“Módulo de grabación y transmisión de audio para el software del aula tecnológica ATcnea”

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

Autor: Daniel Blanco Isidrón

Tutores: Ing. Carlos Montenegro Amador

Ing. Carlos Arturo Rondón Quinta

La Habana, Julio de 2017

“Año 59 de la Revolución”

Declaración de Autoría

Declaro ser el autor del presente trabajo de diploma y otorgo a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____

Firma del Autor
Daniel Blanco Isidrón

_____

Firma del Tutor
Ing. Carlos Montenegro Amador

_____

Firma del Tutor
Ing. Carlos Arturo Rondón Quinta

"El objetivo principal de la educación es crear personas capaces de hacer cosas nuevas y no simplemente repetir lo que otras generaciones hicieron"

Jean Piaget

Agradecimientos

Primero quiero agradecer a todas aquellas personas que de alguna forma u otra en la realización de mi tesis, además de que por encima de todo agradecer a mis compañeros de apartamento que me apoyaron en todo momento y nunca permitieron que me rindiera. Quiero darle y dedicarle este título a mi familia que en las buenas y en las malas contribuyeron a que cumpliera esta meta tan deseada por todos y más que todo a mis padres dedicarle esta meta ya alcanzada y que los quiero mucho.

Resumen

Con el propósito de introducir las tecnologías de la información y las comunicaciones en el proceso de enseñanza y aprendizaje en la educación cubana; el Centro de Tecnologías para la Formación de la Universidad de las Ciencias Informáticas, se encuentra desarrollando un *software* educativo que formará parte de la primera aula tecnológica del país. Esta versión del aula tecnológica denominada ATcnea, fue creada fundamentalmente para lograr una mayor interacción entre el estudiante y el profesor. La misma, ayuda a los profesores a aumentar el nivel de colaboración e interactividad en el aula haciendo uso de diferentes recursos de información. Este software aún no ha logrado completamente la participación e interacción entre los estudiantes y el profesor mediante la vía oral, debido a que no se pueden comunicar mediante mensajes de voz en tiempo real, ni a través de grabaciones. La presente investigación tiene como objetivo desarrollar un módulo de transmisión y grabación de audio para el *software* del aula tecnológica ATcnea, el cual contribuirá a elevar la interacción entre el profesor y el estudiante. Como parte de la investigación se realizó un estudio de algunas aulas tecnológicas existentes a nivel internacional, se evaluaron las principales tecnologías y herramientas usadas para el desarrollo de aplicaciones para la transmisión y grabación de audio. Seguidamente se transitó siguiendo la metodología de desarrollo AUP en su versión UCI por algunos de sus flujos de trabajo; obteniéndose artefactos que eran necesarios para un mejor entendimiento y comprensión de las tareas a realizar. Además de realizarse pruebas al módulo a desarrollar con el objetivo de evaluar su correcto funcionamiento.

Palabras clave:

Aula Tecnológica, módulo, transmisión, grabación, proceso de enseñanza-aprendizaje

Contenido

Introducción

Las Tecnologías de la Información y las Comunicaciones (TIC) posibilitan la creación de un nuevo espacio social-virtual para las interrelaciones humanas. Las TIC como concepto general se refiere a la utilización de múltiples medios tecnológicos o informáticos para almacenar, procesar y difundir todo tipo de información con diferentes finalidades. Las mismas son utilizadas como forma de gestionar y organizar procesos en el plano educativo; ya que su uso como herramienta didáctica facilita la explicación de los contenidos y la búsqueda de un tipo de educación donde el estudiante esté más involucrado en proyectos multidisciplinarios. (Blanco Sánchez, 2012)

Este entorno origina nuevos procesos de aprendizaje, educación y transmisión del conocimiento a través de las redes modernas de comunicaciones, permitiendo que los estudiantes aprendan a desenvolverse en un ambiente de colaboración y desarrollo cooperativo. Esta nueva filosofía de aprendizaje, está obligando a reestructurar las actuales aulas tradicionales, para dar espacio al surgimiento de una nueva modalidad de aprendizaje basado en el uso de tecnologías.

A este tipo de enseñanza basada en el uso de dispositivos electrónicos y de redes modernas de comunicaciones se le denomina *e-learning*, el cual permite la interacción del usuario con el material, mediante la utilización de diversas herramientas informáticas. Esto posibilita que las clases se puedan impartir a distancia y no sea necesario la presencia de un educador en un salón de conferencia. El término *e-learning* viene dado por la simplificación de *Electronical Learning*, el mismo reúne a las diferentes tecnologías y a los aspectos pedagógicos de la enseñanza y el aprendizaje, mediante el uso de dispositivos electrónicos (Sarmiento,2010).

A partir de esta concepción de tener una educación basada en el uso de la tecnología en el Proceso de Enseñanza y Aprendizaje (PEA), da paso al surgimiento de las aulas tecnológicas las cuales tienen como propósito reemplazar las actuales aulas tradicionales. En el aula tecnológica se incorporan recursos informáticos y no informáticos para enriquecer la enseñanza, es espacio innovador que permite realizar sesiones de trabajo en régimen de multisesión abierta, entre usuarios, que pueden intercambiar y comparar contenidos diversos (Gonzalo,2009).

Introducción

El aula tecnológica se concibe como una oportunidad pedagógica y una estrategia didáctica para el docente, es una herramienta de apoyo y de mejoramiento continuo al desarrollo del PEA. Dentro de los elementos que componen un aula tecnológica se encuentran: dispositivos *Tablet* para los alumnos y una *laptop* para el profesor. Desde la *laptop* del profesor se puede monitorear y controlar todas las operaciones realizadas por los estudiantes en sus *Tablet*. Los profesores, instructores y conferencistas pueden mejorar la eficacia de la formación en la clase, enseñando a los estudiantes de manera centralizada desde su ordenador portátil.

Las actividades desarrolladas en un aula tecnológica son mostradas en una pizarra digital interactiva mediante el uso de un proyector. La pizarra electrónica, es una herramienta didáctica de uso reciente en la enseñanza y con grandes potenciales pedagógicos. Esta incrementa los niveles de motivación y participación de los alumnos, además de facilitar la interacción del estudiante con gráficos, animaciones, simulaciones, videos, imágenes y otros materiales multimedias. El uso de los distintos recursos en el aula tecnológica contribuye a mejorar la calidad educativa. Estos elementos favorecen la atención del estudiante durante la clase, monitoreando las aplicaciones, mejorando el soporte con la ayuda en línea y ahorrando tiempo mediante la realización de preguntas y respuestas rápidas a toda la clase. (Lorena,2012)

Cuba, no se ha quedado ajena a estas trasformaciones, por lo que se crea en la Facultad 4 de la Universidad de las Ciencias Informáticas (UCI), el Centro de Tecnologías para la Formación (FORTES) con el propósito de introducir las TIC en el PEA en la educación cubana.

FORTES está encargado de desarrollar tecnologías y soluciones de formación para todo tipo de instituciones. En el mismo se está desarrollando un *software* educativo que formará parte de la primera aula tecnológica del país. Esta versión del aula tecnológica denominada ATcnea, fue creada fundamentalmente para lograr una mayor interacción entre el estudiante y el profesor. Esta es una solución de capacitación y formación que proporciona a los docentes o conferencistas, la capacidad de interactuar con sus estudiantes individualmente como un grupo definido o con toda la clase.

ATcnea es un aula tecnológica concebida para transformar la enseñanza y el aprendizaje. Se compone de tecnologías como pueden ser: pizarra digital interactiva, video proyector, cámara de documentos, audio, datos y video. Además de contar con un *software* de gestión del aula, compuesto por dos aplicaciones, una destinada al profesor o moderador de la clase, desarrollada en el lenguaje *Java* y otra destinada a los

estudiantes o receptores, desarrollada en *Android*. Entre ambas aplicaciones existe comunicación mediante conexión inalámbrica, para que el profesor desde su *laptop* pueda guiar el PEA usando los medios que proveen las aulas tecnológicas, y los estudiantes desde el *Tablet* recibir e interactuar con el contenido.

El *software* cuenta con funcionalidades, que permiten a un profesor guiar el PEA. Entre estas se encuentran: autenticación de los usuarios, atención diferenciada a los estudiantes, herramienta chat, creación de actividades grupales, creación de preguntas con varias tipologías y exámenes, registro de asistencia, compartir video, envío de archivos, control de las terminales de los estudiantes de manera remota por parte del profesor, reportes de asistencia y evaluaciones.

Estás nuevas tecnologías representan un desafío para el profesor, porque le exige articular una herramienta innovadora en su quehacer pedagógico con los contenidos, los recursos disponibles, la motivación en procura de experiencias significativas para el desempeño de los estudiantes. El uso de variedad de recursos en un mismo espacio favorece la comprensión, visualización de conceptos y la construcción del conocimiento.

Aún en esta primera versión, no se ha logrado completamente la participación e interacción entre los estudiantes y el profesor mediante la vía oral, debido a que no se pueden comunicar mediante mensajes de voz en tiempo real, ni a través de grabaciones. Otro elemento que no se incluye, es el incremento de sus capacidades de análisis crítico de los textos o pronunciar criterios mediante exposiciones orales a través de la transmisión de voz en tiempo real. También se ve limitado la ayuda de los profesores a los estudiantes a superar las barreras que a menudo los separan del aprendizaje de la lengua inglesa.

A partir de la situación problemática descrita anteriormente se propone como **problema a resolver:** ¿Cómo contribuir a la interacción entre los estudiantes y el profesor a través del *software* para el aula tecnológica ATcnea durante el desarrollo del PEA?

Con el fin de solucionar el problema, se define como **objeto de estudio** de la presente investigación: la transmisión y grabación de audio en las aulas tecnológicas. Enfocando el **campo de acción:** en la transmisión y grabación de audio en el aula tecnológica ATcnea.

Teniendo en cuenta el problema a resolver se define como **objetivo general:** desarrollar un módulo de grabación y transmisión de audio para *software* del aula tecnológica ATcnea que contribuya a la interacción

entre los estudiantes y el profesor en el desarrollo del PEA. Complementando el objetivo general se definen los siguientes **objetivos específicos:**

- Construir los referentes teóricos relacionando los aspectos fundamentales que sustentan la investigación, mediante los cuales se consulta, extrae y recopila la información relevante sobre el problema a investigar.
- Realizar el diseño del módulo de grabación y transmisión de audio para el *software* del aula tecnológica ATcnea basado en los requisitos descritos.
- Implementar el módulo de grabación y transmisión de audio para el *software* del aula tecnológica ATcnea.
- Realizar las pruebas al módulo de grabación y transmisión de audio para el *software* del aula tecnológica ATcnea.

La siguiente investigación tendrá como **resultado esperado:** un módulo de grabación y transmisión de audio para el *software* del aula tecnológica ATcnea que contribuya a la interacción entre los estudiantes y el profesor en el desarrollo del PEA.

Métodos investigativos

Para realizar la investigación se utilizaron métodos teóricos y empíricos que a continuación se relacionan:

Métodos teóricos

- Analítico-Sintético: se empleó para el análisis de la teoría y extracción de los principales conceptos a incluir en el marco teórico. Además, para el estudio y análisis de la información, lo que permitió obtener los conceptos y elementos principales relacionados con las aulas tecnológicas.

- Análisis Histórico-Lógico: fue utilizado para el estudio del estado del arte, para construir los referentes teóricos relacionados con los aspectos fundamentales que sustentan la investigación. Dígase, el estudio de otras soluciones o aplicaciones similares con la investigación, así como de las metodologías de desarrollo y lenguajes de programación a utilizar en el desarrollo de la propuesta de solución.

Métodos empíricos

- Observación: este método es el instrumento que permitió estudiar el objeto de la investigación, las acciones, causas y consecuencias.
- Entrevista: se utiliza para precisar cuáles son las principales funcionalidades con que cuenta el aula tecnológica y los diferentes subprocesos con que presenta el mismo.

Descripción Capitular

El siguiente trabajo cuenta con una estructura capitular que quedó constituida de la siguiente manera:

Capítulo 1: Fundamentación Teórica.

En este capítulo se aborda lo referente a los temas teóricos necesarios que podrán dar solución al problema planteando. Además, se hace referencia a los conceptos y elementos relacionados con la transmisión de audio en las aulas tecnológicas. Se realiza un estudio de herramientas similares, así como la selección de las tecnologías y herramientas a utilizar para desarrollar la propuesta de solución.

Capítulo 2: Descripción de la propuesta solución.

En el capítulo se hará una descripción de la propuesta de solución que permitirá transmitir y grabar audio en el *software* del aula tecnológica ATcnea. Se presentará el modelo de dominio y se especificarán los requisitos que debe cumplir el sistema. Además, se realizarán los artefactos del diseño correspondientes al escenario de la metodología definida.

Capítulo 3: Desarrollo de la propuesta de solución.

En este capítulo se abordará todo lo referente a la implementación del módulo de grabación y transmisión de audio propuesto. Se realizará el diagrama de componentes del sistema y se definira´n los estándares de codificación empleados. Se especificarán los niveles y técnicas de pruebas que serán aplicadas a dicho módulo. Además, se realizarán las pruebas al *software* utilizando casos de pruebas que servirán para llevar a cabo las comprobaciones de calidad.

Capítulo 1: Fundamentación Teórica

1.1 Introducción

La base teórica del presente trabajo está sustentada por las tecnologías, metodología, lenguajes de programación y de modelado necesarios para el desarrollo del módulo de grabación y transmisión de audio para el *software* del aula tecnológica ATcnea. Este capítulo presenta, además, una descripción del estado del arte acerca de los sistemas similares desarrollados internacionalmente.

1.2 Aula Tecnológica

Un aula tecnológica es una solución educativa que brinda una experiencia única de aprendizaje. Tiene como objetivo la creación de un ambiente colaborativo, donde la tecnología enriquece el contenido académico de cada asignatura y permite al profesor-estudiante establecer una amplia comunicación (Ricardo Bahena,2000).

Muchos son los autores que han definido el aula tecnológica. A continuación, se citan algunos conceptos:

- Segovia define aula tecnológica como una comunidad de aprendizaje, cuyo objetivo principal es el desarrollo de la inteligencia de los alumnos bajo la mediación de los profesores, por medio de métodos didácticos diversificados en un espacio multiuso abierto, tecnológicamente equipado y organizado (Ricardo Bahena,2000).

- Antonia Lozano Díaz precisa que el sistema educativo aula inteligente es un constructo creativo, un conjunto de saberes que se plasman en una pedagogía singular. Propugna un cambio de modelo de educación a través de la reingeniería total del sistema educativo; lo hace partiendo de una determinada conceptualización de lo que sería la calidad en educación (Olmo,2003).

- Antonia Lozano expone: un aula tecnológica o interactiva reúne aspectos arquitectónicos, ambientales, de acabado o mobiliario, como elemento fundamental el equipamiento físico y lógico

básico, considerándose como tal las PC o dispositivos móviles, el *software* compatible y la conectividad adecuada que garantice la integridad del equipamiento (Olmo,2003).

- Las redes inalámbricas son un elemento que permite acercar a las clases una gran cantidad de recursos que no eran imaginables a un mínimo costo y de fácil acceso. Este proceso ha dado lugar a la creación de espacios y herramientas pensadas para la enseñanza, con la idea de hacer un uso educativo basado en el uso de la tecnología. Esto es lo que algunos especialistas de la temática han llamado aulas tecnológicas. (Gonzalo,2009)

- (Navarro y Soto, 2006) sostienen que el aprendizaje mediante dispositivos electrónicos, necesita un soporte tecnológico, pedagógico y social el cual lo constituye el aula tecnológica.

- El término que se le atribuye a Roxanne Hiltz; quién la define como el empleo de sistemas comunicacionales mediadas por ordenadores, para crear un ambiente análogo electrónico de las formas de comunicación que normalmente se producen en un aula convencional. (Rodolfo,2001)

- El aula tecnológica es el espacio donde se concentrará el proceso de aprendizaje. Más allá del modo en que se organice la educación: sea presencial, semipresencial o remota, sincróna o asíncrona, esta será el medio de intercambio donde la clase tendrá lugar. Dicho esto, es importante que el diseño o la elección de un sistema de aula tecnológica quede claro; para ello se deberá tener en cuenta que elementos de esta herramienta se utilizarán para lograr que el aprendizaje de los alumnos sea productivo. (Juadon,2010)

Una vez analizados los conceptos anteriores, el autor de la presente investigación entiende por aula tecnológica: un ambiente educativo compuesto por tecnología, donde el profesor hace uso de los métodos didácticos para desarrollar de una forma ágil e interactiva el PEA, permitiendo que los estudiantes asimilen el contenido de una manera dinámica.

1.2.1 Características y ventajas de las aulas tecnológicas

Las características básicas que presentan las aulas tecnológicas son las siguientes:(Ayala,2015)

- Una organización menos definida del espacio y el tiempo educativo.
- Uso más amplio e intensivo de las TIC.
- Planificación y organización del aprendizaje más guiado en sus aspectos globales.
- Contenidos de aprendizaje apoyados con mayor base económica.
- Forma telemática de llevar a cabo la interacción social.
- Desarrollo de las actividades de aprendizaje más centrado en el alumnado.

Tabla 1. Formación en las aulas tecnológicas vs. Formación en las aulas tradicionales

Formación en el aula tecnológica	Formación en aula tradicional
Posibilitan que los estudiantes vayan a su ritmo de aprendizaje.	Se proponen ciertos conocimientos adquiridos por lo que los estudiantes deben ajustarse a los mismos.
Puede recurrirse a las mismas cuando se necesita.	El docente establece el momento en que los alumnos recibirán el material docente.
El conocimiento representa un proceso activo de construcción.	Tiende a un modelo líneal de comunicación.
Es de carácter preferentemente interactivo entre el profesor y el alumno y también respecto a los conocimientos.	La comunicación tiene lugar entre el docente y el alumno.
Suele realizase de manera individual, sin que ello suponga renunciar a propuestas colaborativas.	La enseñanza se realiza de manera grupal.
No siempre se tienen los recursos estructurales y organizativos para su puesta en práctica .	Existen varios recursos estructurales y organizativos para su puesta en funcionamiento.

Ventajas de las aulas tecnológicas

El aula tecnológica aporta múltiples beneficios a estudiantes y profesores. Mejora la calidad del PEA en las instituciones educativas, condicionado por el avance tecnológico que en estas se emplea. A continuación, en la siguiente tabla se muestran algunos de estos beneficios (Lavin,2010):

Tabla 2. Beneficios de un aula tecnológica para el profesor y los estudiantes

Profesor	Estudiante
• Uso y comprensión de la tecnología • Programación de contenidos • Seguimiento a los avances individuales y de grupo • Realización de experimentos • Desarrollar más actividades vivenciales	• Vivirá intensamente el aprendizaje en todas sus formas: visual, oral, escrita y auditiva • Desarrollará su creatividad • Estimulará su potencial innovador • Fortalecerá su aprendizaje autónomo • Despertará su interés por la ciencia y la tecnología

1.3 Estudio de sistemas similares

Hoy en día el uso del aula tecnológica está revolucionando el PEA trayendo consigo el surgimiento de nuevos proyectos educativos, con el objetivo de mejorar los métodos y herramientas de enseñanza. Uno de sus principales componentes es el sistema de gestión del aula encargado de establecer interacción profesor-estudiante, así como proporcionar interfaz e interoperabilidad entre estos dos actores.

Dentro del territorio nacional el *software* del proyecto ATcnea se puede considerar como pionera debido a que es la primera de su tipo en Cuba. Fuera del territorio nacional se puede encontrar con disímiles aulas tecnológicas, a continuación, se hace un análisis de algunos de estos sistemas indagando en sus principales funcionalidades.

ITALC

Vendedor: *italcSolutions*
País: Alemania
URL: http://italc.sourceforge.net
Despliegue: Instalado
Soporte: Sí (*2014-07-10*)
Licencia: Libre (GNU GPLv2)

ITALC posee un cúmulo de funcionalidades importantes como: mostrar un demo (ya sea en pantalla completa o en una ventana), la pantalla del profesor se muestra en todas las computadoras de los estudiantes en tiempo real. Permite encender, apagar o reiniciar por control remoto, bloquear las estaciones de trabajo para mover la atención individual. Este *software* no tiene en cuenta el envío y la transmisión de audio dentro del aula tecnológica (ITALC,2016).

NetSupport School

Vendedor: *NetSoporte*
País: Estados Unidos
URL: www.netSoporte-inc.com
Prueba gratis: Sí
Despliegue: Instalado
Soporte: Sí
Licencia: Privativa

Este sistema cuenta con funciones como: gestión de la clase, registro de estudiantes, chat en grupo o a nivel individual, evaluación, módulo de preguntas y respuestas. Esta aula tecnológica sí realiza el envío de archivos de audio, así como la transmisión en tiempo real de mensajes de voz, pero la licencia es privativa (NetSupport,2016).

Mythware Classroom Management Software

Vendedor: *Nanjing Universal Networks*
País: China
URL: www.mythware.com
Fundado: 2007
Prueba gratis: Sí
Despliegue: Web
Soporte: Sí
Licencia: Privativa

Mythware Classroom Management Software está diseñado para ayudar a los profesores a impartir la clase de aprendizaje de idiomas con más interacción profesor-alumno. Con las diversas funciones tales como difusión de voz, intercomunicación, interpretación simultánea, entrenamiento y examen oral, este sistema ayuda a los estudiantes a estar más involucrados en el aprendizaje (Mythware,2016).

Tabla 3. Funcionalidades de otros sistemas similares

Funcionalidades	*Mythware Classroom Management Software*	*iTALC*	*NetSupport School*
Pizarra interactiva	X	X	X
Abrir sitio web de forma remota	X		X
Transmisión por cámara	X	X	
Transmisión de pantalla	X	X	
Bloquear pantalla al estudiante	X	X	

Enviar mensaje a los estudiantes en forma de texto	X	X	X
Entrega y recogida de archivos		X	X
Chat en grupo o a nivel individual	X	X	X
Evaluación módulo de preguntas y respuestas	X		
Pruebas y exámenes para compartir contenido		X	X
Añadir o adjuntar direcciones web e imágenes a mensajes instantáneos	X	X	X
Recursos online para exámenes y compartir contenido y consola técnica	X	X	X
Bloquear las estaciones de trabajo	X		X
Visualización de las pantallas de los estudiantes en tiempo real		X	X
Enviar preguntas rápidas			X
Gestión de la clase	X	X	X
Transmisión por cámara		X	X
Película en red	X		X

Vista en miniatura de las pantallas de los usuarios en tiempo real	X	X	
Gestionar servicios	X		

Conclusiones del estudio de las herramientas similares

Los sistemas anteriormente estudiados permiten reconocer la necesidad de la investigación, ya que no poseen la transmisión de audio como funcionalidades nativas del *software*. Todas estas aulas tecnológicas cuentan con dispositivos tecnológicos al igual que tienen los *softwares* necesarios para guiar el PEA. La diferencia entre estas y la del proyecto ATcnea es que cuentan con un sistema *online* de aprendizaje por medio de sitios web, además de que no presentan un sistema de comunicación a través de voz para interactuar con los estudiantes y algunas de ella presentan licencia privativa. Sin embargo, sirvió como guía para la actual investigación.

1.3.1 Aula tecnológica ATcnea

ATcnea es la primera aula tecnológica de Cuba, desarrollada en el centro FORTES de la UCI. Es una solución de capacitación y formación que proporciona a los docentes o conferencistas, la capacidad de interactuar con sus estudiantes. Proporciona una solución para la creación de un aula interactiva compuesta por una pizarra inteligente, una laptop para el profesor y 16 tabletas: una por cada estudiante en cada mesa garantizando la interacción entre las tabletas y la laptop a través de la conexión inalámbrica. Esta solución cuenta con una aplicación desarrollada en *Java* para la computadora portátil del profesor y una aplicación desarrollada en *Android* para las *Tablet* de los estudiantes. A continuación, se relacionan las tecnologías que hoy están presentes en ATcnea:

NovaDroid: resultado del trabajo conjunto HAIER – UCI, con un entrenamiento de alto nivel suministrado por ingenieros de la empresa HAIER. Se construyó una personalización de *Android* que da lugar a una ROM que hoy está disponible en las tabletas, incluye una experiencia de usuario mejorada más cercana a lo que los usuarios están acostumbrados a ver en las computadoras de escritorio. Actualmente se trabaja en una

personalización más a la medida del *SystemUI* y en otros temas asociados a los requerimientos propios del aula tecnológica.

NovaNAS: el aula está equipada con un servidor de almacenamiento gestionado con la solución NAS que se lanzó con Nova en la Feria de Informática pasada celebrada del 20 al 24 de junio del 2016. NovaNAS almacena toda la información del aula, recursos de aprendizaje, multimedia, etc. Todos estos recursos son, a su vez, operados por la plataforma Zera, el Entorno Virtual de Aprendizaje (EVA) desarrollado en la Facultad 4.

Nova Unificado: el servidor unificado provee dominio para el entorno del aula, así como el servicio de mensajería instantánea.

Nova: sistema operativo desarrollado en el Centro de Desarrollo de Soluciones Libres (CESOL) de la UCI instalado en la laptop del profesor, provee soporte para la ejecución de la aplicación ATcnea.

App de Realidad Aumentada: el Centro de Entorno Interactivo 3D (VERTEX) de la Facultad 4 provee a las tabletas una aplicación de realidad aumentada que complementa los recursos de aprendizaje de algunas de las clases hoy instaladas en el aula. Un marcador disponible en cada mesa de los estudiantes permite (de conjunto con la *app* instalada en las tabletas) visualizar contenidos en 3D aplicando la técnica de la realidad aumentada (Yoandy,2016).

1.4 Metodología de desarrollo

Es un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo en sistemas de información. Las metodologías definen con precisión los artefactos, roles y actividades involucrados, junto con prácticas y técnicas recomendadas para guiar el proceso de desarrollo de *software* (Roger S,2010).

Las metodologías de desarrollo de *software* son un conjunto de procedimientos, técnicas de ayudas a la documentación para el desarrollo de productos de *software*. Estas van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además, detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. (Brito,2016)

Para lograr una planificación eficiente y predecir resultados durante el proceso de desarrollo de *software* han surgido varias metodologías. Estas últimas son una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de *software* en sus esfuerzos por implementar nuevos sistemas de información. De acuerdo a la filosofía de desarrollo, estas metodologías se clasifican en tradicionales o ágiles.

Para el desarrollo del módulo se hace uso de la metodología de desarrollo ágil AUP UCI, de acuerdo a que fue la que el proyecto seleccionó para el desarrollo del *software,* teniendo en cuenta que es la utilizada en la actividad productiva de la UCI.

1.4.1 Proceso unificado ágil

El Proceso Unificado Ágil (AUP, por sus siglas en inglés) es una versión simplificada del Proceso Unificado de *Rational* (RUP, por sus siglas en inglés). Este describe de manera fácil la forma de desarrollar aplicaciones de *software* de negocio usando técnicas ágiles de XP y conceptos que aún se mantienen válidos en RUP (Roger S,2010).

Esta metodología tiene como característica principal que posee tres fases que se ejecutan de manera consecutiva, siete disciplinas asociadas, once roles y cuatro posibles escenarios, que permiten su adaptación al proyecto.

Características de AUP

- **Iterativo e Incremental:** compuesto de cuatro fases denominadas Inicio, Elaboración, Construcción y Transición. Cada una de estas fases es a su vez dividida en una serie de iteraciones (la de inicio puede incluir varias iteraciones en proyectos grandes). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo. Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto.

- **Dirigido por los casos de uso:** los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración tome un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de las distintas disciplinas.
- **Centrado en la arquitectura:** no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura de *software* de un sistema.
- **Enfocado en los riesgos:** requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero (AUP-UCI,2017).

Fases de metodología AUP (AUP-UCI,2017)**:**

- **Inicio:** durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto.
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el software.
- **Cierre:** En esta se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.4.2 AUP en su versión UCI

El proyecto ATcnea pertenece a una de los centros productivos de la UCI por lo es necesario que se ajuste a los estándares y metodologías empleadas para el desarrollo de *software* en la universidad. Por esos motivos la metodología AUP antes descrita, sufre modificaciones para que se ajuste al marco de trabajo de los centros productivos. (Brito,2016)

Los proyectos de la UCI deben mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llama Ejecución y se agrega la fase de Cierre.

El ciclo de vida de los proyectos de la UCI debe tener 7 disciplinas también, pero a un nivel más atómico que el definido en AUP. Los flujos de trabajos: Modelado de negocio, Requisitos, Análisis y Diseño en AUP están unidos en la disciplina Modelo, en la variación para la UCI se consideran a cada uno de ellos

disciplinas. Se mantiene la disciplina Implementación, en el caso de Prueba se desagrega en 3 disciplinas: Pruebas Internas, de Liberación y Aceptación. Las restantes 3 disciplinas de AUP asociadas a la parte de gestión para la variación UCI se cubren con las áreas de procesos que define CMMIDEV v1.3 para el nivel 2, serían CM (Gestión de la configuración), PP (Planeación de proyecto) y PMC (Monitoreo y control de proyecto).

Los proyectos de la UCI tienen 11 roles, manteniendo algunos de los propuestos por AUP y unificando o agregando otros. Los roles propuestos son: jefe de proyecto, planificador, analista, arquitecto de proyecto, desarrollador, administrador de la configuración, tenedor de apuestas de requisitos, administrador de la calidad, probador, arquitecto de *software* y administrador de base de datos.

Entre las técnicas ágiles que utiliza AUP se encuentra el Modelado ágil, se hará uso de esta técnica para los proyectos que necesiten por sus características encapsular sus requisitos funcionales en Historias de Usuarios (HU) o en Descripción de requisitos por procesos. La otra forma de encapsular los requisitos se mantiene por Casos de Uso (CU).

Para el desarrollo de este *software* la metodología a utilizar es AUP en su versión UCI en el escenario número 4 (encapsulamiento de requisitos funcionales en HU), puesto que el proyecto ha evaluado el negocio a informatizar y lo tiene muy bien definido. El cliente estará siempre acompañando al equipo de desarrollo para convenir los detalles de los requisitos y así poder implementarlos, probarlos y validarlos. Se recomienda en proyectos no muy extensos, ya que una HU no debe poseer demasiada información.

1.5 Lenguaje de modelado

Es un conjunto de símbolos estandarizados usados para diseñar un sistema orientado a objetos. Su utilización depende generalmente de la combinación de una metodología de desarrollo de software, para obtener una especificación inicial a un plan de implementación y para comunicar dicho plan a todo un equipo de desarrolladores (Cáceres ,2015).

1.5.1 Lenguaje unificado del modelado

El Lenguaje Unificado del Modelado (UML) se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software. Además, es utilizada para entender, diseñar, configurar, mantener y controlar la información sobre los sistemas que se desean desarrollar.

Está pensado para usarse con todos los métodos de desarrollo, etapas del ciclo de vida y dominios de aplicación. La especificación de UML no define un proceso estándar, pero está pensado para ser útil en un proceso de desarrollo iterativo. Pretende dar apoyo a la mayoría de los procesos de desarrollo orientados a objetos (Cáceres,2015). En otras palabras, es el lenguaje en el que está descrito el modelo.

1.6 Lenguaje de programación

Un lenguaje de programación es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora; puede ser usado para controlar el comportamiento de una máquina, o para definir una secuencia de instrucciones para su procesamiento (PHP,2015).

1.6.1 Java en su versión 8.0

Lenguaje de programación de propósito general que es concurrente, basado en clases, orientado a objetos y específicamente diseñado para tener pocas dependencias de implementación como sea posible. Su intención es permitir que los desarrolladores de aplicaciones escriban el programa una vez y lo ejecuten en cualquier dispositivo. *Java* es también el lenguaje principal a la hora de desarrollar aplicaciones nativas de *Android* para teléfonos inteligentes y tabletas. Posee mecanismos para garantizar la seguridad durante la ejecución, comprobando antes de ejecutar código, que este no viole ninguna restricción de seguridad del sistema donde se va a ejecutar (Dominguez,2005).

1.6.2 Tecnologías para el Marco de Trabajo de Desarrollo

***JavaFX* está basado en Java.** La plataforma *JavaFX* permite a los desarrolladores de la aplicación crear e implementar fácilmente Aplicaciones de *Internet* Enriquecidas (RIA) que se comportan de la misma forma en distintas plataformas. *JavaFX* amplía la potencia de *Java* permitiendo a los desarrolladores utilizar cualquier biblioteca de *Java* en aplicaciones *JavaFX*. Los desarrolladores pueden ampliar sus capacidades

en *Java* y utilizar la tecnología de presentación que *JavaFX* proporciona para crear experiencias visuales que resulten atractivas. Permite el desarrollo de aplicaciones web que tienen las características y capacidades de aplicaciones de escritorio, incluyendo aplicaciones multimedia interactivas (Hildebrandt,2014).

1.7 Entorno de desarrollo

Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDE pueden ser aplicaciones por sí solas o ser parte de aplicaciones existentes. Es posible que un mismo IDE pueda funcionar con varios lenguajes de programación.

Un IDE debe tener las siguientes características:

- Multiplataforma
- Soporte para diversos lenguajes de programación
- Integración con Sistemas de Control de Versiones
- Reconocimiento de Sintaxis
- Extensiones y Componentes para el IDE
- Integración con Framework populares
- Depurador
- Importar y Exportar proyectos
- Múltiples idiomas
- Manual de Usuarios y Ayuda

Entorno de desarrollo para la aplicación del profesor

NetBeans v8.0 es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje de programación *Java*. Permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de *software* llamados módulos. Un módulo es un archivo *Java* que contiene clases para interactuar con las APIs de *NetBeans* y un archivo especial (*manifest fil*e) que lo identifica como módulo. Existe además un número importante de módulos para extenderlo. (Netbeans,2016).

Entorno de desarrollo para la aplicación de los estudiantes

Android Studio es un IDE oficial para el desarrollo de aplicaciones para *Android* y se basa en *IntelliJ IDEA*. Además del potente editor de códigos y las herramientas para desarrolladores de *IntelliJ, Android Studio* ofrece aún más funciones que aumentan la productividad durante la compilación de aplicaciones para *Android*, como las siguientes:

- ✓ Sistema de compilación flexible basado en *Gradle*.
- ✓ Un emulador rápido con varias funciones.
- ✓ Un entorno unificado en el que puedes realizar desarrollos para todos los dispositivos *Android*.
- ✓ Integración de plantillas de código y *GitHub*, para ayudarte a compilar funciones comunes de las aplicaciones e importar ejemplos de código.
- ✓ Gran cantidad de herramientas y *frameworks* de prueba.
- ✓ Herramientas *Lint* para detectar problemas de rendimiento, uso, y compatibilidad de versión (Android,2014).

1.8 Herramienta de modelado Visual Paradigm

Las herramientas CASE (*Computer Aided Software Engineering*, o Ingeniería de *Software* Asistida por Computadora) son un conjunto de métodos, utilidades y técnicas que ayudan al desarrollo de *software*. Aumentan la productividad y logran una reducción en el costo de tiempo y dinero. Al utilizar estas herramientas se puede abstraer al código en un nivel donde la arquitectura y el diseño son más fáciles de entender y modificar (Pressman,2002).

Visual Paradigm v8.0 es una herramienta que utiliza como lenguaje de modelado UML. Soporta el ciclo de vida completo del desarrollo de *software*: análisis y diseño orientados a objetos, construcción, pruebas y despliegue.

El *software* de modelado UML ayuda a una más rápida construcción de aplicaciones de mayor calidad y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (Pressman,2002).

Se ha escogido *Visual Paradigm* como herramienta de modelado debido a la constante sincronización del modelo de diseño y el código fuente durante todo el ciclo de desarrollo del software y soporte para UML.

1.9 Herramienta para el control de versiones

Un sistema de control de versiones permite guardar “fotografías” del estado del proyecto en ese instante del tiempo, dándote la capacidad de restaurar ese estado en cualquier momento. Es simple: tomas una de las “fotos”, trabajas en el proyecto y si algo sale mal puedes volver a atrás, a algún punto donde todo funcionaba (Julio, 2015).

El control de versiones es la gestión de los diversos cambios que se realizan sobre los elementos de algún producto o una configuración del mismo. La versión de un producto es el estado en el que se encuentre el mismo en un documento dado de su desarrollo o modificación (Pilato,2004).

Git en su versión 8.16.1

Software de control de versiones diseñado por *Linus Torvalds*, pensando en la eficiencia y la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente (Git,2016). Es un sistema de control de versiones distribuido que no depende de acceso a la red o un repositorio central.

Cualidades de Git:

- ✓ Diseñado para manejar proyectos grandes.
- ✓ Es extremadamente rápido.
- ✓ Autenticación criptográfica del historial.
- ✓ Formato de archivo muy sencillo y compacto.
- ✓ Se puede sincronizar por cualquier medio.

Conclusiones parciales

Como parte del desarrollo del presente capítulo se establecen las siguientes conclusiones parciales:

- La identificación de los conceptos relacionados con las aulas tecnológicas ATcnea, contribuyó a un mejor entendimiento del entorno en que se desenvuelve la investigación.
- El estudio de sistemas similares arribó a que estos no brindan aplicaciones de transmisión de voz como aplicaciones nativas del software, trayendo como desventaja para cualquier institución, ya que no pueden establecer diálogos a través de voz entre el profesor y los estudiantes. Por lo que se demuestra la necesidad de dicho módulo.
- Se describieron las herramientas, tecnologías y la metodología que se utilizará en el desarrollo de la propuesta de solución.

Capítulo 2: Descripción de la solución propuesta

2.1 Introducción

En el presente capítulo se muestra el modelo de dominio para relacionar los conceptos asociados a la investigación. Se realiza el levantamiento y especificación de los requisitos funcionales y no funcionales, para el desarrollo del módulo. Se plantea, además, la propuesta de solución al problema principal de la presente investigación. Se describe la arquitectura a utilizar y los patrones de diseños empleados para el desarrollo de esta propuesta.

2.2 Descripción de la propuesta solución

El módulo de grabación y transmisión de audio para el *software* del aula tecnológica ATcnea, permitirá tanto como para el profesor como para el estudiante la transmisión de voz en tiempo real, ambos usuarios tendrán la posibilidad de grabar la voz y transferirse archivos entre ellos. El módulo se divide en dos submódulos, uno para la aplicación del estudiante desarrollado en *Android* y un submódulo para la aplicación del profesor desarrollada en *Java*. A continuación, se realiza la descripción de las principales funcionalidades de la propuesta de solución:

Funcionalidad grabar voz:

Esta funcionalidad a través del uso del paquete *Sound* propio de *JavaFX* permitirá grabar la voz utilizando métodos de la clase *AudioRecorder*, el mismo permitirá crear archivos de audio, así como las funcionalidades de comenzar y detener la grabación. Una vez comenzada dicha grabación se crea un archivo en el cual se va grabando la voz emitida, luego de detenerse se guarda con el nombre que defina el usuario en la carpeta correspondiente en el almacenamiento interno del dispositivo. Además, se actualiza el registro que contiene todas las grabaciones realizadas.

Funcionalidad *streaming* de voz y audio:

Haciendo uso de la clase *Media Player* y la librería *VLCJ*, las cuales permiten realizar la transmisión de voz entre ambos submódulos para lograr la comunicación. La clase *HeadlessMediaPlayer* permite a los estudiantes realizar la petición de hablar, donde el profesor será el encargado de aceptar o no dicha petición.

También cuando el profesor desee dirigirse a los estudiantes solo tendrá que accionar en el botón de transmisión, y se mostrara en los estudiantes una pantalla que representa la transmisión la cual podrá detener cuando desee. Además, el profesor podrá reproducir un audio a todos los estudiantes o a varios estudiantes de la clase presionando el botón *play,* para comenzar la reproducción este debe haber seleccionado el audio que va a reproducir. Mientras que en la pantalla de los estudiantes se mostrará una imagen que represente la reproducción del audio y cuando el profesor presione el botón *stop* se quitará automáticamente la imagen que visualizaba el estudiante.

Funcionalidad enviar archivo de audio:

Tanto como para el estudiante como el profesor se utilizan la misma vía para la transferencia de archivos, para ello se utiliza la clase *Socket* de *Java*. Esta clase permite realizar la acción de envío, además de que es necesario que uno funcione como servidor y otro como cliente dependiendo de la transferencia que se vaya a realizar. También se envía una notificación de confirmación una vez que se haya completado el enviado de dichos paquetes.

2.3 Modelo de dominio

El modelo de dominio es una representación visual de los conceptos u objetos que se manejan en el dominio del sistema. Se utiliza para capturar y expresar el entendimiento ganado en un área bajo análisis como paso previo al diseño de un sistema. Este puede ser usado como punto de partida para el diseño del sistema. (Larman,2005).

2.3.1 Descripción de los objetos asociados al dominio

Para la posterior confección del modelo, se procede a esclarecer las terminologías asociadas al dominio, definiendo los términos más importantes que serán de utilidad.

- **Estudiante:** usuario que cumple con los requisitos funcionales basados en el sistema operativo *Android* haciendo función de cliente; o sea, que se va a encargar de recibir todas las operaciones que sean orientadas por el profesor.
- **Profesor:** usuario encargado de montar el aula tecnológica y que controla el acceso a cada uno de los usuarios estudiantes haciendo uso de la aplicación ATcnea de ordenador.

- **Aula tecnológica:** espacio o lugar diseñado y preparado para cada profesor o conferencista imparta clase a su grupo de estudiantes que recibirán la clase a impartir.
- **Audio:** recurso que se utiliza para la comunicación entre el profesor y el estudiante.
- **Grabación:** funcionalidad que debe cumplir el aula tecnológica, tiene que almacenar el audio grabado en forma de archivo para poder enviarlo o reproducirlo.
- **Transmisión de audio:** funcionalidad que debe cumplir el aula tecnológica, tiene que ser capaz de transmitir audio en tiempo real.

2.3.2 Diagrama del modelo de dominio

El siguiente diagrama muestra la relación entre los conceptos identificados del problema descrito anteriormente.

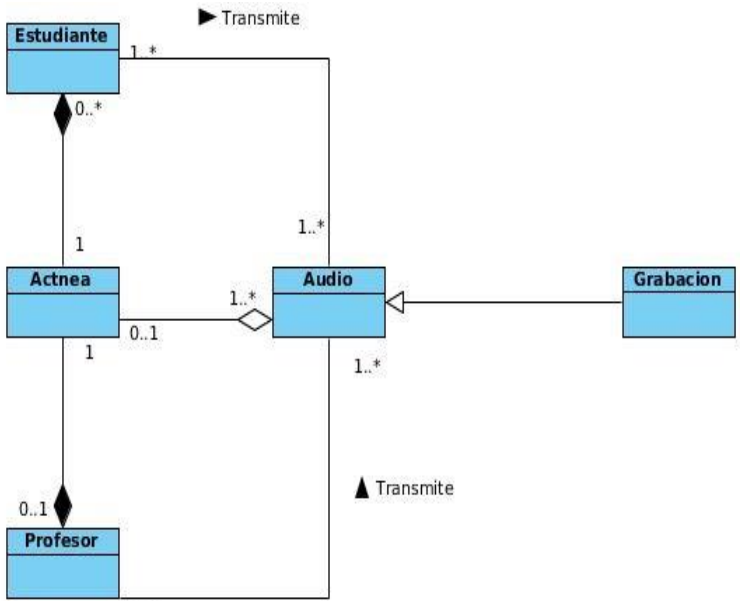


Ilustración 1. Modelo de dominio

2.4 Requisitos del sistema

El proceso de desarrollo de *software* comprende en sus etapas tempranas la definición de tareas orientadas a captar las necesidades o características para satisfacer el sistema que se vaya a crear o modificar. Como resultado de esta captación, se obtendrán los requisitos con los que deberá cumplir la solución (Roger,2010).

2.4.1 Requisitos funcionales

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares (Olivera,2010). A continuación, se listan los requisitos funcionales identificados:

-**RF1 Grabar voz del estudiante:** al presionar el botón grabar de la aplicación de audio del estudiante, esta comenzará a grabar la voz del estudiante.

-**RF2 Almacenar grabación de voz del estudiante:** al presionar el botón detener, la grabación se almacena en la memoria del *Tablet*.

-**RF3 Enviar grabación de voz del estudiante al profesor:** al presionar el botón enviar de la aplicación de audio del estudiante, esta enviará la pista de audio grabada al profesor.

-**RF4 Grabar voz del profesor:** al presionar el botón grabar de la aplicación de audio del profesor esta comenzará a grabar la voz del profesor.

-**RF5 Almacenar grabación de voz del profesor:** al presionar el botón detener, la grabación se almacena en la memoria del ordenador y se muestra en la interfaz del módulo del profesor.

-**RF6 Enviar grabación de voz del profesor al estudiante:** el profesor tendrá que seleccionar previamente el audio a enviar**;** al presionar el botón enviar esta enviará la pista de audio grabada al estudiante.

-**RF7 Transmitir audio del profesor a los estudiantes:** al presionar el botón de transmisión, el profesor podrá comenzar a hablar y orientar verbalmente cualquier actividad que desee al estudiante de manera oral.

-**RF8 Transmitir audio de los estudiantes al profesor:** al presionar el botón de transmisión, el estudiante podrá comenzar a hablar y responder cualquier actividad orientada por el profesor.

2.4.2 Requisitos no funcionales

Los requerimientos no funcionales son requisitos que imponen restricciones en el diseño o implementación. Son propiedades o cualidades que el *software* debe tener.

Tabla 4. Requisitos no funcionales

No	Requisitos no funcionales	Descripción
1	Usabilidad	La aplicación tanto del estudiante como del profesor debe poseer una interfaz intuitiva para cualquier tipo de usuario con conocimientos básicos de informática, en el manejo de ordenadores y móviles *Android*.
2	Hardware	Permitir la instalación de la herramienta en Tablet que cuenten con NovaDroid. Se recomienda 1 GB RAM o superior, 7 pulgadas o superior de pantalla, Procesador Dual Core 1.0GHz o superior. Permitir la instalación de la herramienta en computadoras que cuenten con sistema operativo Nova 2015 o superior. Se recomienda que tenga 4GB RAM o superior, CPU Core 2 E6300 o superior, 1GB tarjeta de video o superior.
3	Interfaz	Cumplir con el diseño aplicado en ATcnea que contenga colores como azul y amarillo.
4	Seguridad	Garantizar el acceso a las funcionalidades definidas para los usuarios de acuerdo al tipo de usuario (profesor y estudiante).

2.4.3 Descripción de historias de usuarios

Una Historia de Usuario o (HU) es una forma rápida de administrar los requisitos de los usuarios sin tener que elaborar gran cantidad de documentos formales, comúnmente utilizada en las metodologías de desarrollo ágiles para la especificación de requisitos. (Reynoso,2004)

En el cuarto escenario de AUP, en su variante UCI, los requisitos se administran utilizando HU, por lo que para la solución propuesta se describieron un total de 8 HU, de las cuales se presenta la correspondiente al RF: Grabar voz del estudiante. Las restantes se podrán encontrar en el **Anexo I.**

Tabla 5. Historia de Usuario RF 1

Número: 1	Nombre del Requisito: Grabar la voz del estudiante.	
Programador: Daniel Blanco Isidrón		**Iteración Asignada:** 1era
Prioridad: Alta		**Tiempo Estimado:** N/A
Riesgo en Desarrollo: N/A		**Tiempo Real:** N/A

Descripción:

El usuario en este caso tendrá la opción de presionar en el botón grabar e inmediatamente dicha aplicación móvil comenzará a grabar la voz del estudiante.

1- Objetivo:

Grabar la voz del estudiante.

2- Acciones para lograr el objetivo (precondiciones y datos):

Para grabar la voz de un estudiante al utilizar la aplicación de grabación hay que:

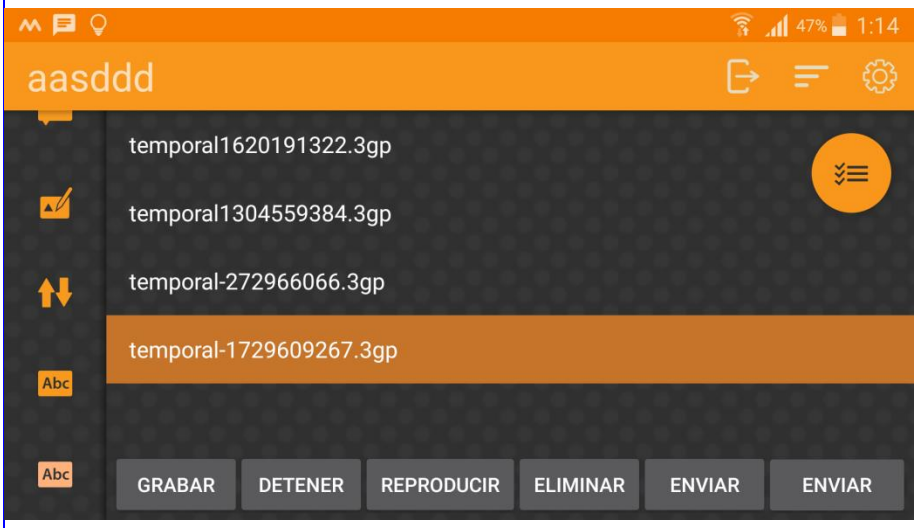
- Tener un Tablet con la aplicación de ATcnea instalada.
- Estar conectado al aula tecnológica creada.
- Acceder a la opción de grabar audio.
- Debe comenzar a hablar para poder grabar el audio.

3- Comportamientos válidos y no válidos (flujo central y alternos):

4- Flujo de la acción a realizar:

- El sistema debe permitir grabar una voz, esta acción puede realizarse presionando el botón grabar de la aplicación *Android* del estudiante.
- Cuando el usuario presiona el botón grabar se muestra un contador de tiempo para la grabación.

Prototipo de interfaz:



2.5 Patrón Arquitectónico

Los patrones arquitectónicos, o patrones de arquitectura, ofrecen soluciones a problemas de ingeniería de software. Reflejan una descripción de los elementos y el tipo de relación que tienen, seguido de un conjunto de restricciones sobre cómo pueden ser usados. (Pressman, 2005).

2.5.1 Patrón Modelo-Vista-Controlador

El patrón Modelo-Vista-Controlador o MVC es un patrón de arquitectura de *software* encargado de separar la lógica de negocio de la interfaz del usuario, ya que facilita la funcionalidad, mantenibilidad y escalabilidad del sistema, de forma simple y sencilla, a la vez que permite no mezclar lenguajes de programación en el mismo código (Pressman,2010).

Como base para el desarrollo de la aplicación propuesta se utilizó el marco de trabajo *JAVA FX* y en el desarrollo de la aplicación *Android* que implementa el patrón arquitectónico Modelo-Vista-Controlador (MVC). Este estilo de arquitectura separa los datos de una aplicación, interfaz de usuario, y lógica de control en tres componentes distintos:

Modelo: recoge la información (la lógica de la aplicación). Por ejemplo, la base de datos. Es la parte más reutilizable, se puede portar fácilmente todo el modelo de una aplicación a otra.

Vista: la vista es la parte más sencilla de entender, porque se refiere a los *layouts*, a lo que el usuario ve por pantalla en cuánto ejecuta la aplicación. Lenguaje XML en *Android*.

Controlador: el controlador es la "clave" de la aplicación. Por ejemplo, las funcionalidades presentes en una aplicación. Toda la maquinaria que hace algo al ejecutarla. Ej. el código del botón que detiene el audio.

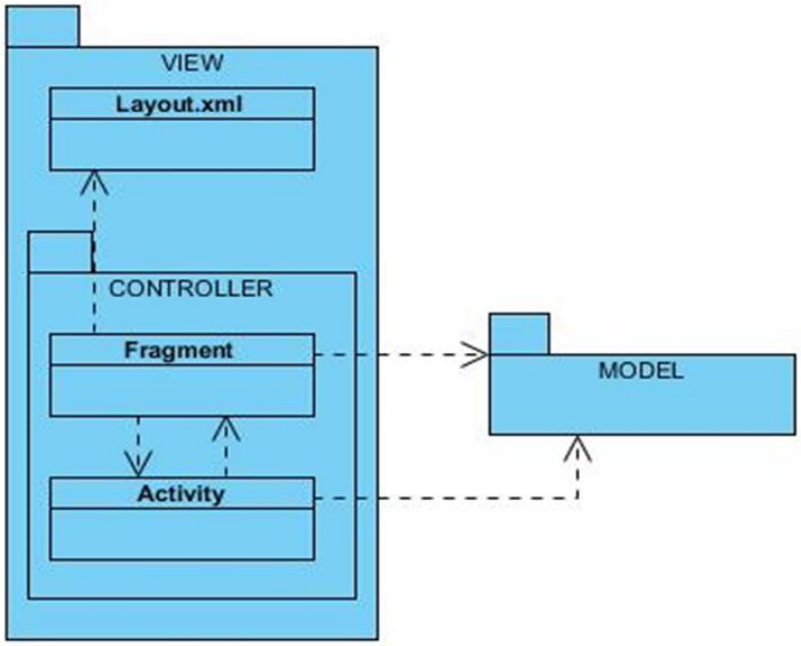


Ilustración 2. Modelo-Vista-Controlador

2.6 Patrones de diseño

Los patrones de diseño son soluciones a problemas repetidos en la construcción de *software* y en ocasiones pueden incluir sugerencias para aplicar estas soluciones en diversos entornos. (Michael,2010)

De los patrones de diseño estudiados, se seleccionaron los considerados que era pertinente utilizar en desarrollo del módulo.

2.6.1 Patrones GRASP

Los Patrones de Asignación de Responsabilidades (GRASP por sus siglas en inglés) describen los principios fundamentales del diseño de objetos y la asignación de responsabilidades, expresados como patrones (*Carmona,2012)*.

- **Creador:** se encarga de guiar la asignación de responsabilidades relacionadas con la creación de objetos. Su intención es encontrar un creador que necesite conectarse al objeto creado en alguna situación.

```
public class AudioRecordController extends AnchorPane implements Initializable,
ControlledScreen {

Hilo hilo;

……..

public void initialize(URL location, ResourceBundle resources) {

……..

hilo= new Hilo();

}

}
```

- **Bajo acoplamiento:** la colaboración entre clases debe mantenerse en un mínimo aceptable. Es la idea de tener las clases lo menos ligadas entre sí que se pueda de tal forma que en caso de producirse una modificación en alguna de ellas, se tenga la mínima repercusión posible en el resto de clases.

```
public class AudioRecordController extends AnchorPane implements Initializable,
ControlledScreen {

……..

soloButton.setOnMouseClicked(event -> {

   if(solo) {

……

            soloButton.setTooltipText("Para uno solo");

   }else{

       ……

       soloButton.setTooltipText("Para todos");

   }

});

}
```

- **Alta cohesión**: las responsabilidades que almacena una clase deben ser pequeñas y enfocadas. Los métodos y atributos deben ser de ser sencillos para implementar dichas responsabilidades.

```
public class AudioObservable extends CheckBoxModel {

.....

public void setId(int id) {

    this.id.set(id);

  }

  public String getUrl() {

    return url.get();

  }

  public void setUrl(String url) {

    this.url.set(url);
```

- **Controlador:** establece una clara separación entre la interfaz de usuario y núcleo de procesamiento de la aplicación, donde se halla la lógica de negocios. La idea básica es crear una clase que implemente métodos dedicados a escuchar o atender los eventos del sistema (Larman,2005).

```
public class AudioRecordController extends AnchorPane implements Initializable,
ControlledScreen {

Hilo hilo;

……….

public void initialize(URL location, ResourceBundle resources) {

……….

hilo= new Hilo();

}

}
```

2.6.2 Patrones GOF

Los patrones GOF favorecen la reutilización de código y ayudan a construir *software* basados en la reutilización (Gamma,1995).

- **Singleton:** garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. Es un patrón de diseño para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella, dicho objeto se ve reflejado en el objeto *player* de la clase *Media Player* dentro de la clase controladora AudioRecorderFragment.

2.7 Modelado del diseño

En el diseño se modela el sistema y se encuentra la forma de que soporte todos los requisitos, incluyendo los requisitos no funcionales y cualquier otra restricción. El modelo de diseño crea un punto de partida para las actividades de implementación subsiguientes. Permite descomponer los trabajos de implementación en

partes más manejables que puedan ser llevadas a cabo por diferentes equipos de desarrollo. Da una forma al sistema mientras que intenta preservar la estructura definida por el modelo de análisis (*Jacobson,2000)*.

2.7.1 Diagrama de clase de diseño

Representa todas las clases del diseño, subsistemas, paquetes, colaboraciones y las relaciones entre ellos, constituyendo la entrada principal a las actividades de la fase de implementación (*Jacobson,2000)*.

A continuación, se muestra el diagrama de clase del diseño correspondiente a la **RF1** y **RF4**. Para el estudio de los demás diagramas remitirse al **Anexo 3**.

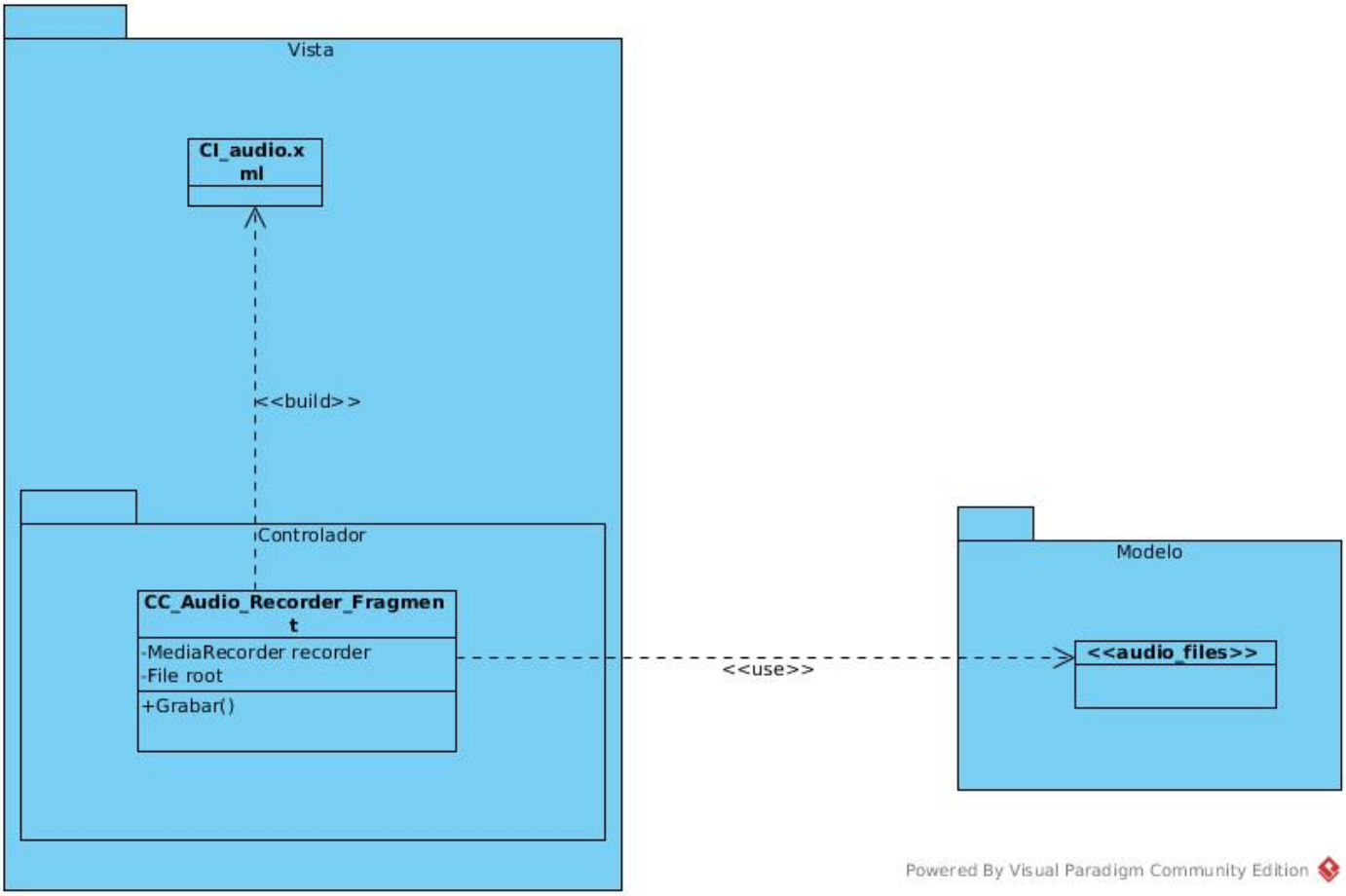


Ilustración 3. Diagrama de clase de diseño grabar voz del estudiante

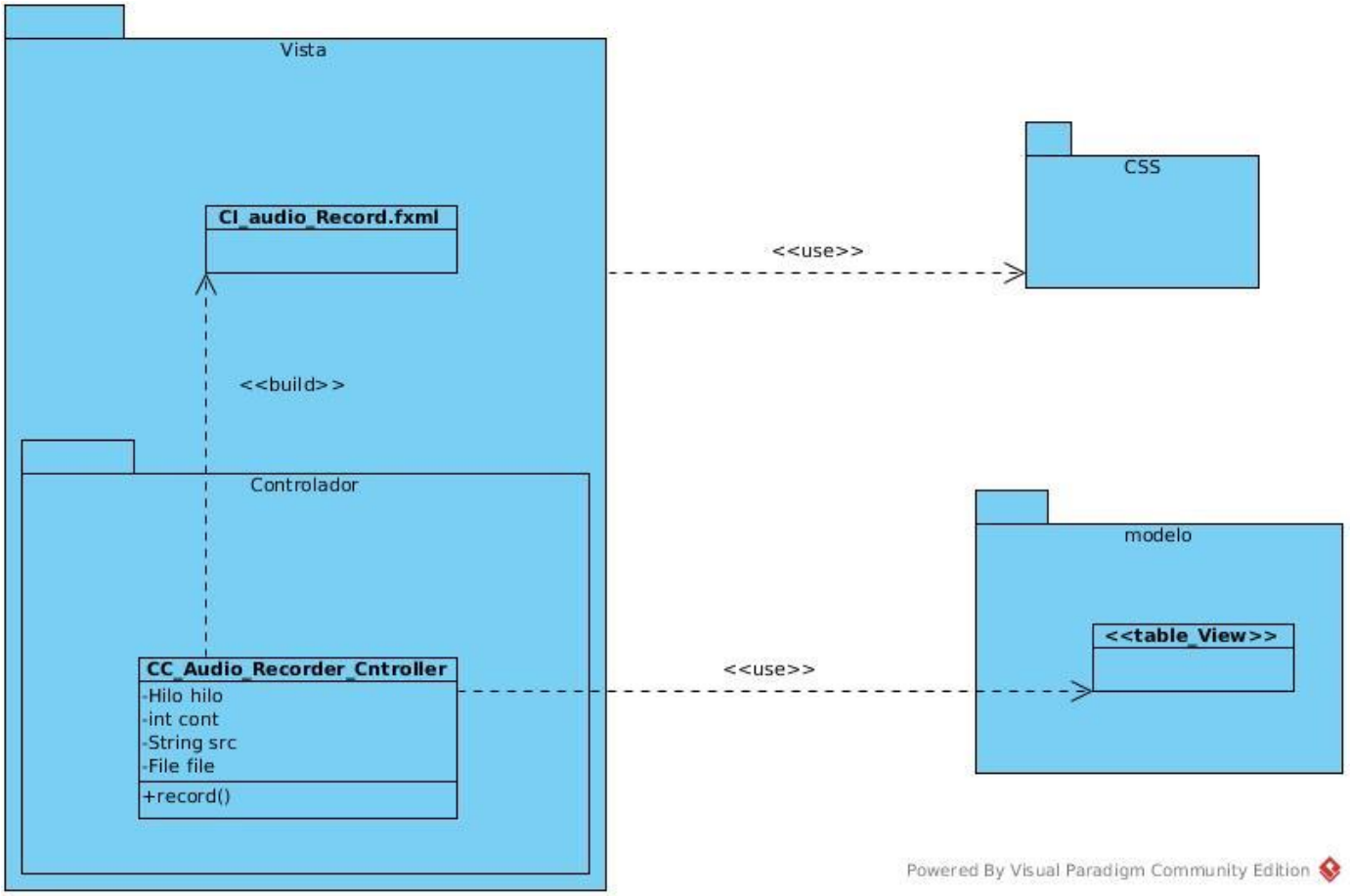


Ilustración 4. Diagrama de clase de diseño grabar voz del profesor

2.7.2 Diagrama de secuencia

Los diagramas de secuencia ilustran las interacciones entre objetos con el transcurso del tiempo. Estos muestran los objetos participantes de la interacción y la secuencia de los mensajes intercambiados (*Jacobson,2000)*.

A continuación, se muestran los diagramas de secuencias correspondientes a las HU: Grabar voz del estudiante, los restantes diagramas se pueden consultar en el **Anexo 2.**

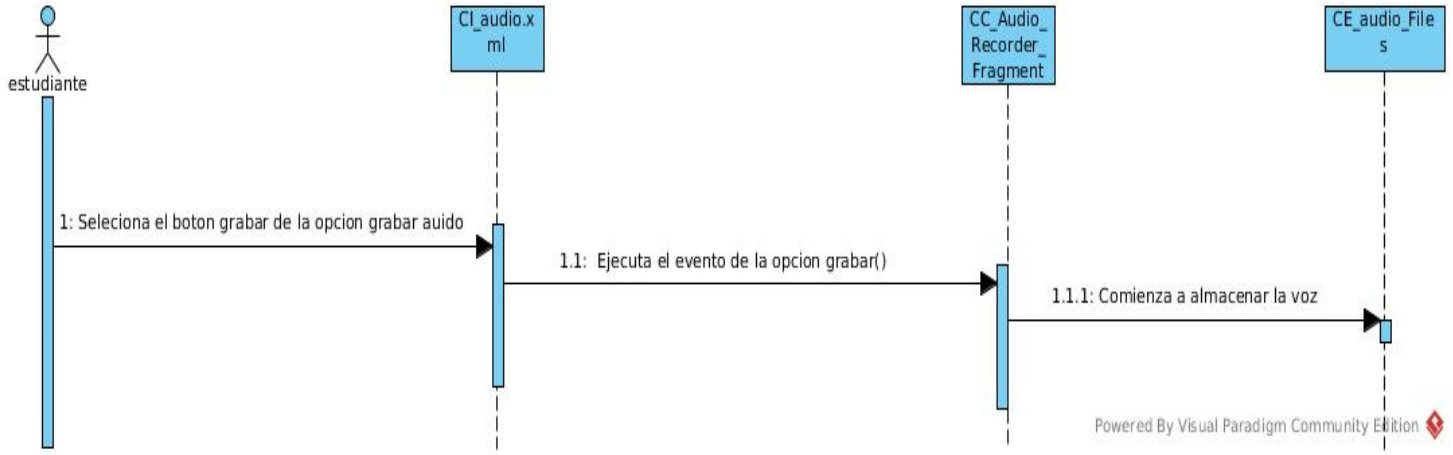


Ilustración 5. Diagrama de secuencia grabar voz del estudiante

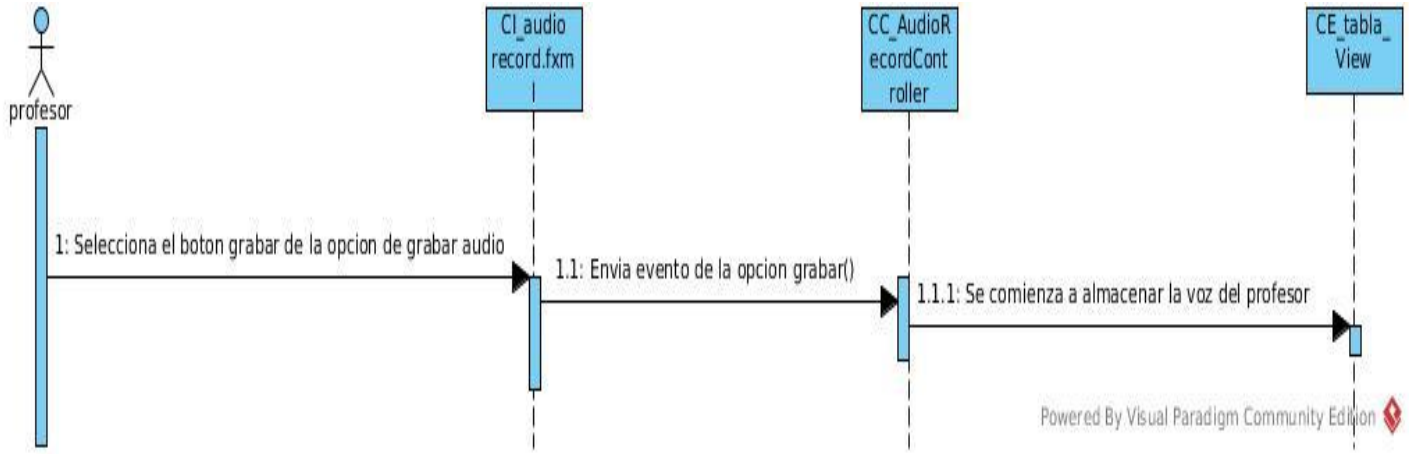


Ilustración 6. Diagrama de secuencia grabar voz del profesor

2.8 Diagrama de despliegue

El diagrama de despliegue modela la arquitectura en tiempo de ejecución y muestra la disposición física de los nodos que componen el sistema. El mismo presenta la configuración de los elementos de hardware (nodos) y muestra como los elementos y artefactos del software se trazan en esos nodos y se encuentran conectados por enlaces de comunicación. (Roger,2010)

Capítulo 2: Descripción de la propuesta de solución

Para establecer la transmisión de datos dentro del aula tecnológica se hace uso de los protocolos UDP y TCP. La comunicación entre el estudiante y el profesor se realiza por el protocolo de transmisión UDP y en el caso del profesor con el estudiante por el protocolo TCP.

TCP (*Transmission Contorl Protocol*), el fin de TCP es proveer un flujo de *bytes* confiable de extremo a extremo sobre una *internet* no confiable. TCP puede adaptarse dinámicamente a las propiedades de la *internet* y manejar fallas de muchas clases (Julio,2010).

UDP (*User Datagram Protocol*, protocolo de datagrama de usuario) permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera.

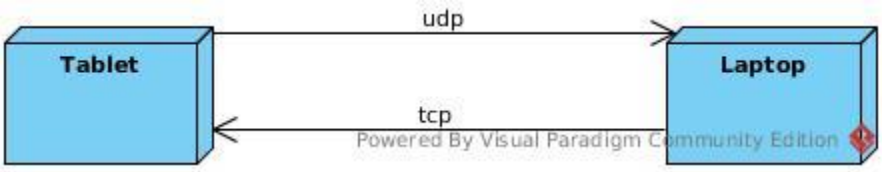


Ilustración 7. Diagrama de despliegue

2.9 Conclusiones parciales

En el presente capítulo se llevó a cabo el diseño del sistema a partir de la metodología de trabajo AUP en su variante UCI arrojando los siguientes resultados:

- La identificación de los requisitos funcionales y no funcionales del sistema, y su administración mediante historias de usuario, sirvió de guía para la implementación de las distintas funcionalidades de la herramienta propuesta.

- El modelo del diseño obtenido permitió describir la realización física y dinámica de las historias de usuario y crear de esta manera, una entrada apropiada para las actividades de implementación.

- La selección de los patrones arquitectónicos y de diseño aportó sencillez al proceso de mantenimiento y permitirá la obtención de una herramienta robusta.

- Los artefactos generados durante el diseño de la solución contribuyeron al mejor entendimiento del sistema para dar paso a la implementación de la solución propuesta.

Capítulo 3: Implementación y prueba de la propuesta de solución

Capítulo 3. Implementación y prueba de la propuesta solución

3.1 Introducción

En este capítulo se implementará el módulo de grabación y transmisión de audio para el *software* del aula tecnológica ATcnea. Se realiza el diagrama de componentes del sistema y se definen los estándares de codificación empleados. Se definen los niveles y técnicas de pruebas que se le serán aplicadas a dicho módulo. Además, se realizan las pruebas al *software* utilizando los casos de pruebas que servirán para llevar a cabo las comprobaciones de calidad.

3.2 Modelo de implementación

El proceso de implementación parte como resultado del análisis y diseño de la propuesta de solución planteada, se tiene como objetivo llevar la arquitectura y el sistema como un todo. Describe como se organizan los componentes de acuerdo con los mecanismos de estructuración y modelación disponibles en el entorno de implementación. Además de los lenguajes de programación utilizados, y cómo dependen los componentes unos de otros (*Jacobson,2000)*. Como parte del modelo de implementación se obtienen los diagramas de componentes que a se presentan en el siguiente epígrafe.

3.3 Diagrama de componentes

El diagrama de componentes muestra los elementos de diseño de un sistema de software. Permite visualizar con facilidad la estructura general del sistema y el comportamiento de los servicios que estos componentes.

Los diagramas de componentes se utilizan para modelar la vida estática de un sistema. Muestra las organizaciones y las dependencias entre un conjunto de componentes de software. Además, organiza los subsistemas de implementación en capas. Donde los componentes constituyen su elemento central, un componente es el empaquetamiento físico de los elementos de un modelo, como son las clases en el modelo de diseño (*Jacobson,2000)*.

Capítulo 3: Implementación y prueba de la propuesta de solución

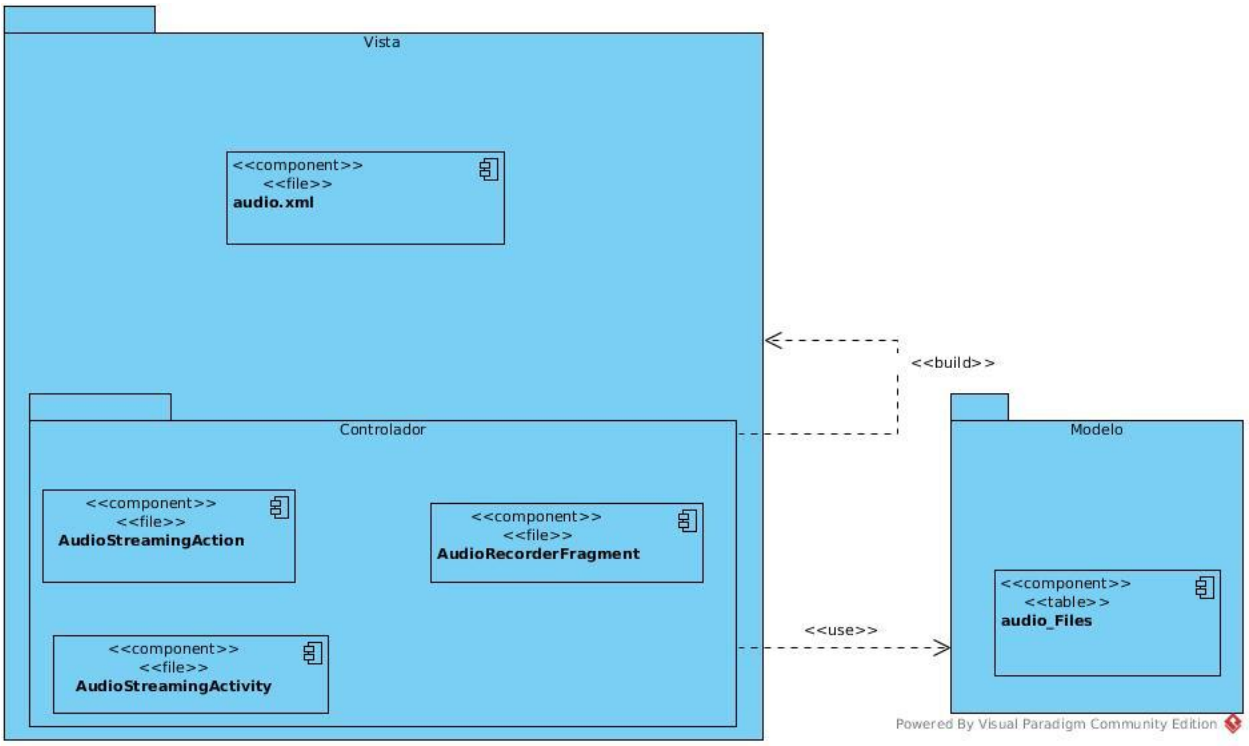


Ilustración 8. Diagrama de componentes del estudiante

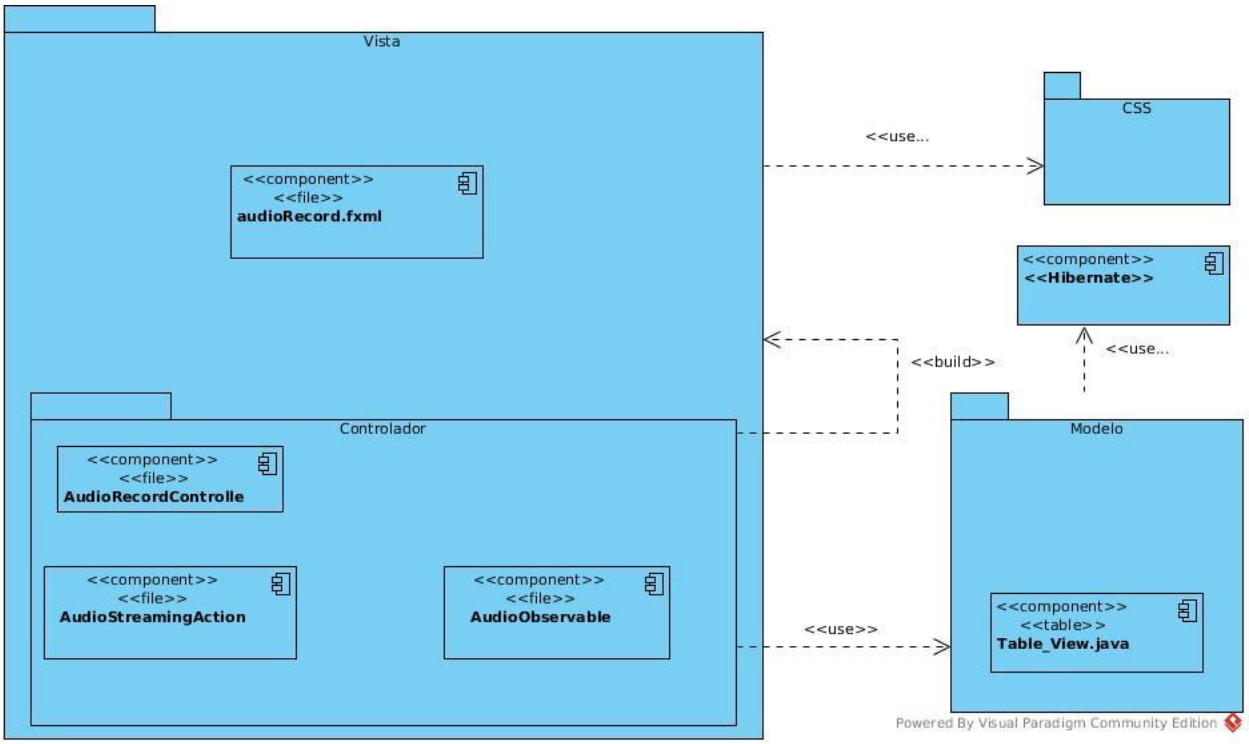


Ilustración 9 Diagrama de componentes del profesor

Capítulo 3: Implementación y prueba de la propuesta de solución

3.4 Estándares de codificación

Un estándar de codificación se refiere a los aspectos relacionados con la generación de código. La legibilidad del código fuente no influye en la capacidad funcional de la aplicación, pero sí, en la forma en que un programador pudiera comprender el sistema. El uso de buenas técnicas de programación y codificación sólida ayuda a generar un código de alta calidad, lo cual influye en la calidad del *software*.

Paquetería

Los nombres de los paquetes estarán dados de la siguiente manera: *com.empresa.nombreaplicacion.modulo* siendo esta la ruta básica.

Cuando se crea otro paquete deberá ir dentro del paquete módulo con un nombre significativo y en minúscula. Aquellos paquetes cuyo nombre está compuesto por dos palabras deberán estar unidos y en minúscula, ejemplo, *com.empresa.modulo.nombreaplicacion.comprasporpagar*

Nombre de actividades o clases Java

Cada actividad debe contener al inicio el nombre del autor, la fecha creada, y una descripción breve de su funcionalidad.

/*

* esta actividad hace el llamado a la lista de contactos

*/

Todo nombre de una actividad debe iniciar en mayúscula *Actividad.java*

Para aquellos nombres que sean compuesto de dos palabras, cada palabra deberá iniciar con la primera letra en mayúscula, ejemplo: NumeroAleatorio.java

public class NumeroAleatorio{

…

}

Creación de métodos

Los nombres de los métodos deberán ser significativos y deberán nombrarse en minúscula.

public void metodo(){

….

}

En caso de que el nombre sean dos palabras o más, deberán estar unidas y la primera letra de cada palabra deberá estar en mayúscula.

public void NombreCompuesto(){

….

}

Cada método debe estar comentado antes de crearlo, describiendo brevemente su funcionalidad.

Public void metodo(){

/*

* Comentario sobre el método

*/

}

El formato dado para comentarios de más de una línea es:

/*

* comentario

*/

En caso de que el comentario sea de una sola línea deberá utilizarse dos barras inclinadas, ejemplo:

// comentario

Sentencias

Normas básicas son:

- Una sentencia por línea de código.
- Todo bloque de sentencias entre llaves, aunque sea una sola sentencia después de un *if*.

La declaración de los bucles *for* serán usualmente de la forma:

for (int i=0; i < condición; i++)

Son obligatorias las tres condiciones del bucle *for*: inicialización, condición de finalización y actualización del valor de la variable de avance.

La variable de avance del bucle nunca podrá ser modificada dentro del propio bucle.

3.5 Pruebas de software

La fase de pruebas del sistema tiene como objetivo verificar el *software* para comprobar si este cumple sus requisitos. Dentro de esta fase pueden desarrollarse varios tipos de pruebas en función de los objetivos de las mismas (Hernández,2014).

Las pruebas de *software* son un elemento importante dentro del ciclo de vida de un proyecto, que permiten detectar defectos en el software, verificar que todos los requisitos se hayan implementado correctamente y verificar la integración adecuada de los componentes.

3.5.1 Niveles de pruebas

Los niveles de pruebas son diferentes formas de verificar y validar un producto de software. A continuación, se plantean los tipos de pruebas empleados para

mostrar la calidad del módulo desarrollado. Se pretende además verificar y evaluar las funcionalidades del mismo.

Prueba de Sistema: son las pruebas que verifican el comportamiento del sistema en su conjunto. Funcionan para verificar que los elementos del sistema se hayan integrado de manera adecuada y que se realicen las funciones asignadas. De manera general este nivel de prueba es preparado y ejecutado por un grupo independiente al desarrollador, y consiste en validar que el *software* cumpla con los requerimientos especificados por el cliente. Las pruebas al sistema se realizan con la técnica de caja negra (Roger,2010).

Prueba de Integración: es una técnica sistemática para construir la estructura del programa mientras que al mismo tiempo se lleva a cabo pruebas para detectar errores asociados a la interacción. Es ejecutada para garantizar que en el modelo de implementación los componentes operen correctamente cuando son combinadas para ejecutar un requisito funcional. Es realizada también para el desarrollo del *software* y consiste fundamentalmente en validar la integración entre dos o más componentes de *software* haciendo uso de la técnica de caja blanca. Para realizar pruebas de integración a un sistema es necesario establecer una estrategia de integración. A continuación, se muestran dichas estrategias (Maria,2010):

- De *big-bang*: se integran todos los componentes y entonces se prueba el sistema como un todo. Esta estrategia se basa en acoplar todos los módulos del proyecto de una vez con el objetivo de reducir la cantidad de pruebas.
- De arriba abajo (*top-down*): consiste en empezar la integración y la prueba por los módulos que están en los niveles superiores de abstracción, e integrar incrementalmente los niveles inferiores.
- De abajo a arriba (*bottom-up*): consiste en empezar la integración y la prueba por los módulos que están en los niveles inferiores de abstracción, e integrar incrementalmente los niveles superiores.

Prueba de aceptación: se realiza una vez que el sistema se ha implantado en su entorno real de funcionamiento, y su objetivo es demostrar al usuario que el sistema satisface sus necesidades (Pressman,2010).

3.5.2 Métodos de pruebas

Un método de prueba es un procedimiento definitivo que produce un resultado de prueba. Se puede probar cualquier producto de ingeniería en dos formas: conociendo la función específica para la cual fue diseñado el mismo y conociendo el funcionamiento del producto. Al primer enfoque se le denomina prueba de caja negra y al segundo, prueba de caja blanca (Roger,2010).

Pruebas de caja blanca: es donde se comprueban los componentes internos. Según Pressman, se basan en un examen detallado de los procedimientos y caminos lógicos del sistema para determinar si el estado real coincide con el esperado (Roger,2002).

Pruebas de caja negra: son nominadas pruebas funcionales o de comportamiento y se centran en los requisitos funcionales del software. Se llevan a cabo sobre la interfaz del *software* buscando errores en cada una de las funcionalidades (Roger,2002).

Para validar la propuesta de solución se empleará el método de caja blanca que permitirá determinar si el estado real coincide con el esperado. También va a ser utilizado el método de caja negra teniendo en cuenta la estrategia de partición equivalente, para comprobar la validez en las respuestas de las funcionalidades antes de las acciones del usuario, y la calidad de las salidas en dependencias de las entradas.

3.5.3 Diseño de casos de prueba

El diseño de caso de prueba consiste en probar el sistema, se incluyen los datos de entrada y los resultados esperados. Éstos parten de las descripciones de requisitos por procesos, poseen la finalidad de encontrar la mayor cantidad de definiciones en las funcionalidades implementadas (Ian,2005).

Capítulo 3: Implementación y prueba de la propuesta de solución

A continuación, se presenta el diseño del caso de prueba perteneciente a la HU Grabar voz del profesor, el resto de los artefactos de este tipo se encuentran en el **Anexo 4**.

Tabla 6. Caso de prueba del requisito Grabar voz del profesor

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción de grabar la voz	Una vez en la vista de grabación de voz selecciona la opción grabar.	El sistema debe permitir especificar el siguiente dato: (*) Opciones de menú para la grabación de audio. (*)Permita realizar la acción de grabar la voz.	AudioRecorderController/Grabar

3.5.4 Resultados de las pruebas de integración

Con el objetivo de lograr las pruebas de integración al módulo de grabación y transmisión de audio para el *software* ATcnea, se utilizó la estrategia de *big-bang*. Esta estrategia fue seleccionada después de analizar las siguientes ventajas:

Ventajas

- Útil para la detección de errores cuando se encuentren todos los módulos ya en construcción.
- Aplicable antes de la entrega de proyecto para evaluar el trabajo de todo el sistema con diversos escenarios.
- Se puede evaluar la interacción entre módulos para agilizar los procesos.
- Es apta para aplicar diversos escenarios para poder analizar el trabajo de todos los módulos en diversas situaciones (Zapata, 2010).

La realización de las pruebas integración permitió establecer un conjunto de No Conformidades (NC) en 3ra iteración, estas fueron agrupadas en la siguiente tabla

siendo evaluadas en un rango comprendido entre: Alta, Media, Baja y No procede. A continuación, se presentan los resultados arrojado por estas pruebas:

Tabla 7. Resultado de prueba de integración

Iteraciones	No conformidades				
	Alta	Media	Baja	No procede	Total
1	1	2	2	-	5
2	2	-	2	-	4
3	-	-	-	-	-
	3	**2**	**4**	**-**	**9**

En la tabla anterior se muestra la ejecución de tres iteraciones de pruebas de integración, donde en cada iteración fueron resueltas cada una de las no conformidades identificadas, hasta llegar a la total corrección de las mismas.

3.5.5 Resultados obtenidos con los casos de prueba

Para evaluar la solución se probó el *software* íntegramente, prestando gran atención a evaluar las funcionalidades que presenta el módulo. A continuación, se especifican los casos de pruebas realizados a la aplicación haciendo uso del método de caja negra, estas pertenecen al nivel de prueba de sistema.

Tabla 8. No conformidades encontradas

No. CP	Caso de prueba	No conformidades				
		Alta	Media	Baja	No procede	Total
1	RF Grabar voz del estudiante	-	-	1	-	1
2	RF Grabar voz del profesor	1	-	-	-	1

3	RF Enviar grabación del estudiante	-	-	-	-	-
4	RF Enviar grabación del profesor	-	-	1	-	1
5	RF Almacenar grabación del estudiante	-	1	-	-	1
6	RF Almacenar grabación del profesor	-	1	1	-	2
7	RF Transmitir voz del estudiante al profesor	1	-	-	-	1
8	RF Transmitir voz del profesor al estudiante	1	-	-	-	1
Total		**3**	**2**	**3**	**-**	**8**

Para verificar las NC detectadas se presenta una tabla con los siguientes datos: el Requisito funcional (RF), NC detectada, Descripción (clasificada en Alta, Media o Baja) y estado con respecto a la Solución (RA: Resuelta y Aprobada por el revisor, PD: Pendiente por solución del equipo de desarrollo, NP: No Procede, AV: Aplazada para resolver en próximas versiones, NR: Nuevo Requisito).

Tabla 9. Como serán resueltas las no conformidades

No. NC	No. CP	Descripción	Complejidad	Estado
1	1	El tiempo de grabación está limitado para solo 10 min.	Baja	RA
2	2	La función de grabar consuene muchos recursos y no permite utilizar otro evento.	Alta	RA
3	4	El audio almacenado no es posible enviarlo puesto a que	Baja	RA

		el evento enviar no realiza dicha acción.		
4	5	La voz del estudiante una vez almacenada no se muestra en la tabla de *Files*.	Media	RA
5	6	La voz del profesor una vez almacenada no se muestra en la tabla de *View* que corresponde a los elementos grabados.	Media	RA
6	6	Los archivos almacenados no poseen una localización válida.	Baja	RA
7	7	No se logra emitir la voz del estudiante hacia el profesor.	Alta	RA
8	8	No se logra emitir la voz del profesor hacia el estudiante.	Baja	RA

Capítulo 3: Implementación y prueba de la propuesta de solución

3.6 Conclusiones parciales

- Los artefactos generados en la implementación del módulo, describen las especificidades de las técnicas de programación que se utilizaron.
- Las pruebas realizadas validaron que las funcionalidades desarrolladas satisfacen los requisitos especificados, dejando listo la funcionalidad de audio del *software* con vista a su implantación.
- Mediante la realización de pruebas integración y pruebas de caja negra usando la partición equivalente, quedó validado que el módulo contribuyó a la interacción entre el profesor y el estudiante en el *software* para el aula tecnológica ATcnea en la UCI.

Conclusiones Generales

Con la culminación del presente trabajo se ha dado cumplimiento a los objetivos trazados en la investigación, obteniéndose como resultado principal, el módulo de transmisión y grabación de audio para el *software* del aula tecnológica ATcnea. A continuación, se exponen las conclusiones generales:

- El estudio realizado como parte de la investigación sirvió de apoyo en la toma de decisiones con vista a un desarrollo eficiente del sistema, demostrando que no existe un sistema similar que utiliza la transmisión y la grabación de audio como otra vía de interacción entre el estudiante y el profesor.
- El empleo de la metodología, las herramientas, tecnologías y lenguajes de desarrollo seleccionados, soportaron todo el proceso de desarrollo de la propuesta de solución. Además, los artefactos generados correspondientes a la metodología de desarrollo AUP-UCI facilitaron la implementación del módulo.
- La implementación del módulo, contribuye a utilizar una vía de comunicación directa con el estudiante mediante la vía oral utilizando la transmisión de voz, así como la posibilidad de emitir criterios de los estudiantes.
- A partir de las diferentes iteraciones de pruebas practicadas al módulo, se demostró que este satisface los requisitos funcionales y no funcionales obtenidos durante el flujo de trabajo de los requerimientos.

Recomendaciones:

- Investigar nuevas librerías para la transmisión y grabación de audio que pueden ser utilizadas en el módulo de grabación y transmisión para el software del aula tecnológica ATcnea.

Referencias bibliográficas

Referencias:

-Olmo, Felipe Segovia. El aula inteligente. 2003.

-The impact of classroom technology on student behav. Angeline M. Lavin, Leon Korte, Thomas L. Davies. Dakota : s.n., 2010.

-Reviews of AB Tutor : Free Pricing & Demos : Classroom Management Software.

-Pressman, Roger S. Ingeniría del Software. Un enfoque práctico. España: McGraw-Hill, 2002

-Cáceres González, Abdiel. Lenguajes de Programación. Instituto tecnológico de Monterrey, México.

-Netbeans. [En línea] 2016. https://netbeans.org/index_es.html.

- ROGER.Ingeniería del Software:Un enfoque práctico. 5ta Edicion. 2002.

-Pilato, C. Michael, Fitzpatrick, Brian W. and Collins-Sussman, Ben. Version Control with Subversion. s.l.:O'Reilly. 2004.

-Git. [En línea] 2016. https://git-scm.com/.

-Mythware. Mythware sitio oficial. [En línea] [Citado el: 10 de Diciembre de 2016.] http://www.mythware.com.

-ITALC. ITALC sitio oficial. [En línea] [Citado el: 10 de Diciembre de 2016.] http://italc.sourceforge.net/.

-NetSupport. [En línea] NetSupport sitio oficial. [Citado el: 10 de diciembre de 2016.] www. NetSupport – inc.com

-Brito, Kerenny Acuña. Metodología de Desarrollo. 2005 Cited Febrero 3, 2016.

-Domínguez-Dorado, M. Todo Programación. Madrid : Editorial Iberprensa , 2005.

-JavaFX. Jasper Potts, Nancy Hildebrandt, Joni Gordon, Cindy Castillo. August 2014. E50607-02.

-Sommerville, Ian.Ingeniería del software Séptima edición . Madrid : s.n., 2005.

-S.PRESSMAN., ROGER.Ingeniería del Software:Un enfoque práctico. 5ta Edicion. 2002.

-Pressman, Roger S.Software Engineering and practitioners approach. Septima. New York: Hair Education : s.n., 2010.

Referencias bibliográficas

-Pressman, Roger S.Ingeniería de Software. Un enfoque práctico. Mexico : Maria Tereza Zapata, 2010. 978-607-15-0315-5.

-E.V.A., UCI. Coferencia#5 Protocolos de la capa de transporte UDP. Teleinformática II.

-Julio César Chavez Urrea,2010 http://www.monografias.com/trabajos/protocolotcpip/protocolotcpip.shtml

-Calendamaia, Enero 2014 https://www.genbetadev.com/autor/calendamaia

-Brito, Kerenny Acuña. Metodología de Desarrollo. 2005 Cited Febrero 3, 2016.

-Netbeans. Netbeans. Netbeans. [En línea] 2016. https://netbeans.org/index_es.html.

-Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides,. "Design Patterns: Elements of Reusable Object-Oriented Software",. s.l. : Addison-Wesley, 0-201-63361-2, 1995.

-Larman., Craig. UML y Patrones. 2ª Edición. 2003.

-Fabian Ayala, Beneficio que genera un Aula Virtual, 4 de agosto del 2015.

- Jacobson, IvarBooch, et al. El proceso unificado de desarrollo de software/The unified software development process. El proceso unificado de desarrollo de software/The unified software development process. s.l.: Pearson Educación, 2000.

- Tapia Hernández, Areli. MODELO DE IMPLEMENTACION. 2014.

-Carmona, Juan García. Solid y GRASP. Buenas prácticas hacia el éxito en el desarrollo de software. 2012