseño de una interfaz de usuario	18
Figura 4. Esquema de proceso de modelado de Arquitectura Software	21
Figura 5. Ejemplo de diagrama de secuencia para la construcción de un MVC	22
Figura 6. Modelo de interfaz de usuario	27
Figura 7. Modelo de proceso para la creación de un editor DSL	30
Figura 8. Variables críticas en la formación en red.	33
Figura 9. Creación de un proyecto EMF.	49
Figura 10. Creación de Diagrama Ecore	50
Figura 11. Paleta de herramientas EMF.	51
Figura 12. Herencia de atributos comunes de componentes.	54
Figura 13. Metamodelo de componentes basados en distribución de video.	55
Figura 14. Bosquejos de modelo de definición gráfica de los componentes	56
Figura 15. Modelo de arquitectura software 1.	62
Figura 16. Modelo de arquitectura software 2.	63
Figura 17. Diagrama de clase Reproductor de video	64
Figura 18. Diagrama de clase Visualizador de documentos	65
Figura 19. Diagrama de clase Quiz	66
Figura 20. Diagrama de clase de Tareas	67
Figura 21. Diagrama de clase de Chat.....	68
Figura 22. Diagrama de clase Recursos relacionados.....	68
Figura 23. Diagrama de clase Tabla de contenido.....	69
Figura 24. Archivos iniciales del reproductor de video	70
Figura 25. Carpeta de estilos de reproductor de video	70
Figura 26. Fuentes utilizadas en el reproductor de video	70
Figura 27. Imágenes utilizadas en el reproductor de video.....	70
Figura 28. Archivos Javascript del reproductor de video.	71
Figura 29. Carpeta de metadatos del reproductor de video.....	71
Figura 30. Carpeta de subtítulos de reproductor de video.	71
Figura 31. Carpeta de videos de reproductor de video.	72

Figura 32. Prototipo final componente de reproductor de video	72
Figura 33. Prototipo final sección de transcripción de texto del componente de reproductor de video.	73
Figura 34. Prototipo final componente de visualizador de documentos	73
Figura 35. Prototipo final Quiz.....	74
Figura 36. Prototipo final de componente de tareas	74
Figura 37. Prototipo final Chat usuario en línea	75
Figura 38. Prototipo final Recursos relacionados.....	75
Figura 39. Prototipo final componente Tabla de Contenido	75
Figura 40. Plug-ins de modelado instalados en Eclipse.....	78
Figura 41. Plug-ins GEF instalados en Eclipse.	79
Figura 42. Composición y modelos GMF	80
Figura 43. Importación de un modelo en EMF	81
Figura 44. Selección de metamodelo a importar EMF	81
Figura 45. Presentación de metamodelo seleccionado	82
Figura 46. Configuración de paquete base de modelo	82
Figura 47. Creación de modelo de definición gráfica	84
Figura 48. Selección de folder para modelo de definición gráfico.....	84
Figura 49. Selección de archivo con modelo de dominio ecore.....	85
Figura 50. Definición de nodos, enlaces y etiquetas MDG	85
Figura 51. Estructura de modelo de definición gráfica.	86
Figura 52. Configuración de propiedades de imagen SVG	87
Figura 53. Configuración de propiedades de etiqueta MDG	87
Figura 54. Configuración de imagen y etiqueta de nodos MDG	88
Figura 55. Creación de modelo definición de herramientas.....	89
Figura 56. Modelo de definición de herramientas	90
Figura 57. Especificación de nodos y enlaces <i>Mapping</i>	91
Figura 58. Validación de creación de <i>Mapping</i>	91
Figura 59. Relación de modelos en el Nodo <i>Mapping</i>	92
Figura 60. Árbol del modelo de relación de elementos	92
Figura 61. Explorador de paquetes generación de código.....	93
Figura 62. Aspectos de evaluación de software.....	103
Figura 63. Media aritmética de dimensiones de evaluación	104
Figura 64. Diagrama de barras de resultados de evaluación de dimensiones	104

Figura 65. Media aritmerica para los tres aspectos de calidad pragmática y calidad hedónica.....	105
Figura 66. Media y desviación estándar de evaluación de componentes.....	106
Figura 67. Ubicación de los componentes en caso de estudio.....	107
Figura 68. Interfaz final caso de estudio.	108

Índice de tablas

Tabla 1. <i>Frameworks</i> MBUID estudiados.....	12
Tabla 2. Sub proyectos Eclipse para desarrollo basado en modelos.	16
Tabla 3. Tecnologías existentes para desarrollo de componentes UI.	23
Tabla 4. Aporte del e-learning a la mejora e innovación de la enseñanza.....	32
Tabla 5. Herramientas para la educación en línea.....	34
Tabla 6. Presentación de componente Reproductor de video	38
Tabla 7. Presentación de componente Visualizador de documentos.	40
Tabla 8. Presentación de componente Chat	41
Tabla 9. Presentación de componente Quiz.	43
Tabla 10. Presentación de componente Tareas.	44
Tabla 11. Presentación de componente Recursos relacionados	45
Tabla 12. Presentación de componente Contenido	46
Tabla 13. Descripción de modelo del elemento “Player” (Reproductor de video).....	51
Tabla 14. Descripción de modelo del elemento “Visualizador de componentes”.....	52
Tabla 15. Descripción de modelo del elemento “Quiz”.	52
Tabla 16. Descripción de modelo del elemento “Chat”.	52
Tabla 17. Descripción de modelo del elemento “Tareas”.....	53
Tabla 18. Descripción de modelo del elemento “Repositorio”.....	53
Tabla 19. Bosquejos para la definición de paleta de herramientas.....	57
Tabla 20. Comparación entre metodologías ágiles y tradicionales.....	59
Tabla 21. Validación componente de reproductor de video	97
Tabla 22. Validación componentes de visualizador de documentos.....	98
Tabla 23. Validación editor gráfico	99
Tabla 24. Características de calidad de software norma ISO/IEC 9126	101
Tabla 25. Resultado de evaluación caso de estudio.....	107

Capítulo 1

Introducción

La distribución de contenido de video ha impulsado a los desarrolladores la creación de nuevas aplicaciones cada vez más robustas aumentando el consumo de este tipo de información. Los desarrolladores buscan el apoyo de *frameworks* para la elaboración de sus proyectos obteniendo un trabajo más productivo. Gran parte de estos *frameworks* son los basados en modelos, esto permite la transformación de un modelo de diseño de alto nivel en una descripción de un conjunto abstracto de los elementos que lo componen y seguidamente la transformación de estos elementos en el código para una plataforma específica. Existe una gran variedad de sistemas interactivos con contenido de video con aplicación en diferentes contextos, que entre los más importantes están los del contexto de educación, comunicación, entretenimiento, industrial, vigilancia y médico.

Con el propósito de contribuir con la elaboración de componentes más usados por las últimas tecnologías quiere presentarse el siguiente trabajo de investigación en donde no sólo son tomadas en cuenta los componentes convencionales sino también los que permiten obtener interfaces de usuario más completas.

El presente capítulo aborda el planteamiento general del trabajo de grado, que dentro de ello está incluido el problema y motivación del mismo, además de los objetivos tanto generales como específicos planteados. Igualmente, se presenta la hipótesis y experimentación correspondiente a la investigación junto con las contribuciones al estado actual del conocimiento. Es importante mostrar también en este capítulo, la metodología implementada durante todo el proyecto con la estructura del documento, es decir la especificación de la organización por secciones de este mismo.

1.1. Problema y motivación

El uso de video es cada vez más popular en internet, según algunos análisis y predicciones el tráfico de video IP fue el 70% de todo el tráfico IP global en el 2015 y será del 82% en el 2020. Este porcentaje no incluye el intercambio de video P2P (*Peer to peer*) pero abarca las demás formas de video las cuales son: TV, Video bajo demanda (VoD) e internet (Index). Esta tendencia en el consumo de video ha hecho que fábricas de software y otro tipo de empresas tiendan a apostar por aplicaciones en donde el negocio gira en torno a la distribución de contenido de video¹(Odden). Facebook por ejemplo (una de las redes sociales con más usuarios activos en el planeta), en el 2016 logró ser el centro del *streaming* de video casero. Para lograrlo, la compañía amplió su plataforma para permitir la publicación de transmisión en directo en su red desde cualquier cámara con conexión a Internet ("Facebook apuesta por ser el centro del streaming de video casero ", 2016). Este es tan sólo un ejemplo de los cambios que están haciendo las empresas en sus modelos de negocio para atraer a los usuarios que consumen y distribuyen contenido de video. A medida que la tecnología avance, las aplicaciones basadas en distribución de contenido de video serán más robustas. Es de esperarse que el internet del futuro soporte videos con resolución 4K, llamadas de voz en alta definición y hasta hologramas (Dergarabedian, 2016). Como consecuencia de esto, incrementará el desarrollo de nuevas aplicaciones alrededor del contenido de video.

Existe una gran variedad de sistemas interactivos con contenido de video con aplicación en diferentes contextos (Hannington & Reed, 2002). Algunos de estos contextos son expuestos a continuación:

- *A nivel de comunicación:* aplicaciones de trabajo colaborativo o GroupWare (JCMS (Dourgnon-Hanoune, Dang, Dissert, & Reguigui, 2006), Google Apps (Railean, 2012), eGroupWare (GmbH & Becker, 2007), entre otras), videoconferencia (Skype (Anderruthy, 2007), Hangouts (Xu et al., 2010), ooVoo(McEvoy), TANGO (Kurniawati, Celetto, Capovilla, & George, 2012)), streaming media y tele servicios multimedia.
- *A nivel de entretenimiento:* juegos de PC en 3D, Juegos en red para múltiples jugadores, *infotainment* (Información y entretenimiento) y producción audiovisual interactivo.
- *A nivel educativo y de entrenamiento:* libros electrónicos, materiales flexibles de enseñanza, sistemas de simulación, pruebas automatizadas y educación a distancia.
- *A nivel de vigilancia y satelital:* aplicaciones que incluyen investigaciones de video satelital sobre el planeta para diferentes entornos de habidad y operaciones militares. Hay dos tipos comunes

¹ Contenido de video. Fundamentalmente referido a información que es presentada en formato de video para que otros lo consuman.

de video de vigilancia, la primera en seguridad utilizada, generalmente, para las propiedades o áreas públicas y la otra aplicada al automovilismo utilizada para controlar el flujo de tráfico.

- *A nivel médico:* desarrollo de aplicaciones enfocadas a la salud, principalmente para la organización jerárquica en clínicas y hospitales, también aplicaciones para el diagnóstico del paciente como “*Medical video*” (Zhu, Aref, Fan, Catlin, & Elmagarmid, 2003) donde a través de técnicas de procesamiento de audio y de vídeo es integrada la información a eventos, como diálogo, presentación clínica y funcionamiento, a partir de escenas detectadas.
- *A nivel industrial:* los sistemas de inspección industrial son usados principalmente en capacidades respecto al control de calidad.

De acuerdo a lo expuesto anteriormente, el contenido de video tiene un alto potencial y uso en los sistemas interactivos. De otro modo, las empresas que desarrollan este tipo de aplicaciones, usualmente están apoyadas en *frameworks*, con el fin de aumentar la productividad en su trabajo. Dentro de los *frameworks* utilizados están aquellos que son basados en modelos. El Desarrollo Basado en Modelos (MDD) es uno de los enfoques que trata de reducir la complejidad de la creación de aplicaciones multimedia interactivas (Kulesza et al., 2012). El MDD permite la transformación de un modelo de diseño de alto nivel en una descripción de un conjunto abstracto de los elementos que lo componen y seguidamente la transformación de estos elementos en el código para una plataforma específica. El Diseño de la Interfaz de Usuario basados en Modelo (MBUID) es la vertiente de la ingeniería dirigida por modelo (MDE) que enfoca el desarrollo de la interfaz de usuario a partir de modelos. El propósito del diseño basado en modelos es identificar modelos de alto nivel que permiten a los diseñadores especificar y analizar las aplicaciones de software interactivos a partir de un nivel de orientación más semántica en lugar de comenzar de inmediato a hacer frente al nivel de aplicación. Esto les permite concentrarse en los aspectos más importantes sin confundirse inmediatamente por muchos detalles de implementación (W. C. W. Group, 2013).

A partir del análisis de un conjunto de *frameworks* MBUID, los componentes y/o *Widgets* en este tipo de herramientas no han tenido mucha evolución, es decir, los widget siguen siendo los convencionales (ej. botones, check-box, combo-box). Es de notar que las aplicaciones basadas en contenido de video tienen ciertas particularidades a nivel de interfaz que hacen que los componentes o *widget* convencionales no sean suficientes para su desarrollo. La carencia de componentes y/o *widget* específicos para aplicaciones basada en contenido de video en los *frameworks* representa para los desarrolladores mayor tiempo en el desarrollo y por consiguiente una menor productividad. De igual manera, la ausencia de este tipo de componentes puede llevar a generar interfaces de usuario débiles. Una interfaz de usuario débil o mal hecha trae consecuencias perjudiciales como inducir al usuario a cometer errores o frustrar sus esfuerzos para alcanzar las metas, entre otras (Ponencia sobre Diseño de Interfaces y Usabilidad: cómo hacer productos más útiles).

Los *frameworks* basados en modelos pueden tener una evolución notable al enriquecerlos con nuevos componentes, que necesariamente especifican tres dimensiones: negocio, aplicación y tecnológica. La dimensión de negocio describe los componentes en forma de modelos. La dimensión de aplicación describe la notación de cada componente y su integración dentro del ambiente de desarrollo. Finalmente, la dimensión tecnológica es asociada con la implementación del componente en un lenguaje de programación específico que es el insumo para la construcción de generadores de código de los *frameworks* basados en modelos.

1.2. Objetivos

1.2.1. Objetivo general

Proponer componentes de interfaz de usuario para sistemas basados en distribución de contenido de video a nivel de negocio, aplicación y tecnológico en un *framework* basado en modelos, con el fin de apoyar el desarrollo de la interfaz de usuario de este tipo de sistemas.

1.2.2. Objetivos específicos

- Identificar los principales componentes de interfaz de usuario que caracterizan a los sistemas basados en distribución de contenido de video en el contexto educativo.
- Definir la sintaxis abstracta (modelos) y concreta (notación) de los componentes de interfaz de usuario de los sistemas basados en distribución de contenido de video que describan sus propiedades, acciones y relaciones.
- Implementar los componentes en el *framework* seleccionado.
- Evaluar la funcionalidad de los componentes a través del desarrollo de un caso de estudio en el contexto educativo.

1.3. Hipótesis

Como hipótesis principal para este trabajo de grado se plantea la siguiente:

Es posible la creación de nuevos componentes para interfaces de usuarios tecnológicas de sistemas que involucran contenido de video en *frameworks* con ingeniería basada en modelos, brindando a los desarrolladores no solamente optimización y productividad en sus trabajos si no también amplios recursos para el desarrollo de aplicaciones enfocadas a dichos sistemas.

Con base a lo planteado, las interfaces de usuario pueden fortalecerse en términos de componentes con la obtención de sus sintaxis abstracta y concreta, asimismo el avance de desarrollo puede llegar a ser bastante notable si los enfoques de estudio son a nivel de aplicación, negocio y tecnológico.

1.4. Experimentación

Como parte de la experimentación, está la búsqueda de componentes de interfaces de usuario en sistemas de distribución de video y la implementación de estos. La experimentación también cuenta con un grupo seleccionado de personas, con propósito de probar y evaluar cada componente de una interfaz de usuario desarrollada a partir de modelos. Como primera medida se definieron algunos criterios de selección del grupo de experimentación y seguidamente se desarrolló una contextualización con el personal seleccionado para introducirlos al tema. De esta forma los resultados obtenidos en las pruebas, validan el trabajo realizado.

1.5. Conclusiones y Divulgación

Para las conclusiones hay un proceso de síntesis de los resultados más relevantes, de recolección de experiencias obtenidas y elementos investigativos para ser tenidos en cuenta a futuro. Como actividad de divulgación este trabajo es sometido a un artículo científico que describe de manera precisa los aportes más importantes de esta investigación.

1.6. Metodología de Trabajo

La metodología a seguir para el desarrollo del presente trabajo de investigación, es construida de manera sistemática con el fin de obtener altas posibilidades de éxito en todas las diferentes etapas propuestas. Esta metodología está basada en una adaptación del modelo de construcción de soluciones telemáticas (Cantone, 2006), el cual plantea las siguientes fases:

1. Especificaciones.
2. Análisis.
3. Diseño.
4. Implementación.
5. Depuración. (Garantizar la no presencia de errores).
6. Validación.

El primer objetivo específico propuesto en este trabajo es abordado por las dos primeras fases de la metodología planteada, los objetivos 2 y 3 son abordados en la fase 3, encargada del diseño basado en modelos de la interfaz, además de la selección del *framework*. El objetivo 4 es abordado por la fase 4 de la metodología en el cual consiste en la implementación de los componentes de la interfaz. Teniendo en cuenta que la implementación es equivalente a desarrollo de software y ese desarrollo está ligado a otra metodología software propia para el desarrollo que será presentada en dicho capítulo al lector. El último objetivo es abordado por las última dos fases en donde son las correcciones de errores y además valida el trabajo realizado a través de pruebas.

1.7. Estructura del Documento

El presente documento está estructurado de la siguiente forma:

- El capítulo 1, incluye la introducción al documento, el planteamiento del problema y la estructura general para el desarrollo del trabajo de grado.
- El capítulo 2, denominado “Marco de referencia”, hace alusión a los conceptos, tecnologías y experiencias previas de otros investigadores relacionados con interfaces de usuario para el uso componentes con distribución de contenido de video, *frameworks* desarrollados en dicho ámbito y componentes para la interfaz.
- El capítulo 3 denominado “Identificación y selección de componentes” hace referencia a los entornos de investigación seleccionados, las herramientas revisadas, y un catálogo de componentes encontrados. También plantea un proceso de selección de componentes propuestos para su implementación.
- El capítulo 4 denominado “Sintaxis abstracta y concreta”, muestra las propiedades, acciones y relaciones de los componentes seleccionados en la investigación a través de modelos y una notación específica.
- El capítulo 5, denominado “Implementación”, contiene una descripción de las herramientas que soportan la integración. Este capítulo consta de dos partes, la primera es la implementación de los componentes en términos de funcionalidad sin involucrar la herramienta de modelado. La segunda, es la creación de la herramienta de modelado y la introducción de los componentes a modelar.
- El capítulo 6, denominado “Validación y Evaluación” define todas las pruebas realizadas a nivel de funcionalidad del desarrollo software realizado y a nivel de usabilidad al realizar pruebas con usuarios.
- El capítulo 7, denominado “Conclusiones y trabajos futuros”, presenta unas conclusiones generales del trabajo realizado y propone trabajos futuros a realizar.

Capítulo 2

Marco de Referencia

Como es descrito en el planteamiento del problema de este trabajo de grado, gran parte de las herramientas dedicadas a la construcción de interfaces de usuario, no involucran componentes específicos relacionados a los sistemas interactivos basados en distribución de contenido de video, lo cual representa para los desarrolladores mayor tiempo en el desarrollo y por consiguiente una menor productividad. Por tal razón este trabajo propone componentes de interfaz de usuario para sistemas basados en distribución de contenido de video a nivel de negocio, aplicación y tecnológico que pueden ser incorporados en *frameworks* para el diseño de la interfaz de usuario basados en modelos, con el fin de apoyar el desarrollo de la IU de este tipo de sistemas. En aras de conseguir el objetivo propuesto, este trabajo esta apoyado en la revisión de investigaciones similares, permitiendo así identificar las tecnologías existentes en diferentes contextos y los conocimientos necesarios para su desarrollo.

Los conceptos, tecnologías y trabajos relacionados más representativos son tomadas en cuenta en este capítulo. De acuerdo al objetivo propuesto, este marco de referencia abarca dos tópicos fundamentales: 1) el diseño de la interfaz de usuario basado en modelos, sus herramientas y componentes asociados y 2) aproximaciones metodológicas para el desarrollo de editores para lenguajes específicos de dominio (DSL). Estos dos tópicos son abordados tanto en el estado del arte como en el marco teórico.

Debe agregarse que dichas investigaciones dan a conocer diferentes soluciones planteadas a problemáticas similares. A partir de esto es construido el estado del arte que analiza las diferentes propuestas y concluye con el planteamiento de una solución según las necesidades identificadas y las fortalezas y/o debilidades encontradas en las demás propuestas.

2.1. Estado del arte

A continuación son relacionadas las propuestas para el diseño de la interfaz de usuario basadas en modelos (MBUID).

2.1.1. Experiencias previas de herramientas para el diseño de interfaz de usuario basado en modelos MBUID

Desde las primeras propuestas relacionadas al desarrollo de interfaces de usuario basado en modelos (MBUID) en 1990 a las propuestas más recientes (2016), son muchos los trabajos desarrollados en esta temática. Ellos son trabajos clasificados en cuatro generaciones (Meixner, Paterno, & Vanderdonckt), cada una con características particulares: en la 1°, las herramientas usaban principalmente un modelo de interfaz de usuario (IU) declarativo universal que integraba los aspectos más relevantes de esta; la 2° caracterizada por la extensión de los modelos de IU a través de la integración de otros; en la 3°, MBUID cobra relevancia debido a la proliferación de diversos dispositivos y la 4° enfocada en el desarrollo de la IU sensitiva al contexto para plataformas, dispositivos, modalidades diferentes y la integración de aplicaciones web.

Esta sección presenta un conjunto de *frameworks* MBUID que han sido explorados durante la investigación, seleccionando los más significativos conforme a criterios como existencia de documentación (información), actualidad y disponibilidad de herramientas.

MARIAE (Paterno', Santoro, & Spano, 2009). Esta herramienta está basada en el desarrollo de modelos de interfaz de usuarios y modelos de tareas, diseñado en una plataforma de Servicios Web para aplicaciones interactivas. Produce dos implementaciones estructuradas: gráfica e implementación de interfaz vocal. La herramienta permite la construcción de interfaces a través de modelos específicos con un seguimiento desde las CTT (*Concur Task Trees*) *Task Model*, y seguidamente genera las AUI (*Abstract User interface*) y las CUI (*Concret User Interface*). En cada una de las etapas del desarrollo, está permitido la creación, importación y edición de algún modelo necesario. Finalmente, existen varios tipos de escenarios para correr la aplicación final (*Home and Weather, Educational, Car Rental y Multimodal*), en donde hay archivos propios del programa que adaptan la aplicación creada a la necesidad del entorno de ejecución.

MOFLON (Amelunxen, Königs, Röttschke, & Schürr, 2006). La herramienta MOFLON está diseñada para el soporte en definición, transformación, análisis e integración de metamodelos, a través de combinación de notaciones visuales y textuales de modularización. MOFLON es compatible con el estándar que proporciona soporte completo de la nueva modularización MOF 2.0. La herramienta proporciona un lenguaje de transformación gráfico visual con control de flujo y diagramas para estrategias de reglas de aplicación adoptadas por FUJABA (Burmester et al., 2004) para realizar transformaciones de modelos locales. Para la integración modelo a modelo, usa un enfoque QVT (*Query/View/Transformation*) declarativo basado en *Triple Graph Grammer*.

CASSIS (Bergh & Coninx, 2014). Es una herramienta que utiliza un lenguaje conciso para la creación de componentes de UI usando modelos en lugar de extensiones de lenguajes. Permite la creación de una sintaxis gráfica que no está completamente definida en el lenguaje mismo. CASSIS apoya la

creación de bibliotecas de componentes, conceptos y eventos que pueden especificar su propio icono. Este lenguaje soporta la interacción del sistema de especificación de una manera adaptable. Esto permite que un equipo de diseño de interfaz de usuario y la ingeniería, puedan comunicarse con mayor eficiencia sobre la memoria descriptiva.

SUPPLE (Gajos, Weld, & Wobbrock, 2010). Herramienta que interpretando limitaciones en diferentes dispositivos, enfoca su estudio en reducción de esfuerzos y acciones en la creación de una interfaz de usuario a través de un patrón de uso particular y una eficiente representación algorítmica de interfaces específicas que usan la ingeniería basada en modelos. SUPPLE adapta la interfaz al estilo de trabajo individual del usuario así como a las preferencias personales. La generación y la adaptación son en tiempo de ejecución.

SerCHo (Blumendorf, Feuerstack, & Albayrak, 2008). Tiene como tarea principal el desarrollo de una plataforma de interfaz de usuario pensada en incrementar la calidad de vida en el hogar a través de la implementación de un framework. El propósito es generar servicios multimodales propios con aplicación en el hogar. La realización de los servicios de asistencia en SerCHo está especialmente apoyada por la combinación de un entorno de desarrollo con una plataforma tanto en la red como en la casa. Como un ejemplo de contexto, existe el "Asistente Inteligente de la Salud", un servicio que promueve el mantenimiento de una buena salud soportado en SerCHO; explica cómo los servicios en el software son desarrollados, instalados y proporcionados en un entorno real de un moderno apartamento de cuatro habitaciones amuebladas, totalmente acondicionado por la tecnología de interfaces MBUID.

CAMELEON (Calvary et al., 2002). Framework basado en modelos que permite el diseño de la interfaz de usuario en diferentes niveles de abstracción. A nivel de modelos ontológicos contiene un modelo de dominio, contextos de uso y adaptación. En un espacio taxonómico abarca cuestiones fundamentales en la multi-segmentación para las etapas de desarrollo, así como para el tiempo de ejecución, es así como propone un marco conceptual que ayuda a estructurar el proceso de desarrollo de interfaz de usuario de manera multi-objetivo, así como el apoyo en tiempo de ejecución, Cameleon puede caracterizar claramente la cobertura funcional de herramientas existentes y determinar los requisitos para futuras herramientas.

CIAT (W. J. Giraldo, 2010). Es una herramienta para la generación automática de una interfaz de usuario dando soporte a la metodología TD-MBUID. Esta metodología es caracterizada en el soporte para la generación automática de IU, el uso de modelos declarativos y la adopción de una metodología para soportar el desarrollo de la interfaz. Adicionalmente, incorpora una serie de guías, reglas y aspectos que están asociados a la presentación y usabilidad. Esta propuesta de desarrollo de la IU da libertad al desarrollador para continuar el diseño a partir de los datos o iniciarlo a partir de las tareas interactivas. El proceso no es completamente automático y requiere la intervención constante de los usuarios; los cuales participan en todo el desarrollo de la IU. Sólo para casos sencillos sugieren obtener IU de manera automática por medio de las herramientas de generación automática y las bases de conocimiento que soportan el proceso.

COMET (Demeure, Calvary, & Coninx, 2008). Esta herramienta principalmente centra su objetivo en la capacidad de reacción de interfaces de usuario para adaptarse dinámicamente al contexto de uso (Usuario, plataforma, ambiente) esto le llaman plasticidad, Comet (*Context Mouldable Widget*) proporciona al diseñador herramientas para la creación de interfaces de usuario plásticas tanto en tiempo de diseño como de ejecución donde existen Cometas, los cuales interactúan, y en grupo apoyan tareas particulares.

GUMMY (Meskens, Vermeulen, Luyten, & Coninx, 2008). Este entorno da un soporte para el desarrollo de interfaz de usuario concreta, proporciona la capacidad de generar un diseño en una nueva plataforma mediante la adaptación y combinación de características existentes, Gummy genera un soporte independiente de la interfaz de usuario y la va actualizando según cada cambio con esto facilita la migración a diferentes plataformas ya que puede generar un diseño inicial.

CEDAR (Akiki, Bandara, & Yu, 2013). Entorno integrado de desarrollo (IDE) que proporciona un soporte orientado a ambientes empresariales tanto en la fase desarrollo como la del post desarrollo, provee el desarrollo de interfaces de usuario adaptativas para aplicaciones empresariales, este IDE es un apoyo donde da acceso a todas las herramientas de diseño visual y edición de código en un mismo lugar, CEDAR maneja el patrón de arquitectura del modelo vista-controlador. Adicional a esto CEDAR está basado en CAMELEON.

Esta sección presenta un conjunto de *frameworks* MBUID que han sido explorados durante la investigación, seleccionando los más significativos conforme a criterios como existencia de documentación (información), actualidad y disponibilidad de herramientas.

Los *frameworks* MBUID anteriormente presentados, fueron analizados en relación a los componentes de interfaz que soportaban y a su capacidad para ser mejorados o ampliados a nivel de componentes. Para ello, están definidos los siguientes criterios de clasificación:

- *Herramienta (H)*: evaluar si el *framework* tiene herramienta a la que pueda accederse
- *Libre*: evaluar si la herramienta es de código abierto y de libre uso para trabajo de investigación
- *Componentes*: listados de componentes de interfaz de usuario
- *Lenguaje Generado*: listado de lenguaje de programación en los cuales puede generarse el código
- *Modificación (MOD)*: evaluar si es posible modificar las herramientas
- *Documentación (DOC)*: evaluar si está disponible la documentación de la herramienta

Los *framework* MBUID clasificados son presentados en la Tabla 1. Tener en cuenta que en algunos campos de esta tabla, especifica la sigla ND, esto

quiere dar a entender, que el criterio correspondiente para esa herramienta ya no estaba disponible o fue imposible acceder a este.

Al realizar la clasificación hay evidencia que los componentes de IU de los *framework* analizados incluyen los componentes convencionales de los sistemas interactivos y no aquellos que son específicos del dominio de las aplicaciones basadas en distribución de contenido de video. De igual manera, no todos los *frameworks* tienen la herramienta y documentación disponible. También puede apreciarse que son pocos los *frameworks* que permiten su modificación o extensión.

Tabla 1. Frameworks MBUID estudiados.

Propuesta	Referencia	He*	Libre	Componentes	Lenguaje Generado	Mod*	Doc*
MARIAE	(Paterno, Santoro, & Spano, 2009)	Si	Si	Button, MailTo, Link, ImageMap, ImageButton, DuiPanel, Image, Label	HTML, Voice XML	Si	Si
CTTE	(Mori, Paternò, & Santoro, 2002)	Si	Si	ND	ND	No	No
Comet	(Demeure et al., 2008)	Si	No	Radio Button, List, ComboBox, PieMenu.	OpenGL, HTML, SAPI	No	No
Dynamo	(Clerckx, Luyten, & Coninx, 2004)	Si	No	ND	ND	No	Si
Cameleon	(Calvary et al., 2002)	No	No	Text Input, Button, Label, ComboBox, ListBox.	JAVA, HTML	No	Si
Moflon	(Amelunxen et al., 2006)	Si	Si	TextView, TextBox, Buton, RadioButton, ListBox.	XML, XSLT	No	Si
Gummy	(Meskens et al., 2008)	No	No	Label, Container, Image, List, Frame, ToggleButton, Horizontal Scroll, Vertical Range, Calendar, Frame, Horizontal Range, Text, Vertical Scrool, RadioButton.	XML	No	Si
Cedar Studio	(Akiki et al., 2013)	No	No	PictureBox, ComboBox, RadioButton, TabControl, ProgressBar, Button, Checkbox, DataTimerPicker, RichTextBox, GroupBox, Listbox, Panel, DataGridView	C#, VB.NET, Iron Phyton,	No	Si

CASSIS	(Bergh & Coninx, 2014)	Si	No	ND	ND	Si	Si
MyUI	(Peissner et al., 2012)	No	No	Button, Grid, TextView, CheckBox,	HTML, JavaScript	No	No
CIAT	(W. J. Giraldo, 2010)	Si	Si	UserInterface, Group, TextComponent, TextView, ListView, Button, DataView, DialogText, CheckBox, RadioButton, ImageView, RadioGroup, Spinner, VideoView.	JAVA	Si	Si
SerCho	(Blumendorf et al., 2008)	No	No	ListBox, Button, ListView, ComboBox, Grid, RadioButton, CheckBox.	HTTP/XML	No	No
Damask	(Lin & Landay, 2008)	Si	No	Button, CheckBox, Label, RadioButton, Listbox, Drop Down Box, TextBox, Panel, Split, Merge	XML	No	No
IFML 1.0	(Brambilla, Mauri, & Umuhoza, 2014)	Si	Si	Screen, Message, Containers, mpcrophone	OCL (Acceleo), C, JAVA	Si	Si
SUPPLE	(Gajos et al., 2010)	No	No	RadioButton, Combobox, ListView, CheckBox, TextBox.	HTML.	Si	Si

Nota: * He = Herramienta, Mod = Modificación, Doc = Documentación

2.1.2. Aproximaciones metodológicas para la creación de editores para lenguajes específicos de dominio (DSL).

A nivel mundial, distintas grupos de investigación están dedicadas en los últimos años al desarrollo de editores para el modelado en diferentes contextos, utilizando diferentes aproximaciones metodológicas. A continuación son presentadas algunas alternativas metodológicas y tecnológicas para la construcción de editores de modelado.

2.1.2.1. Propuesta Metacase

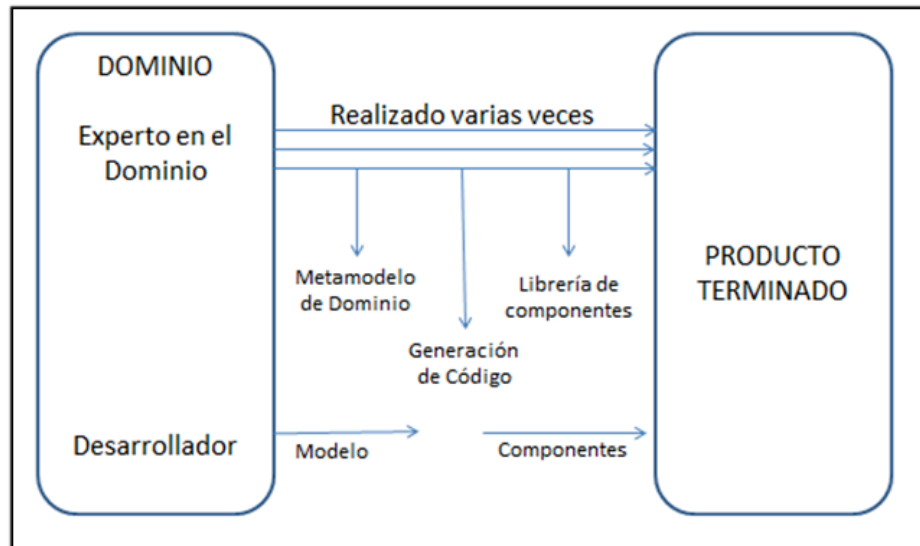


Figura 1. Overview de MetaEdit+, la tecnología que permite elaborar herramientas de modelado en un entorno de Metacase.

La propuesta Metacase (Alderson, 1991) es identificada como un proveedor de entornos de modelado disponible para todos los usuarios que buscan herramientas para construir software dirigido por modelos. Esto es logrado gracias a que dispone de lenguajes de modelados y generadores en su composición. A continuación es presentada la tecnología denominada metaEdit+ con la cual está fundamentada esta propuesta, acompañada de un modelo de proceso para la creación de dichas herramientas:

MetaEdit+ (Tolvanen & Kelly, 2009) es una herramienta comercial, basada en repositorios que permite crear modelos de lenguajes de dominio sin escribir una sola línea de código. Esta herramienta dirige un metamodelo y especifica el *Mapping* desde este hasta generar su código. Como lo muestra en la figura 1, la herramienta utiliza lenguajes de dominio en su proceso de modelado. Para definir el metamodelo MetaEdit+ usa un lenguaje de meta-metamodelado llamado GOPPRR (*Graph-Object-Property-Port-Role-Relationship*) que contiene conceptos de dominio y las reglas que se deben cumplir. A través de esta herramienta es identificado el siguiente proceso de modelado:

1. Definir Meta-Metamodelo.
2. Definir conceptos de dominio o metalenguaje.
3. Definir reglas de dominio.
4. Definir símbolos de notación.

5. Configurar el generador de código.
6. Configurar editor gráfico.

2.1.2.2. Propuesta Microsoft

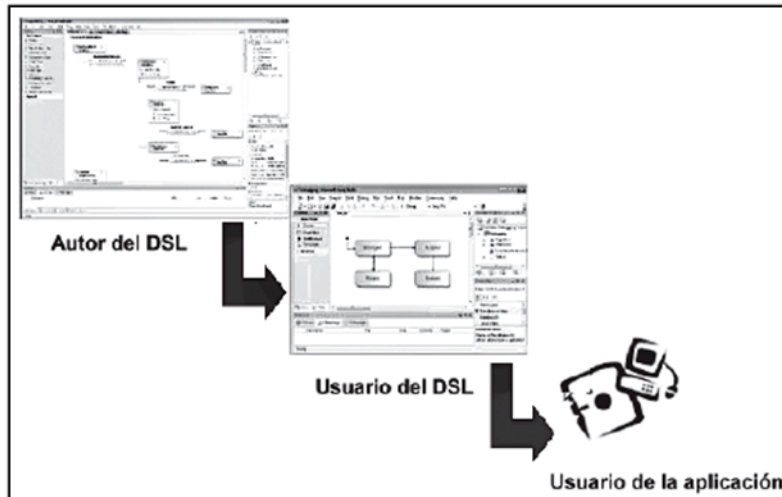


Figura 2. Roles definidos en DSL (Domain Specific Language)

Esta propuesta está fundamentada en un producto de Microsoft denominado Visual Studio (Cook, Jones, Kent, & Wills, 2007) el cual tiene las herramientas necesarias para construir DSL (Lenguajes de Dominio Específicos) para la construcción de sus modelos. También cuenta con la capacidad de producir código con generadores asociados.

Según un seguimiento hecho al proceso de modelado que Visual Studio utiliza y con base a los roles identificados en la Figura 2. Son descritos los siguientes pasos como el proceso propio de esta tecnología:

1. Definir Meta-metamodelo
2. Definir un lenguaje DSL:
 - Definir modelo de dominio.
 - Definir forma de los elementos (Notación).
 - Establecer conexión entre elementos de modelo.
3. Generar código de modelo.
4. Expresar validaciones y restricciones sobre los modelos.

2.1.2.3. Propuesta Eclipse

La herramienta Eclipse permite crear plataformas de desarrollo abiertas. Eclipse es identificado como multi-plataforma² y posee un *framework* pensado para los desarrolladores que construyen sus herramientas basadas en modelos.

La Fundación Eclipse es una organización independiente sin fines de lucro que fomenta una comunidad de programación software de código abierto. Esta comunidad cuenta actualmente

² Que dicha plataforma puede ejecutarse en múltiples sistemas operativos.

con 60 proyectos y para este trabajo está enfocado en el proyecto dedicado a promover tecnologías de desarrollo basadas en modelos, denominado Eclipse Modelling Project. Dicho proyecto está compuesto por otros sub-proyectos relacionados los cuales son presentados en la Tabla 2 y son las bases para desarrollar aplicaciones basadas en modelos.

Tabla 2. Sub proyectos Eclipse para desarrollo basado en modelos.

Tarea	Proyecto
Desarrollo de sintaxis abstracta	EMF (Eclipse Modelling Framework)
Desarrollo de Sintaxis concreta	GMF (Graphic Modelling Framework) TMF (Textual Modelling Framework)
Transformación de modelo	QVT(Query/View/Transformation) JET (Java Emitter Templates) MOF2Text (Modelling Object Framework To Text)
Implementación de estándar	UML2 OCL OMD XSD

La construcción de una herramienta de modelado a través de EMF también plantea un proceso con pasos específicos como las otras propuestas. La comunidad Eclipse busca la forma de brindar a los desarrolladores la oportunidad de construir sus herramientas de modelados del nivel más completo posible y el mejor resultado es obtenido por medio de los siguientes pasos que conforman el proceso de desarrollo de una herramienta basada en modelos en Eclipse:

1. Definición de un modelo de dominio.
2. Construcción del modelo de definición gráfica.
3. Definición de herramientas para el editor gráfico.
4. Definición de relaciones entre elementos.
5. Generación de código.

2.2. Marco teórico

La sección anterior presentó diferentes trabajos paralelos a la temática de este trabajo buscando construir una propuesta concreta como solución a la problemática planteada, pero es necesario también presentar algunos conceptos que ayuden al lector a comprender mejor las diferentes tecnologías y metodologías utilizadas. Por lo tanto, esta sección presenta un marco teórico que abarca las definiciones consideradas como relevantes a tener en cuenta durante el proceso de desarrollo de todo el trabajo. Este a su vez consta de dos partes: Primero, la teoría referente a las interfaces de usuario y sus componentes; y segundo, la teoría referente a la ingeniería dirigida por modelos. A continuación son expuestos los conceptos en mención.

2.2.1. Conceptos generales involucrados con interfaces de usuario y sus componentes

2.2.1.1. Definición y composición de una Interfaz de Usuario

El usuario es comunicado con un desarrollo software en una máquina, equipo o dispositivo a través de una interfaz de usuario (UI). La UI permite interactuar a los usuarios con todo tipo de actividades y procesos que el desarrollo ofrezca y su objetivo es permitir el funcionamiento y

control más efectivo de la máquina o programa desde la interacción con el humano. Partiendo de este concepto y analizando el entorno informático en el que estamos sometidos, el estudio más completo de una interfaz de usuario es logrado reconociendo la disparidad de usuarios, lenguajes, aplicaciones, además de la velocidad con la que todos estos factores están cambiando (Mercovich & Aires, 1999). Uno de los desafíos más interesantes en el entorno de desarrollo de software es la implementación de una interfaz de usuario útil y ergonómica. Es decir, la interfaz se adapta a las capacidades y necesidades de los usuarios de la manera más fácil posible. Cuando un desarrollador comienza su trabajo de diseño de software debe tomar los desarrollos similares como punto de referencia para el suyo, y así siendo cuidadoso en la creación, alcanzará un gran éxito en todos los factores.

La interfaz de usuario está compuesta de elementos de interacción con el usuario a través de un lenguaje sensorial, es manifestado mediante composiciones gráficas, sonidos, respuestas táctiles, etc. Estos elementos son distinguidos con estilos y diseños personalizables para el desarrollador como un sistema unificado construido por partes (fondos, ventanas y paneles, botones y controles, iconos, imágenes, textos, videos, sonidos, animaciones, etc.), todo esto con el fin de construir una UI completa, organizada y de calidad. De hecho existen investigaciones que aportan a los desarrolladores estudios para definir la composición de sus interfaces según su contexto (Gómez, Marín, & Díaz, 2014). Dado que las interfaces no son el objetivo en la informática sino solamente un medio para llegar a él, la interfaz mejor diseñada es la que no se ve. Sin embargo las interfaces fallan en este campo por dos motivos, o es nueva y desconocida, o es conocida pero mal diseñada. A continuación es expuesto el concepto de una interfaz mal diseñada en el tema de desarrollo software o también conocida como interfaz débil.

2.2.1.2. Interfaz de usuario débil

Una interfaz de usuario mal diseñada o débil trae consecuencias serias para el futuro uso de la misma. Una interfaz débil es aquella que no es capaz de alcanzar con los niveles de calidad de desarrollo en el momento de la interacción con el usuario. Este nivel de calidad es medible a través de un factor denominado usabilidad, este factor permite detectar que tan buena es la comunicación entre el usuario y la interfaz. Así pues, el factor usabilidad es el que debe tenerse en cuenta antes de diseñar e implementar una UI. La Figura 3 muestra la definición más clara de usabilidad conectando diferentes elementos.

El valor de un cliente insatisfecho es muy importante para un desarrollo de una UI pues es tarea de los desarrolladores identificar todos los errores que puedan tener la misma. Medible o no medible dicho error tiene un costo que ningún programador o administrador de software lo quisiese pagar más tiene que ser solucionado, pues todo error es consecuente.

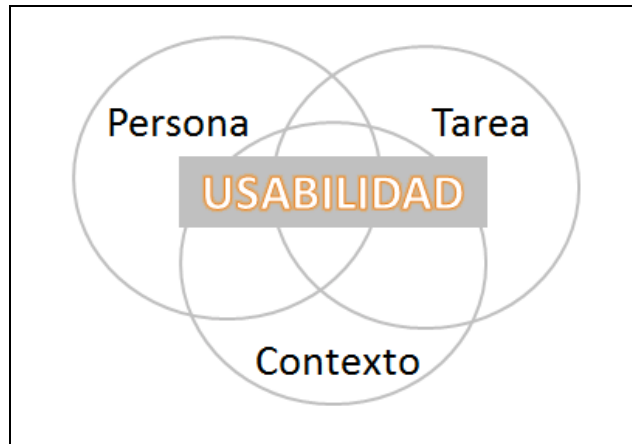


Figura 3. Entorno para el diseño de una interfaz de usuario

En conclusión, aumentar los recursos destinados al desarrollo de la interfaz es una excelente inversión, teniendo en cuenta la relación costo/beneficio medible y segura. Además pueden obtenerse beneficios no medibles en dinero como el aumento de la satisfacción, siempre y cuando sea hecho un buen estudio previo en torno a la usabilidad de la misma. El concepto que viene a continuación, precisamente expande todo o que se debe de tener en cuenta en cuanto a la usabilidad de una interfaz de usuario.

2.2.1.3. Usabilidad

La usabilidad es definida como la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso (Standardization & Commission, 2001). Para el estándar ISO 9241, que trata los requerimientos ergonómicos, la usabilidad es “el grado en el que un producto puede ser utilizado por usuarios para conseguir objetivos específicos con efectividad, eficiencia y satisfacción en un determinado contexto de uso” (Standardization, 1998). Para obtener un desarrollo software de calidad es medido a partir de la usabilidad teniendo en cuenta 4 factores básicos:

1. Atracción del software

La primera impresión del usuario con un software es tan importante que depende de este, el nivel de aceptación y uso del mismo. La eficacia y comodidad de los procesos al realizar una actividad, la visibilidad en cualquiera que sea el tipo de dispositivo y tecnología, juegan un papel muy importante en la valoración de la experiencia de usuario.

2. Entender el software

En el marco de desarrollo de una interfaz de usuario, cada uno de sus componentes debe familiarizarse fácilmente con el usuario y esto es medido a través de la “facilidad de aprendizaje” que indica cómo un usuario es capaz de realizar correctamente las tareas (Iso, 1998).

3. Usar el software

Si el sistema no es percibido como una herramienta que ayuda al usuario a realizar sus tareas, se dificulta la aceptación del sistema. Puede ocurrir que el sistema no llegue a usarse en absoluto, o que sea usado con escasa eficiencia. Las tareas del usuario deben ser respaldadas convenientemente por el sistema (Mascheroni, Greiner, Petris, Dapozo, & Estayno, 2012). El hecho de usar constantemente una interfaz le brinda la oportunidad al usuario de detectar

errores o falencias en el sistema, o por el contrario expresar la conformidad que este tiene con la interfaz.

4. Permanecer con el software

El usuario ideal para el desarrollador de software no solamente es el que le atraiga y use su interfaz sino el que la prefiera por encima de otras ya revisadas, esto le da un alto nivel de certificación y demuestra a través de estudios estadísticos la periodicidad con la que los usuarios ingresan y usan el trabajo desarrollado en este caso: una interfaz de usuario. Al haber alcanzado este objetivo, el desarrollador debe enfocarse en buscar la manera de solucionar los errores identificados y estar en una constante actualización y modernización según cada tecnología le permita.

Además de garantizar el diseño de una buena interfaz con la usabilidad, también es importante que esta cuente con componentes funcionales capaces de satisfacer las necesidades del usuario en la aplicación. Así, en la siguiente sección es introducido el concepto de componente de una interfaz en el contexto del presente trabajo de grado.

2.2.1.4. Componentes de una interfaz de usuario

Toda interfaz de usuario contiene componentes gráficos que permiten la comunicación entre el usuario y una función específica que él mismo quiera realizar. Existen componentes de diferentes tipos que según el contexto de la aplicación, han evolucionado a través del tiempo y de esta manera entra más a fondo al estudio de los mismos. Esta sección pretende analizar los conceptos necesarios para construir un componente de interfaz de usuario en cada uno de sus niveles según un modelo específico de arquitectura. El desarrollo software de un componente parte desde la definición de un modelo de arquitectura, es por eso que a continuación es introducido este primer concepto:

- **Modelo de arquitectura**

Al desarrollar una UI hay que tener en cuenta que la elaboración de sus componentes siga un modelo de desarrollo apropiado que guíe el proceso desde plasmar la idea principal hasta la presentación final del mismo. Esto es denominado modelo de arquitectura y le permite al desarrollador abarcar todos los posibles errores que puedan presentarse en el desarrollo y además garantizar que el nivel satisfacción del usuario final sea alto en la interactividad con el componente. Un gran número de componentes son afectados cuando la interfaz es pensada con fines de modificaciones a futuro, los modelos de arquitecturas permitirán al desarrollador rediseñar y alterar los componentes previamente desarrollados sin afectar el resto.

Establecer un modelo de este tipo trae mucha relevancia en el desarrollo de una aplicación, existen factores que demuestran el éxito de los componentes sólo por el hecho de basarse en un proceso de modelado y son presentados a continuación según un estudio realizado a la arquitectura de software (Reynoso, 2004):

- Comunicación mutua. La AS representa un alto nivel de abstracción común que la mayoría de los participantes, si no todos, pueden usar como base para crear entendimiento mutuo, formar consenso y comunicarse entre sí. En sus mejores expresiones, la descripción arquitectónica expone las restricciones de alto nivel sobre el diseño del sistema, así como la justificación de decisiones arquitectónicas fundamentales.
- Decisiones tempranas de diseño. La AS representa la encarnación de las decisiones de diseño más tempranas sobre un sistema, y esos vínculos tempranos tienen un peso

fuera de toda proporción en su gravedad individual con respecto al desarrollo restante del sistema, su servicio en el despliegue y su vida de mantenimiento.

- Restricciones constructivas. Una descripción arquitectónica proporciona planos parciales para el desarrollo, indicando los componentes y las dependencias entre ellos. Por ejemplo, una vista en capas de una arquitectura documenta típicamente los límites de abstracción entre las partes, identificando las principales interfaces y estableciendo las formas en que unas partes pueden interactuar con otras.
- Reutilización, o abstracción transferible de un sistema. La AS encarna un modelo relativamente pequeño, intelectualmente tratable, de la forma en que un sistema es estructurado y sus componentes se entienden entre sí; este modelo es transferible a través de sistemas; en particular, puede aplicarse a otros sistemas que exhiben requerimientos parecidos y puede promover reutilización en gran escala. El diseño arquitectónico soporta reutilización de grandes componentes o incluso de *frameworks* en el que pueden integrarse componentes.
- Evolución. La AS puede exponer las dimensiones a lo largo de las cuales puede esperarse que evolucione un sistema. Haciendo explícitas estas “paredes” perdurables, quienes mantienen un sistema pueden comprender mejor las ramificaciones de los cambios y estimar con mayor precisión los costos de las modificaciones. Esas delimitaciones ayudan también a establecer mecanismos de conexión que permiten manejar requerimientos cambiantes de interoperabilidad, prototipado y reutilización.
- Análisis. Las descripciones arquitectónicas aportan nuevas oportunidades para el análisis, incluyendo verificaciones de consistencia del sistema, conformidad con las restricciones impuestas por un estilo, conformidad con atributos de calidad, análisis de dependencias y análisis específicos de dominio y negocios.
- Administración. La experiencia demuestra que los proyectos exitosos consideran una arquitectura viable como un logro clave del proceso de desarrollo industrial. La evaluación crítica de una arquitectura conduce típicamente a una comprensión más clara de los requerimientos, las estrategias de implementación y los riesgos potenciales.

Según la literatura revisada, desde los años 90 existen propuestas para la elaboración de componentes de interfaz de usuario a partir de modelos, una de ellos es la de Dieterich (Dieterich, Malinowski, Kühme, & Schneider-Hufschmidt, 1993), en donde propone partir de cinco niveles de adaptación: léxico, sintáctico, semántico, tarea y meta. En 2001 es presentado un modelo más reciente definiendo los niveles léxico sintáctico y semántico usando ejemplos como definiciones (Eisenstein, Vanderdonckt, & Puerta, 2001) y así poco a poco diferentes diseñadores de UI proponen sus propios modelos para la construcción de sus componentes aunque depende mucho del tipo de aplicación y el entorno donde es desarrollado. Hay que tener en cuenta bajo cualquier modelo propuesto, que los componentes deben organizarse a través de una sintaxis abstracta y una sintaxis concreta que son conceptos importantes presentados más adelante.

El diseño de un modelo de arquitectura de software no solo implica implementarlo y nada más sino que es un proceso de modelado que como lo muestra la Figura 4, podrían identificarse tres etapas:

- Definir los requerimientos: involucra crear un modelo desde los requerimientos que guiarán el diseño de la arquitectura basado en los atributos de calidad esperados.
- Diseño de la arquitectura: involucra definir la estructura y las responsabilidades de los componentes que comprenderán la Arquitectura de Software.

- Validación: significa evaluar la arquitectura, pasando a través del diseño contra los requerimientos actuales y cualquier posible requerimiento a futuro.

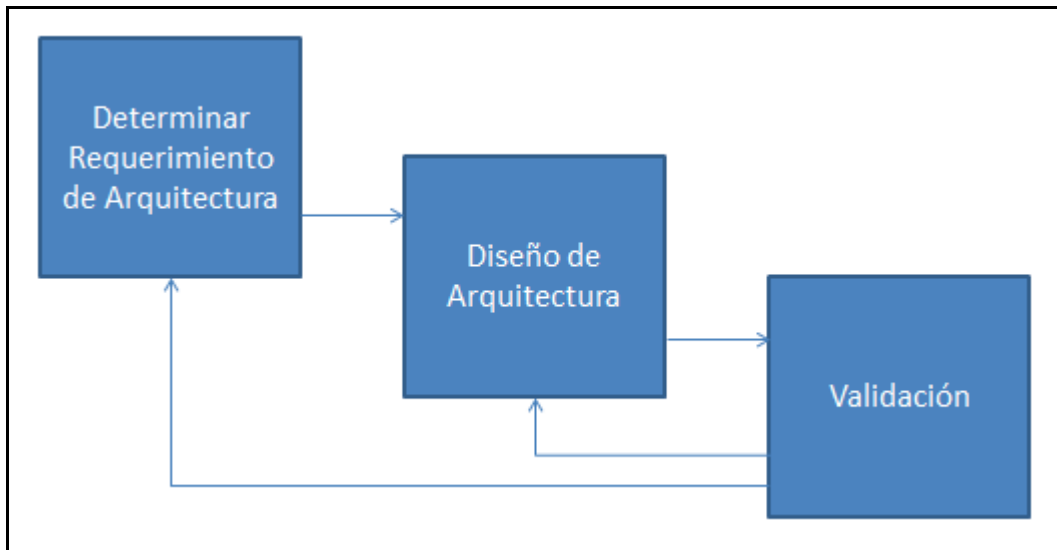


Figura 4. Esquema de proceso de modelado de Arquitectura Software

El diseño del modelo de arquitectura expuesto anteriormente es aplicable para cualquier tipo de componente, pero con el fin de hacer énfasis al contexto del presente trabajo de grado, es traído el concepto de componentes basados en distribución de contenido de video presentado a continuación.

- **Componentes basados en distribución de contenido de video**

Las interfaces de usuario han ido creciendo al nutrirse de nuevos componentes según las necesidades que encuentran los desarrolladores en las aplicaciones. Como fue plasmado en el planteamiento del problema, siempre van ha encontrarse que los desarrollos en su gran mayoría utilizan los componentes convencionales, pero es muy poca la documentación centrada en los componentes UI enfocados en distribución de contenido de video o relacionados con la interactividad de este tipo de sistemas con el usuario.

El concepto de componentes basados en distribución de video es un concepto que hace referencia a la variación de componentes relacionados a sistemas interactivos en el contexto de aplicación. Por mencionar un ejemplo, a nivel de educación, puede utilizarse un reproductor de video para ubicar contenido en una clase, en este componente pueden encontrarse diversas funcionalidades que permiten al docente hacer de su clase más interactiva. Este tipo de componentes también son relacionados entre sí con otros en una misma interfaz, es decir, es posible relacionar el contenido de un reproductor de video con el contenido de una presentación de diapositivas. Otro ejemplo, muy común a nivel de entretenimiento, es evidenciar componentes como listas de reproducción con funcionalidades de *raking* o redes sociales.

El desarrollo o implementación de este tipo de componentes tiene un nivel de complejidad más alto que los widgets convencionales debido a que es considerado como un desarrollo software completo. Es por eso que la siguiente sección está dedicada a la teoría que debe tenerse en cuenta para lograr desarrollar completamente un componente de interfaz de usuario con distribución de contenido de video.

2.2.1.5. Desarrollo de un componente basado en distribución de contenido de video

El desarrollo de estos componentes debe basarse en una arquitectura software que permita distinguir la composición del componente, para esto es traído a colación el concepto de modelo vista controlador como parte de la metodología de desarrollo. También posteriormente son presentadas las tecnologías existentes que permiten la implementación de dichos componentes.

- **Modelo vista controlador**

La arquitectura de software tiene un patrón fundamental que busca separar los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo que gestiona los eventos y las comunicaciones. Esto es denominado MVC (Modelo Vista Controlador). El concepto central detrás de las librerías de interfaz de usuario provistas por SmallTalk (Goldberg, 1984) está basado en el patrón de diseño MVC, creado por el profesor Trygve Reenskaug; en donde se propone diseñar un desarrollo software dividiendo el trabajo en tres módulos claramente identificables y con funcionalidad bien definida: la vista, el modelo y el controlador (Deacon, 2009).

El hecho de establecer un patrón MVC garantiza la actualización y mantenimiento del software de forma sencilla y reduce el factor tiempo. La implementación de cada elemento por separado y los pasos que plantea el patrón para el diseño y desarrollo de aplicaciones que presenten una gran interactividad con el usuario, puede reflejarse en el ejemplo que presenta la Figura 5 (Gutiérrez, 2014), la cual muestra cómo a partir de los tres elementos es construido un patrón de implementación a través de un diagrama de secuencia.

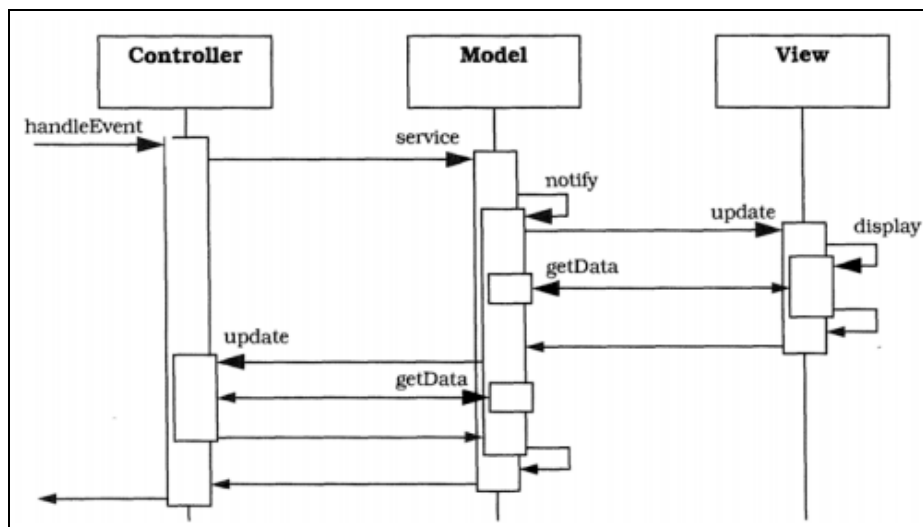


Figura 5. Ejemplo de diagrama de secuencia para la construcción de un MVC

El modelo es un conjunto de clases que representan la información o datos del mundo real que el sistema debe procesar, también incluye las reglas de negocio para implementarse.

Las vistas son el conjunto de clases encargadas de mostrar al usuario la información contenida en el modelo. Una vista está asociada a un modelo, pudiendo existir varias vistas asociadas al mismo modelo. Gracias a esta, existe la capacidad de realizar transformaciones gráficas entre diferentes niveles de jerarquía de la aplicación. Una vista obtiene del modelo solamente la información que necesita para desplegar y actualizarse cada vez que el modelo del dominio cambia por medio de notificaciones generadas por el modelo de la aplicación (Bergin, 2007).

Finalmente el controlador principalmente es dedicado a responder eventos e invoca peticiones al modelo cuando solicita alguna información. Los eventos comúnmente son realizados por el usuario y a partir de estas peticiones, el controlador es encargado de modificar el modelo o de abrir y cerrar vistas. Un fundamento esencial, es que el controlador tiene acceso al modelo y las vistas, pero ni el modelo ni las vistas tienen acceso al controlador.

- **Tecnologías existentes**

Esta sección recoge todas aquellas tecnologías que permitirán el desarrollo de los componentes en términos tanto de lenguajes, *frameworks*, librerías y programas de entornos de desarrollo. Es importante tener en cuenta que es buscada tanto la eficiencia en ejecución como la facilidad de interpretación de código. A continuación en la Tabla 3 son presentadas las tecnologías seleccionadas.

Tabla 3. Tecnologías existentes para desarrollo de componentes UI.

Tecnología	Descripción
HTML5(Network; Network)	El lenguaje de marcado para hipertextos o html hace referencia al lenguaje de más bajo nivel para la construcción de páginas web, este lenguaje permite la creación y representación visual del contenido sin funcionalidad. Este lenguaje le proporciona al navegador cómo se tiene que mostrar el contenido (palabras, documentos, audio, video, imágenes, etc.) identificando los distintos tipos de elementos predefinidos mediante sus etiquetas encapsuladas (“<>”), algunas de estas etiquetas no incrustan su información directamente en su propio código si no que hacen referencia a la ubicación de sus elementos a través de texto.
Javascript (Network; Network)	JavaScript es un lenguaje altamente dinámico utilizado principalmente del lado del cliente pero también usado en el lado del servidor, JavaScript es un lenguaje que puede operar de forma orientada a objetos basados en prototipos en lugar de clases o de manera procedimental que trabaja en un flujo de operación mediante pasos para llegar a un resultado. JavaScript trabaja del lado del cliente en las páginas web agregando interactividad y funcionalidad a la página web.
Jquery	JQuery es una librería de JavaScript utilizada para el manejo y manipulación de elementos HTML, también sirve para el manejo de eventos, efectos y animaciones. Esta librería es utilizada en combinación con el código JavaScript tradicional y trabaja de la mano con este.
PHP (P. Group; Welling & Thomson, 2005)	PHP es un lenguaje que opera del lado del servidor diseñado para la programación web, este lenguaje es combinado directamente con el lenguaje HTML o también opera de manera independiente. Una de sus principales ventajas es su fácil conectividad de motores de bases de datos a páginas web dinámicas, entre ellas PostgreSQL y MySQL. Este lenguaje es depurado de manera anónima ya que no muestra una interfaz al usuario hasta que tenga un resultado de su ejecución.

	<p>Este lenguaje es diseñado principalmente para la ejecución de scripts del lado del servidor, esto hace que tenga mucha potencia para recopilar datos de formularios, generar páginas con contenidos dinámicos, o enviar y recibir cookies.</p>
<p>CSS3 (Network)</p>	<p>CSS es el lenguaje de hojas de estilo en cascada, este lenguaje proporciona el diseño gráfico de una página web a partir de la creación de estilos para cada elemento HTML. En su última versión CSS3 tiene algunas nuevas funcionalidades y mejoras como: transiciones, animaciones, layouts con multicolumnas, cajas flexibles y grid layouts. Lo que hace este lenguaje es dividir el contenido de su representación visual, y es enfocado en el estilo del contenido, dando así una modularización de la vista del contenido.</p>
<p>JAVA (Oracle; Perry, 2012)</p>	<p>El lenguaje de programación Java es un derivado del lenguaje C por eso tiene muchas similitudes, pero Java es un lenguaje de nivel superior que C. Su programación es orientada a objetos basada en clases, por lo que dentro de estas clases son compuestas con sus variables, constantes y métodos. Java es un lenguaje universal utilizado en muchos entornos de funcionamiento como: celulares, servidores, aplicaciones de escritorio y en navegadores web, entre otros.</p>
<p>Materialize CSS (Network)</p>	<p>Materialize es un framework para el desarrollo web para la capa de presentación (front-end) el cual está basado en material design, este es un lenguaje de diseño creado por Google. Materialize es un framework moderno y responsivo el cual acelera el desarrollo ya que posee componentes, transiciones y animaciones, con una interfaz de usuario final. Este framework posee ejemplos específicos de código para cada uno de sus componentes y animaciones con el fin de enfocar al desarrollador en programar la funcionalidad de la página web y no dedicar tiempo a la interfaz de la misma.</p>
<p>NodeJS</p>	<p>NodeJS es un entorno multiplataforma en tiempo de ejecución y de código abierto creado para la capa del servidor pensando en el desarrollo de programas involucrados con servidores web.</p>
<p>SocketIO (SOCKET.IO)</p>	<p>Es una librería de javaScript que permite la comunicación bidireccional en tiempo real basada en eventos, la comunicación está dada por medio de un único socket TCP (WebSocket). Tiene una buena compatibilidad, fiabilidad y velocidad. El lado del cliente funciona sobre el navegador y el lado del servidor sobre NodeJS.</p>

2.2.2. Conceptos relacionados a la ingeniería dirigida por modelos.

Este trabajo busca el camino más eficiente e innovador por parte de la ingeniería de software para plantear una solución en cuanto al desarrollo software y la ingeniería basada en modelos es una propuesta adoptada y es respaldada por los argumentos que exponen los conceptos a continuación.

Esta sección recoge los conceptos generales a tener en cuenta en la ingeniería dirigida por modelos en el marco de desarrollo de un *framework*. A continuación son expuestos los conceptos más relevantes.

2.2.2.1. Framework de desarrollo

Un *framework* es un modelo de un dominio en particular o un aspecto importante del mismo, tiene como objetivo ofrecer una funcionalidad definida, auto contenida, siendo construido a través de patrones de diseño, y su característica principal es su alta cohesión y bajo acoplamiento (Riehle, 2000). El núcleo de un *framework* es definido en términos de clases y modelos con un rol específico. Un modelo a seguir puede ser definido nuevamente por el *framework*, o ser importado desde otro *framework*, o ser importado desde una librería de clases. En general, con el término *framework*, es referido a una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. En otras palabras, un framework puede considerarse como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta.

Un desarrollo software fundamentado en *frameworks* es una importante herramienta para desarrollar soluciones a problemas con características específicas, y es la segunda generación de metodologías para el desarrollo de sistemas informáticos (Ambler et al. 2005). Este tipo de desarrollo debe separar cada elemento de los modelos tanto a nivel de negocio, como de aplicación y tecnología, para así asegurar la agilidad y flexibilidad de los modelos del sistema. En la sección a continuación es expuesta la composición de un *frameworks* dirigido por modelos, planteado como una solución al desarrollo software.

2.2.2.2. Ingeniería dirigida por modelos –MDE

Este concepto hace referencia al desarrollo software con un enfoque en la transformación de un modelo abstracto creado a implementaciones concretas. En esta ingeniería es realizada una generación de código automática prácticamente, pues utiliza transformaciones o traslaciones de modelos. Un sistema con estas características cuenta con un grupo de modelos del mismo, ampliando los puntos de vista del mismo. El propósito de utilizar este tipo de ingeniería esta centrada en la reducción de la brecha existente entre el problema planteado y la implementación de su solución. Esto es, a través de técnicas, métodos, procesos y herramientas de soporte.

La ingeniería dirigida por modelos (MDE) tiene un alcance más amplio que las actividades de desarrollo de MDA combinando procesos y análisis con diferentes arquitecturas (Kent, 2002).

2.2.2.3. Diseño de interfaz de usuario basado en modelo -MBUID

Dado que MDE pretende elevar el nivel de abstracción de las aplicaciones de software, puede servir como base para diseñar interfaces de usuario adaptativas debido a la posibilidad de aplicar diferentes tipos de adaptaciones en los distintos niveles de abstracción. El criterio fundamental para el desarrollo de dichas interfaces es basado en utilizar modelos declarativos para conseguir la automatización del proceso de desarrollo (Calvary & Coutaz, 2014). Este

enfoque ha recibido la mayor atención en la literatura. A continuación son identificados los siguientes enfoques basados en modelos utilizados para desarrollar interfaces de usuario.

- El modelado estático

Basado en modelos para el diseño de UI y eventualmente termina en una fase antes de la generación de código. Por definición, los modelos estáticos no pueden cambiar en tiempo de ejecución, por lo tanto no son útiles más allá de la fase de desarrollo.

- Modelado generativo en tiempo de ejecución

Mantiene los modelos vivos en tiempo de ejecución para adaptar los artefactos de interfaz de usuario basados en código que se generaron en tiempo de diseño.

- Modelado interpretado en tiempo de ejecución

No requiere generación de código para crear la interfaz de usuario. En su lugar, los modelos se interpretan en tiempo de ejecución para representar la interfaz de usuario.

Uno de los enfoques que puede dar solución a la situación problemática presentada en este trabajo de investigación es el desarrollo de la interfaz de usuario basada en modelos (MBUID). MBUID es un enfoque que tiene como objetivo hacer frente a los problemas de heterogeneidad relacionados a dispositivos, usuarios y contextos de uso, reduciendo el esfuerzo necesario para el desarrollo de la interfaz de usuario (IU) asegurando su calidad (Calvary & Coutaz, 2014).

MBUID parte de modelos independientes de la computación (CIM) para el desarrollo de la interfaz de usuario. Una vez desarrollados los modelos CIM se incorporan una serie de guías, reglas y aspectos relacionados con la presentación y usabilidad de la interfaz. Posteriormente es llevado a cabo una integración de estas especificaciones con el código de la aplicación que soportará la funcionalidad.

El proceso descrito anteriormente permite identificar tres características principales de los entornos MBUIDE: soporte para la generación automática de interfaces de usuario, uso de métodos declarativos para la especificación de las interfaces y adopción de una metodología para soportar el desarrollo de la interfaz (Orozco & Joseph, 2010).

Las metodologías de desarrollo de la interfaz de usuario deben ser centradas en el usuario, es decir, deben involucrar aspectos de HCI como es la usabilidad o más ampliamente, la experiencia de usuario. La inclusión de mecanismos de especificación de usabilidad en ambientes MBUID para potenciar el modelado de la interacción conllevará a generar interfaces de más calidad, de tal forma que la usabilidad esté incluida en la especificación de los modelos, desde los requisitos hasta el código resultante. Estos aspectos garantizarán la facilidad de uso de la interfaz, satisfacción de los usuarios y enriquecer la experiencia del usuario en el proceso interactivo (Law, Roto, Hassenzahl, Vermeeren, & Kort, 2009; Rosson & Carroll, 2002).

Los modelos de tiempo de ejecución constituyen un importante campo de investigación en la ingeniería dirigida por modelos (France & Rumpe, 2007), pues estos suelen ser más adecuados para soportar el comportamiento adaptativo. Aunque si es verdad que en ciertos escenarios al usar modelos de este tipo no es suficiente mantener los artefactos basados en el código generado, sigue siendo una opción óptima para el desarrollo de interfaces.

Al comparar el enfoque MDE para la construcción de IU con las técnicas tradicionales (Myers, Hudson, & Pausch, 2000), indican que este enfoque sufre de una alta curva de aprendizaje. Sin embargo, aunque la curva de aprendizaje es generalmente más alta para MDE que las técnicas tradicionales, los desarrolladores podrían acostumbrarse rápidamente a MDE para diseñar

interfaces de usuario si es proporcionado el soporte de herramientas apropiado. Además, al evaluar si esta curva de aprendizaje es justificable, podemos ver que el MDE está agregando valor a los enfoques tradicionales basados en herramientas, mejorando la trazabilidad, la independencia tecnológica y la capacidad de realizar comprobaciones sobre el resultado general de la adaptación de la IU. Podemos decir que el MDE es un enfoque viable según el trabajo de investigación a llevarse a cabo, por ejemplo Myers (Myers et al., 2000) considera que MDE sufre de un comportamiento impredecible y tiene un bajo tope que indica que es incapaz de producir UI avanzadas. Sin embargo, esta consideración es hecha con enfoques basados en MDE completamente automatizados en mente. Otros enfoques basados en MDE aplican procedimientos casi automatizados que permiten que las UIs avanzadas y predecibles sean producidas apoyando la entrada del diseñador en todos los niveles de abstracción, especialmente en la IU concreta. Por lo tanto, es considerada la capacidad de MDE para proporcionar una buena integridad y el control sobre la interfaz de usuario que depende de la aplicación.

2.2.2.4. Composición de un *framework* de desarrollo dirigido por modelos

El concepto anteriormente presentado permite introducir la idea de un *framework* como base a la solución del problema planteado. Es decir, es posible plantear un desarrollo basado en modelos para un *framework* específico en el marco de desarrollo de una interfaz de usuario. Más específicamente, esta sección presenta un bosquejo para aplicar dichos conceptos en la creación dicho *framework*. La Figura 6 (William J Giraldo, Collazos, & Giraldo, 2009) presenta la composición de una interfaz de usuario construida a partir de modelos.

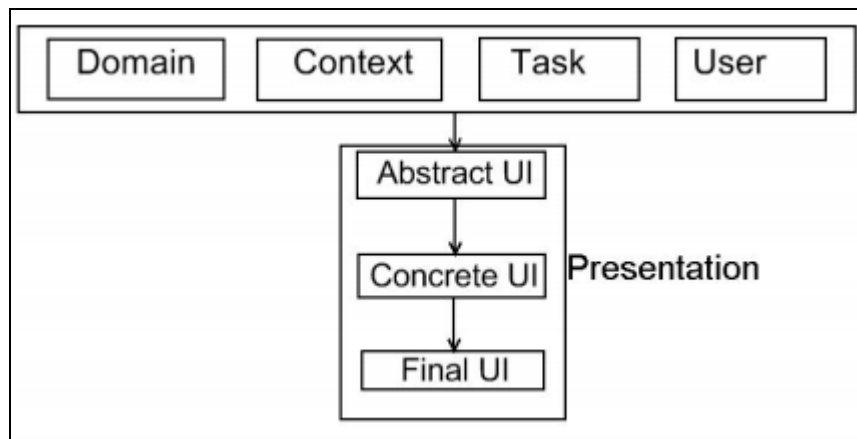


Figura 6. Modelo de interfaz de usuario

En la Figura 6 es posible distinguir que una interfaz de usuario puede ser generada a partir de al menos 4 modelos particulares: Dominio, contexto, tareas y usuario. A partir de cada modelo se captura la semántica del contexto del sistema a desarrollar y además define los requisitos de información para el desarrollo de la interfaz. A continuación se explican de forma breve cada uno de estos modelos.

- Modelo de dominio

Aquí son especificados los datos mediante el uso de diagramas de clase UML en donde son plasmados los objetos accesibles para el usuario en la interfaz. Este tipo de objetos son desplegados con sus atributos y relaciones asignadas. Los objetos del dominio son considerados como instancias de clases que representan los conceptos manipulados y que son totalmente independientes de la forma en que son mostrados a la interfaz.

- **Modelo de contexto**

Este modelo representa los recursos y capacidades que la interfaz de usuario debe ofrecer al usuario. Un modelo de contexto es un conjunto de elementos de la interfaz de usuario abstracta que representa los elementos necesarios o deseados y las áreas de trabajo que debe proporcionar un sistema para apoyar a uno o más casos de uso esencial. Los modelos de contexto son construidos a través de transformaciones partiendo de los objetos del negocio que representan a cada contexto de uso. El propósito es mantener una interfaz de usuario consistente que pueda ser compartida por diferentes casos de uso y evitar duplicaciones innecesarias (William J Giraldo et al., 2009).

- **Modelo de tareas**

Este modelo proporciona una descripción de las tareas que el usuario puede realizar a través de la interfaz. Adicionalmente este modelo permite especificar las relaciones existentes entre las tareas. Cada una de estas tareas a su vez, son divididas en acciones atómicas que permiten identificar cada paso para lograr los objetivos de la misma.

- **Modelo de usuario**

Este modelo es compuesto por un listado jerárquico de usuarios en estereotipos que comparten un mismo rol. La información que el usuario pueda presentar es compuesta de intereses comunes, conductas y responsabilidades. Un modelo de usuario es definido por la comprensión del uso del sistema y son muy indicados para alcanzar fines de usabilidad. Ellos identifican y representan los aspectos esenciales de las necesidades y requerimientos de los usuarios

Los niveles abstracto y concreto que hacen parte del módulo de presentación según la Figura 6, son relacionados exclusivamente con la interfaz de usuario y son definidos de la siguiente forma:

- **Nivel abstracto de interfaz de usuario**

El nivel abstracto describe la información de manera independiente de la modalidad. Es decir es independiente tanto del dispositivo donde se pretende desplegar como también del lenguaje de programación que se vaya a utilizar para su implementación. Este nivel es representado a través de modelos que describen

- **Nivel Concreto de interfaz de usuario**

El nivel concreto materializa la definición abstracta anterior en cada contexto de uso la información de una manera independiente de la implementación de la interfaz de usuario.

Finalmente, la interfaz de usuario final es representada por el código de ejecución en la plataforma seleccionada para la interfaz. Por tanto, el nivel final de la interfaz es totalmente dependiente de la plataforma y del lenguaje de programación con el cual es implementado.

2.2.2.5. Lenguaje Específico de Dominio (DSL)

Un lenguaje de modelado es un DSL, un lenguaje que está centrado en unos pocos conceptos dentro de todo el dominio. Este está compuesto por tres elementos (Kleppe, 2008): sintaxis abstracta, sintaxis concreta y semántica. La sintaxis abstracta abarca los conceptos principales del lenguaje, sus relaciones y un conjunto de restricciones y usualmente es implementado de la forma de un metamodelo. La sintaxis concreta define la notación que usa el lenguaje, que puede ser textual y/o gráfica. Finalmente, la semántica del lenguaje es definida generalmente por medio de una transformación aunque en la práctica es realizada informalmente. Los conceptos a continuación amplían dichas definiciones.

Sintaxis abstracta

Dado que los lenguajes de modelado actuales en general no están basados en texto, sino en gráficos, es conveniente recurrir a un mecanismo propio para este tipo de modelos. En los últimos años es desarrollada una técnica específica para facilitar la definición de los lenguajes gráficos, llamada “*metamodelado*” (Pons, Giandini, & Pérez, 2010). El metamodelo describe la sintaxis abstracta del lenguaje y constituye la base para el procesamiento automatizado de los modelos. Pero entonces, ¿a qué hace referencia una sintaxis abstracta? La sintaxis abstracta define los elementos del lenguaje y las reglas que establecen cómo pueden ser combinados los mismos. Esta sintaxis pretende mostrar la estructura de la aplicación previa al desarrollo y la forma más eficiente de plasmar esto es a través de los diagramas de clases de cada componente y la interacción entre las clases. Las clases, como bien lo define la informática se encarga de recoger todos los atributos que describen a los componentes y las funciones que el usuario puede ejecutar a partir de este.

Sintaxis concreta

En los lenguajes textuales la sintaxis concreta coincide (casi) exactamente con la sintaxis abstracta mientras que en los lenguajes gráficos se presenta una marcada diferencia entre ambas. La sintaxis concreta permite presentar el modelado del componente de una forma terminada al usuario a través de diagramas pero no es el desarrollo final. Es decir, aquí pueden encontrarse la vista del componente en la interfaz en cualquiera de las fases. Una sintaxis concreta es definida mediante diferentes mecanismos y no es relevante para las herramientas de transformación de modelos. La sintaxis concreta es la interfaz para el modelador e influye fuertemente en el grado de legibilidad de los modelos. Como consecuencia de este desacoplamiento, el metamodelo y la sintaxis concreta de un lenguaje pueden mantener una relación 1: n, es decir que la misma sintaxis abstracta (definida por el metamodelo) puede ser visualizada a través de diferentes sintaxis concretas. Inclusive un mismo lenguaje puede tener una sintaxis concreta gráfica y otra textual (Pons et al., 2010).

Semántica

La semántica es definida como la interpretación de un lenguaje, lo que le da sentido a la sintaxis concreta construida a partir de la sintaxis abstracta. Es el campo que concierne al estudio del significado de los lenguajes. La semántica formal de un lenguaje es definida por la estructura matemática que describe todas las posibles computaciones expresadas por el lenguaje (Pérez, Irazábal, Pons, & Giandini). Existen algunas propuestas para definir la semántica formal, entre ellas la semántica dada por la traducción a otro lenguaje. La ventaja de esta propuesta es que si existe una maquinaria que ejecute el lenguaje destino, es posible obtener una semántica ejecutable para el lenguaje por medio de la traducción (Izquierdo & Trujillo).

A partir de los conceptos anteriormente presentados para DSL y los conceptos de la sección anterior referente a la composición de un *framework* dirigido por modelos, es presentada en la Figura 7 un modelo de proceso para la creación de una herramienta basada en modelos adaptado para este trabajo. Este modelo de proceso fue construido también con base a la revisión de las diferentes tecnologías existentes que son presentadas en el estado del arte, obteniendo así una aproximación metodológica.

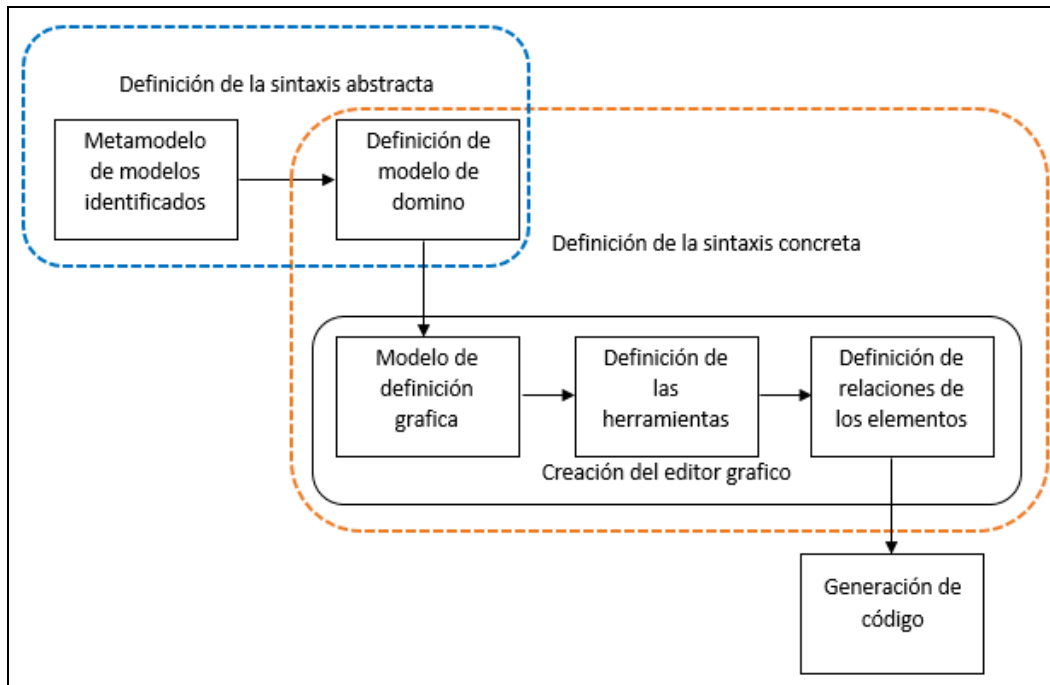


Figura 7. Modelo de proceso para la creación de un editor DSL

La Figura 7 presenta la aproximación metodológica propuesta, la cual pretende ser la base conceptual de un marco de desarrollo enfocado en los principios de MBUID. La aproximación metodológica describe el proceso para crear herramientas de modelado sin atarlo a una tecnología específica. El proceso consta de tres fases: i) definición de la sintaxis abstracta, ii) definición de la sintaxis concreta y iii) generación de código.

Capítulo 3

Identificación y selección de componentes

Esta sección enfoca la exploración de componentes específicos de los sistemas interactivos basados en distribución de contenido de video de acuerdo a diferentes contextos. La exploración tiene como fin identificar aquellos componentes más utilizados que posteriormente puedan integrarse a un *framework* MBUID, con el fin de facilitar el diseño de interfaces de este tipo de sistemas. La exploración es realizada a través de la interacción con diferentes herramientas de libre acceso. De cada herramienta son abstraídos los componentes y sus características y finalmente como una conclusión de este proceso, son presentados los componentes seleccionados para su implementación y posteriormente su incorporación en una herramienta MBUID. En las secciones siguientes son detallados cada uno de estos aspectos.

3.1. Selección de contexto de investigación

Los sistemas interactivos basados en distribución de contenido de video tienen muchos contextos de aplicación como: comunicación, entretenimiento, educación, vigilancia y satélites, medicina, la industria, entre otros. En cada contexto existe una gran variedad de componentes, Para el logro de los objetivos propuestos en este contexto es seleccionado uno en específico: la educación. La selección de este contexto fue escogido principalmente por el entorno en el que es desarrollado el presente trabajo y por el alto nivel de aplicación y avances tecnológicos que ha tenido en los últimos tiempos, sin desmeritar los componentes existentes con sus propios avances en los otros contextos.

Hay que tener en cuenta que el proceso de aprendizaje no es ajeno a los cambios tecnológicos, así pues el aprendizaje a través de las TIC (llamado en adelante *e-learning*) es el último paso de la evolución de la educación a distancia. El *e-learning* proporciona la oportunidad de crear ambientes de aprendizaje centrados en el estudiante (Boneu, 2007). Estos escenarios son caracterizados además por ser interactivos, eficientes, fácilmente accesibles y distribuidos. La visión de la educación virtual es que los estudiantes obtengan sus conocimientos de la forma más fácil y didáctica posible usando las nuevas tecnologías, esto requiere que en los dispositivos implementen interfaces de usuario robustas con componentes sencillos de manejar pero potentes en sus tareas. La Tabla 4 muestra los beneficios que trae la educación virtual con el apoyo de componentes en el contexto de educación aprendiendo y enseñando en espacios virtuales (Area & Adell, 2009). Esta tabla recoge la relación identificada entre algunos componentes con los principales aportes en la educación. Además permite justificar los beneficios que trae apuntar hacia la educación envuelta en la tecnología obteniendo resultados mejorables no solamente para los estudiantes sino también para los docentes.

Tabla 4. Aporte del e-learning a la mejora e innovación de la enseñanza.

Componentes	Beneficios
Recursos de archivos en línea	Extender y facilitar el acceso a la formación a colectivos e individuos que no pueden acceder a la modalidad presencial.
Tareas e investigaciones	Incrementar la autonomía y responsabilidad del estudiante en su propio proceso de aprendizaje.
Difusión de clases a través de un reproductor de video	Superar las limitaciones provocadas por la separación en espacio y/o tiempo del profesor-alumnos.
	Flexibilidad en los tiempos y espacios educativos
Quiz interactivo	Gran potencial interactivo entre profesor-alumno.
Enlaces de recursos relacionados a la temática de la clase	Acceder a multiplicidad de fuentes y datos diferentes de los ofrecidos por el profesor en cualquier momento y desde cualquier lugar.
Foro o chat entre alumnos y profesores	Aprendizaje colaborativo entre comunidades virtuales de docentes y estudiantes.

El proceso más adecuado para la construcción de entornos virtuales de aprendizaje es realizado a través de la gestión de contenido (Lara & Duart, 2005); y habla de contenido educativo no solamente refiriéndose a los materiales o documentos empleados como soporte para los estudiantes, sino todas las herramientas informáticas que una plataforma o aplicación puede brindar tanto a los estudiantes cómo a los profesores para elevar el nivel de aprendizaje a una alta calidad. Esto incluye la información, los espacios de interacción, las facilidades de comunicación en tiempo real o en diferido, y es con base a esto son analizados cuáles son los componentes UI que permiten alcanzar este nivel esperado.

Los programas de enseñanza de instituciones que usen herramientas informáticas con distribución de contenido de video traen beneficios tanto para los profesores como para los estudiantes. Entre ellos son mencionados, la interactividad entre la relación estudiante-docente, el ahorro de dinero en cuánto al desplazamiento físico, la oportunidad de adaptar las clases de acuerdo a los tiempos de los alumnos, la posibilidad de brindar recursos informáticos adicionales a los estudiantes, entre otros. Estos beneficios elevan la calidad de la educación y motivan a los desarrolladores a seguir apuntando en el desarrollo software en el contexto de la enseñanza.

La manera más clara de comprender cuáles son las necesidades de las herramientas informáticas educativas es llevar a cabo un buen proceso de gestión de contenidos, para esto es tenido en cuenta algunas variables importantes para hacer de este trabajo el más eficaz. Existen muchas herramientas cargadas de componentes con un excelente desarrollo de software, pero que lastimosamente al estar saturadas de componentes o información, pierden la comunicación con los usuarios ocasionando confusiones en las que el estudiante o el profesor deciden abandonar la herramienta. También está el otro extremo, en donde a falta de exploración, las interfaces son muy pobres y son dedicadas a compartir archivos pero no existe una interactividad entre los usuarios. El objetivo es poder buscar un equilibrio exitoso en el momento de desarrollar, y este equilibrio pretende lograr en esta investigación a partir de dos pasos:

1. Revisión de herramientas software dedicadas a la educación y exploración de sus interfaces de usuario.
2. Identificación de componentes tecnológicos de acuerdo a las variables críticas de formación en red.

La Figura 8 (Cabero-Almenara, 2006) plasma las variables críticas identificadas para basarse en la construcción de los componentes de la UI en el contexto educativo. Estas variables deberían de percibirse en la interacción y no de forma aislada. Sin entrar en detalle en cada una de las variables pero sí es necesario tenerlas en cuenta en el momento de interactuar con las herramientas y obviamente al momento de diseñar interfaces con componentes de contenido de video.

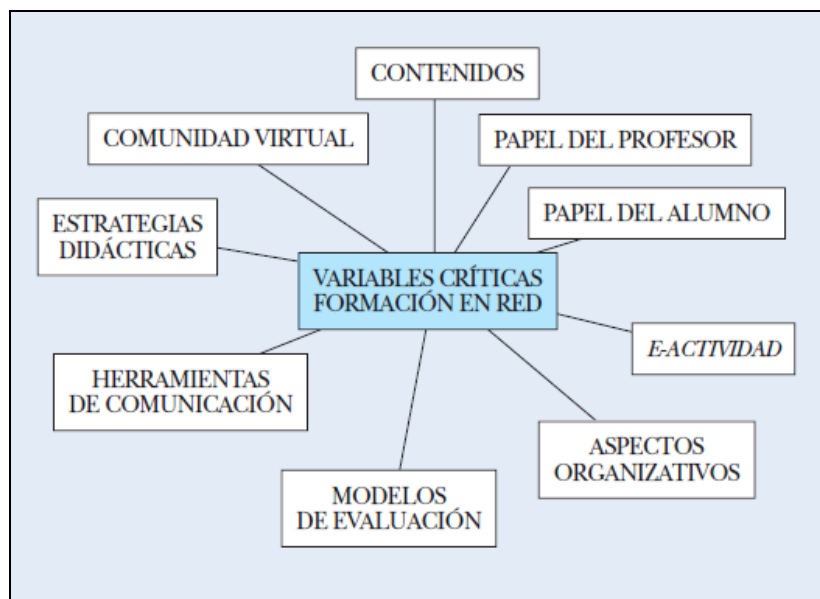


Figura 8. Variables críticas en la formación en red.

Estas variables reconocen a los componentes en diferentes entornos en el contexto de educación virtual. La búsqueda de los componentes no solamente se debe limitar en la acción de dictar una clase y difundir información, sino también en todos los otros ambientes en los que puede someterse tanto el alumno como el docente en la herramienta. Es decir se deben considerar aspectos como los de evaluación, trabajo en equipo (el cual necesita de componentes de comunicación entre los estudiantes), Organización, entre otros.

Finalmente, teniendo claro el contexto seleccionado de educación virtual y los mecanismos claros de identificación de los componentes en las herramientas en dicho contexto, se puede continuar con la exploración de los componentes, la cual es presentada en la siguiente sección.

3.2. Exploración de componentes de sistemas interactivos en el contexto educativo.

Una vez seleccionado el contexto de educación, luego son exploradas herramientas web enfocadas en educación y comunicación (estas últimas utilizadas para interacción entre el docente y los estudiantes), con el fin de identificar componentes específicos de este tipo de sistemas interactivos. A continuación es presentado el listado de las herramientas exploradas y los componentes más representativos que maneja cada una.

Tabla 5. Herramientas para la educación en línea.

Herramienta	Referencia	Descripción	Contenido encontrado
Com8s		Establece una colaboración muy estrecha y fácil entre alumnos y docentes. Además trata de una red de contactos pero con numerosos aplicativos integrados, que nos permiten establecer diferentes espacios colaborativos con diferentes permisos	<ul style="list-style-type: none"> • Almacenamiento de archivos personales • Agenda • Chat • Foro • Referencias • Desarrollo de proyectos cooperativos
Schoology	(Besana, 2012)	Plataforma de educación virtual en línea, en donde instructores pueden crear cursos para la comunicación con sus estudiantes y compartir contenido en él. Los estudiantes acceden a sus perfiles y a un curso específico a través de un código.	<ul style="list-style-type: none"> • Agregar Actividades • Crear calendario • Consultar perfiles de estudiantes • Evaluación en línea • Foros de discusión
Udemy	("Udemy Online Courses - Learn Anything, On Your Schedule," 2010)	Udemy es un mercado global para el aprendizaje y la enseñanza en línea donde los estudiantes están dominando las nuevas habilidades y logran sus objetivos mediante el aprendizaje de una amplia biblioteca de más de 45.000 cursos impartidos por instructores expertos.	<ul style="list-style-type: none"> • Reproductor de video • Seguimiento a los cursos gráficamente • Quiz interactivo • Evaluaciones en línea • Consultas a Instructores
Coursera	(Levin, 2012)	Es una plataforma de educación virtual nacida en octubre de 2011 y desarrollada por académicos de la Universidad de Stanford con el fin de brindar oferta de educación masiva a la población con cursos en diferentes idiomas.	<ul style="list-style-type: none"> • Reproductor de video • Foro de discusión • Evaluación a estudiantes

EdX	(University, 2012)	Es una plataforma de cursos en línea masivos y abiertos (MOOC) fundada por el Instituto Tecnológico de Massachusetts y la Universidad de Harvard en mayo de 2012 para hospedar cursos en línea de nivel universitarios de una amplio rango de disciplinas para todo el mundo sin costos para propiciar la investigación y el aprendizaje	<ul style="list-style-type: none"> • Reproductor de video • Quiz interactivo • Foro de discusión • Evaluación a estudiantes
Khan Academy	(Khan, 2007)	Khan Academy es una plataforma que ofrece ejercicios de práctica, videos instructivos y un panel de aprendizaje personalizado que permite a los alumnos aprender a su propio ritmo, dentro y fuera del salón de clases. Se abordan las matemáticas, la ciencia, la programación de computadoras, la historia, la historia del arte, la economía y más.	<ul style="list-style-type: none"> • Reproductor de video • Pizarra virtual • Quiz interactivo • Evaluación online • Panel de preguntas • Comentarios en cada sección
Udacity	(Sebastian Thrun, 2012)	Organización educativa que permite cursos online masivos y abiertos que cuenta con contenido educativo y un entorno de aprendizaje comunitario y colaborativo.	<ul style="list-style-type: none"> • Reproductor de video • Editor de código online • Quiz interactivo • Administración de archivos.
UniMooc	("UniMOOC,")	UniMooc es una plataforma de formación a través de cursos masivos online, fomenta la innovación, es de habla hispana y lo conforman alrededor de 350.000 estudiantes, nació en el Instituto de Economía Internacional de la Universidad de Alicante gracias al apoyo de	<ul style="list-style-type: none"> • Reproductor de video • Archivos compartidos para lecciones o tareas • Evaluación online

		instituciones de prestigio como Google, Banco Santander o la Escuela de Organización Industrial.	
Open2study	((OUA), 2013)	Esta es una iniciativa de educación eficiente que ofrece cursos en línea cortos y eficientes en muchas variedades temáticas. Cuentan con un equipo de facilitadores de aprendizaje social, que están con los estudiantes en los foros, así como en las redes sociales, para compartir las últimas investigaciones y opiniones y así ayudarlo a llevar su aprendizaje al siguiente nivel.	<ul style="list-style-type: none"> • Reproductor de video • Quiz Interactivo • Foro de colaboración
Open Learning	(Global, 2013)	Es una institución de tecnología educativa con ánimo de lucro establecida en Australia que ofrece una plataforma de aprendizaje social en línea que puede ofrecer cursos en línea masivos abiertos.	<ul style="list-style-type: none"> • Reproductor de video • Colaboraciones en grupo • Comentarios en clases

La mayoría de las herramientas exploradas e investigadas han logrado su impacto gracias a la calidad y eficiencia de sus contenidos, como puede observarse en la Tabla 5 hay algunos contenidos importantes en común entre las interfaces de usuario. Entre ellas el reproductor de video, como el método más eficiente para transmitir la información de formación para los estudiantes, los foros de colaboración, los mensajes y el chat como la forma en que los estudiantes discuten los temas aprendidos y aportan a los cursos. Finalmente en los temas de evaluación son encontrados los *Quiz* interactivos y los test en línea como los métodos más comunes para probar que los estudiantes adquirieron la información y así comprobar que así cumplieron con los objetivos del curso. Cabe resaltar que las herramientas anteriormente presentadas han logrado convertirse en las más comunes debido al creciente número de estudiantes que son registrados. Sin embargo hay otros desarrollos web y móvil centrados en la educación virtual que también fueron explorados. A continuación son presentados algunos de ellos:

- Futurelearn
- Canvas network
- Miriadax
- Skilledup
- Telescopio
- TED
- Versal
- Stanford
- Acamica
- Alisona
- Apna course
- Aquent Gimnasio

- Initiative
- FX Academia
- Instreamia
- Iversity
- MRUniversity
- OpenHP

Con esta revisión, la tarea de seleccionar los componentes para este trabajo de grado tendrá fundamentos más sólidos. Trata de recoger los componentes que son considerados esenciales en cada aplicación o herramienta pensada para la educación virtual y además estudiar sus características principales tanto a nivel de vista y elementos gráficos como a nivel de funcionalidad.

3.3. Componentes seleccionados

En esta sección son presentados los componentes seleccionados a partir de la revisión realizada. Debido a que la cantidad de herramientas revisadas fue un número considerable son seleccionados los componentes más representativos de cada plataforma. Esta sección expone un listado de tablas donde es detallado cada componente seleccionado para su implementación. Adicionalmente el anexo 1, muestra los componentes que hicieron parte de esta investigación pero que no fueron considerados para su implementación.

A continuación se presentan los componentes seleccionados:

- Reproductor de video
- Visualizador de documentos
- Chat
- Quiz interactivo
- Tareas virtuales
- Tabla de contenido
- Recursos relacionados

De la lista anterior, a continuación son detallados dos componentes que aportan a una interfaz más interactiva para el estudiante:

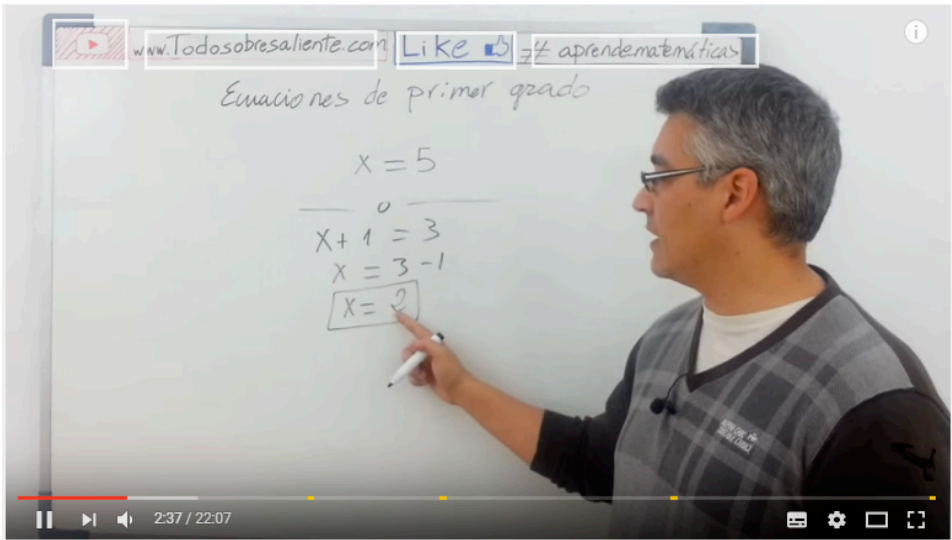
- **Recursos Relacionados:** son enlaces clasificados según el formato, que proporcionan información adicional a los estudiantes respecto al tema que estén tratando. Los estudiantes al acceder a estos enlaces son redirigidos a otros recursos en la web escogidos previamente por los docentes como un complemento de su educación. Este componente puede complementar cualquier otro componente que esté pensado para la difusión de contenido formativo o que esté utilizando alguna estrategia de enseñanza de un tema específico.
- **Tabla de Contenido:** muestra en pantalla la organización secuencial de la temática que el docente haya determinado o el seguimiento de una clase virtual clasificados a través de temas y subtemas. En algunas herramientas si son distinguidos este tipo de componentes para guiarse en las clases pero la propuesta de este trabajo es que no solamente muestre la información como el nombre de tema y subtema. Sino que sea totalmente interactivo con el resto de componentes. Por ejemplo, supongamos que un estudiante está recibiendo una clase en la cual el profesor ha dispuesto un reproductor de video y un visualizador de documentos para transmitir su conocimiento. Si el estudiante en el componente de contenido presiona clic al título del tema o del subtema, el reproductor de video ajustará su video al tiempo exacto donde comience ese tema o

subtema e igualmente el visualizador lo hará mostrando la página o diapositiva en donde puede visualizarse dicho tema o subtema.

La presentación de los componentes se hará a través de una tabla en donde se consideraron los siguientes factores:

- Imagen: descripción gráfica del componente
- Título: nombre formal del componente
- Descripción: definición y contextualización del componente
- Atributos: atributos propios de los componentes.
- Funcionalidades: todo tipo de acción que pueda hacer el usuario usando el componente.
- Entradas: todo tipo de datos que utilice el componente identificados como indispensables para probar su funcionamiento.
- Configuraciones: características configurables que describen gráficamente al componente identificados como modificables por el desarrollador en la creación del componente través de la herramienta.
- Observaciones: observaciones varias a tener en cuenta al implementar el componente.

Tabla 6. Presentación de componente Reproductor de video

	
REPRODUCTOR DE VIDEO	
Descripción	Componente que permite la reproducción de un contenido del curso en formato de video
Atributos	Tamaño de componente Volumen predeterminado Calidad predeterminada

	Estilo de componentes internos
Funcionalidades	Reproducir video Pausar video Retroceder Avanzar Elegir momento de reproducción Subir Volumen Bajar Volumen Transcripción de texto Cambiar subtítulos
Acciones de propiedades	Ajustar a pantalla completa Editar velocidad de reproducción Ajustar calidad de video Activar subtítulos
Atributos personalizados	Tamaño Color Activar/Desactivar opciones del menú desplegable
Entradas	Metadatos del video (nombre, ruta, tipo, otra calidad y recursos relacionados)
Observaciones	Crear una carpeta por cada video y la carpeta debe tener los archivos de metadatos y los archivos de subtítulos Archivos de sincronización. (Nombre del tema, numero de página y tiempo del video)

Tabla 7. Presentación de componente Visualizador de documentos.

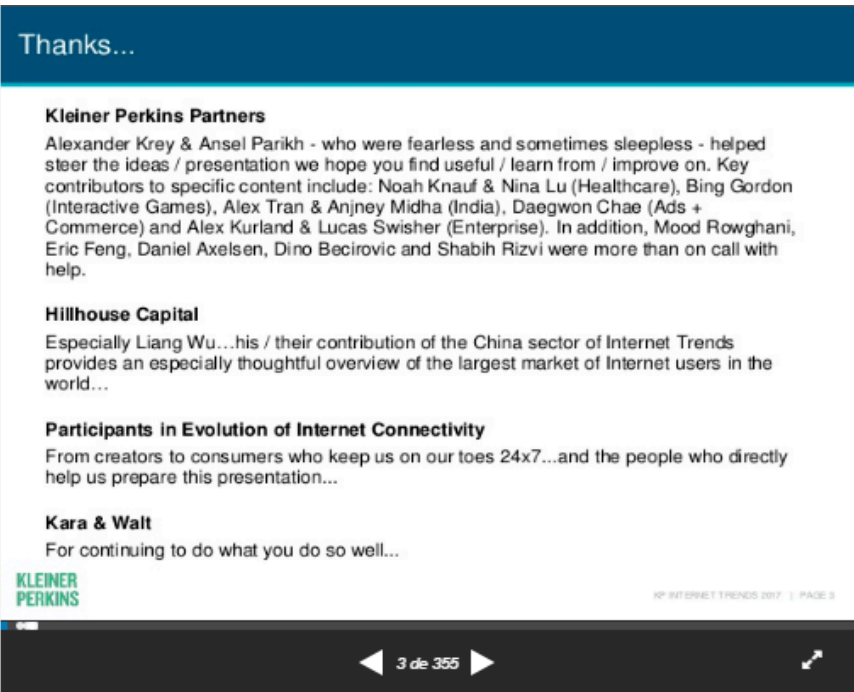
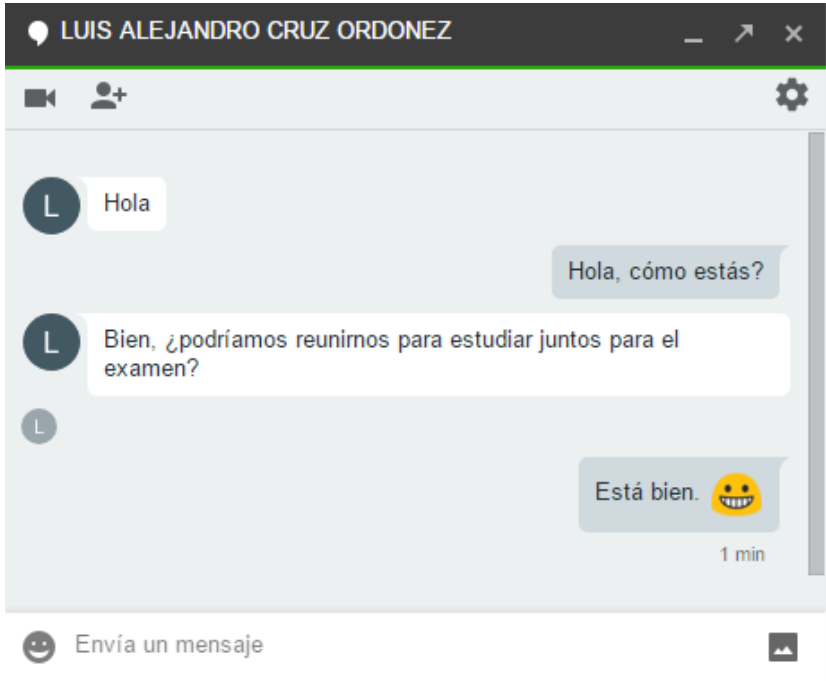
	
VISUALIZADOR DE DOCUMENTOS	
Descripción	.
Atributos	<ul style="list-style-type: none"> Tamaño Orientación Ruta del Archivo Color de botones
Funcionalidades	<ul style="list-style-type: none"> Ir a página siguiente Ir a página anterior Mostrar página Actual
Entradas	<ul style="list-style-type: none"> Seleccionar el archivo
Configuraciones	<ul style="list-style-type: none"> Definir tamaño Definir orientación Color de botones

Tabla 8. Presentación de componente Chat

	
CHAT	
Descripción	Comunicación bidireccional a través de mensajería instantánea entre 2 o más usuarios conectados entre sí por una red. El tipo de información que puede compartirse ha evolucionado según la herramienta lo permita.
Atributos	<ul style="list-style-type: none"> Tamaño de componente Número de caracteres máximos en un mensaje Tipos de archivos aceptables para enviar Nombre de usuario Estado de usuario Foto de perfil Archivar conversación Tipo de aviso de notificación (Audio/ Vibración/ Silencio) Mensaje enviado/ mensaje No enviado Mensaje entregado/ mensaje No entregado Mensaje leído/ mensaje No leído Fondo de la conversación predeterminado (Imagen) Tipo de letra Color de texto Tipo de pincel para escribir

	Color de pincel para escribir
Funcionalidades	<p>Enviar texto</p> <p>Enviar vínculo</p> <p>Enviar emoticones</p> <p>Enviar <i>Gift</i></p> <p>Enviar notas de voz</p> <p>Compartir Contactos</p> <p>Enviar ubicación</p> <p>Realizar llamada (cambio de componente)</p> <p>Adjuntar y enviar imagen guardada</p> <p>Adjuntar y enviar video guardado</p> <p>Adjuntar y enviar documentos</p> <p>Adjuntar y enviar audio</p> <p>Enviar pincelazos desde pizarra</p> <p>Añadir Participante a la conversación</p> <p>Sincronizar aplicaciones externas para compartir información más fácil.</p>
Acciones de propiedades	<p>Eliminar mensajes</p> <p>Eliminar historial</p> <p>Activar corrector ortográfico</p> <p>Cambiar lenguaje de escritura</p> <p>Activar/Desactivar notificaciones</p> <p>Configuración de privacidad (Los mensajes los pueden leer algunos usuarios en específico o todos)</p> <p>Cambiar tipo de aviso de notificación (Audio/ Vibración/ Silencio)</p> <p>Seleccionar y modificar foto de perfil</p> <p>Archivar chat</p> <p>Bloquear Contacto</p> <p>Marcar conversación como no leída</p> <p>Elegir fondo de la conversación (Imagen)</p> <p>Ocultar conversación</p> <p>Editar mensaje de la conversación (Borrar, guardar o reenviar)</p>

Tabla 9. Presentación de componente Quiz.

QUIZ INTERACTIVO	
Descripción	Componente que permite la gestión de un examen rápido mediante la selección de elementos o ingreso de texto.
Atributos	Tamaño Colores Fuente
Funcionalidades	Seleccionar opciones de pregunta Validar respuesta Cambiar respuesta Solicitar respuesta Reportar mala pregunta Solicitar ayuda Compartir resultado en redes sociales
Propiedades de acciones	Ajustar auto calificación Seleccionar nivel de dificultad

Símbolo y nombre de los elementos químicos

Por Jesús Marcos Segura

En este quiz puedes revisar las convenciones recomendadas para los elementos químicos. correcta

1. Las valencias de los elementos Mg, Ca y Al son, respectivamente:

- 3, 2 y 2
- 2, 3, 2
- 2, 3, 3
- 2, 2 y 3

2. ¿Cuáles de los siguientes elementos poseen valencia o estado de oxidación variable? S, P,

- Solamente Cu y Cl
- Solamente S, P, Cl y Fe
- Todos los señalados poseen valencia variable
- Cu, Fe, Mn y P

Tabla 10. Presentación de componente Tareas.

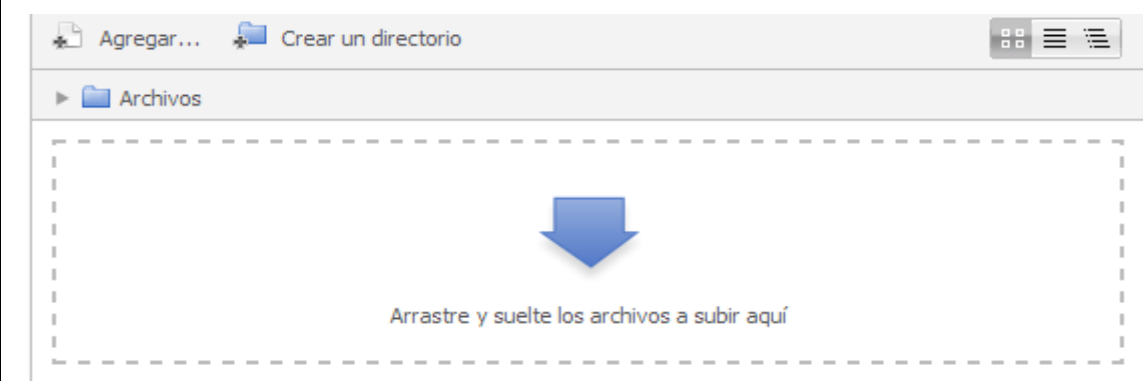
	
TAREAS	
Descripción	Componente pensado para gestionar las tareas que los educadores dejan a los estudiantes en ambientes virtuales de aprendizaje.
Atributos	<ul style="list-style-type: none"> Título Descripción Tipo de tarea Calificación Tamaño del archivo Fecha de entrega
Funcionalidades	<ul style="list-style-type: none"> Subir archivo Agregar comentario Especificar retroalimentación que el docente da (cuadro texto) Asignar calificación a la tarea
Entradas	Metadatos XML
Configuraciones	<ul style="list-style-type: none"> Color, tipo y tamaño de letra Layouts (temas)

Tabla 11. Presentación de componente Recursos relacionados

<ul style="list-style-type: none"> • Ejemplos: https://spark.apache.org/examples.html • Cassandra - Spark: https://github.com/datastax/spark-cassandra-connector • Exploracion: http://lightning-viz.org/ • Deep learning : https://github.com/deeplearning4j/deeplearning4j 	
RECURSOS RELACIONADOS	
Descripción	Los recursos relacionados le permiten al estudiante acceder por medio de enlaces a otra fuente de información virtual que contribuya a la enseñanza que esté recibiendo en alguna clase.
Atributos	Atributos por cada ítem: Título Ruta del recurso Tipo de recurso
Funcionalidades	Acceder a recurso por medio del enlace
Entradas	Metadatos: Archivos de tipo XML para construir los enlaces que el educador haya dispuesto.
Configuraciones	Color y tipo de letra Ruta de metadatos XML

Tabla 12. Presentación de componente Contenido

PROGRESS SECTION

- Lecture 0: Before You Get Started
- Lecture 1: What We Now Know
- Lecture 1.5A: Business Models and Customer Development
- Lecture 1.5B: Business Models and Customer Development
- Lecture 2: Value Proposition
- Lecture 3: Customer Segments
- Lecture 4: Channels
- **Lecture 5: Customer Relationships**
- Lecture 6: Revenue Model
- Lecture 7: Partners
- Lecture 8: Resources, Activities & Costs
- Secret Notes for Instructors/Coaches

TABLA DE CONTENIDO	
Descripción	Proporciona la información del contenido o índice de la clase que el educador lleva a clase, está conformado por enlaces que permiten mayor interacción del usuario con otros componentes. Más específicamente con el componente de video y de presentación de documentos.
Atributos	Atributos por cada ítem: Título Páginas de referencia Tiempo de referencia
Funcionalidades	Acceder a diferentes temáticas
Entradas	Metadatos XML (título, tiempo y pagina)
Configuraciones	Color y tipo de letra y ruta Metadatos XML
Observaciones	El archivo de metadatos es creado aparte y en cada carpeta ya sea de video o de presentación.

Capítulo 4

Sintaxis abstracta y concreta del editor DSL para el diseño de la interfaz de usuario

Este capítulo recoge todos los elementos involucrados para la creación de la sintaxis abstracta y concreta del editor para el diseño de la interfaz de usuario. Empleando estos dos elementos pretende hacer la construcción de la herramienta para la creación de interfaces de usuario dotada con los componentes seleccionados en el capítulo anterior. Dicha herramienta será construida siguiendo el modelo de proceso presentado en la sección 2.2.2.5, el cual propone una aproximación metodológica para la creación de editores DSL.

Inicialmente es presentada la selección de la tecnología a partir de las aproximaciones metodológicas para la creación de editores para lenguajes específicos de dominio (DSL) presentadas en el estado del arte, seguidamente es realizada la creación la sintaxis abstracta y finalmente la sintaxis concreta.

4.1. Elección de tecnología

En el marco de las tecnologías existentes en el desarrollo de editores DSL expuestas en el estado del arte, se debe elegir una tecnología específica para este trabajo de grado. Todas las tecnologías cumplen con el objetivo de mostrar una alternativa en la construcción de editores de modelado a través de un proceso específico, pero son detectadas algunas desventajas que sustenta la elección de una tecnología específica. Dichas desventajas identificadas son presentadas a continuación:

- No son herramientas de código abierto.
- Para que funcionen los editores generados necesitan de *MetaEdit+* y Visual Studio en el caso de *Metacase* y *Microsoft* respectivamente.
- Dificulta la portabilidad de proyectos debido a que usa un lenguaje de metamodelado propio que no cumple con el estándar.

A continuación son presentados algunos criterios fundamentales que son buscados en las tecnologías dedicadas en la construcción de dichas herramientas de modelado y que llevan a la decisión de elegir la propuesta Eclipse como la tecnología de desarrollo de este trabajo:

- Fundamentarse en una herramienta de código abierto.
- La generación de código debe obtenerse a partir de la especificación de un modelo.

- Establecer soporte para interoperabilidad con otras herramientas a través de versiones simplificadas de lenguaje de metamodelado MOF.
- Integración de los *plugins* de especificación de la sintaxis concreta con EMF facilitando la construcción de los editores gráficos.
- Existencia de gran disponibilidad en documentación, asistentes y tutoriales sobre EMF y GMF.

4.2. Sintaxis abstracta

4.2.1. Creación de Metamodelo a través de ECORE

Esta subsección llevará a cabo el proceso de creación del metamodelo de la herramienta. Para la creación de este metamodelo es utilizado Ecore en Eclipse. Para esto, es necesario instalar el *plugin* de EMF (Eclipse Modeling Framework), este *plugin* provee básicamente dos herramientas para construir un modelo basado en Ecore: Una es el Ecore Model que es un editor manual de tipo árbol de navegación para la creación del modelo y la otra es el Ecore Diagram, un editor gráfico para la creación de diagramas de clases tipo UML. Las dos formas mencionadas anteriormente son válidas para crear el diagrama basado en Ecore. Para este trabajo es seleccionado la forma del Ecore Diagram construyendo el metamodelo como un editor que obtiene un diagrama de clases UML. El siguiente párrafo muestra los pasos como guía para la construcción del metamodelo propuesto:

Primeramente es abierta la herramienta Eclipse, es creado un proyecto EMF vacío cliqueando en “File”, luego “New” y luego “Other”. En el panel presentado, seleccionar la opción “Empty EMF Project” tal cual como lo muestra la Figura 9. Al continuar debe asignársele un nombre al proyecto y después pulsar “Finish”.

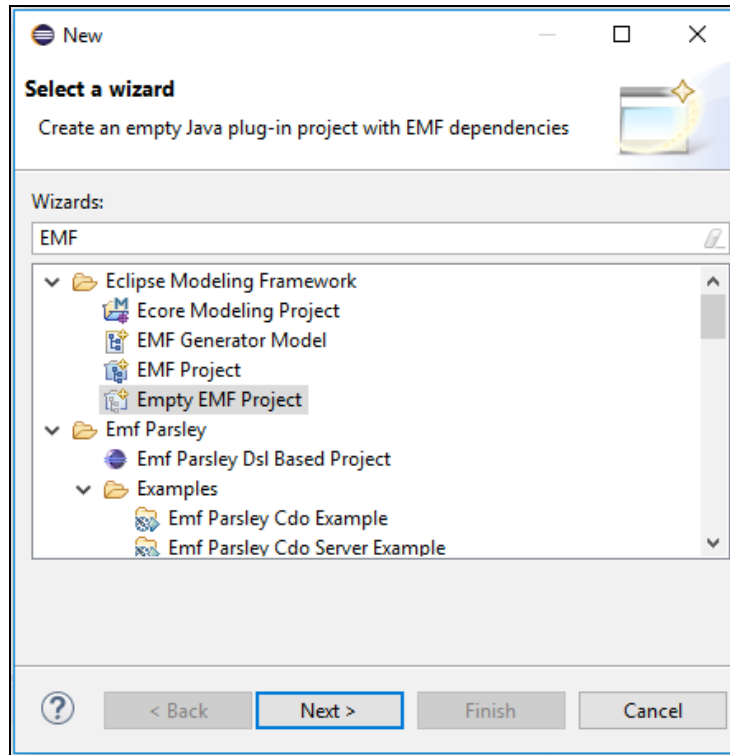


Figura 9. Creación de un proyecto EMF.

El proyecto EMF creado, está compuesto por algunos ficheros, entre ellos una carpeta denominada “model”, la cual contendrá el modelo Ecore que se va a construir. Para esto, una vez más realizamos clic en “File”, luego “New”, luego “Other” y en el panel presentado, seleccionar “Ecore Diagram” tal cual como lo indica la Figura 10. Después la carpeta “model” debe ser seleccionada como la ubicación del diagrama y finalmente al pulsar “Finish” es creado el diagrama Ecore.

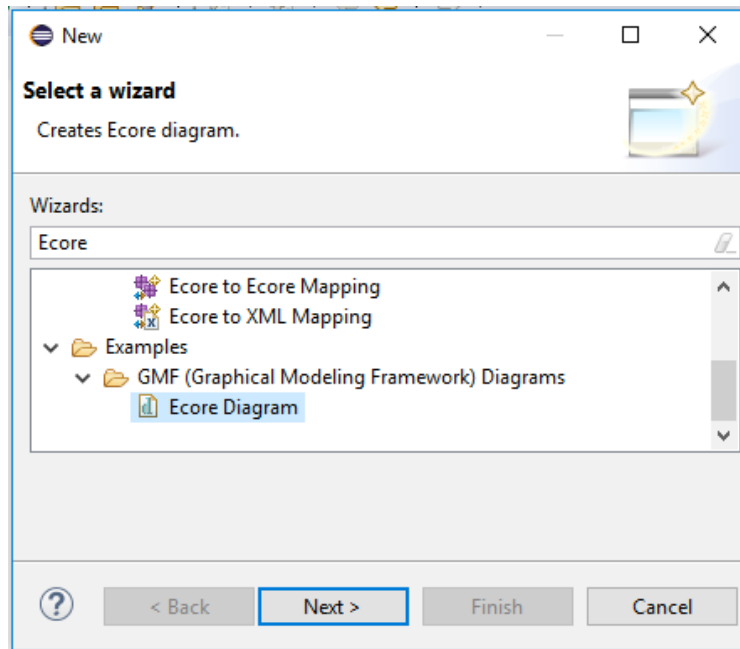


Figura 10. Creación de Diagrama Ecore

Una vez creado el “Ecore Diagram”, permite contar con un entorno de edición para la construcción de los modelos de tipo diagrama de clase con todas las herramientas a disposición. También cuenta con una paleta de herramientas tal como lo muestra la Figura 11, en donde son distinguidas las herramientas de *Eclass*, para crear nuevas clases, la herramienta *Epackage*, para crear nuevos paquetes, *EAnnotation*, para crear anotaciones y comentarios, la herramienta *EDataType* con la cual son creados nuevos tipos de datos, el *EAttribute* que permite agregar atributos a las clases, *EOperation* que permite añadir operaciones a las clases y por último, son distinguidas las herramientas de *Asociation*, *Agregation* y *Generalitazion* para establecer las relaciones de asociación, agregación y herencia respectivamente.

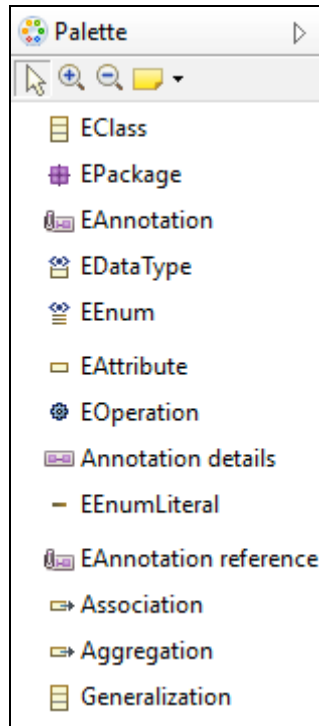


Figura 11. Paleta de herramientas EMF.

Finalmente, esta herramienta genera un fichero XML³ (MOF, 2007) que es una especificación para el intercambio de diagramas.

4.2.2. Descripción de elementos

Cada modelo implementado en la sintaxis abstracta es identificado con una clase de tipo UML la cual muestra el título asignado y los atributos del elemento seleccionados como modificables en la creación de dicho elemento. A continuación es presentado cada elemento con su respectivo título y sus respectivos atributos:

Tabla 13. Descripción de modelo del elemento "Player" (Reproductor de video).

Player	
Atributo	Tipo de dato
BackgroundColorBar	String
elementColorBar	String
progressColorBar	String
BackgroundColorMenu	String
fontColorMenu	String
borderColorMenu	String
BackgroundColorMenuHover	String

³ XML Metadata Interchange

fontColorMenuHover	String
BackgroundColorSubmenu	String
fontColorSubmenu	String
timelsEnabled	Boolean
speedIsEnabled	Boolean
qualityIsEnabled	Boolean
subtitlesEnabled	Boolean
backgroundColorTranscription	String
fontColorTranscription	String
fontColorTitleTranscription	String

Tabla 14. Descripción de modelo del elemento “Visualizador de componentes”.

Visualizador	
Atributo	Tipo de dato
NextColorButton	String
backColorButton	String
fontColorButon	String

Tabla 15. Descripción de modelo del elemento “Quiz”.

Quiz	
Atributo	Tipo de dato
fontFamily	String
BackgroundColorButton	String
BackgroundColorAd	String
BackgroundColor	String
fontColorTitleTranscription	String

Tabla 16. Descripción de modelo del elemento “Chat”.

Chat	
Atributo	Tipo de dato
fontColor	String
BackgroundColorButton	String
fontColorButton	String

BackgroundColor	String
-----------------	--------

Tabla 17. Descripción de modelo del elemento “Tareas”.

Tareas	
Atributo	Tipo de dato
fontColorTitle	String
fontColorSubtitle	String
fontFamily	String
BackgroundColorButton	String
BackgroundColorHead	String
BackgroundColorBody	String

Tabla 18. Descripción de modelo del elemento “Repositorio”.

Repositorio (Recursos relacionados)	
Atributo	Tipo de dato
fontColor	String
fontFamily	String

Existen algunos atributos comunes entre los elementos. Es por esto que es creado una clase con dichos atributos, de la cual los demás elementos los heredan tal cual como lo muestra la Figura 12. Los atributos identificados son: nombre, posición (X, Y), alto y ancho. Dichos atributos están relacionados con la organización en el aspecto visual en la interfaz y son sumados a los atributos independientes de cada elemento presentados anteriormente.

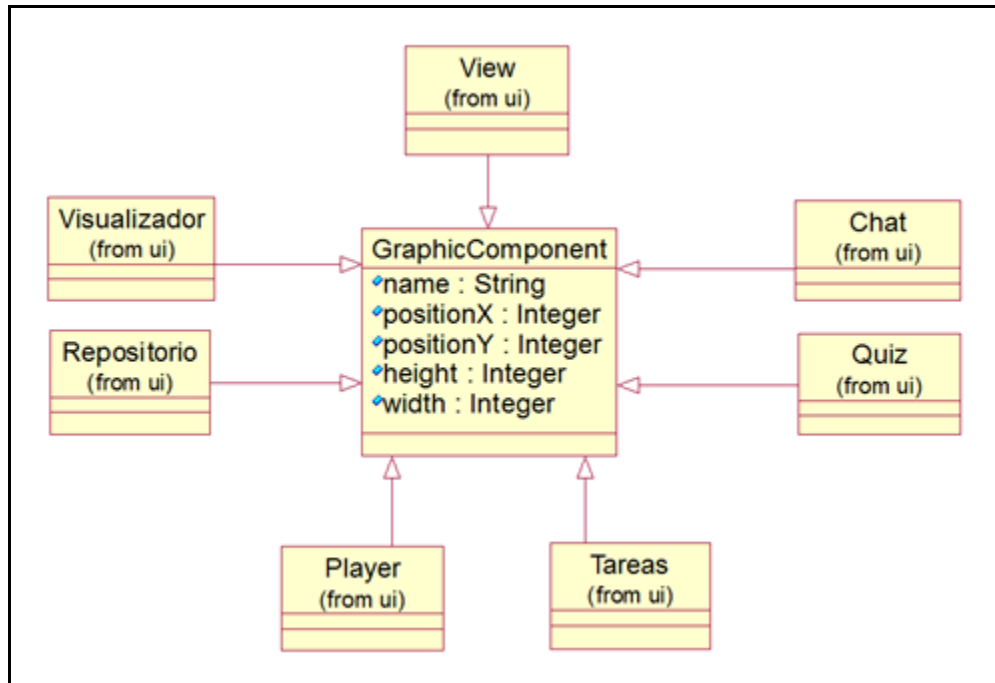


Figura 12. Herencia de atributos comunes de componentes.

4.2.3. Relaciones identificadas entre clases

Las interfaces de usuario que pretenden construirse con estos elementos están representadas en la sintaxis abstracta con una clase de un nivel superior denominada "View". Esta clase está compuesta por todos los elementos seleccionados y en términos del editor gráfico es el *Canvas* principal en donde serán arrastrados los elementos.

Teniendo en cuenta lo anterior, la relación identificada entre las clases es la composición en donde una clase "View" está compuesta por los elementos. Esta relación es de tipo 1: N (uno a muchos), en donde un "View" puede componerse del mismo elemento muchas veces. Esto indica por ejemplo, que podrían existir interfaces de usuario con dos visualizadores de documentos o más configurados de formas distintas o iguales.

Otra relación identificada es la Herencia, en donde las clases de los componentes heredan los atributos de una clase padre creada debido a que son atributos que se perciben en todos los componentes.

4.2.4. Presentación de modelo final

La Figura 13 presenta el modelo final construido, en él están representados todos los elementos con sus respectivos atributos y las relaciones existentes. Este modelo final es el presentado como la sintaxis abstracta para el desarrollo de la herramienta basada en modelos propuesta en este trabajo.

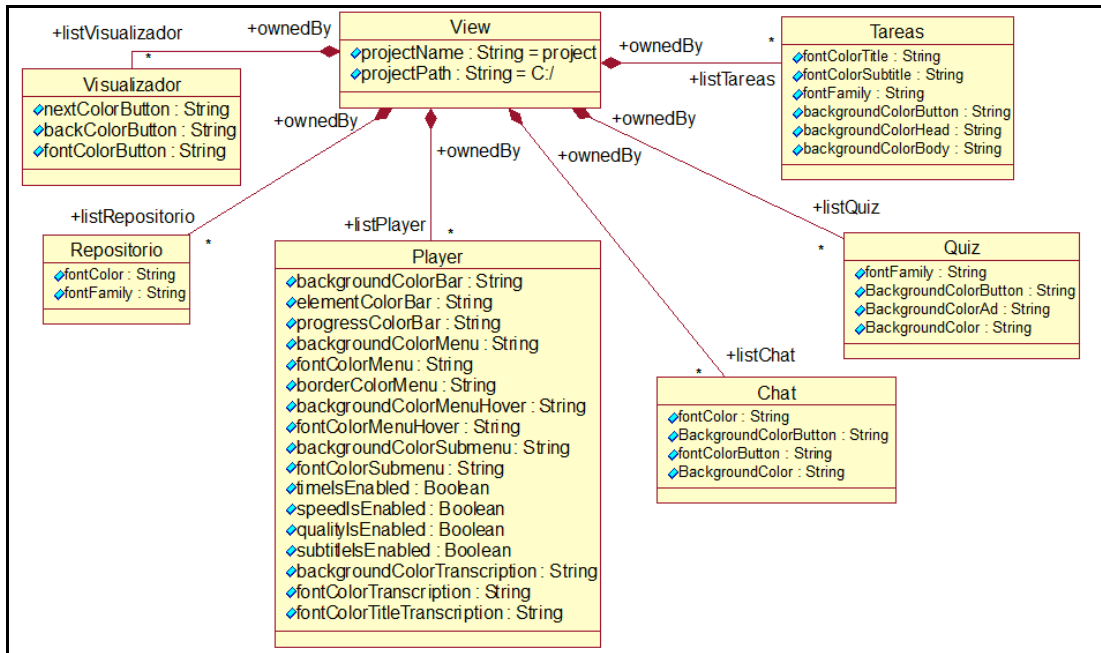


Figura 13. Metamodelo de componentes basados en distribución de video.

4.3. Sintaxis concreta

En esta sección es presentada la definición de la sintaxis concreta elaborada para aportar a la construcción de la herramienta basada en modelos. Como lo mostró el estado del arte, la sintaxis concreta recoge los elementos influyentes en la vista de los componentes en la interfaz de usuario en cualquiera de las fases. El metamodelo construido en la sección anterior está representado con un diagrama de clases y es perfectamente compatible con muchas notaciones gráficas. De dichas notaciones gráficas, debe definirse una en específico, la cual será la sintaxis concreta de la herramienta. La sintaxis concreta presenta el conjunto de símbolos gráficos utilizados para dibujar los diagramas y no es independiente de la sintaxis abstracta, pues a partir de los modelos de la sintaxis abstracta son construidas las representaciones gráficas de los mismos.

Según el modelo de aproximación metodológica presentado en la Figura 7 la definición de la sintaxis concreta plantea tres pasos que a nivel de implementación necesitan recursos visuales (*Muckups*) los cuales son presentados a continuación.

4.3.1. Bosquejos para el modelo de definición gráfica.

La Figura 14 presenta los bosquejos asignados a los componentes, en ella es posible distinguir de izquierda a derecha los siguientes componentes: reproductor de video, visualizador de documentos, quiz, chat, recursos relacionados y tareas. Estos bosquejos gráficos han sido elaborados con una herramienta de edición gráfica (Corel, 2017) y cumplen la misión de proporcionar al usuario una idea para identificar el componente a través de un dibujo simplificado. Así, el usuario podrá familiarizarse con cada componente de la interfaz de usuario al momento de construir la interfaz en el modelo de definición gráfica de la herramienta.

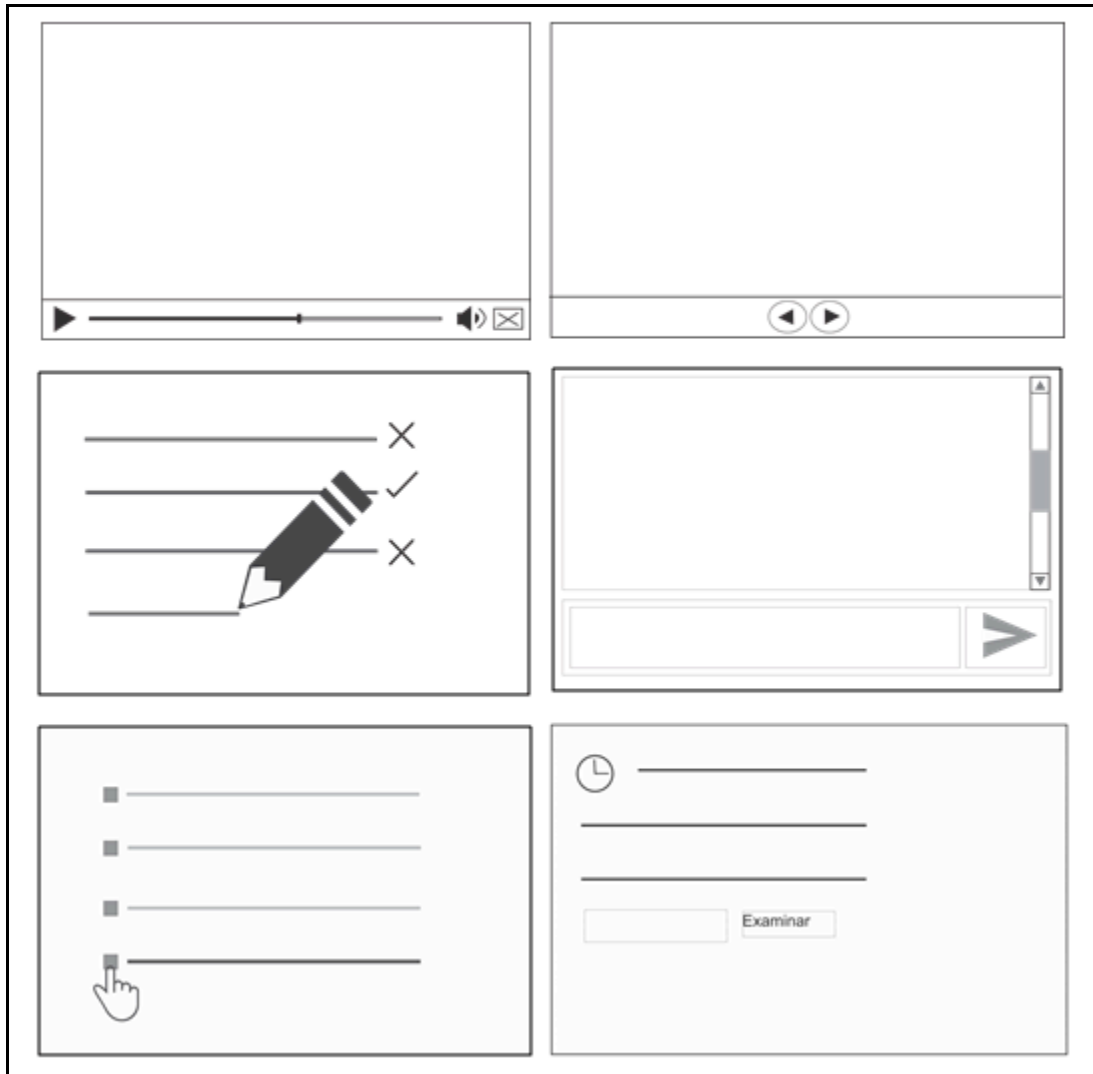









Figura 14. Bosquejos de modelo de definición gráfica de los componentes

4.3.2. Bosquejos para la definición de la herramienta.

En esta sección son presentados los bosquejos diseñados para la definición gráfica de la herramienta. Es decir, los iconos que serán utilizados para la paleta de herramientas del *framework* construido. Diseñando un icono para cada componente con base a las notaciones gráficas más utilizadas en demás herramientas con el fin de obtener una buena comunicación con el usuario

En la Tabla 19 son listados dichos íconos.

Tabla 19. Bosquejos para la definición de paleta de herramientas

Componente	Notación gráfica
Reproductor de video	
Visualizador de documentos	
Recursos relacionados (Repositorio)	
Tareas	
Quiz	
Chat	
Contenedor	

Capítulo 5

Implementación

Este capítulo presenta la implementación tanto de los componentes como la de la herramienta para modelarlos. Teniendo en cuenta lo realizado en los capítulos anteriores es realizada la implementación del desarrollo software, este es sin duda la sección más relevante de este trabajo de grado. El proceso de implementación consiste en desarrollar todos los componentes, además de modelarlos a través del *framework* que es construido. En esta sección son seleccionadas las tecnologías según el requerimiento de cada componente, además de la arquitectura software a utilizar y los respectivos diagramas de clase planteados.

El presente capítulo está dividido en dos fases generales: la primera consta del desarrollo de los componentes de interfaz de usuario con la tecnología seleccionada. Al obtener los prototipos finales sigue la segunda fase en la cual es presentada paso a paso la creación de la herramienta de modelado de principio a fin. A partir de esta herramienta es modelado cada componente accediendo a todos sus atributos respecto a la vista del mismo (tamaño, color, fuente, etc.).

5.1. Desarrollo elemental de componentes

La implementación de los componentes es llevada a cabo a partir de un desarrollo software que sigue una metodología definida. Esta sección presenta los criterios para seleccionar la metodología software a utilizar y las características de esta.

La metodología de desarrollo es la base fundamental para obtener el producto final con una alta calidad. Es por eso que antes de comenzar a desarrollar es planteado el hecho de optar por una metodología ágil (Grupo, 2003), que a diferencia de las metodologías tradicionales impulsan a los desarrolladores a hacer su trabajo de una manera más profesional y eficiente. La Tabla 20 muestra una comparación entre las metodologías ágiles y las tradicionales con el objetivo de evaluarlas en diferentes aspectos y de esa manera sustentar el hecho de optar por una metodología ágil.

Tabla 20. Comparación entre metodologías ágiles y tradicionales.

Metodologías Ágiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Metodología que proyecta cierta resistencia a los cambios
Impuestas internamente (por el equipo)	Impuestas externamente

Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (menos de 10 integrantes) y trabajando en un mismo sitio	Grupos grandes y posiblemente distribuidos
Pocos artefactos	Más artefactos
Pocos roles	Más roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y es expresada mediante modelos

A partir de los beneficios que son identificados al utilizar una metodología ágil y según las metodologías más comunes para desarrollos con este tipo de enfoque, es propuesta la metodología SCRUM, la cual es presentada y aplicada a este trabajo en la siguiente sección.

5.1.1. Definición de la metodología SCRUM

Dentro de las metodologías de desarrollo ágiles, existe la metodología SCRUM, una metodología que comenzó a tomar fuerza por los desarrolladores emprendedores gracias a su visión ligera y fácil de entender. SCRUM no es un proceso o una técnica para construir productos; en lugar de eso, es un marco de trabajo dentro del cual es posible emplear varias técnicas y procesos.

Esta metodología es basada en la teoría de control de procesos empírica o empirismo. El empirismo asegura que el conocimiento procede de la experiencia y de tomar decisiones basándose en lo que es conocido. SCRUM emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control del riesgo.

El desarrollo de los componentes seleccionados como cualquier otro desarrollo software debe partir de una previa documentación de planeación establecida, que a través de un proceso, haya un seguimiento detallado desde el requerimiento inicial hasta el producto final. Con base a la metodología establecida anteriormente para los desarrollos que son llevados a cabo en este trabajo de grado, es planteado construirse el proceso a seguir para la implementación de estos componentes. A continuación son presentados los pasos a llevar a cabo teniendo en cuenta los criterios software a partir de la metodología aquí propuesta:

1. Definición del requerimiento.

La implementación de cada componente debe interpretarse como el producto requerido por parte de un cliente a un desarrollador, por eso, debe establecerse un requerimiento concreto enfocado en las principales características del componente y sus funcionalidades. El requerimiento establece los límites de trabajo del desarrollador, es por eso que terminado el desarrollo, debe evaluarse el componente conforme a los criterios establecidos en su requerimiento.

2. Elección de tecnologías y herramientas.

Las tecnologías utilizadas son clasificadas según el modelo vista controlador, esta sección presenta todas las tecnologías usadas para el desarrollo de los componentes:

- **HTML5.**

Es seleccionada la tecnología HTML con el fin de lograr el mejor alcance web en los componentes. Esta selección es sustentada a través de la documentación revisada (Musciano & Kennedy, 1996) y la comparación con otros tipos de lenguaje. A continuación son presentados los criterios de selección de dicha tecnología:

- Lograr que la información, y la forma de presentarla estén lo más separadas posible.
- Resumir, simplificar y hacer más sencillo el código utilizado.
- Un lenguaje que haga las páginas compatibles con todos los navegadores web, incluyendo los de los teléfonos móviles y otros dispositivos modernos usados en la actualidad para navegar en Internet.
- Eliminar restricciones que hagan el código más popular y asequible.

Una vez claras las razones del porqué del lenguaje HTML, es especificado también a continuación, la elección de la versión HTML5 (Pilgrim, 2010), que aunque parezca obvio usar la última versión, ésta trae algunos beneficios claros en cuanto al presente desarrollo:

- Al ser el código más sencillo y simplificado, cargan más rápido las páginas en el navegador.
- Las páginas y los elementos que contienen, son perfectamente vistos en todos los navegadores. La gran mayoría de los navegadores de los teléfonos Smartphone y las tabletas, son compatibles con HTML5.
- Los *plugins*, *widgets* y botones funcionan excelente, con más opciones que los clásicos en XHTML o que los *iframes*.
- Es posible insertar directamente videos en las páginas sin tener que acudir a los *iframes* o usar la etiqueta *object*.

- **JAVASCRIPT.**

La funcionalidad del componente es otro de los pilares fundamentales de este desarrollo, Javascript es un lenguaje de scripting precisamente enfocado en el entorno web que gracias a su dinamismo con eventos y la relación con formularios cumple los requisitos para todas las funciones solicitadas por estos componentes.

Para ser más eficientes en algunas funciones específicas, también se deciden usar las siguientes librerías de Javascript: JQuery y AJAX

- **PHP.**

A través de PHP es implementada toda la programación por parte del servidor y organización de la navegación entre páginas. PHP le permite al desarrollador crear una comunicación con la base de datos a través de consultas. Esta tecnología será utilizada en los componentes en los que es necesario almacenar datos y donde es identificada una estructura de aplicación web.

- **MySQL.**

Para los componentes que necesitan la gestión de bases de datos, es seleccionada la tecnología MySQL que es la más utilizada para este tipo de desarrollo por sus bases de datos relacionales, multihilo y multiusuario.

- **CSS3.**

El último pilar fundamental es la vista, el aspecto visual de la interfaz está dirigido por CSS3 y más específicamente es elegido un framework denominado Materializecss. Muchos de los factores identificados como modificables en el aspecto de la vista, serán controlados a través de CSS3 ya sea en archivos propios de lenguaje tipo .css o a través de funciones Javascript.

A continuación son presentadas las herramientas que fueron seleccionadas para llevar a cabo la programación de los componentes según las tecnologías anteriormente presentadas:

- **Visual Code Studio.**

Visual Code Studio es el editor de código seleccionado para la programación, es caracterizado por ser poderoso y ligero. Este editor soporta gran número de lenguajes de programación, tiene integrado Git para mayor productividad de trabajo y lo más importante es Open Source.

- **WAMPSEVER.**

El servidor local que ha sido seleccionado es Wampserver debido a que ahí es posible pre visualizar los componentes como un contenido web. Esta herramienta además incluye un administrador de base de datos (phpMyAdmin) para gestionar tablas, hacer consultas y todo tipo de funciones relacionadas con bases de datos.

3. Modelo de arquitectura software

Los modelos de arquitectura permiten visualizar la organización software de los componentes. Para este trabajo son identificados 2 modelos: el primero abarca las tecnologías que no necesitan programación por parte del servidor ni almacenamiento de información, y el segundo abarca los componentes un poco más robustos que si incluyen estas características. Estos modelos pueden apreciarse en la Figura 15 y Figura 16 respectivamente.

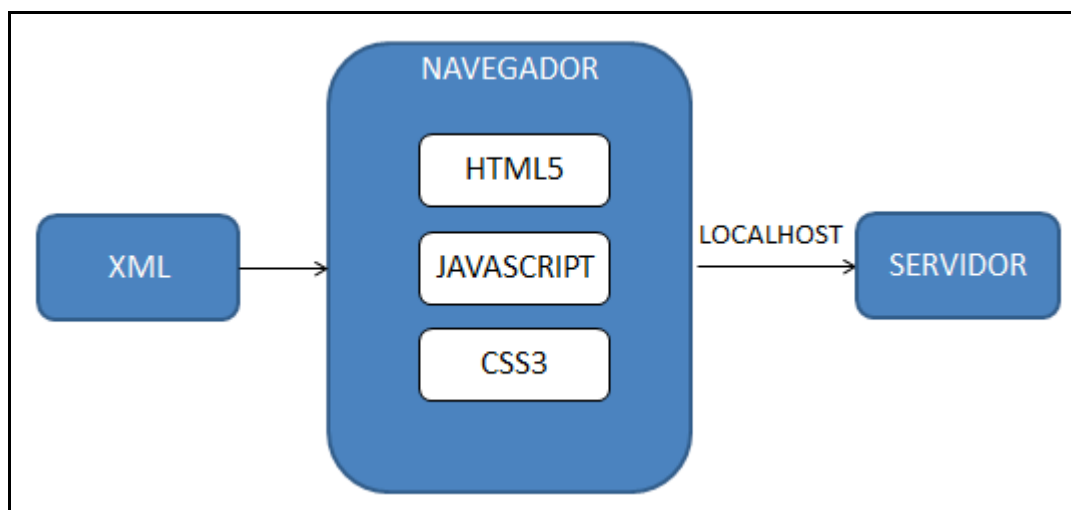


Figura 15. Modelo de arquitectura software 1.

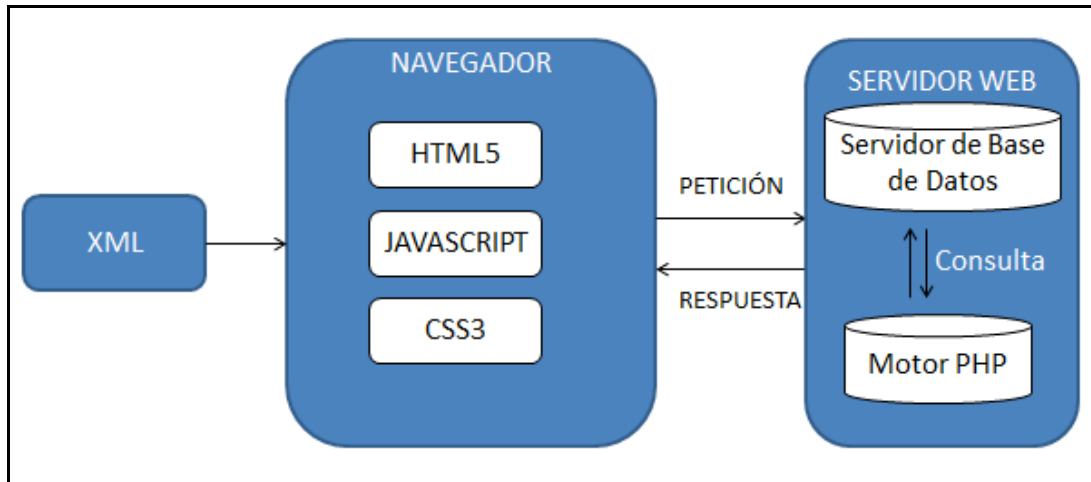


Figura 16. Modelo de arquitectura software 2.

4. Diagrama de clases.

Los diagramas de clases permiten representar a cada componente como una estructura definida para su desarrollo software. Cada componente es representado con una clase la cual esta compuesta de atributos y métodos. Los atributos muestran las características visibles del componente y los métodos muestran todas aquellas acciones que el usuario puede lograr utilizando dicho componente. A continuación son presentados los diagramas de clase mencionados.

5.1.2. Aplicación de la Metodología SCRUM

Teniendo en cuenta cada uno de los aspectos identificados en la metodología seleccionada, son identificados dichos aspectos en cada componente. A continuación en las siguientes secciones son presentados los aspectos en mención.

Reproductor de video

1. Requerimiento.

Desarrollar un reproductor de video que recoja las principales funcionalidades de los reproductores existentes reconocidos y que además permita personalizar aspectos en cuanto a sus estilos.

El reproductor de video debe proporcionar un archivo de tipo XML que permita configurar los aspectos identificados como modificables en el componente.

2. Tecnologías seleccionadas.

- HTML.
- Javascript.
- CSS.

3. Herramientas seleccionadas.

- Visual Studio Code.
- Google Chrome.

4. Modelo de arquitectura software.

Debido a que el presente componente no necesita el uso de bases de datos en su desarrollo ni tampoco desarrollo por parte del servidor, el modelo de arquitectura software seleccionado es el presentado en la Figura 15.

5. Diagrama de clases.

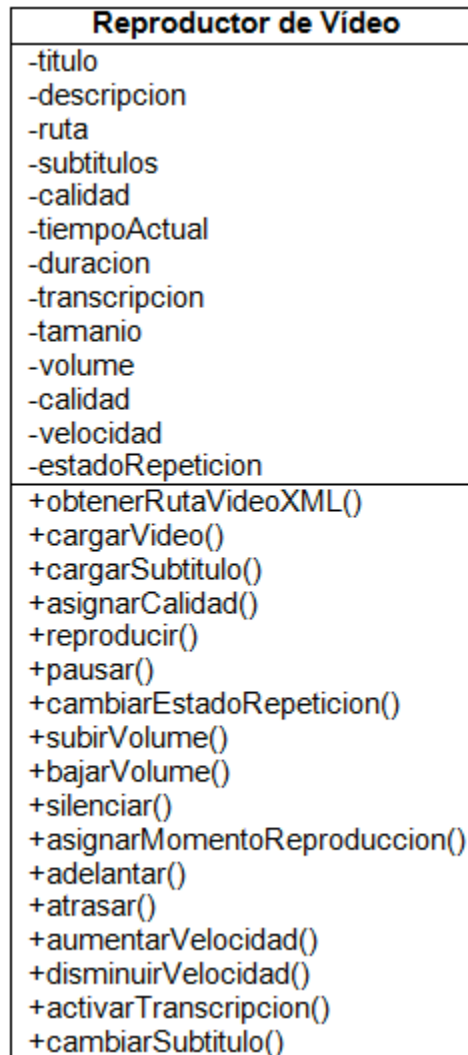


Figura 17. Diagrama de clase Reproductor de video

Visualizador de documentos

1. Requerimiento.

Desarrollar un componente de presentación de documentos donde el usuario pueda visualizar información a modo de diapositivas con las funcionalidades de navegación de páginas a través de botones de “Siguiente” y “Atrás”. El componente debe proporcionar un archivo de tipo XML que permita configurar los aspectos identificados como modificables en el componente.

2. Tecnologías seleccionadas.

- HTML.

- Javascript.
 - CSS.
3. Herramientas seleccionadas.
 - Visual Studio Code.
 - Google Chrome.
 4. Modelo de arquitectura software.

Debido a que el presente componente no necesita el uso de bases de datos en su desarrollo ni tampoco desarrollo por parte del servidor, el modelo de arquitectura software seleccionado es el presentado en la Figura 15.

5. Diagrama de clases.

Visualizador de Documentos
-ruta -escala -canvas -numeroPagina -numeroPaginas
+mostrarPagina(numeroPagina) +irPaginaSiguiente() +irPaginaAnterior() +obtenerDocumento()

Figura 18. Diagrama de clase Visualizador de documentos

Quiz interactivo

1. Requerimiento.

Desarrollar un componente para que en un entorno de educación virtual, los profesores puedan proporcionar a los estudiantes un Quiz interactivo como uno de los métodos de evaluación del aprendizaje. El cuestionario del Quiz debe ser construido a partir de un documento XML, deben existir distintos tipos de preguntas y debe desplegarse en la interfaz de usuario de una manera interactiva incluyendo su calificación.

2. Tecnologías seleccionadas.
 - HTML.
 - Javascript.
 - PHP.
 - MYSQL.
 - CSS.
3. Herramientas seleccionadas.
 - Visual Studio Code.
 - Google Chrome.
 - WAMP SERVER

- PHPMyAdmin.

4. Modelo de arquitectura software

Debido a que el presente componente necesita el uso de bases de datos en su desarrollo y además desarrollo por parte del servidor, el modelo de arquitectura software seleccionado es el presentado en la Figura 16.

5. Diagrama de clases.

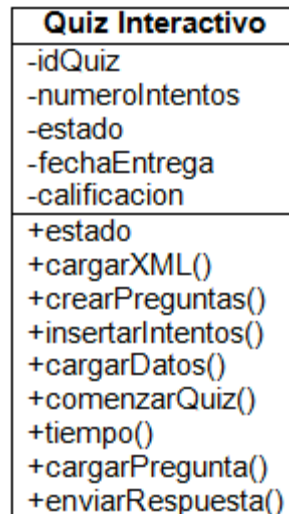


Figura 19. Diagrama de clase Quiz

Tareas

1. Requerimiento.

Desarrollar un componente para que en un entorno de educación virtual, los estudiantes puedan gestionar sus tareas con una comunicación interactiva con el docente. El componente debe mostrar como primera pantalla la tarea asignada por el profesor: (Titulo, descripción y/o archivos adjuntos). Por parte del estudiante está la funcionalidad de adjuntar sus archivos adjuntos con algunas observaciones o comentarios dirigidos al profesor. El componente también debe permitir editar y eliminar respuestas enviadas y finalmente como respuesta del profesor debe desplegarse una calificación y una retroalimentación por parte del docente al estudiante.

2. Tecnologías seleccionadas.

- HTML.
- Javascript.
- PHP
- MYSQL
- CSS.

3. Herramientas seleccionadas.

- Visual Studio Code.
- Google Chrome.

- WAMP SERVER
- PHPMyAdmin.

4. Modelo de arquitectura software.

Debido a que el presente componente necesita el uso de bases de datos en su desarrollo y además desarrollo por parte del servidor, el modelo de arquitectura software seleccionado es el presentado en la Figura 16.

5. Diagrama de clases.

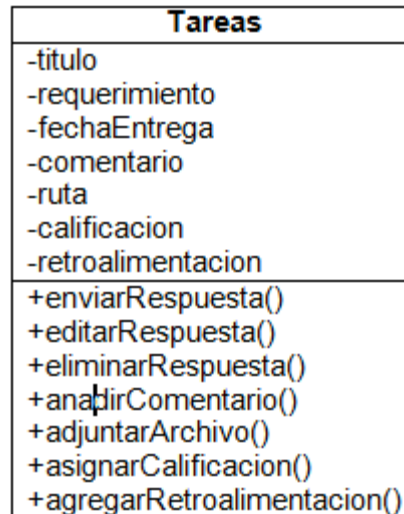


Figura 20. Diagrama de clase de Tareas

Chat

1. Requerimiento.

Desarrollar un componente para que en un entorno de educación virtual, los estudiantes puedan comunicarse entre sí a través de mensajería instantánea. El Chat debe permitir el envío de texto y emoticones. Los usuarios deberán identificarse con un nombre de usuario y una contraseña para el ingreso de sesión.

2. Tecnologías seleccionadas.

- HTML.
- Javascript.
- NodeJS
- Sockets
- CSS.

3. Herramientas seleccionadas.

- Visual Studio Code.
- Google Chrome.

4. Modelo de arquitectura software.

Debido a que el presente componente necesita el uso de bases de datos en su desarrollo y además desarrollo por parte del servidor, el modelo de arquitectura software seleccionado es el presentado en la Figura 16.

5. Diagrama de clases.

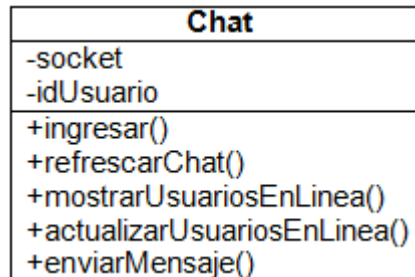


Figura 21. Diagrama de clase de Chat

Recursos Relacionados

1. Requerimiento.

Desarrollar un componente para que en un entorno de educación virtual, los estudiantes puedan acceder a través de enlaces a otros recursos de educación brindados por parte del docente.

2. Tecnologías seleccionadas.

- HTML.
- Javascript.
- CSS.

3. Herramientas seleccionadas.

- Visual Studio Code.
- Google Chrome.

4. Modelo de arquitectura software.

Debido a que el presente componente no necesita el uso de bases de datos en su desarrollo ni tampoco desarrollo por parte del servidor, el modelo de arquitectura software seleccionado es el presentado en la Figura 15.

5. Diagrama de clases.

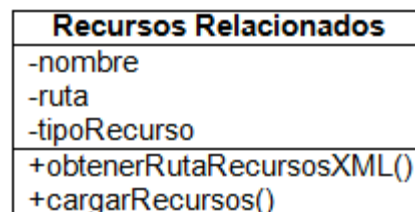


Figura 22. Diagrama de clase Recursos relacionados

Tabla de Contenido

1. Requerimiento.

Desarrollar un componente para que en un entorno de educación virtual, los estudiantes puedan apreciar el contenido de la clase que el docente esté dictando. El componente debe desplegar los temas de la clase en enlaces que tienen la funcionalidad de relacionar al estudiante interactivamente con dicho tema en componentes que proporcionen información de difusión de información formativa como lo es el reproductor de video y el componente de presentación de documentos.

2. Tecnologías seleccionadas.

- HTML.
- Javascript.
- CSS.

3. Herramientas seleccionadas.

- Visual Studio Code.
- Google Chrome.

4. Modelo de arquitectura software.

Debido a que el presente componente no necesita el uso de bases de datos en su desarrollo ni tampoco desarrollo por parte del servidor, el modelo de arquitectura software seleccionado es el presentado en la Figura 15.

5. Diagrama de clases.

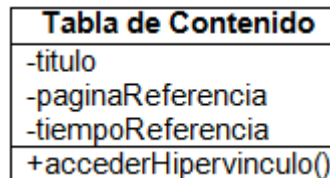


Figura 23. Diagrama de clase Tabla de contenido

5.1.3. Proceso de desarrollo

En la sección anterior son identificados aspectos previos a la implementación de código mediante la metodología SCRUM. En esta sección es llevado a cabo un análisis del desarrollo implementado para uno de los componentes. A través de esta sección es posible identificar la metodología utilizada a través de los lenguajes de programación y las herramientas seleccionadas. El componente seleccionado para este análisis es el reproductor de video, ya que es el más representativo en el contexto del trabajo. A continuación es presentada la composición de las carpetas que lo compone con los archivos involucrados:

• Archivos iniciales:

La Figura 24 presenta 3 archivos iniciales en el editor de código. El primero, denominado “debug.log”, es creado por el propio editor para presentar en pantalla algún error presentado con la herramienta durante la programación. El segundo, archivo “index.html”, es el archivo inicial del proyecto de tipo html. Aquí se encuentra todo el código html presente en el componente. En este archivo son invocados también los demás archivos relacionados con la funcionalidad y la vista. El último archivo llamado “LICENSE” sólo es uno de los tantos archivos que son insertados automáticamente, al utilizar alguna librería o framework en el desarrollo, especificando la licencia respectiva de este.



Figura 24. Archivos iniciales del reproductor de video

- **Carpeta “css”:**

En la Figura 25 es posible apreciar la carpeta donde son encontrados los archivos relacionados con la vista y estilo del componente. Ya que es seleccionada la tecnología css, los archivos son de tipo css. El primer archivo denominado “estilo.css” contiene los estilos definidos por el desarrollador y los otros dos archivos que es posible apreciar hacen parte de un *framework* llamada “Materializecss” que le permite al desarrollador contar con estilos más profesionales y dinámicos en el desarrollo.

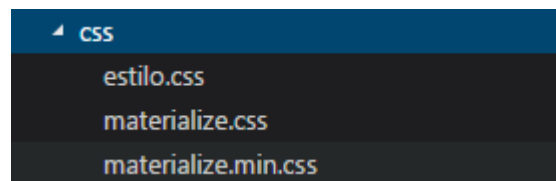


Figura 25. Carpeta de estilos de reproductor de video

- **Carpeta “fonts”:**

En la Figura 26 es posible apreciar una carpeta dedicada exclusivamente a las fuentes de texto utilizadas en el componente. También es seleccionada una familia de fuentes denominada “roboto” donde hay suficientes fuentes para conseguir el aspecto deseado.

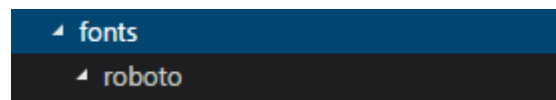


Figura 26. Fuentes utilizadas en el reproductor de video

- **Carpeta “images”:**

En la Figura 27 es posible apreciar una carpeta dedicada exclusivamente a las imágenes utilizadas en el componente. Las imágenes son recursos utilizados para diseñar de la manera más clara el aspecto visual del componente. Muchas imágenes de botones son representadas a través de iconos de fuentes, pero existen algunos casos en los que es necesario incluir imágenes propias para componentes con funciones novedosas con no cuentan con gran variedad de iconos.

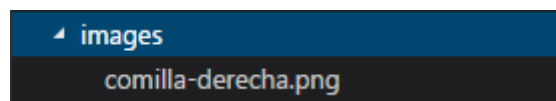


Figura 27. Imágenes utilizadas en el reproductor de video

- **Carpeta “js”:**

La Figura 28 muestra la carpeta denominada “js” que recoge toda la funcionalidad del componente. El primer archivo denominado “controller.js” contiene todas las funciones de modificar aspectos gráficos del componente como el tamaño, colores de fondo, tipo de fuente,

etc. El archivo "jquery.min.js" es el archivo requerido por jquery para el uso de esta librería al igual que los dos archivos siguientes necesarios para usar la librería materializecss.

El archivo denominado "mireproductor.js" contiene toda la funcionalidad del componente, es posible decir que este es el nucleo del componente en donde es creado y controla las funciones del reproductor de video tales como reproducir, pausar, subir o bajar volume, etc. Por último, la carpeta "sincronizacion.js" contiene funciones relacionadas con otros componentes que son involucradas funcionalmente con este, por ejemplo, el caso de sincronizar el video con un documento del componente de presentación de documentos.

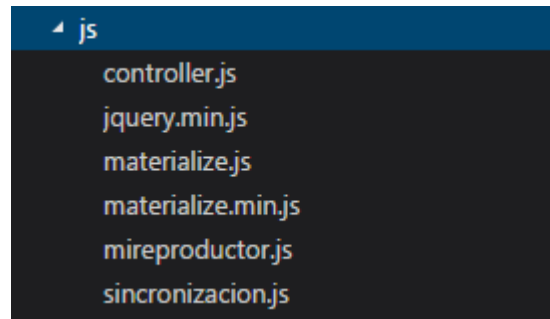


Figura 28. Archivos Javascript del reproductor de video.

- **Carpeta "metadatos":**

La Figura 29 muestra una carpeta denominada "metadatos" la cual tiene almacenada un archivo de tipo XML en donde son recogidos todos los recursos de información que el componente necesita para ejecutarse tales como el título, descripción, url del video, etc.

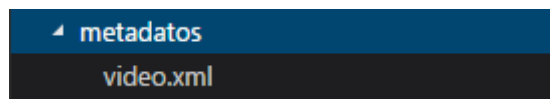


Figura 29. Carpeta de metadatos del reproductor de video.

- **Carpetas de recursos:**

La Figura 30 muestra una carpeta dedicada a los archivos de subtítulos y la Figura 31 muestra una carpeta donde estan los videos que son presentados en el reproductor. Este tipo de carpetas son utilizadas como recursos para probar el funcionamiento del componente.

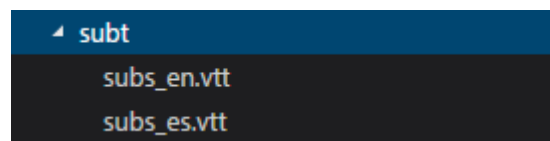


Figura 30. Carpeta de subtítulos de reproductor de video.

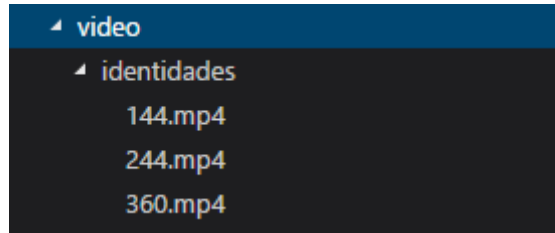


Figura 31. Carpeta de videos de reproductor de video.

5.1.4. Prototipo final de desarrollo de componentes

A partir de los desarrollos de código implementados es construido cada componente hasta obtener un prototipo final del mismo. En esta sección es presentado un bosquejo del prototipo final obtenido de cada componente mediante una imagen. Estos prototipos son presentados en las Figuras Figura 32 a Figura 39.

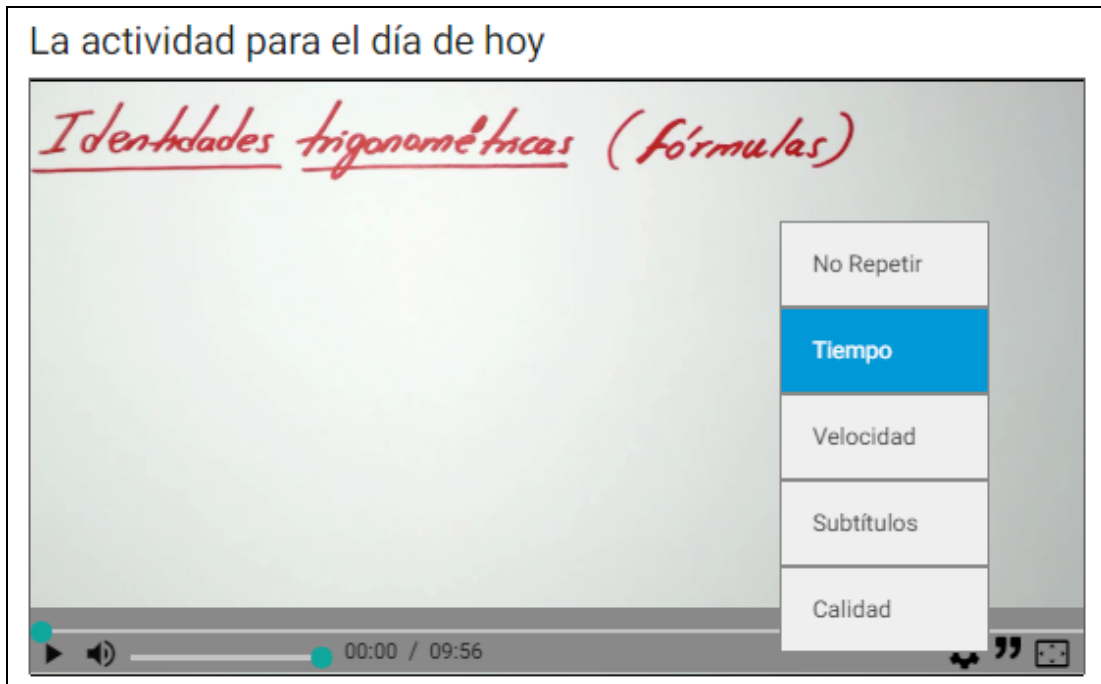


Figura 32. Prototipo final componente de reproductor de video

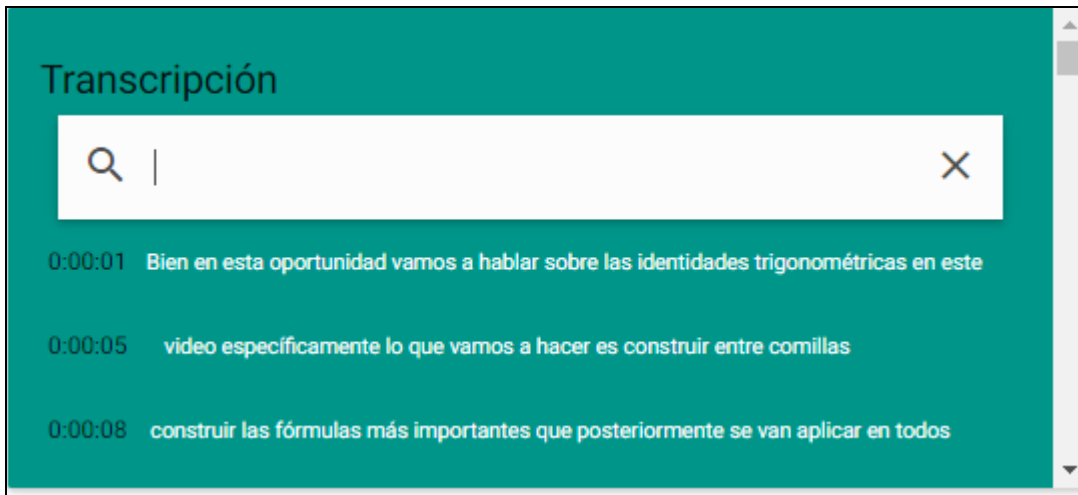


Figura 33. Prototipo final sección de transcripción de texto del componente de reproductor de video.

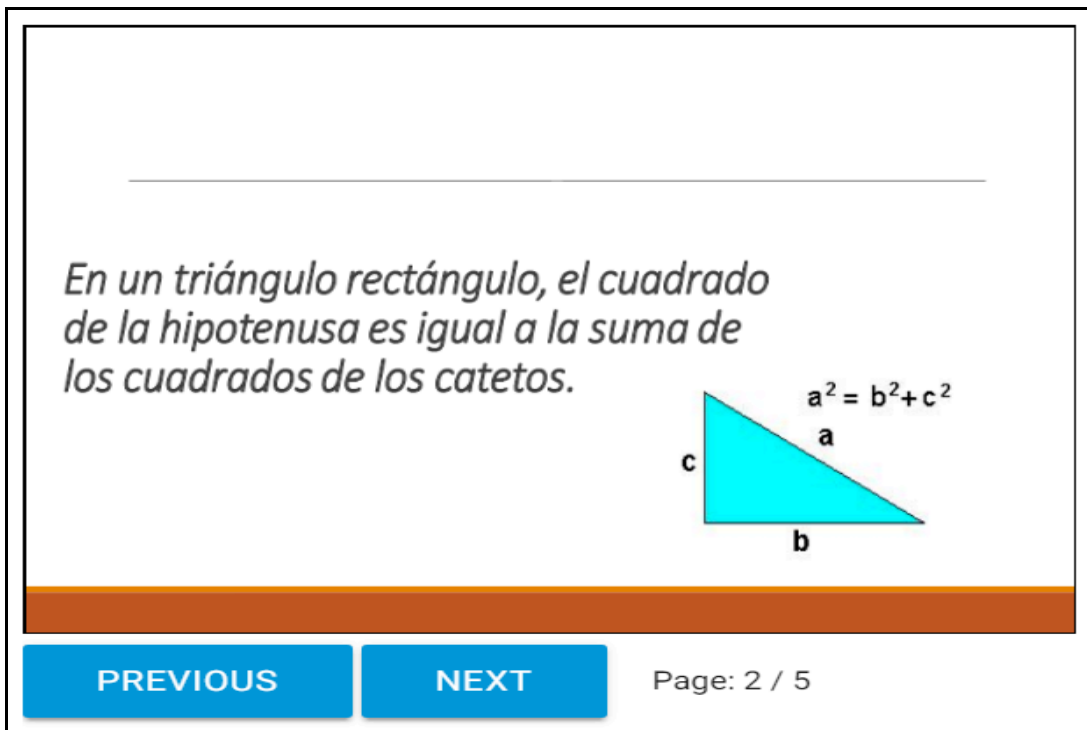


Figura 34. Prototipo final componente de visualizador de documentos

figura y diagonales

Seleccione ▼

cuadrado

Seleccione ▼

hexagono

Seleccione ▼

triangulo

Un triangulo equilatero tiene todos los lados iguales?

Verdadero

Falso

Un angulo llano es un angulo de 90 grados?


Verdadero

Falso

Figura 35. Prototipo final Quiz

Tareas pendientes

27/4/2017

 Demostración de identidades trigonométricas

Fecha de entrega: 2017-10-02

Descripción:

Demostrar cada una de las identidades trigonométricas vistas en clase.

Enviar una Respuesta:

Observaciones:

Subir Archivo: Ningún archivo seleccionado

Figura 36. Prototipo final de componente de tareas

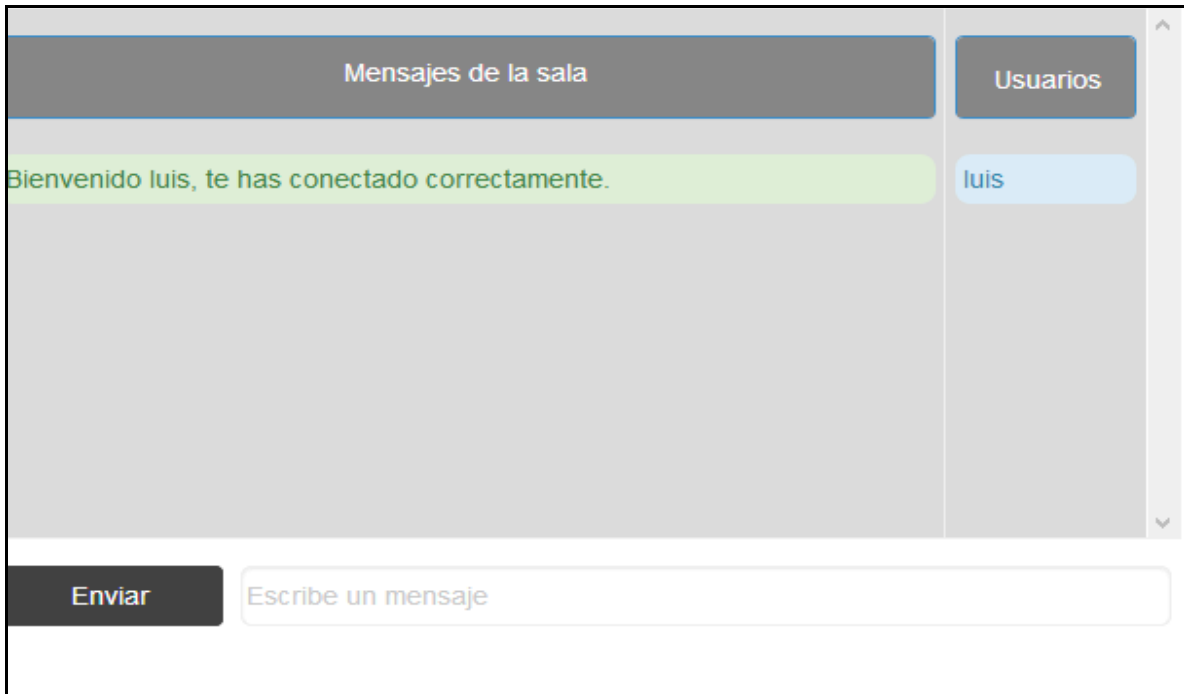


Figura 37. Prototipo final Chat usuario en línea

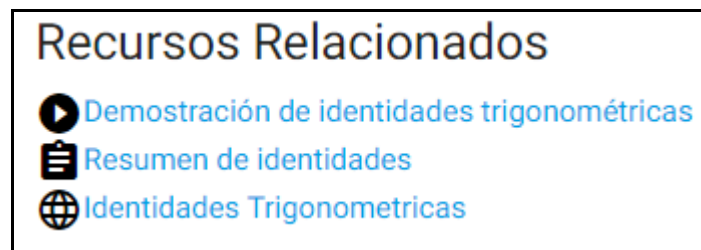


Figura 38. Prototipo final Recursos relacionados

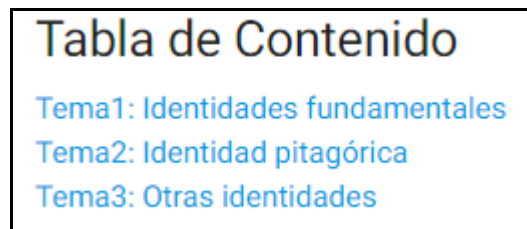


Figura 39. Prototipo final componente Tabla de Contenido

5.2. Implementación del Framework basado en modelos

En esta sección inicia el proceso de desarrollo software de la herramienta planteada. También explica detalladamente cada uno de los pasos de este proceso partiendo desde la creación del proyecto hasta la generación del código final. Este tipo de herramientas con esta tecnología propuesta son denominadas herramientas de modelado, que como puede apreciarse en el marco teórico, está sustentada en ingeniería de desarrollo a través de modelos. También es

necesario en algunos pasos, introducir algunos nuevos conceptos que serán identificados en la construcción de la misma.

La implementación de una herramienta de modelado es lograda guiándose por un proceso a seguir establecido por GMF el cual está compuesto por 3 etapas importantes: la primera es el dominio de modelo o también llamado metamodelo. En este caso de estudio, el metamodelo utilizado es el construido en el Capítulo 4 como la sintaxis abstracta y generará un código propio de modelo. La segunda etapa, es la creación del editor gráfico en la cual está el modelo de definición gráfica, la definición de herramienta y la relación entre elementos los elementos. La tercera y última etapa es la generación de código final. Cada una de estas etapas está propiamente especificada en esta sección mostrando paso a paso la construcción de la herramienta.

5.2.1. Requisitos previos

Para poder llevar a cabo esta implementación es necesario algunos requisitos mínimos. En la parte hardware es necesario un ordenador con al menos 256 MB de memoria RAM y como requisitos software es necesario:

- JDK 1.5.
- Eclipse build S-3.2.
- EMF 2.2.0.
- GEF 3.2.
- GMF 1.0.

Las subsecciones siguientes amplía la información correspondiente de cada uno de los requisitos software respecto a su instalación y configuraciones a tener en cuenta para su uso.

5.2.1.1. JDK⁴

- **Sustento de requerimiento**

JDK es utilizado para compilar los archivos de tipo .java a archivos de tipo .class, este proceso puede explicarse simplícidamente de la siguiente forma: al crear los archivos .java, el ordenador no es capaz de interpretar este código, aquí es cuando entra la tarea de *JDK*. Si esta garantizado la no existencia de errores de sintaxis en la clase .java, *JDK* realiza una compilación la cual traduce el lenguaje java a lenguaje de máquina, es decir en "ceros y unos". Una vez realizada la compilación es generado un archivo con extensión .class. Este archivo generado se puede ejecutar mediante comandos propios de la JVM⁵.

- **Instalación**

Con la instalación de *JDK* es conseguido el entorno de desarrollo para la construcción de aplicaciones a través del lenguaje de programación *JAVA*. Al acceder al sitio web oficial de Oracle⁶ puede hacerse la descarga del archivo ejecutable aceptando los términos de licencia e identificando el sistema operativo de la máquina.

⁴ Java Development Kit.

⁵ Java Virtual Machine

⁶ <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>.

5.2.1.2. Eclipse Build

- **Instalación**

En el sitio web oficial de Eclipse es posible descargar esta plataforma de desarrollo. Eclipse es una plataforma *open-source* y la web del proyecto facilita la descarga sin ninguna complejidad. Accediendo a la web, son presentados muchas versiones disponibles para los usuarios, para este caso en específico, es descargada la versión Eclipse Modelling Tools (*Includes incubating components*), dicha versión, incluye todos los recursos necesarios que sera utilizados en el desarrollo de la herramienta. Seguidamente es elegido el servidor con el cual va a realizar la descarga y obteniendo un fichero, es descomprimido y escogido un lugar donde ejecutar la plataforma.

- **Ejecución de eclipse**

Como primera medida, antes de comenzar el trabajo con Eclipse, debe crearse una carpeta identificada como el espacio de trabajo, ahí estan ubicados todos los demás archivos que seran creados tanto manual como automáticamente. Es muy importante la buena organización de las carpetas en términos de saberlas identificar con nombres claros y acordes al desarrollo además de tener una jerarquía clara entre ellas.

Al ejecutar la plataforma, esta pide seleccionar el *workspace* (el espacio de trabajo creado), en realidad un espacio de trabajo simplemente establece en qué carpeta de disco seran almacenados los diferentes ficheros que compongan todo el proyecto en Eclipse.

5.2.1.3. EMF y GMF

Antes de continuar con la ejecución del eclipse, es importante verificar que el Eclipse tiene las librerías necesarias para utilizar EMF y GMF. Esto puede verificarse siguiendo los siguientes pasos: Click en “Help”, “About Eclipse” y seguidamente pulsar el botón “Installation details” y en el botón de “Plug-ins”. Una vez realizado esto, en la lista de *plug-ins* instalados debe comprobarse que los plug-ins estén correctamente instalados tal cual como lo muestra Figura 40. EMF y GMF son las herramientas con las cuales seran construidas las sintaxis abstracta y concreta respectivamente.

Provider	Feature Name	Version	Feature Id
Eclipse Modeling Project	EMF - Eclipse Modeling Fram...	2.12.0.v2016052...	org.eclipse.emf
Eclipse Modeling Project	EMF Code Generation	2.11.0.v2016052...	org.eclipse.emf.codegen
Eclipse Modeling Project	EMF Code Generation UI	2.8.0.v20160526...	org.eclipse.emf.codegen.ui
Eclipse Modeling Project	EMF Common	2.12.0.v2016042...	org.eclipse.emf.common
Eclipse Modeling Project	EMF Common UI	2.11.0.v2016052...	org.eclipse.emf.common.ui
Eclipse Modeling Project	EMF Compare Core	3.2.1.201608311...	org.eclipse.emf.compare
Eclipse Modeling Project	EMF Compare Core	3.2.1.201608311...	org.eclipse.emf.compare.sou
Eclipse Modeling Project	EMF Compare Diagram Com...	3.2.1.201608311...	org.eclipse.emf.compare.diaç
Eclipse Modeling Project	EMF Compare IDE UI	3.2.1.201608311...	org.eclipse.emf.compare.ide.
Eclipse Modeling Project	EMF Compare IDE UI	3.2.1.201608311...	org.eclipse.emf.compare.ide.
Eclipse Modeling Project	EMF Data Binding	1.4.0.v20160526...	org.eclipse.emf.databinding
Eclipse Modeling Project	EMF Documentation	2.11.0.v2016052...	org.eclipse.emf.doc
Eclipse Modeling Project	EMF Ecore	2.12.0.v2016042...	org.eclipse.emf.ecore
Eclipse Modeling Project	EMF Ecore Code Generator	2.12.0.v2016052...	org.eclipse.emf.codegen.ecor
Eclipse Modeling Project	EMF Ecore Code Generator UI	2.12.0.v2016052...	org.eclipse.emf.codegen.ecor
Eclipse Modeling Project	EMF Ecore Edit	2.9.0.v20160526...	org.eclipse.emf.ecore.edit
Eclipse Modeling Project	EMF Ecore Mapping	2.9.0.v20160526...	org.eclipse.emf.mapping.eco
Eclipse Modeling Project	EMF Ecore Mapping Editor	2.10.0.v2016052...	org.eclipse.emf.mapping.eco
Eclipse Modeling Project	EMF Edit	2.12.0.v2016052...	org.eclipse.emf.edit
Eclipse Modeling Project	EMF Edit Data Binding	1.4.0.v20160526...	org.eclipse.emf.databinding.ε

Figura 40. Plug-ins de modelado instalados en Eclipse.

5.2.1.4. GEF

Eclipse proporciona un *framework* que les permite a usuarios finales la posibilidad de realizar editores gráficos y vistas como parte de las aplicaciones. Esta herramienta esta dividida internamente en tres componentes:

- Draw2d: un componente de visualización 2D basado en el conjunto de herramientas estándar de Widget (SWT)
- GEF (MVC) : un componente basado en el modelo vista controlador que puede utilizarse para realizar editores gráficos como parte de las aplicaciones de Eclipse Rich Client Product (RCP)
- Zest: un juego de herramientas de visualización basado en gráficos que se puede utilizar para realizar vistas para la visualización de estructuras de datos tipo gráfico como parte de las aplicaciones Eclipse RCP

Mientras que las aplicaciones gráficas pueden construirse directamente sobre los componentes de GEF, Draw2d y GEF (MVC) también son utilizados por el Framework de Modelado Gráfico (GMF), que los combina con el Marco de Modelado de Eclipse (EMF) para crear el código tanto para el modelo de datos como para el editor gráfico.

En las configuraciones de Eclipse también es importante verificar la instalación de este *framework*, estas deben presentarse en las características de Eclipse tal cual como lo muestra la Figura 41. Con esta verificación, es considerado que ya están todos los elementos necesarios para comenzar la construcción de la herramienta de modelado planteada en este trabajo.

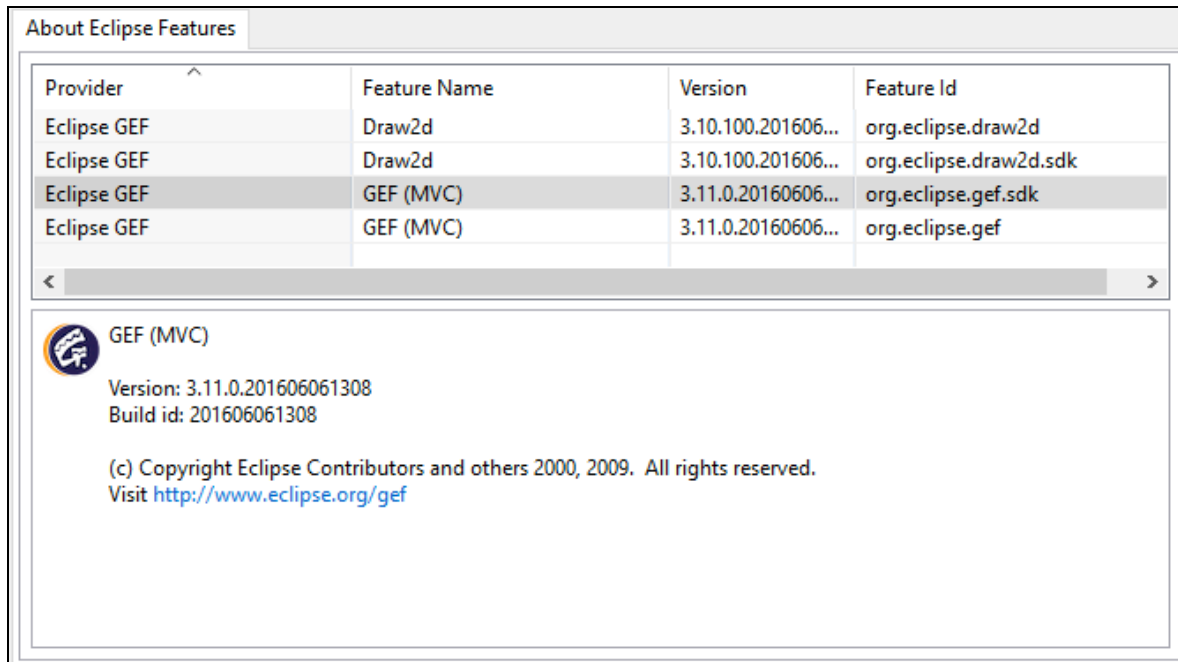


Figura 41. Plug-ins GEF instalados en Eclipse.

Teniendo a disposición los requisitos software instalados y configurados previamente, luego esta la creación de un proyecto GMF. Este proceso es presentado en la siguiente sección.

5.2.2. Creación de un proyecto GMF

El modelo de dominio a menudo necesita ser visualizado gráficamente, tal cual como un editor de diagramas, esto es muy útil cuando esta desarrollandose herramientas de modelado. Eclipse proporciona algunos *frameworks* para esto, GMF es un *framework* con arquitectura dirigida por modelos que sirve como un editor gráfico completamente funcional.

El proyecto GMF permite construir la sintaxis concreta definida en el Capítulo 4 este proyecto permite la generación de un editor gráfico basado en EMF y GEF. La principal característica de GMF es la reutilización de las definiciones gráficas definidas por medio de modelos de cada elemento específico. En este trabajo los modelos son definidos a partir de cada componente de interfaz de usuario seleccionado como caso de estudio.

Para la creación del proyecto GMF de Eclipse deben seguirse los siguientes pasos:

1. Ejecutar la herramienta Eclipse con los requisitos previos ya explicados en este capítulo.
2. Pulsar clic en *File*, luego en *New*, y luego en *Project*: Seguidamente, buscar la carpeta de *Graphical Modelling Framework* y seleccionar *Graphical Editor Project*.
3. Al pulsar *Next*, Nombrar el proyecto (No se debe asignar ningún nombre que coincida con alguna de las clases, pues puede generar error en la generación de código). Finalmente, activar la casilla "*Show dashboard view for created project*" y pulsar *Finish*.

La Figura 42, muestra un esquema de composición del proyecto y el proceso de construcción del editor gráfico. Se puede observar que siempre se parte desde el modelo de dominio, a partir de este, se derivan otros modelos que son necesarios para la generación del editor: Por

un lado esta el modelo de definición gráfica que especifica los gráficos que pueden ser dibujados en el editor y por otro lado esta el modelo de definición de herramientas “*tooling*”, el cual contiene información de los elementos en la paleta del editor. Al definir estos modelos, son relacionados a través de un modelo denominado *Mapping*. Finalmente, al definir cada uno de los modelos y relacionarlos, GMF provee un generador de modelo que permite definir los detalles de implementación anteriores a la fase de generación de código. La generación de código generará un *plugin* que interrelaciona la notación con el modelo de dominio. También provee persistencia y sincronización de ambos.

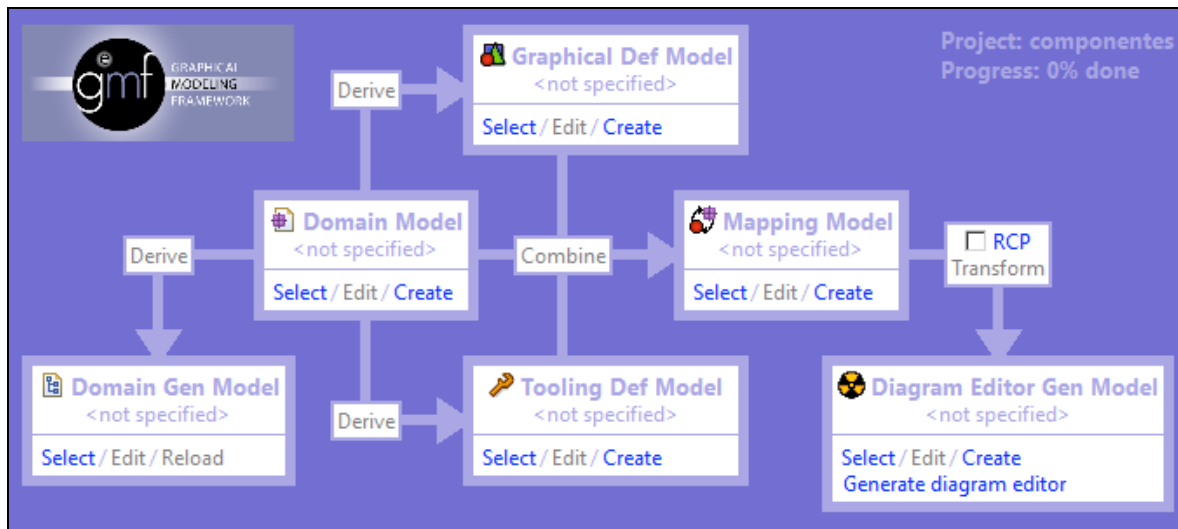


Figura 42. Composición y modelos GMF

5.2.3. Generación de código a partir de un modelo

5.2.3.1. Definición del metamodelo

La característica principal de EMF consiste en valerse de un metamodelo y proporcionárselo a GMF. El proceso consiste en transformar los modelos elaborados tipo “diagramas de clase” en clases java para Eclipse. Este proceso es denominado generación de código de modelo. A continuación es presentado el proceso utilizado para traer el metamodelo definido en la sintaxis abstracta para la construcción de esta herramienta.

Como primera medida cabe recordar que al crear el diagrama con Ecore del Capítulo 4 EMF genera un archivo con extensión *.ecore* que asocia el diagrama del modelo. Este archivo es el utilizado para esta generación de código. El proceso de importación de modelo es realizado de la siguiente manera: Primero dando clic derecho sobre el proyecto GMF creado, elegir la opción *New*, luego *Other* y en la carpeta de *Eclipse Modelling framework*. Después, seleccionar *EMF Project* y al pulsar *Next*, debe nombrarse el proyecto. Seguidamente debe presentarse un panel como el de la Figura 43, en donde son dispuestos varios tipos de modelo a importar. En este caso, pulsar en *Ecore model*, escoger el modelo con la extensión *.ecore* ya creado tal cual como lo muestra la Figura 44, luego es seleccionado el paquete como lo indica la Figura 45 y finalizar pulsando *finish*. De esta manera ya queda importado al proyecto el metamodelo seleccionado.

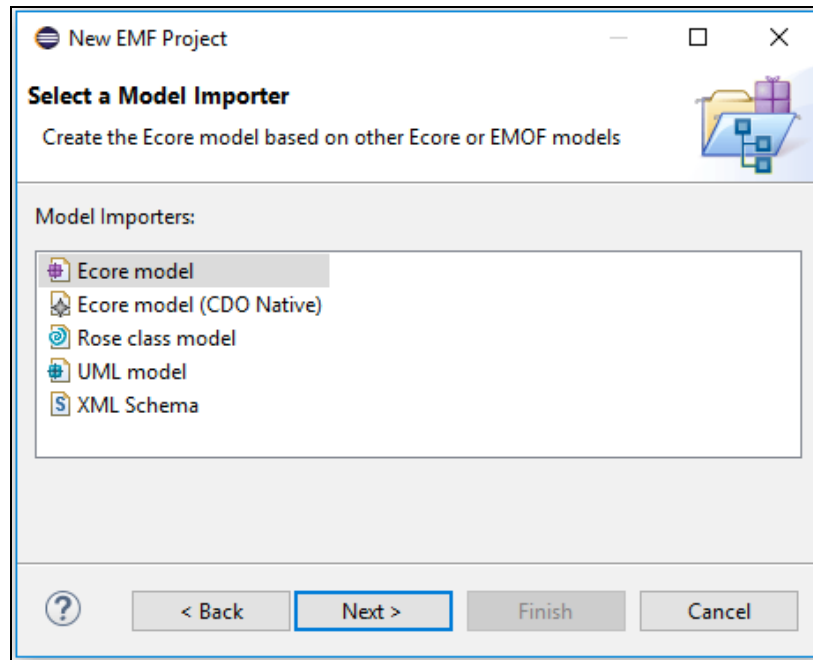


Figura 43. Importación de un modelo en EMF

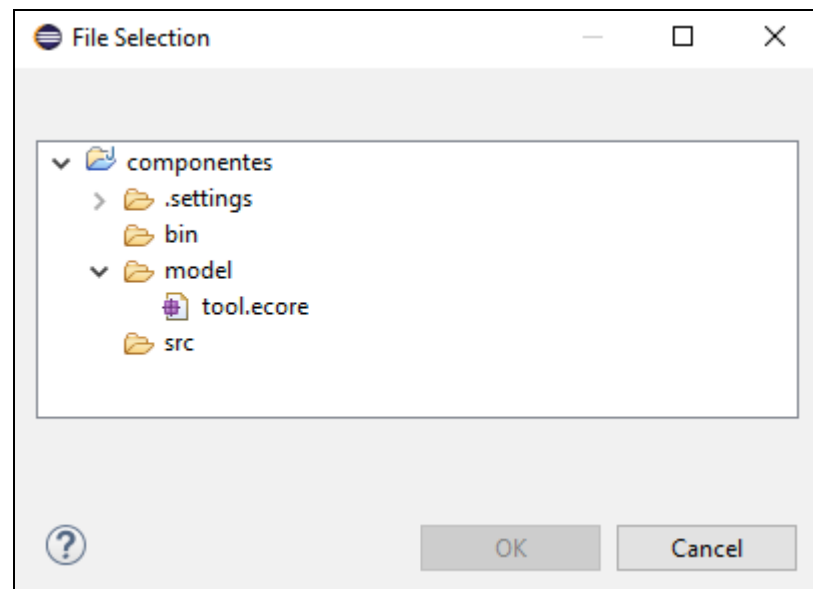


Figura 44. Selección de metamodelo a importar EMF

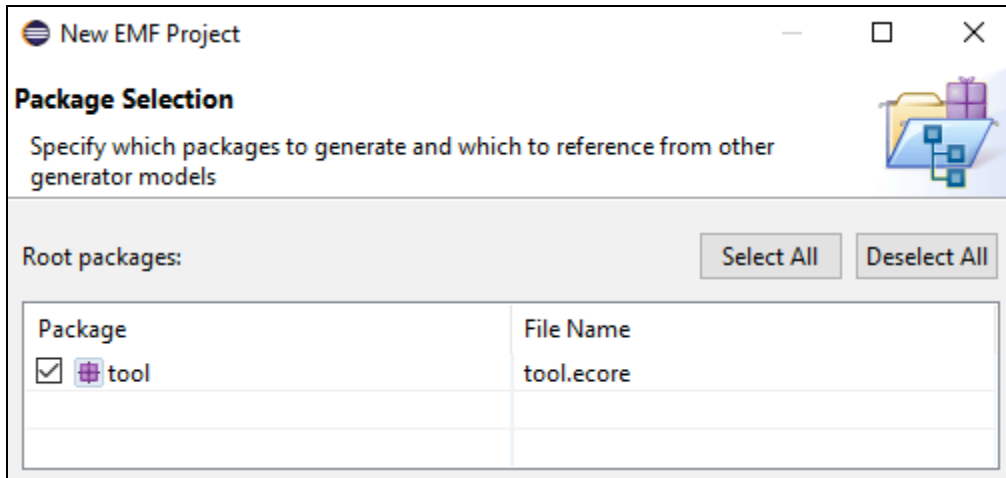


Figura 45. Presentación de metamodelo seleccionado

5.2.3.2. Generación de código

Teniendo el modelo en el proyecto sigue su generación de código. Para esto, debe identificarse un archivo creado automáticamente al importar el modelo el cual tiene una extensión *.genmodel*. Este archivo permite que a través de patrones de transformación son generadas las clases java de dicho modelo. Estas clases java serán utilizadas posteriormente en el proceso de creación de la herramienta de modelado.

Antes de generar el código, es necesario definir el paquete base del modelo *.genmodel*, para ello, pulsamos clic con el botón derecho y es seleccionada la opción "show properties View". De esta manera es configurado como lo muestra la Figura 46.

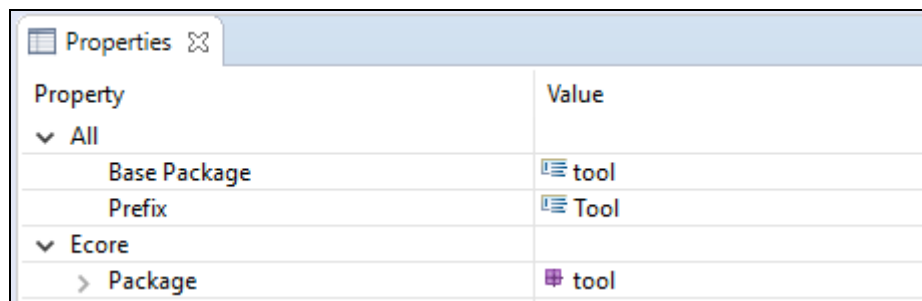


Figura 46. Configuración de paquete base de modelo

Con esto, ya es posible proceder a la generación de código, para esto debe pulsarse clic derecho sobre el paquete que compone el archivo de tipo *.genmodel* y seleccionar la opción *Generate all*. Una vez realizado esto, es creado automáticamente el *plugin* del código de modelo junto con los *plugins* correspondientes a la generación de código *edit*, *editor* y *tests*.

Al generar el código pueden identificarse las clases Java completamente manipulables a nivel de instancias, también es compuesto de clases adaptadoras para visualizar y editar las propiedades de las instancias desde la vista "propiedades" de Eclipse. Además provee un editor básico en forma de árbol para crear instancias del modelo. Por último, incluye un conjunto de casos de prueba para permitir verificar propiedades.

5.2.4. Creación del editor gráfico

En esta sección es presentada la creación del editor gráfico a través de la definición de modelos, cada modelo es definido en archivos separados, con formato XMI. GMF provee un editor para hacer más amigable cada una de estas definiciones. Por organización, es creada una carpeta denominada “*app*” la cual está compuesta por estos modelos. Dichos modelos de definición son presentados en las subsecciones a continuación.

5.2.4.1. Modelo de Definición gráfica

A través del modelo de definición gráfica puede configurarse el aspecto visual de los elementos en el editor gráfico. En el diagrama pueden definirse figuras, nodos, links, compartimientos y demás elementos gráficos que el desarrollador guste agregar. Para iniciar la creación de este modelo debe seguirse el siguiente proceso:

1. Clic derecho sobre el fichero con extensión *.ecore* (el metamodelo importado); después *New*, *Other*, y ubicarse en el fichero denominado “*Simple Graphical Definition Model*” tal cual como lo muestra la Figura 47.
2. Al pulsar *Next*, seleccionar la carpeta “*app*” (como lo muestra la Figura 48), en donde será guardado este modelo. También es definido un nombre para este fichero con la extensión *.gmfgraph*.
3. Seguidamente debe seleccionarse el archivo *Model Factory* (ver Figura 49) que es con la cual está definido el modelo de dominio.
4. Después debe elegirse qué elementos del metamodelo actuarán como nodos, cuáles como enlaces y cuáles como etiquetas. La ventana que Eclipse presenta tiene representado a través de tres columnas en la cabecera así: cuadrado con rayas horizontales (nodos), línea inclinada (enlaces), y letra A (etiquetas). Cada elemento del modelo tiene asociada una casilla para seleccionar, para así elegir el tipo de representación gráfica que guste.
5. Finalmente, pulsar *finish* y el modelo de definición gráfica será creado. El fichero con la extensión “.gmfgraph” debe tener el aspecto como el de la Figura 51.

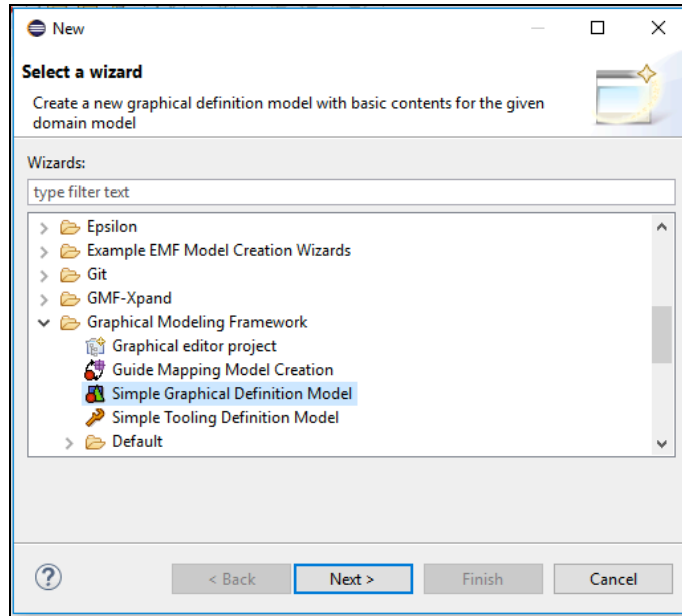


Figura 47. Creación de modelo de definición gráfica

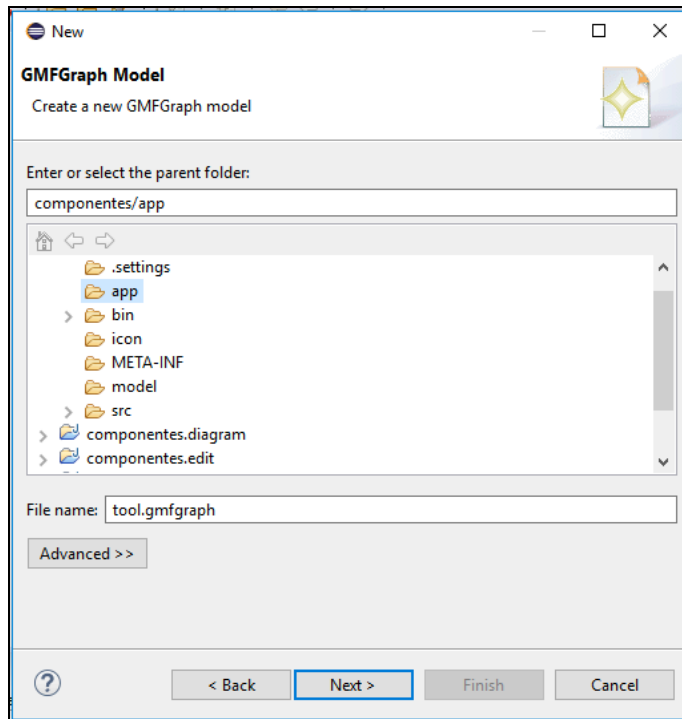


Figura 48. Selección de folder para modelo de definición gráfico

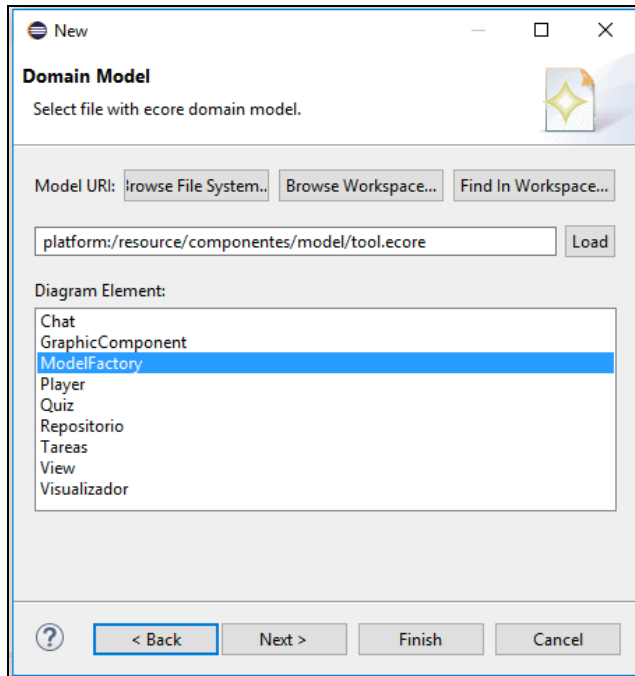


Figura 49. Selección de archivo con modelo de dominio.ecore

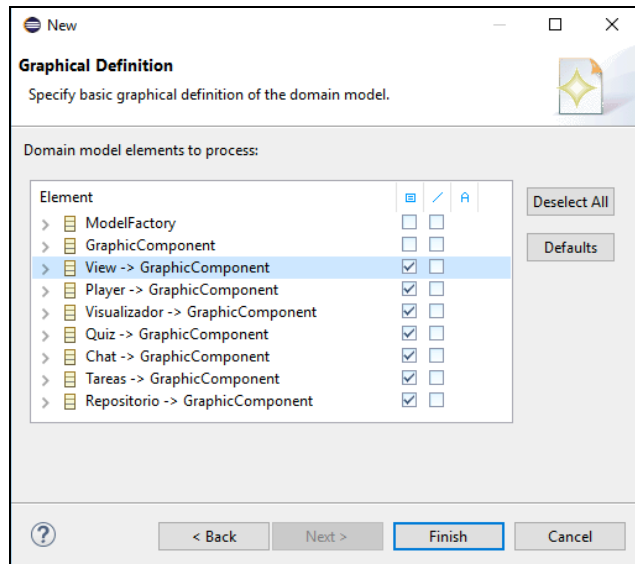


Figura 50. Definición de nodos, enlaces y etiquetas MDG

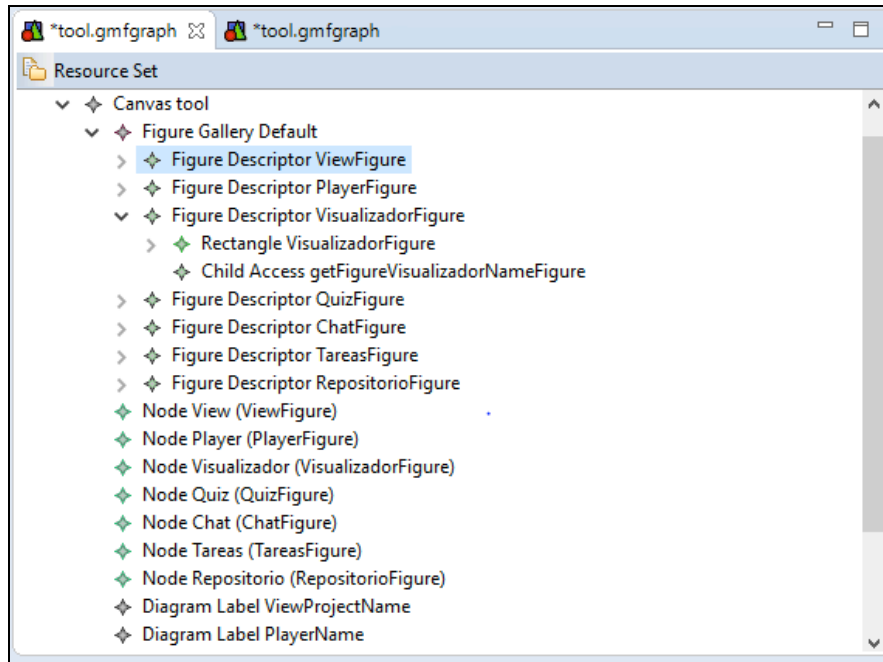


Figura 51. Estructura de modelo de definición gráfica.

Como puede notarse, el MDG⁷ consta de un fichero de nombre “*tool.gmfgraph*”. Este fichero contiene un archivo “*Canvas tool*” con todos los elementos involucrados en la edición gráfica de los componentes. Los nodos del modelo (*Player*, *Visualizador*, *Chat*, etc) están definidos por un descriptor de figura que contiene un “*Rectangle*” por defecto y además contiene unas funciones de acceso a etiquetas denominadas “*Child Access getFigure*”. Al desplegar la figura “*Rectangle*” de cada nodo, es posible identificar las etiquetas *Label* y la alineación de las etiquetas *Flow Layout*, adicionalmente por cada figura existe un nodo de acceso al *FigureDescriptor*.

Para este trabajo, esta definido la vista de cada nodo haciendo uso de los bosquejos de los componentes definidos en la sintaxis concreta del Capítulo 4 y adicional una etiqueta denominada “*label*” para disponer el nombre.

La personalización planteada en el párrafo anterior, es lograda abriendo el fichero del nodo que quiere configurarse, luego eliminando el rectángulo que el fichero tiene por defecto y es reemplazado por un archivo de tipo “*SVG Figure*”. Con el fin de importar la imagen del *mockup* del componente. En las propiedades de este archivo SVG debe definirse la ruta y el nombre de la imagen (ver Figura 52). Finalmente, es creado un nuevo elemento hijo dentro de la imagen SVG y es seleccionada una etiqueta denominada “*label*”. Esta etiqueta también debe configurarse definiéndole un nombre y un texto a mostrar (ver Figura 53).

⁷ Modelo de Definición Gráfica

Property	Value
Descriptor	Figure Descriptor PlayerFigure
Document URI	platform:/plugin/componentes/icon/player.svg
Name	PlayerSVG
No Canvas Height	false
No Canvas Width	false

Figura 52. Configuración de propiedades de imagen SVG

Property	Value
Descriptor	Figure Descriptor PlayerFigure
Name	PlayerNameFigure
Text	<...>

Figura 53. Configuración de propiedades de etiqueta MDG

La herramienta necesita también la creación de un nuevo nodo de compartimiento con el fin de obtener un panel para introducir los componentes arrastrados y asociarlos a una interfaz de usuario. Este compartimiento tiene la función de acceso del nodo "View" y las propiedades para identificarse con una etiqueta.

Al finalizar, la estructura del fichero debe estar organizada como lo muestra la Figura 54. Puede percibirse en dicha figura la configuración del nodo *View* y el reproductor de video *Player* de esa manera están configurados los demás nodos. Nótese que los nodos poseen otras propiedades que el desarrollador agrega para adecuar la vista del nodo a su gusto. Así mismo, cada componente tiene su configuración con sus respectivas imágenes de sus *mockups* y sus respectivas configuraciones.

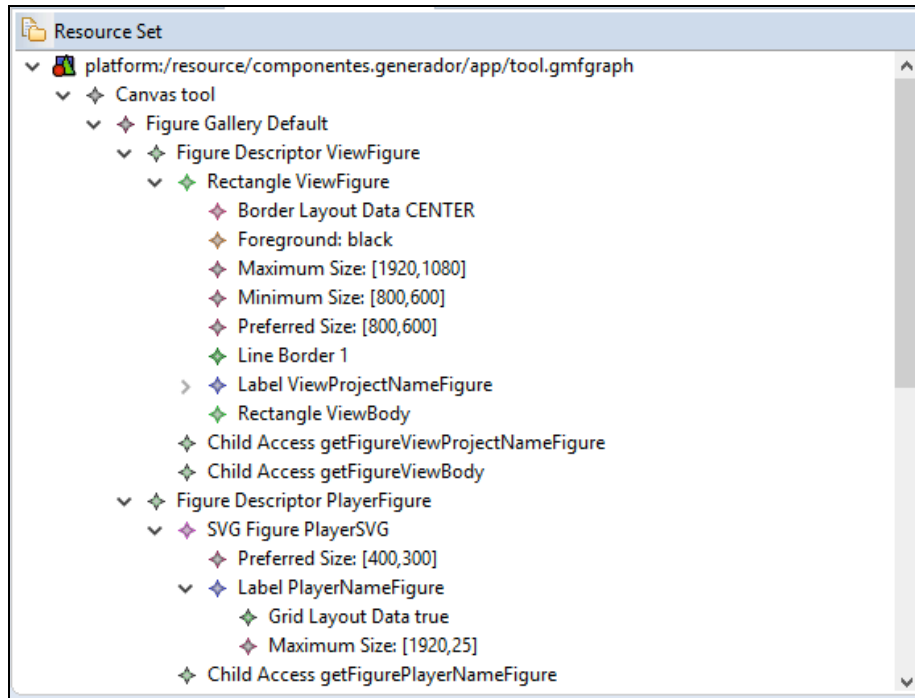


Figura 54. Configuración de imagen y etiqueta de nodos MDG

5.2.4.2. Definición de herramientas.

A través de este modelo es especificada la creación de la paleta de herramientas para cada nodo haciendo referencia a los elementos gráficos, además son definidas las acciones para cada nodo identificando cuál nodo es principal y cuál es secundario. También son definidas las acciones que desencadenan detrás de un elemento de la paleta y las imágenes de los botones que van a utilizarse en el editor gráfico.

Para crear este modelo de definición de herramientas es realizado el siguiente proceso:

1. Pulsar clic derecho sobre el metamodelo con extensión “.ecore”, seguidamente seleccionar *New*, luego *Other*, ubicar el fichero denominado “*Simple Tooling Definition*” en la carpeta “*Graphical Modeling Framework*” (ver Figura 55).
2. Al pulsar *Next* es seleccionada la carpeta app como destino y es asignado un nombre al proyecto con la extensión “.gmftool”.
3. Seleccionar el modelo de dominio del proyecto identificado como “ModelFactory”.
4. Al pulsar *Next*, al igual que en el modelo de definición gráfica es presentada una ventana de identificación y selección para los nodos y enlaces. En este caso, son identificados como nodos los componentes. Finalmente, es pulsado *Finish* y de esta manera es creado el modelo de definición de herramientas.

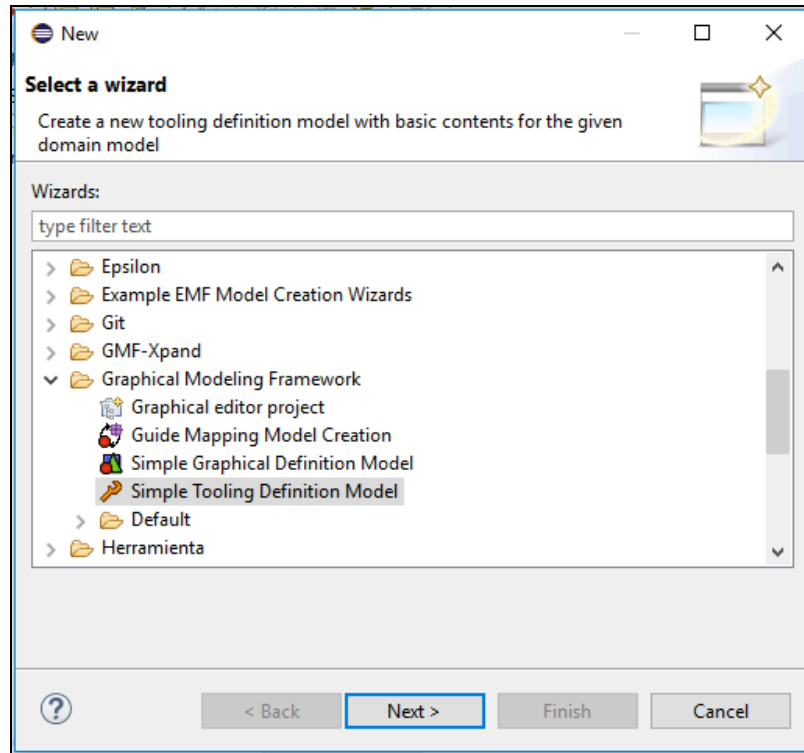


Figura 55. Creación de modelo definición de herramientas

Para darle distinción a los elementos de la paleta de herramientas, son definidos unos iconos específicos para cada componente. Los iconos deben relacionar directamente al usuario con el componente que va a crear. Para esto, son dispuestas las figuras diseñadas para el bosquejo de definición gráfica de herramientas diseñado en la sintaxis concreta del Capítulo 4. La asignación de este aspecto visual para la paleta es realizado de la siguiente manera:

1. Eliminar las dos imágenes por defecto “*default image*” que son encontradas en el nodo desplegado del fichero “*Creation Tool Node*”.
2. Pulsar clic derecho en *Creation Tool Node*, seleccionar *New Child* y crear un *Small Icon Bundle Image* y un *Large Icon Bundle Image*.
3. Definir las propiedades de ambos.

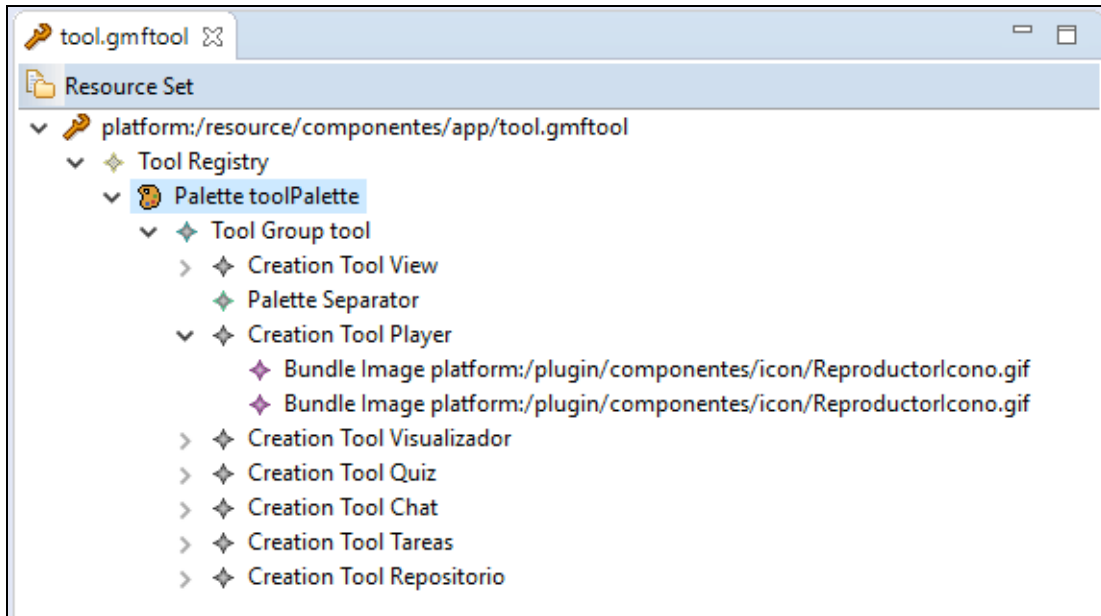


Figura 56. Modelo de definición de herramientas

5.2.4.3. Definición de relación entre elementos

Esta etapa es llevada a cabo a través de un proceso denominado “Mapping”, el cual consiste en crear un archivo para relacionar cada uno de los elementos. Esta relación está dada entre el modelo de dominio, el modelo de definición gráfica y la definición de herramientas. Este se autogenera y crea las conexiones automáticamente, pero debe hacerse una revisión manual ya que no es totalmente exacto.

El proceso de “*Mapping*” es realizado a través de los siguientes pasos:

1. Pulsar clic derecho sobre el metamodelo de extensión *.ecore*, seleccionar *New, Other*, y en la carpeta denominada *Graphical Modeling Framework*, seleccionar el fichero *Guide Mapping Model Creation*.
2. Asignar un nombre al fichero con la extensión “.gmfmap”: Al pulsar *Next* debe especificarse el modelo de dominio como en los anteriores modelos.
3. Seleccionar el *modelo de definición gráfica con extensión “.gmfgrph”* (si no aparece por defecto, pulsar *Find in Workspace* para seleccionarlo). Pulsar *Next*.
4. Seleccionar el *modelo de definición de herramientas con extensión “.gmftool”* (si no aparece por defecto, pulsar *Find in Workspace* para seleccionarlo). Pulsar *Next*.
5. Seguidamente, deben especificarse los nodos y enlaces (ver Figura 57).
6. Una vez realizado el paso anterior, pulsar *Finish*. Al crearse el fichero debe validarse pulsando clic derecho sobre “*Mapping*” y seleccionar la opción *Validate*. Esto es realizado con el fin de comprobar que el proceso este realizado correctamente. Al realizar esto deben visualizarse los ficheros tal cual como lo muestra la Figura 58.

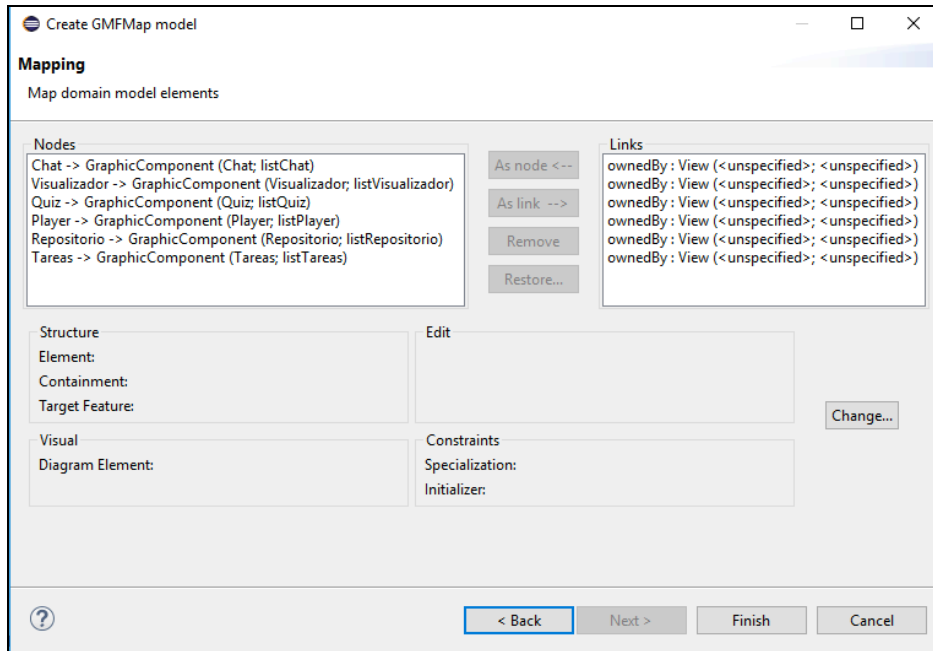


Figura 57. Especificación de nodos y enlaces *Mapping*

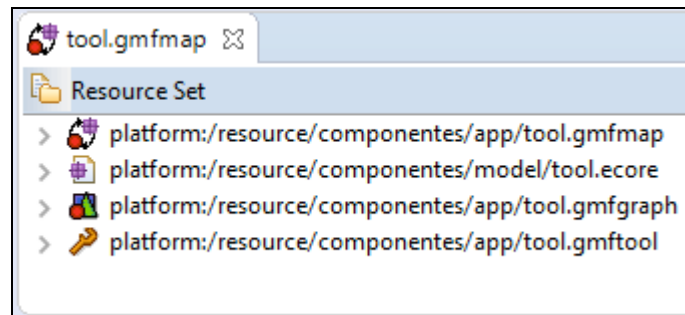


Figura 58. Validación de creación de *Mapping*

Una vez creado el fichero *Mapping*, comienza la creación de toda la relación entre los modelos para cada nodo. Debe crearse cada Nodo en el árbol de despliegue y especificar su relación en sus propiedades tal cual como lo muestra la Figura 59.

Es necesario configurar manualmente cada una de las etiquetas establecidas en el modelo de dominio para cada nodo en específico. Por ejemplo para el caso del reproductor de video, dar clic derecho en *Node mapping*, seleccionar *New, Child, Feature label mapping*. Después en sus propiedades debe seleccionarse el nombre de la etiqueta en el campo *Features* y pulsar *add*. Finalmente pulsar *OK*. Debe cambiarse también en el campo *Diagram label* por la etiqueta que corresponda, en este caso *Diagram Label playerName*. Este proceso debe realizarse similarmente para cada uno de los nodos definiendo sus propias etiquetas. La Figura 60 muestra el árbol final obtenido del proceso para este caso de estudio. En ella, puede apreciarse claramente la especificación de las relaciones entre el modelo de definición gráfica y el modelo de definición de herramientas para cada nodo.

Visual representation	
Appearance Style	
Context Menu	
Diagram Node	◆ Node Player (PlayerFigure)
Tool	◆ Creation Tool Player

Figura 59. Relación de modelos en el Nodo *Mapping*

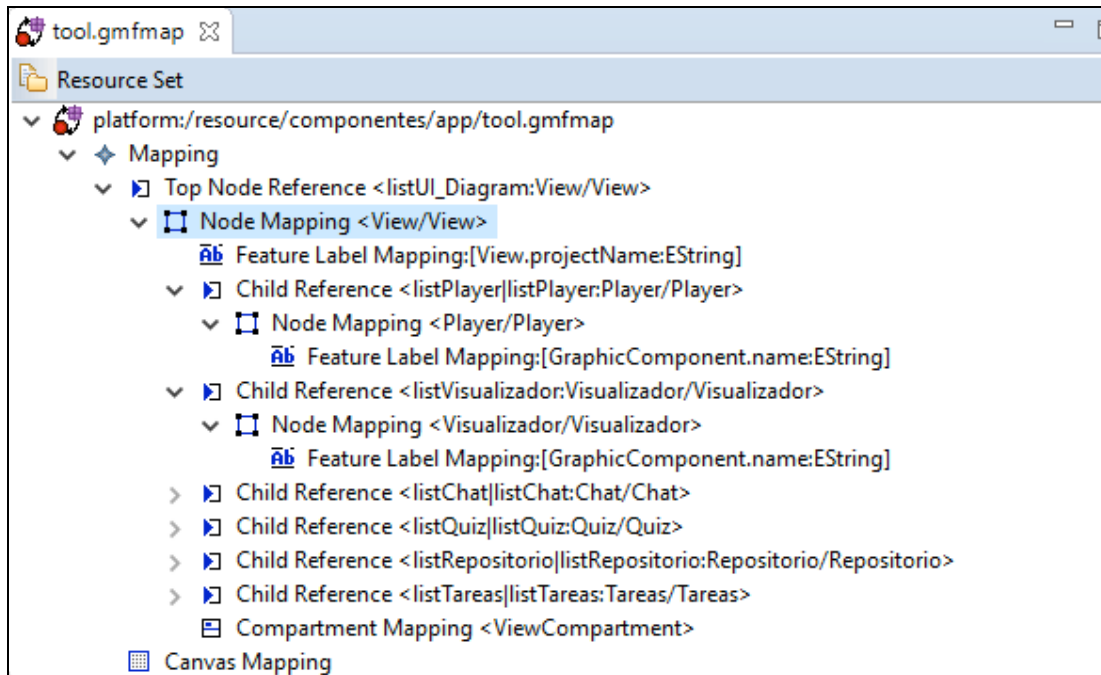


Figura 60. Árbol del modelo de relación de elementos

5.2.5. Generación de código

Al terminar el modelo de relación entre los elementos visto en la sección anterior, es terminada la creación del editor gráfico. Solamente falta la generación del código para poder ejecutar la herramienta. Para esta generación es necesario un generador de código que el *framework* es capaz de crear permitiendo generar varias instancias del editor gráfico. Para la creación de este generador de código es seguido el siguiente proceso:

1. Dar clic derecho sobre el fichero de Mapping y seleccionar *Create generator model*.
2. Asignarle un nombre al generador con la extensión “.gmfgen”.
3. Seleccionar el modelo de relación de elementos con la extensión “.gmfmap”.
4. Seleccionar el modelo de dominio con la extensión “.genmodel”.
5. Pulsar *Finish*.

Al crear el generador de código, el explorador de paquetes es visualizado tal cual como lo muestra la Figura 61.

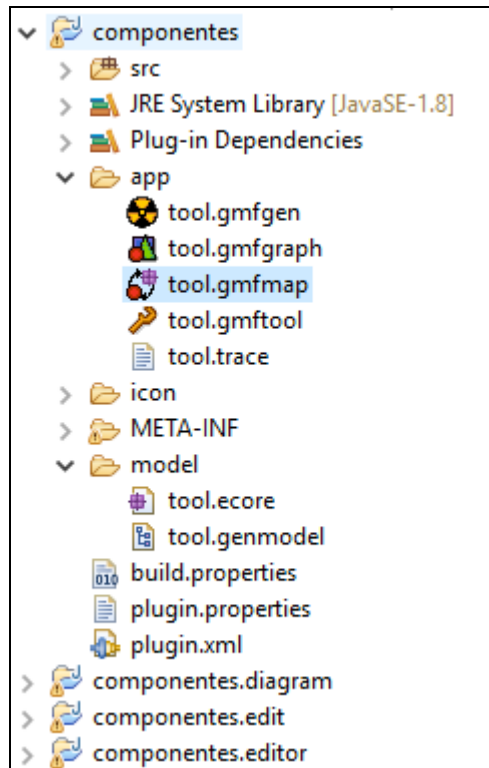


Figura 61. Explorador de paquetes generación de código

Finalmente, es pulsado clic derecho sobre *tool.gmfgen* y es seleccionada la opción *Generate Diagram Code*. Si es generado correctamente aparecerá un mensaje de confirmación, luego genera una carpeta con el nombre *componentes.diagram*.

Una vez creado el editor gráfico para la construcción de las interfaces, la herramienta debe ser capaz de obtener el valor de cada propiedad modificada creando archivos de tipo XML para disposición de los componentes. Este proceso se logra implementando un código Java que permita leer las propiedades del editor y escribirlas en nuevos archivos.

De la misma manera a través de código Java son obtenidos los resultados de la interfaz creada. Son identificados los componentes seleccionados y son creados los archivos correspondientes. El usuario al generar el código final, contará con una carpeta de ejecución en donde son construidos los archivos correspondientes a los componentes involucrados en su interfaz.

Capítulo 6

Validación y Evaluación

El presente capítulo presenta la validación y evaluación del trabajo realizado, esta compuesto de pruebas internas realizadas para comprobar el logro de los objetivos propuestos y además cuenta con sesiones de pruebas externas para evaluar el software tanto en funcionalidad como en usabilidad. A través de este capítulo es buscado medir el alcance del desarrollo software con base a los resultados obtenidos.

Este capítulo está compuesto por dos partes: La primera, abarca la validación tanto de los componentes como del editor gráfico construido. La segunda, abarca la evaluación de los mismos dos aspectos y finalmente es presentada una prueba realizada a un usuario con el fin de evaluar la funcionalidad de los componentes y el editor gráfico en un caso de estudio específico en el contexto educativo creando interfaces sus propias interfaces de usuario para sistemas basados en distribución de contenido de video.

6.1. Validación

La validación consiste en comparar los resultados obtenidos con los objetivos propuestos en cuanto al desarrollo software en un contexto específico. Es decir, en el caso de los componentes es poder comprobar que los componentes cuenten con las funcionalidades propuestas. En el caso del editor gráfico es comprobar su funcionamiento en cada una de las etapas del proceso de creación de una interfaz de usuario en el contexto de los sistemas basados en distribución de contenido de video para educación.

Para llevar a cabo este proceso de validación, son realizadas pruebas utilizando una plantilla presentada por la metodología SCRUM para validación de software. Las pruebas consisten en una serie de reportes en donde son validadas las funcionalidades del trabajo realizado y seguidamente son representados en una tabla. El contenido de cada columna de dicha tabla está presentado a continuación:

- Título: título del desarrollo a validar
- N° de Prueba: numero de prueba realizada (identificación).
- Fecha: fecha en la que es realizada la prueba.
- Acción: acción a través de la cual es validada la funcionalidad.
- Resultados esperados: funcionalidades esperadas según los requerimientos.
- Resultados actuales: resultados obtenidos al finalizar la prueba
- Referencia: visualización de la prueba realizada
- Nivel de aprobación: aprobado/no aprobado.

En las siguientes subsecciones son presentados los resultados obtenidos en las validaciones a través de las tablas correspondientes.

6.1.1. Validación de los componentes

Teniendo en cuenta la información brindada en la sesión anterior, es realizada la validación de cada componente. Las Tabla 21 y Tabla 22 muestran las validaciones del componente reproductor de video y visualizador de documentos respectivamente y en el Anexo B se pueden apreciar las validaciones del resto de componentes.

Puede notarse que las tablas de validación en general, muestran resultados obtenidos favorables con respecto a los objetivos planteados. El proceso de validación ha permitido también realizar algunos ajustes en las funcionalidades mejorando así el desarrollo de cada componente. A continuación, la siguiente sección presenta, de la misma forma, la validación del editor gráfico construido para el diseño de las interfaces gráficas.

6.1.2. Validación del editor gráfico

La validación del editor gráfico y de la herramienta en general puede reflejarse en la Tabla 23. Esta validación consta de algunas pruebas realizadas siguiendo la plantilla de validación presentada anteriormente. Puede evidenciarse que los objetivos propuestos han sido alcanzados satisfactoriamente y ha permitido disponer la herramienta para someterse a pruebas de evaluación con usuarios finales, lo cual está presentado en la siguiente sección.

Tabla 21. Validación componente de reproductor de video

Componente: Reproductor de video							
N° de prueba	Fecha	Funcionalidades Propuestas	Acción	Resultados Esperados	Resultados Actuales	Referencia	Nivel de aprobación
1	16/05/17	Funcionalidades básicas	<ul style="list-style-type: none"> • Reproducir/Pausar • Elegir momento de reproducción • Silenciar • Subir/ Bajar Volumen • Activar/Desactivar Repetición • Ajustar a pantalla completa 	Visualizar funcionalidades básicas sin generación de errores ni alteraciones en el componente	El componente le permite al usuario realizar las funcionalidades básicas	https://youtu.be/PD0aaEJJMco	100%
2	16/05/17	Avanzar y Retroceder	Avanzar y retroceder la reproducción en 10 Segundos	Alteración en tiempo de reproducción	Los botones permiten avanzar y retroceder el tiempo estipulado		

3	16/05/17	Mostrar transcripción de texto	Clic en el botón de transcripción	Visualizar tiempo y texto correspondiente sin alteraciones en el componente	El texto correspondiente al audio del video es visualizado para cada lapso de tiempo		
4	16/05/17	Gestionar subtítulos	Realizar prueba con Español e Inglés	Visualizar subtítulos en pantalla	Los subtítulos correspondientes al idioma seleccionado es percibido en pantalla		
5	16/05/17	Gestionar Velocidad de reproducción	Realizar prueba para lento y rápido	Alteraciones en la velocidad de reproducción	Es alterada la velocidad de reproducción		
6		Ajustar calidad de video	Realizar prueba con dos calidades diferentes	La imagen del video debería cambiar de calidad	El video es auto ajustado a la calidad seleccionada		

Tabla 22. Validación componentes de visualizador de documentos

Componente: Visualizador de documentos							
N° de prueba	Fecha	Funcionalidades Propuestas	Acción	Resultados Esperados	Resultados Actuales	Referencia	Nivel de aprobación
1	16/05/17	Navegación entre páginas	Clic en los botones siguiente y atrás	Redirección automática a la página siguiente o a la anterior según el botón presionado	Los botones permiten avanzar y retroceder las páginas del documento	https://youtu.be/MZ8ccGHWQ4E	100%

2	16/05/17	Probar tanto para documento con orientación vertical como horizontal	Cambiar configuración de la orientación	Visualizar el documento de forma vertical u horizontal según su configuración	El visualizador de documentos cambia la orientación según su configuración		
---	----------	--	---	---	--	--	--

Tabla 23. Validación editor gráfico

Título: Editor gráfico para diseño de interfaces de usuario							
N° de prueba	Fecha	Funcionalidades Propuestas	Acción	Resultados Esperados	Resultados Actuales	Referencia	Nivel de aprobación
1	16/05/17	Ubicación en la interfaz de componentes seleccionados	Arrastrar componentes en el editor gráfico	Los componentes pueden ubicarse en el panel de diseño obteniendo en la interfaz final la misma ubicación	Al seleccionar uno de los iconos correspondiente a un componente en la paleta de herramientas fue posible ubicarlo en cualquier posición del panel principal y además es conservada dicha posición en la interfaz obtenida	https://youtu.be/wWm5QtWHz3E	100%
2	16/05/17	Configuración de propiedades de componentes	Configurar alguna propiedad de los componentes	Las propiedades especificadas deben reflejarse en el resultado de la interfaz final	Al alterar una propiedad en la configuración es reflejado el cambio en la interfaz de usuario final obtenida.		

3	16/05/17	Sincronizar componentes	Sincronizar el componente de contenido con un reproductor de video y un Visualizador de documentos	Al hacer clic en uno de los enlaces del contenido debería de sincronizarse al tiempo y página del reproductor de video y del visualizador respectivamente	El contenido tanto del reproductor de video como del visualizador de componentes son sincronizados al seleccionar uno de los ítems de la tablas de contenido		
---	----------	-------------------------	--	---	--	--	--

6.2. Evaluación

El proceso de evaluación gira en torno a la calidad del software, la cual es medida a través de indicadores de calidad. Los indicadores utilizados en este trabajo son dirigidos bajo los estándares de calidad sugeridos en la norma ISO/IEC-9126 (Losavio, Chirinos, Matteo, Lévy, & Ramdane-Cherif, 2004) de la ISO⁸ y la IEC⁹. Este estándar mide el nivel de calidad del software en diferentes aspectos de evaluación, identificando en el desarrollo unas características, las cuales están descritas en la Tabla 24. Para este trabajo de grado es considerado únicamente dos características: Funcionalidad y Usabilidad, pues se considera que son las necesarias para evaluar el trabajo desarrollado.

Tabla 24. Características de calidad de software norma ISO/IEC 9126

Característica	Sub característica	Explicación
Funcionalidad	Adecuación	Presencia y aptitud de un conjunto de funciones para tareas especificadas
	Exactitud	Disposición de resultados o efectos correctos o acordados
	Seguridad	Habilidad para prevenir acceso no autorizado ya sea accidental o deliberado, a programas y datos
	Interoperabilidad	Habilidad para la interacción con sistemas especificados
Usabilidad	Entendimiento	Esfuerzo de los usuarios para reconocer sus aplicaciones y el concepto lógico, para la operación y el control del software
	Aprendizaje	
	Operabilidad	
	Atracción	

Para medir el nivel alcanzado de las características mencionadas en el párrafo anterior, es posible utilizar distintas formas de evaluación, entre ellas, las pruebas de sesiones guiadas, métodos de seguimiento y protocolos de pensamiento manifestado (“pensar en voz alta”). Otra forma de evaluación es aplicar pruebas no empíricas (Standardization & Commission, 2001). Por ejemplo, para evaluar si una interfaz de usuario es fácil de entender debería ser inspeccionada por un experto en desarrollo de interfaces de usuario. Para este trabajo son realizadas sesiones guiadas de pruebas a usuarios para la creación de actividades en las interfaces. Las sesiones de prueba realizadas para la evaluación de los componentes desarrollados, se presentan en la siguiente sección.

⁸ Organización Internacional de Normalización

⁹ Comisión Electrotécnica Internacional

6.2.1. Prueba de evaluación aplicada

Las pruebas a través de sesiones guiadas a los usuarios evalúan dos aspectos: primero, la funcionalidad de los componentes, es decir que los componentes sean capaces de realizar funciones requeridas por los usuarios. Y segundo la usabilidad, es decir, que los usuarios puedan familiarizarse con la interfaz de cada componente.

Primeramente, el número seleccionado de usuarios para realizar la prueba es de 10 usuarios. Cada usuario debe estar relacionado con conocimientos de desarrollo software y tener experiencia docente en alguna institución. Este criterio va a permitir una mejor adaptación entre el usuario y el desarrollo, es decir, el usuario tiene la oportunidad de familiarizarse más fácil con los contextos específicos del software.

La prueba realizada consiste primeramente en una interacción de los usuarios con los componentes. Esta primera parte consiste en que los usuarios identifiquen cada componente y exploren las funcionalidades dispuestas. A continuación se presentan las tareas asignadas a los usuarios:

1. Explorar cada uno de los componentes e interactuar con las funcionalidades correspondientes.
2. Abrir el editor DSL y crear una interfaz a su gusto para sistemas basados en distribución de contenido de video en el contexto de educación.
3. Analizar los resultados obtenidos para su interfaz de usuario.

Seguidamente, es aplicada una encuesta, la cual consiste en evaluar en una determinada escala diferentes ítems de evaluación. Los ítems de evaluación están inmersos a 6 dimensiones determinadas por estudios de usabilidad y funcionalidad que buscan brindar a los desarrolladores las formas más factibles y eficientes para evaluar sus proyectos. A continuación se presentan dichas dimensiones:

- Atracción: impresión general del producto. ¿A los usuarios les gusta o disgusta el producto?
- Transparencia o claridad: ¿Es fácil entender cómo usar el producto?, ¿Es fácil familiarizarse con el producto?
- Eficiencia: ¿Es posible usar el producto fácil y eficientemente?
- Confianza: ¿El usuario se siente en capacidad de controlar la interacción?, ¿Se siente el usuario seguro al interactuar con el producto?
- Estimulación: ¿Es interesante y emocionante usar el producto?, ¿Se siente motivado a seguir usando el producto?
- Novedad: ¿El diseño del producto es creativo e innovador?

A partir de las dimensiones descritas anteriormente, son derivados en total 24 ítems de evaluación presentados en la Figura 62. Para cada ítem seleccionado los usuarios deberán elegir en una escala de 1 a 7 la favorabilidad de cada uno y antes de responder la encuesta de evaluación es necesario comunicar a los usuarios las siguientes recomendaciones:

- Decidir la manera más espontánea posible. Es importante que usted no piense en los plazos largos, por lo que su evaluación inmediata entra en juego.
- Por favor, compruebe siempre una respuesta, incluso si no está seguro a un par de términos.
- No hay “correcto” o “incorrecto” como respuesta. Su opinión cuenta personal.
- Dé su evaluación actual del producto.
- Marque sólo un círculo por línea.

Una vez comunicadas las recomendaciones es realizada la encuesta cuyos resultados obtenidos y análisis de los mismos son presentados en las secciones siguientes. La encuesta va dirigida a la experiencia de usuario tanto de los componentes como del editor gráfico con el cual construyó la interfaz.

	1	2	3	4	5	6	7		
desagradable	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	agradable	1
no entendible	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	entendible	2
creativo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	sin imaginación	3
fácil de aprender	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	difícil de aprender	4
valioso	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	de poco valor	5
aburrido	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	emocionante	6
no interesante	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	interesante	7
impredecible	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	predecible	8
rápido	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	lento	9
original	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	convencional	10
obstructivo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	impulsor de apoyo	11
bueno	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	malo	12
complicado	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	fácil	13
repeler	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	atraer	14
convencional	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	novedoso	15
incómodo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	cómodo	16
seguro	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	inseguro	17
activante	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	adormecedor	18
cubre expectativas	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	no cubre expectativas	19
ineficiente	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	eficiente	20
claro	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	confuso	21
no pragmático	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	pragmático	22
ordenado	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	sobrecargado	23
atractivo	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	feo	24

Figura 62. Aspectos de evaluación de software

6.2.2. Resultados de la evaluación

Para obtener los mejores resultados minimizando las tendencias de respuestas son organizados los diferentes ítems de manera aleatoria en las encuestas. Los resultados obtenidos serán presentados a través de estadísticas brindadas la herramienta UEQ Analysis Tool (Rauschenberger, Olschner, Cota, Schrepp, & Thomaschewski, 2012) . Los cálculos a realizar con los resultados obtenidos son la media aritmética y la desviación estándar. Con el cálculo de la media aritmética es obtenido un promedio entre los valores seleccionados, esto permite percibir una tendencia para cada una de las dimensiones o ítems más específicamente y la desviación estándar permite verificar si el número de cantidad de respuestas es válida. Es

decir, cuanto menor sea la desviación estándar de las respuestas, menos datos son necesitados para obtener resultados confiables.

En esta sección es presentada la media aritmética de cada dimensión además de la media aritmética y desviación estándar por ítem.

6.2.2.1. Media de dimensiones calculada y análisis.

La Figura 63 presenta la media aritmética de las dimensiones. Se puede percibir que todas muestran un resultado de favorabilidad, lo cual indica que, en un rango de -3 a 3, la media oscila entre 1.5 y 2. Este resultado se puede interpretar más claramente en la gráfica que presenta la Figura 64, en ella se puede observar los resultados obtenidos a través de un diagrama de barras.

UEQ Scales	
Atracción	↑ 1,740
Transparencia	↑ 1,500
Eficiencia	↑ 1,550
Controlabilidad	↑ 1,500
Estimulación	↑ 1,875
Novedad	↑ 1,967

Figura 63. Media aritmética de dimensiones de evaluación

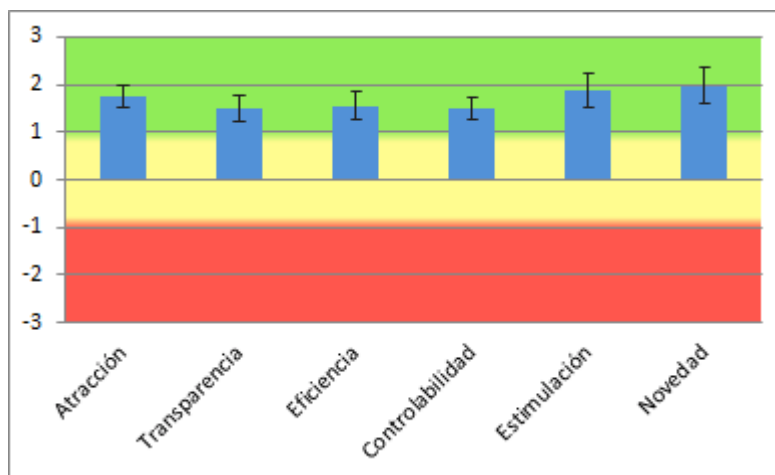


Figura 64. Diagrama de barras de resultados de evaluación de dimensiones

Estos resultados a nivel de dimensiones presentan una visión general del software desarrollado, la novedad es la dimensión con el promedio más alto. Es decir, que los ítems derivados de esta, recibieron el puntaje de favorabilidad más alto. Esto indica que en términos de funcionalidad, el software realiza sus acciones de una manera más novedosa en comparación con otras experiencias. Otra de las dimensiones mejor evaluadas fue la estimulación, lo cual hace que los usuarios sientan motivación a seguir utilizando. Esto significa que el software generó interés y emoción en las experiencias con los usuarios.

Las dimensiones con el promedio más bajo son la transparencia y la confianza (controlabilidad), que aunque no tienen un índice de desfavorable, pueden mejorar. Una de las causas por lo cual son obtenidos dichos resultados puede ser que debido a las diferentes

tecnologías que existen, no le es fácil a los usuarios manipular claramente el software. Otra causa pudo haber sido la claridad de la notación gráfica o el aspecto visual con el que cuenta tanto los componentes como el editor DSL. Una solución viable puede ser estructurar un mejor manual de usuario con el que los usuarios se guíen antes y durante la ejecución del software.

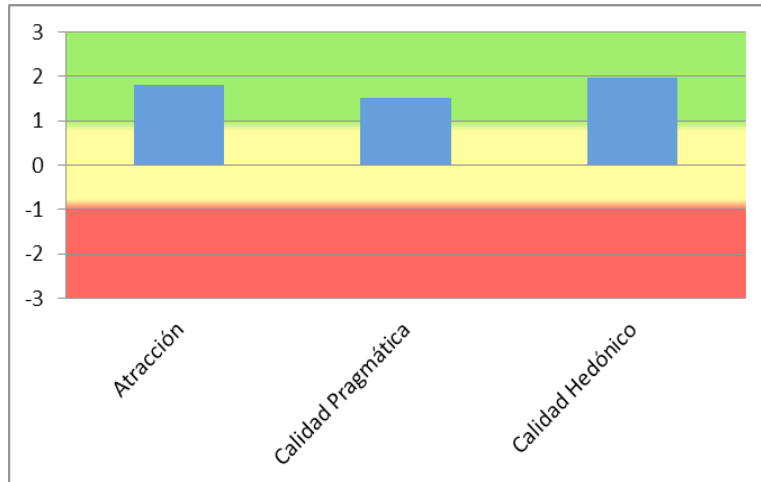


Figura 65. Media aritmerica para los tres aspectos de calidad pragmática y calidad hedónica

Los ítems de la encuesta también se pueden agrupar en calidad pragmática (Perspicuidad, Eficiencia, Confiabilidad) y calidad hedónica (Estimulación, Originalidad). La calidad pragmática describe los aspectos de calidad relacionados con la tarea y la calidad hedónica los aspectos de calidad que no están relacionados con la tarea. La Figura 65 presenta la media para los tres aspectos de calidad pragmática y hedónica, los aspectos de calidad hedónica y atracción tiene los puntajes más altos, es decir que los usuarios encontraron mayores beneficios a nivel de aspectos que no estaban relacionados con la tarea que con los que estaban relacionados. Lo anterior puede ser consecuencia de una poca capacitación en el uso de la herramienta antes de realizar las tareas asignadas.

6.2.2.2. Media y desviación estándar de ítems calculada y análisis.

La **¡Error! No se encuentra el origen de la referencia.** presenta la media y desviación estándar de los resultados obtenidos en la encuesta realizada a los usuarios sometidos a la prueba. La columna identificada de la media representa una escala en el rango de -3 a +3 e indica, como es mencionada anteriormente, un promedio en la favorabilidad de cada ítem. Los valores entre -0.8 y 0.8 representa una evaluación neutral, los mayores a 0.8 representa una evaluación positiva y los menores a 0.8 una negativa. Puede notarse que la media aritmética de los ítems muestra un resultado de favorabilidad, salvo el ítem de seguridad que muestra un resultado de neutralidad. De esta manera se puede comprobar que el desarrollo software realizado responde favorablemente ante pruebas de usabilidad y funcionalidad con usuarios.

Item	Mean	Variance	Std. Dev.
1	↑ 1,6	0,9	1,0
2	↑ 1,5	1,2	1,1
3	↑ 2,1	0,8	0,9
4	↑ 1,8	0,8	0,9
5	↑ 2,0	0,7	0,8
6	↑ 1,5	1,2	1,1
7	↑ 2,0	0,7	0,8
8	↑ 1,5	1,2	1,1
9	↑ 1,0	0,7	0,8
10	↑ 2,0	0,7	0,8
11	↑ 1,9	0,5	0,7
12	↑ 2,3	0,7	0,8
13	↑ 0,9	1,0	1,0
14	↑ 0,9	0,5	0,7
15	↑ 1,8	0,6	0,8
16	↑ 2,0	0,7	0,8
17	⇒ 0,6	0,5	0,7
18	↑ 2,0	0,9	0,9
19	↑ 2,0	0,7	0,8
20	↑ 1,6	0,7	0,8
21	↑ 1,8	0,6	0,8
22	↑ 1,5	0,7	0,8
23	↑ 2,1	0,5	0,7
24	↑ 1,9	0,5	0,7

Figura 66. Media y desviación estándar de evaluación de componentes

La varianza que también está presente en los resultados ayuda a intuir que el resultado individual de cada ítem, es acercado al promedio de los datos y la probabilidad de que esto cambie es poca. La interpretación de estas medidas de dispersión ayuda a los desarrolladores a tener una idea a la tendencia de su desarrollo en los diferentes aspectos impulsándolos a la corrección de errores y la perfección de su desarrollo. Es claro que los aspectos en los cuales es motivado a trabajar mejoras es el de seguridad, complejidad y atracción.

Una vez presentados los resultados de la evaluación es presentado un caso en particular mostrando la interfaz de usuario creada y la experiencia de usuario ante el desarrollo software. Dicho caso de estudio se presenta en la siguiente sección.

6.2.3. Caso de estudio en el contexto educativo

Para el presente caso de estudio se ha seleccionado uno de los usuarios sometidos a la prueba presentada en la sección anterior para evaluar el funcionamiento de los componentes y del editor DSL creando una interfaz de usuario para sistemas basados en distribución de contenido de video en el contexto de educación.

Las tareas asignadas al usuario durante el caso de estudio son exactamente las mismas presentadas en la sección anterior, que básicamente se basa en la interacción y exploración tanto con los componentes como con el editor DSL. A continuación es presentada la experiencia de usuario y la interfaz construida.

1. Resultado de evaluación:

La Tabla 25 muestra los resultados obtenidos individualmente para el caso de estudio en ejecución. Este resultado muestra favorabilidad en cada ítem evaluado y su promedio está por encima del promedio general.

Tabla 25. Resultado de evaluación caso de estudio.

Ítems																							
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
6	6	3	2	2	7	5	6	2	2	6	1	5	5	7	7	3	1	2	7	2	6	1	2

2. Selección de componentes:

La interfaz de usuario que el usuario ha decidido crear es basado en una clase magistral, por lo que ha seleccionado los componentes de reproductor de video como herramienta de difusión de información, un visualizador de documentos para la bibliografía que tiene a disposición y una tabla de contenido para llevar el control de su clase.

3. Ubicación de los componentes:

El usuario ha decidido ubicar los componentes como lo muestra la Figura 67, dándole prioridad en la esquina superior izquierda al reproductor de video, a su lado el visualizador de documentos y en la parte inferior se encuentra ubicada la tabla de contenido.

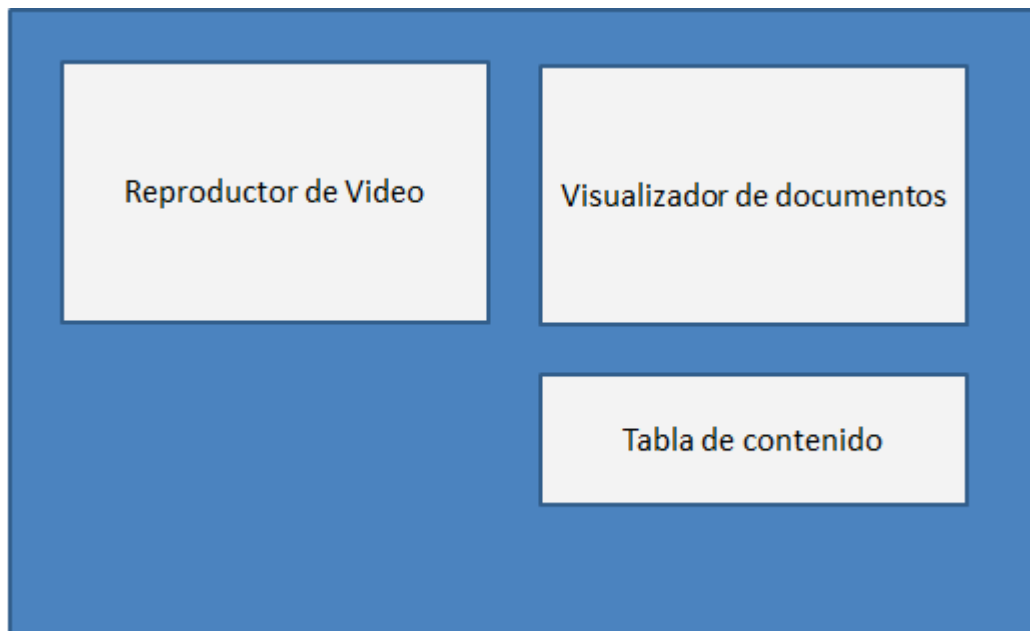


Figura 67. Ubicación de los componentes en caso de estudio

4. Configuración de propiedades:

El usuario ha habilitado todas las funcionalidades de todos los componentes pues las considera importantes para su clase y ha elegido un estilo de colores igual para todos los componentes para darle una buena apariencia a la interfaz.

5. Interfaz de usuario obtenida:

La Figura 68 Presenta la interfaz final obtenida, la cual está compuesta de los tres componentes seleccionados. Con este resultado el usuario obtiene los resultados esperados y tiene a disposición la interfaz para configurar sus videos, documentos y demás recursos a utilizar para su clase.

Teorema de pitagoras



00:00 / 02:36

Teorema de Pitágoras

PREVIOUS NEXT Page: 1 / 5

Tabla de Contenido

- [Tema1: Identidades fundamentales](#)
- [Tema2: Identidad pitagórica](#)
- [Tema3: Otras identidades](#)
- [Tema4: Resumen](#)

Figura 68. Interfaz final caso de estudio.

Capítulo 7

Conclusiones y Trabajos Futuros

El presente capítulo expone las principales conclusiones obtenidas del desarrollo de este trabajo de grado, las lecciones aprendidas tanto en investigación como en desarrollo, recomendaciones y finalmente algunas ideas de investigación y desarrollo para futuros trabajos.

A partir de la propuesta de construcción de interfaces de usuario para sistemas basados en distribución de contenido de video es realizada una investigación previa, la cual aportó la base para la selección de componentes a desarrollar y la construcción de una herramienta dirigida por modelos.

El entorno de exploración seleccionado para determinar los componentes a desarrollar fue el de educación, pues este trabajo de grado quiso hacer énfasis en los nuevos retos presentes que hay en las herramientas utilizadas en la educación virtual.

Considerando que el criterio de investigación fue buscar los componentes más representativos en las diferentes herramientas exploradas, se seleccionaron los siguientes componentes a desarrollar: Reproductor de video, Visualizador de documentos, Quiz, Tareas, Chat, Recursos relacionados y Tabla de contenido.

Se seleccionó la tecnología Eclipse para la implementación de un editor gráfico aplicando el desarrollo dirigido por modelo para implementar cada uno de los componentes considerando la modificación de algunas propiedades de cada uno de ellos.

Finalmente se realizaron pruebas correspondientes a la validación de cada una de las funcionalidades del desarrollo realizado y seguidamente fue sometido a una serie de pruebas enfocadas en evaluar en los aspectos de usabilidad y funcionalidad. En dicha evaluación fue obtenido un resultado con alto nivel de favorabilidad tanto de los componentes desarrollados como del editor gráfico.

7.1. Conclusiones generales

En conclusión, el trabajo realizado es caracterizado como una propuesta tecnológica para el desarrollo de la interfaz de usuario de sistemas interactivos basados en distribución de contenido de video en el contexto educativo a partir de un DSL, el cual constituye una aproximación hacia la construcción de un framework basado en modelos para el mismo fin.

El proceso metodológico empleado para la construcción de la herramienta es uno de los aportes principales del trabajo de investigación debido a que puede ser aplicado en la construcción de otros frameworks basados en modelos o para ampliar los componentes de los ya existentes. Igualmente, la herramienta construida pretende dar soporte a los desarrolladores

en el diseño de la interfaz de usuario de este tipo de sistemas de tal manera que su trabajo sea más productivo.

El proceso de validación estuvo enfocado en evaluar aspectos de funcionalidad y experiencia de usuario tanto de los componentes como del editor DSL a través de artefactos provistos por la metodología SCRUM y de la ejecución de test con usuarios en el contexto de tareas específicas relacionadas al diseño de interfaces para escenarios educativos. Los resultados mostraron que para todas las dimensiones evaluadas fueron obtenidos puntajes favorables tanto en el análisis individual como en el de comparación en relación a otros productos. Entre estos son resaltadas las dimensiones de novedad y estimulación, las cuales recibieron los puntajes más altos.

7.2. Lecciones aprendidas y recomendaciones

Del desarrollo de este trabajo de grado es importante considerar las siguientes lecciones aprendidas y recomendaciones identificadas según algunos enfoques investigativos presentadas a continuación:

1. Según los contextos de investigación.

Las interfaces de usuario con componentes basados en distribución de contenido de video pueden presentarse en diferentes contextos de investigación y cada vez son más los desarrollos para nutrir dichas interfaces. Es por eso, que la exploración de los componentes fue una tarea dispendiosa, pues necesariamente hay que identificar todo tipo de componentes que hoy componen las interfaces que le dan prioridad al contenido de video.

La elección del contexto de la educación fue un reto investigativo, fue el contexto en donde más herramientas y aplicaciones se identificaron. Esto nos permitió descubrir el gran avance tecnológico de este contexto en los últimos tiempos.

2. Según el desarrollo software de los componentes.

El desarrollo de los componentes de interfaz de usuario, pueden representarse en un diagrama de clases con los atributos modificables o no modificables y como métodos las funcionalidades de cada componente. También pueden existir relaciones entre clase en el caso de que es relacionado un componente con otro.

Los metadatos que contienen la información modificable, pueden representarse a través de archivos de tipo XML, json, etc. Esto permitió el fácil acceso por parte de los usuarios a aquellos datos que usarán para los componentes de sus propias interfaces.

También la utilización de herramientas para llevar un control de versiones como Git, agilizan el desarrollo del código y protegen la información en caso de pérdida.

Otra de las experiencias aprendidas fue que la utilización de Frameworks los cuales facilitan el desarrollo y aumentan el nivel de calidad del mismo. Fueron obtenidos más conocimientos de aquellos frameworks que ayudan a mejorar la vista de la interfaz, que ayudan a reutilizar código para las funciones y aquellos que nutren los editores de código para realizar una implementación más ágil.

3. Según el desarrollo del Editor DSL

Un desarrollo dirigido por modelos permite realizar un trabajo más productivo en tiempo y organización. Es notable el alto nivel de producción de software y la reducción de tiempo de

desarrollo. Esto nos deja como experiencia la motivación a seguir apuntando a la ingeniería dirigida por modelos para proyectos de desarrollo software.

El uso de librerías para la gestión gráfica en la construcción de editores DSL como GEF, permitió el desarrollo software de una manera amigable al desarrollador, agilizando el proceso de creación del editor.

4. Según las pruebas realizadas.

Al realizar las pruebas de experiencia de usuario, encontramos interés por parte de los docentes a abrirse a la autonomía en el desarrollo de las interfaces de usuario. Identificamos que era necesario una mejor capacitación previa a la interacción con el prototipo, porque es muy poca la experiencia de los docentes en la interacción con la herramienta.

7.3. Trabajos futuros

A partir de la experiencia adquirida en este trabajo de grado, son sugeridos los siguientes trabajos futuros:

- Realizar el desarrollo de nuevos componentes de interfaz de usuario enmarcados en el mismo contexto de educación e integrarlos al framework basado en modelos construido.
- Explorar otros contextos de investigación como entretenimiento, comunicación, etc. Para que de esta manera sea posible avanzar a un nivel más global la construcción de interfaces de usuario de parte de los desarrolladores un contexto en específico.
- A partir de los componentes desarrollados y haciendo un análisis de la gestión de los componentes en el framework basado en modelos desarrollado, crear la gestión de actividades en el mismo entorno de educación haciendo uso de la combinación de dichos componentes.
- Extender el framework dentro de un servidor para mejorar la gestión y reducir los tiempos de configuración de la herramienta y para tener una generación de código más ágil y sin complicaciones.
- Con los componentes desarrollados crear la gestión de funcionalidad y de interfaz de usuario en paralelo dentro del framework para brindarle al usuario final un ambiente completo para la generación de su aplicación.
- Enriquecer la interfaz generada a partir de la integración de otros modelos como modelos de interacción, modelos de usuario o de contexto, con el fin de promover la usabilidad en la interfaz final
- Aumentar la usabilidad de la interfaz de usuario del editor DSL para obtener una comunicación más amigable con el desarrollador.
- Diseñar una metodología que promueva la experiencia de usuario y que sea apoyada en este tipo de herramientas con el fin de acortar la brecha existente entre los requisitos y el producto final.

Bibliografía

- (OUA), O. U. A. (2013). Open2Study: Free Online Courses For Everyone, from <https://www.open2study.com/>
- Akiki, P. A., Bandara, A. K., & Yu, Y. (2013). *Cedar studio: an IDE supporting adaptive model-driven user interfaces for enterprise applications*. Paper presented at the Proceedings of the 5th ACM SIGCHI symposium on Engineering interactive computing systems, London, United Kingdom.
- Alderson, A. (1991). Meta-CASE technology *Software Development Environments and CASE Technology* (pp. 81-91): Springer.
- Amelunxen, C., Königs, A., Rötschke, T., & Schürr, A. (2006). *MOFLON: a standard-compliant metamodeling framework with graph transformations*. Paper presented at the Model Driven Architecture–Foundations and Applications.
- Anderruthy, J.-N. (2007). *Skype y telefonía IP: Llama gratis por Internet*: Ediciones ENI.
- Area, M., & Adell, J. (2009). E-learning: enseñar y aprender en espacios virtuales. *J. De Pablos*.
- Bergh, J. B., & Coninx, K. (2014) CASSIS: A Modeling Language for Customizable User Interface Designs. *Vol. 8742. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (pp. 243-250).
- Bergin, J. (2007). Building graphical user interfaces with the mvc pattern. *New York*.
- Besana, S. (2012). Schoology: the learning management system goes" social". *TD Tecnologie didattiche*, 20(1), 51-53.
- Blumendorf, M., Feuerstack, S., & Albayrak, S. (2008). *Multimodal user interfaces for smart environments: the multi-access service platform*. Paper presented at the Proceedings of the working conference on Advanced visual interfaces.
- Boneu, J. M. (2007). Plataformas abiertas de e-learning para el soporte de contenidos educativos abiertos. *RUSC. Universities and Knowledge Society Journal*, 4(1).
- Brambilla, M., Mauri, A., & Umuhoza, E. (2014). Extending the interaction flow modeling language (IFML) for model driven development of mobile applications front end *Mobile Web Information Systems* (pp. 176-191): Springer.
- Burmester, S., Giese, H., Niere, J., Tichy, M., Wadsack, J. P., Wagner, R., . . . Zündorf, A. (2004). Tool integration at the meta-model level: the Fujaba approach. *International journal on software tools for technology transfer*, 6(3), 203-218.
- Cabero-Almenara, J. (2006). Bases pedagógicas del e-learning. *RUSC. Universities and Knowledge Society Journal*, 3(1), 1.
- Calvary, G., & Coutaz, J. (2014). Introduction to Model-Based User Interfaces. *W3C Group Note NOTE-mbui-intro-20140107*.
- Calvary, G., Coutaz, J., Bouillon, L., Florins, M., Limbourg, Q., Marucci, L., . . . Thevenin, D. (2002). The CAMELEON reference framework. *Deliverable D1, 1*.
- Cantone, D. (2006). *Implementación y debugging: USERSHOP*.

- Clerckx, T., Luyten, K., & Coninx, K. (2004). DynaMo-AID: a design process and a runtime architecture for dynamic model-based user interface development *Engineering Human Computer Interaction and Interactive Systems* (pp. 77-95): Springer.
- Cook, S., Jones, G., Kent, S., & Wills, A. C. (2007). *Domain-specific development with visual studio dsl tools*: Pearson Education.
- Corel. (2017). CorelDRAW Graphics Suite X8 (Version X8). Retrieved from www.coreldraw.com
- Deacon, J. (2009). Model-view-controller (mvc) architecture. *Online*[Citado em: 10 de março de 2006.] <http://www.jdl.co.uk/briefings/MVC.pdf>.
- Demeure, A., Calvary, G., & Coninx, K. (2008). COMET (s), a software architecture style and an interactors toolkit for plastic user interfaces *Interactive Systems. Design, Specification, and Verification* (pp. 225-237): Springer.
- Dergarabedian, C. (2016). La Internet del futuro soportará videos 4K, llamadas de voz en alta definición y hasta hologramas.
- Dieterich, H., Malinowski, U., Kühme, T., & Schneider-Hufschmidt, M. (1993). State of the art in adaptive user interfaces. *Human factors in information technology*, 10, 13-13.
- Dourgnon-Hanoune, A., Dang, T., Dissert, B., & Reguigui, S. (2006). A Framework for I&C KM using a CMS with Ontology extension. *IEEE/INCOM*.
- Eisenstein, J., Vanderdonckt, J., & Puerta, A. (2001). *Applying model-based techniques to the development of UIs for mobile computers*. Paper presented at the Proceedings of the 6th international conference on Intelligent user interfaces.
- Facebook apuesta por ser el centro del streaming de video casero (2016, 12/04/2016). *La Nación*. Retrieved from <http://www.lanacion.com.ar/1888634-facebook-apuesta-por-ser-el-centro-del-streaming-de-video-casero>
- France, R., & Rumpe, B. (2007). *Model-driven development of complex software: A research roadmap*. Paper presented at the 2007 Future of Software Engineering.
- Gajos, K. Z., Weld, D. S., & Wobbrock, J. O. (2010). Automatically generating personalized user interfaces with Supple. *Artificial Intelligence*, 174(12-13), 910-950. doi: <http://dx.doi.org/10.1016/j.artint.2010.05.005>
- Giraldo, W. J. (2010). *Marco de Desarrollo de Sistemas Groupware Interactivos Basado en la Integración de Procesos y Notaciones - CIAF*. PhD Doctoral, Universidad de Castilla - La Mancha.
- Giraldo, W. J., Collazos, C. A., & Giraldo, F. D. (2009). Desarrollo basado en modelos de la interfaz de usuario de sistemas groupware. *Avances en Sistemas e Informática*, 6(2), 197-204.
- Global, O. L. (2013). OpenLearning: Teach and learn online for free, from <https://www.openlearning.com/>
- GmbH, O. U. T., & Becker, B. (2007). *Manual eGroupWare 1.4*: Outdoor Unlimited Training GmbH.
- Goldberg, A. J. (1984). SMALLTALK-80: the interactive programming environment.
- Gómez, J. M. M., Marín, M. E. H., & Díaz, E. A. (2014). Enfoque metodológico para el diseño de interfaces durante el ciclo de vida de desarrollo de software. *REVISTA GTI*, 12(34), 59-73.
- Group, P. ¿Qué puede hacer PHP? , from <http://docs.php.net/manual/es/intro-whatcando.php>
- Group, W. C. W. (2013). Introduction to Model-Based User Interfaces Meixner, G. Calvary, G.
- Coutaz, J. . Retrieved 17/09/2015, 2015, from <http://www.w3.org/2011/mbui/drafts/mbui-intro/>
- Grupo, I. (2003). Metodologías ágiles en el desarrollo de software.
- Gutiérrez, J. J. (2014). ¿ Qué es un framework Web? Available in: http://www.lsi.us.es/~javierj/investigacion_ficheros/Framework.pdf Accessed May, 12.

- Hannington, A., & Reed, K. (2002). *Towards a taxonomy for guiding multimedia application development*. Paper presented at the Software Engineering Conference, 2002. Ninth Asia-Pacific.
- Index, C. V. N. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015–2020 White Paper: Tech. rep. Cisco, 2016. url: <http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html> (visited on 03/26/2016)(cit. on p. 6).
- Iso, W. (1998). 9241-11. Ergonomic requirements for office work with visual display terminals (VDTs). *The international organization for standardization*, 45.
- Izquierdo, J. L. C., & Trujillo, S. Retos Actuales en el Desarrollo de Lenguajes Especificos del Dominio.
- Kent, S. (2002). *Model driven engineering*. Paper presented at the International Conference on Integrated Formal Methods.
- Khan, S. (2007). Khan Academy | Práctica, lecciones y cursos en línea gratuitos, from <https://es.khanacademy.org/>
- Kleppe, A. (2008). *Software language engineering: creating domain-specific languages using metamodels*: Pearson Education.
- Kulesza, R., Meira, S. R., Ferreira, T. P., Alexandre, E. S., Guido Filho, L., Neto, M. C. M., & San, C. A. (2012). *A Model-driven Approach for Integration of Interactive Applications and Web Services: A Case Study in Interactive Digital TV Platform*. Paper presented at the Multimedia and Expo Workshops (ICMEW), 2012 IEEE International Conference on.
- Kurniawati, E., Celetto, L., Capovilla, N., & George, S. (2012). *Personalized voice command systems in multi modal user interface*. Paper presented at the 2012 IEEE International Conference on Emerging Signal Processing Applications, ESPA 2012 - Proceedings.
- Lara, P., & Duart, J. M. (2005). Gestión de contenidos en el e-learning: acceso y uso de objetos de información como recurso estratégico. *Revista de universidad y sociedad del conocimiento*, 2(2), 6-14.
- Law, E. L.-C., Roto, V., Hassenzahl, M., Vermeeren, A. P., & Kort, J. (2009). *Understanding, scoping and defining user experience: a survey approach*. Paper presented at the Proceedings of the SIGCHI conference on human factors in computing systems.
- Levin, R. (2012). Coursera | Online Courses From Top Universities, from <https://www.coursera.org/>
- Lin, J., & Landay, J. A. (2008). *Employing patterns and layers for early-stage design and prototyping of cross-device user interfaces*. Paper presented at the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems.
- Losavio, F., Chirinos, L., Matteo, A., Lévy, N., & Ramdane-Cherif, A. (2004). ISO quality standards for measuring architectures. *Journal of Systems and Software*, 72(2), 209-223.
- Mascheroni, M. A., Greiner, C. L., Petris, R. H., Dapozo, G. N., & Estayno, M. G. (2012). *Calidad de software e ingeniería de usabilidad*. Paper presented at the XIV Workshop de Investigadores en Ciencias de la Computación.
- McEvoy, A. M. ooVoo 2.0 VoIP Service/ Internet.
- Meixner, G., Paterno, F., & Vanderdonckt, J. Past, present, and future of model-based user interface development. *i-com* 10 (3): 2-11, 2011.
- Mercovich, E., & Aires, B. (1999). *Ponencia sobre Diseño de Interfaces y Usabilidad: cómo hacer productos más útiles, eficientes y seductores*. Paper presented at the SigGraph.
- Meskens, J., Vermeulen, J., Luyten, K., & Coninx, K. (2008). *Gummy for multi-platform user interface designs: shape me, multiply me, fix me, use me*. Paper presented at the Proceedings of the working conference on Advanced visual interfaces.
- MOF, O. (2007). 2.0/XMI Mapping, Version 2.1. 1. *Object Management Group, Needham, MA, USA*.

- Mori, G., Paternò, F., & Santoro, C. (2002). CTTE: support for developing and analyzing task models for interactive system design. *Software Engineering, IEEE Transactions on*, 28(8), 797-813.
- Musciano, C., & Kennedy, B. (1996). *HTML, the definitive Guide*: O'Reilly & Associates.
- Myers, B., Hudson, S. E., & Pausch, R. (2000). Past, present, and future of user interface software tools. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 7(1), 3-28.
- Network, M. D. Acerca de JavaScript, from https://developer.mozilla.org/es/docs/Web/JavaScript/Acerca_de_JavaScript
- Network, M. D. CSS3, from <https://developer.mozilla.org/es/docs/Web/CSS/CSS3>
- Network, M. D. HTML, from <https://developer.mozilla.org/es/docs/Web/HTML>
- Network, M. D. Introducción al HTML, from https://developer.mozilla.org/es/docs/Web/Guide/HTML/Introduction_alhtml
- Network, M. D. JavaScript, from <https://developer.mozilla.org/es/docs/Web/JavaScript>
- Odden, L. What is Content? Learn from 40+ Definitions, from <http://www.toprankblog.com/2013/03/what-is-content/>
- Oracle. ¿Qué es la tecnología Java y para qué la necesito? , from https://www.java.com/es/download/faq/whatis_java.xml
- Orozco, G., & Joseph, W. (2010). Marco de desarrollo de sistemas groupware interactivos basado en la integración de procesos y notaciones.
- Paterno', F., Santoro, C., & Spano, L. D. (2009). MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Trans. Comput.-Hum. Interact.*, 16(4), 1-30. doi: 10.1145/1614390.1614394
- Paterno, F., Santoro, C., & Spano, L. D. (2009). MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 16(4), 19.
- Peissner, M., H, D., #228, be, Janssen, D., & Sellner, T. (2012). *MyUI: generating accessible user interfaces from multimodal design patterns*. Paper presented at the Proceedings of the 4th ACM SIGCHI symposium on Engineering interactive computing systems, Copenhagen, Denmark.
- Pérez, G., Irazábal, J., Pons, C., & Giandini, R. Aplicando herramientas MDE en la definición de un lenguaje específico de dominio para la gestión de modelos.
- Perry, J. S. (2012). Introducción a la programación Java, parte 1: Conceptos básicos del lenguaje Java. IBM Developer Works.
- Pilgrim, M. (2010). *HTML5: Up and Running: Dive into the Future of Web Development*: " O'Reilly Media, Inc."
- Ponencia sobre Diseño de Interfaces y Usabilidad: cómo hacer productos más útiles, e. y. s. Eduardo Mercovich, from <http://www.gaiasur.com.ar/infoteca/siggraph99/disenio-de-interfaces-y-usabilidad.html>
- Pons, C., Giandini, R. S., & Pérez, G. (2010). Desarrollo de software dirigido por modelos.
- Railean, E. (2012). Google Apps for Education—a powerful solution for global scientific classrooms with learner centred environment. *International Journal of Computer Science Research and Application*, 2(2), 19-27.
- Rauschenberger, M., Olschner, S., Cota, M. P., Schrepp, M., & Thomaschewski, J. (2012). *Measurement of user experience: a Spanish language version of the user experience questionnaire (UEQ)*. Paper presented at the Information Systems and Technologies (CISTI), 2012 7th Iberian Conference on.
- Reynoso, C. B. (2004). Introducción a la Arquitectura de Software. *Recuperado de: <http://carlosreynoso.com.ar/archivos/carlos-reynoso-introduccion-a-la-arquitectura-de-software.pdf>*

- Riehle, D. (2000). *Framework design*.
- Rosson, M. B., & Carroll, J. M. (2002). Usability engineering: scenario-based development of human-computer interaction.
- Sebastian Thrun, D. S., Mike Sokolsky (2012). Udacity - Free Online Classes & Nanodegrees, from <https://www.udacity.com/>
- SOCKET.IO. SOCKET.IO, from <https://socket.io>
- Standardization, I. O. f., & Commission, I. E. (2001). *Software Engineering--Product Quality: Quality model* (Vol. 1): ISO/IEC.
- Standardization, I. O. f. (1998). ISO 14598-1 Information Technology - Evaluation of Software Products *ISO*.
- Tolvanen, J.-P., & Kelly, S. (2009). *MetaEdit+: defining and using integrated domain-specific modeling languages*. Paper presented at the Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications.
- Udemy Online Courses - Learn Anything, On Your Schedule. (2010), from <https://www.udemy.com>
- UniMOOC. from <https://unimooc.com/>
- University, M. I. o. T. a. H. (2012). edX | Free online courses from the world's best universities, from <https://www.edx.org/>
- Welling, L., & Thomson, L. (2005). *Desarrollo web con PHP y MySQL*.
- Xu, Y., Yu, C., Li, J., Hu, H., Liu, Y., & Wang, Y. (2010). Measurement study of commercial video conferencing systems. *Technical report Polytechnic Institute of NYU*.
- Zhu, X., Aref, W. G., Fan, J., Catlin, A. C., & Elmagarmid, A. K. (2003). *Medical video mining for efficient database indexing, management and access*. Paper presented at the Data Engineering, 2003. Proceedings. 19th International Conference on.