Universidad de las Ciencias Informáticas

Facultad 4

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

Título: Desarrollo del módulo Pizarra 1.1 del software XAUCE ATcnea para el soporte de ecuaciones matemáticas

Autor: Brian Mena Gómez

Tutor: Ing. William Simón Ramírez

"Suerte es lo que sucede cuando la preparación y la oportunidad se encuentran y se fusionan."

Voltaire. (1694 – 1778)

Escritor, historiador, filósofo y abogado francés.

Dedicatoria

A mi madre Rebeca Gómez Martín por ser la luz que ilumina mi camino, por ser mi guía, mi inspiración, mi ejemplo a seguir y mi mayor tesoro.

Agradecimientos

A Dios por permitirme estar aquí hoy y por todo su cariño, amor, apoyo y protección, gracias.

A mi mamá Rebeca, estoy muy agradecido y feliz de que seas mi madre, te amo con toda la fuerza de mi alma, gracias por todo.

A Lázaro por ser un padre para mí y quererme como un hijo, gracias.

A mi padre Humberto por ayudarme en todo lo que puede, gracias.

A mi tía Yodanis (Pucha como cariñosamente la conocemos) y a Osmany por adoptarme como un hijo más, gracias.

A mi hermano Erick, que estoy seguro que será el próximo ingeniero de la familia por su amor y cariño, gracias.

A Daniel, a Ronald el cumpleañero y a Melanie por su amor, gracias.

A mis abuelos Caridad, Yuya, China, Manuel y Fiñe, gracias.

A Mercedes y Homero por su bondad de acogerme en su casa, gracias.

A mi primo Alejandro y a Alito, gracias.

A mi familia en general que sin ella yo no fuese nadie.

A mi tutor William, gracias.

A Sandy, sin ti el documento no hubiera quedado con la calidad que tiene, gracias.

A los buenos amigos que he hecho en la universidad y que me han ayudado a lo largo de estos años, gracias.

Gracias a todo aquel que ha aportado un granito en mi formación como ser humano, como persona y como profesional.

Gracias mamá, gracias papá…, lo que mediste es perfecto…, de lo demás me encargo yo.

Resumen

XAUCE ATcnea es un software desarrollado por el Centro de Tecnologías para la Formación (FORTES) de la Universidad de las Ciencias Informáticas (UCI) el cual constituye una solución educativa óptima, fácil de operar y de colaboración para el proceso de enseñanza-aprendizaje que brinda una experiencia única en el aula a partir del aprovechamiento de las Tecnologías de la Información y la Comunicación aportando una nueva forma de enseñar y aprender. En esta investigación se propone el desarrollo de la versión 1.1 del módulo Pizarra para el software XAUCE ATcnea con el objetivo de dar soporte a las ecuaciones matemáticas. La investigación estuvo enmarcada en el tipo de investigación científica. Para el cumplimiento del objetivo planteado se hace uso de la metodología de desarrollo de software Variación de AUP para la UCI, de los lenguajes JAVA, FXML, CSS y XML y del entorno de desarrollo integrado (IDE) IntelliJ IDEA y Android Studio. Una vez implementado el módulo se realizan pruebas funcionales y se corrigen los errores detectados, garantizando de esta manera que todas las funcionalidades se comporten de la forma correcta.

Palabras clave: XAUCE ATcnea, módulo Pizarra, ecuaciones matemáticas, aula tecnológica

Abstract

XAUCE ATcnea is a software developed by the Center for Technologies for Training (FORTES) of the University of Computer Sciences (UCI) which is an optimal educational solution, friendly, easy to operate and collaborative for the teaching process- learning that provides a unique experience in the classroom from the use of Information Technology and Communication providing a new way of teaching and learning. In this research we propose the development of version 1.1 of the Board module for the XAUCE ATcnea software with the aim of supporting the mathematical equations. The research was framed in the type of scientific research. For the fulfillment of the proposed objective is made use of software development methodology AUP Variation for the UCI, the JAVA, FXML, CSS and XML languages and the integrated development environment (IDE) IntelliJ IDEA and Android Studio. Once the module is implemented, functional tests are carried out and the detected errors are corrected, thus guaranteeing that all functionalities behave in the correct way.

Key words: XAUCE ATcnea, Board module, mathematical equations, technological classroom.

Contenido

INTRODUCCIÓN

La irrupción de las Tecnologías de la Información y la Comunicación (TIC) en la sociedad actual, está llevando a cabo importantes cambios en nuestra forma de vivir, de relacionarnos y de aprender. Entre todos ellos, el aprendizaje, será en poco tiempo la nota imperante en esta nueva concepción social. Se transitará desde una sociedad informada a una sociedad con un mayor grado de formación, no sólo en el ámbito profesional, sino en un sentido cultural amplio. (1)

En Cuba como parte de la política de informatización de la sociedad se crea en el año 2002 la Universidad de las Ciencias Informáticas (UCI) la cual tiene como misión “formar profesionales comprometidos con su Patria y altamente calificados en la rama de la Informática, producir aplicaciones y servicios informáticos a partir del vínculo estudio – trabajo como modelo de formación – investigación - producción, sirviendo de soporte a la industria cubana de la Informática.” (2)

La Universidad de las Ciencias Informáticas (UCI) cuenta con una serie de proyectos de desarrollo de software mediante los cuales se han informatizado una parte de los principales procesos que se llevan a cabo tanto dentro del centro educacional como en la sociedad cubana e incluso a nivel internacional. El Aula tecnológica XAUCE ATcnea es uno de los productos desarrollados por el Centro de Tecnologías para la Formación (FORTES), “la cual constituye una solución educativa óptima, amigable, fácil de operar y de colaboración para el proceso de enseñanza-aprendizaje que brinda una experiencia única en el aula a partir del aprovechamiento de las Tecnologías de la Información y la Comunicación aportando una nueva forma de enseñar y aprender”. (3)

Actualmente el Aula tecnológica XAUCE ATcnea tiene como principal cliente al Ministerio de Educación de la República de Cuba (MINED) y está siendo desplegado principalmente en escuelas pedagógicas de todo el país donde una de las asignaturas que se imparte es Matemática, y los profesores han planteado la necesidad de contar con alguna funcionalidad que le permita trabajar con ecuaciones matemáticas, que en estos momentos el software no posee. Además, el módulo Pizarra del Aula Tecnológica no permite guardar los cambios que se realizan en este, lo que trae como consecuencia que una vez que se cierra el software se pierde todo el contenido de la pizarra impidiendo que el profesor pueda planificar una clase en la pizarra con antelación.

Debido a la situación existente se propone como **problema a resolver** ¿Cómo dar soporte a las ecuaciones matemáticas en la pizarra del software XAUCE ATcnea?

El **objeto de estudio** definido es los módulos pizarra para los software de aulas tecnológicas, enmarcando como **campo de acción** el soporte de ecuaciones matemáticas en los módulos pizarra para los software de aulas tecnológicas.

El **objetivo general** de este trabajo de diploma es desarrollar el módulo Pizarra 1.1 del software XAUCE ATcnea para dar soporte a las ecuaciones matemáticas.

Para cumplir el objetivo planteado se establecen las siguientes **tareas de investigación**:

1. Elaboración del marco teórico de la investigación.
2. Estudio de los módulos pizarra para los software de aulas tecnológicas existentes.
3. Estudio de las tecnologías utilizadas para el desarrollo de los módulos pizarra para los software de aulas tecnológicas.
4. Selección de la metodología de software y herramientas de desarrollo a utilizar para darle cumplimiento al objetivo propuesto.
5. Identificación de los requisitos funcionales y no funcionales.
6. Análisis y diseño de la solución propuesta.
7. Implementación del módulo propuesto.
8. Ejecución de las pruebas y respuesta a las posibles no conformidades detectadas.

Para la realización del trabajo se tuvieron en cuenta **métodos científicos investigativos** como:

Método 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.

Método Analítico-Sintético: fue utilizado para el análisis de la bibliografía disponible con el objetivo de realizar un estudio sobre el tratamiento de las ecuaciones matemáticas en módulos pizarra para aulas tecnológicas existentes. Posibilitó definir los conceptos principales y analizar otras soluciones existentes. Se sintetizaron las principales características de las herramientas para el desarrollo del sistema y las ventajas del uso de las mismas.

El presente trabajo de diploma se encuentra estructurado en tres capítulos:

Capítulo 1: Fundamentación Teórica

En este capítulo se hace referencia a los elementos teóricos en los cual está basado la investigación y donde se incluye un estudio del estado del arte de este tema. Se exponen los lenguajes, metodologías, herramientas y tecnologías utilizadas en el desarrollo de la solución, así como los principales conceptos relacionados con el contenido.

Capítulo 2: Propuesta de solución

En este capítulo se describe la solución propuesta para llevar a cabo el desarrollo del módulo. Se detallan las principales características, así como la especificación de los requisitos funcionales a implementar y los no funcionales. Se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. Se describe el proceso de implementación del módulo Pizarra 1.1 y se muestran los resultados obtenidos.

Capítulo 3: Pruebas

En este capítulo se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se describen las pruebas que se le realizaron al software y los resultados obtenidos de la aplicación de dichas pruebas con el objetivo de obtener un producto que cumpla con los requerimientos establecidos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En el presente capítulo se describen de forma detallada los conceptos relacionados con el dominio del problema. Se realiza un estudio a nivel nacional e internacional de los módulos de pizarra para aulas tecnológicas existentes que presenten similitud con el módulo propuesto. Se describen las tecnologías, herramientas, lenguajes y metodología que permiten dar solución al problema. Todo esto basado en las características y necesidades del software XAUCE ATcnea.

1.2. Conceptos asociados al problema

Durante el proceso de desarrollo de software tienden a existir problemas en la comunicación y entendimiento entre los clientes, desarrolladores y usuarios finales. Esto se debe al manejo de diferentes términos y conceptos que pueden ser desconocidos para el personal externo, de ahí lo importante que es definir y argumentar los que pueden ser los principales conceptos que se utilizan a lo largo del presente trabajo. (4)

Aula tecnológica

Solución educativa basada en el uso de las nuevas Tecnologías de la Información y la Comunicación (TIC) para facilitar el proceso de aprendizaje de una manera efectiva. Cuenta con un software para apoyar el proceso de enseñanza-aprendizaje con el uso de múltiples dispositivos. (5)

- Tecnología**:** pizarrón interactivo, video proyector, computadora, tableta.
- Conectividad: audio, datos, video, internet.
- Multimedia educativa.
- Programas diseñados con contenidos para diversas áreas del conocimiento.

Beneficios del uso del aula tecnológica para la institución:

Crea ambientes favorables para el aprendizaje a través de los sentidos, logrando así un mejor desempeño tanto en alumnos como en docentes. Fortalece el desarrollo de competencias de alumnos y docentes relacionadas con el manejo de las TIC, al utilizarlas de manera habitual en el proceso de aprendizaje. Convierte a la institución en un agente de cambio al complementar su modelo educativo con tecnología, facilitando el aprendizaje y desarrollo de las habilidades del siglo XXI. (6)

Beneficios del uso del aula tecnológica para el docente:

Utiliza el tiempo de clase para enseñar, guiar y compartir experiencias con los alumnos al apoyarse en recursos previamente desarrollados. Atiende los diferentes estilos de aprendizaje, generando así una mayor participación e interés de los alumnos por aprender. Facilidad para desarrollar sus propios contenidos interactivos, favoreciendo la comprensión de conceptos complejos. (6)

Beneficios del uso del aula tecnológica para el alumno:

Aprende y relaciona conceptos de una manera dinámica e intuitiva. Adquiere conocimientos mientras interactúa con contenidos atractivos que despiertan su interés por aprender. Utiliza la tecnología que tanto llama su atención, pero con fines educativos, mejorando de manera importante su desempeño académico. (6)

Ecuaciones matemáticas

En matemática se llama ecuación a la igualdad entre dos expresiones algebraicas, que serán denominados miembros de la ecuación. En las ecuaciones, aparecerán relacionados a través de operaciones matemáticas, números y letras (incógnitas). (7)

Importancia de las ecuaciones matemáticas en la educación:

Las ecuaciones matemáticas, ayudan a desarrollar la capacidad creativa del intelecto y ayudan a resolver problemas de la vida cotidiana con mayor celeridad. Muchos alumnos con problemas de aprendizaje, especialmente en matemáticas, en los últimos años de la primaria, se vuelven hábiles luego de que aprenden a resolver las primeras, simples y básicas ecuaciones de primer grado. Durante los años de la educación secundaria los alumnos desarrollan una gran capacidad y habilidad con la resolución de problemas con ecuaciones matemáticas. Paulatinamente el alumno adquiere la capacidad de resolver ecuaciones más rápidamente y en poco tiempo resuelve problemas, usando el razonamiento lógico y ya no mecánicamente, como estaba acostumbrado. Los problemas sobre ecuaciones matemáticas contribuyen al desarrollo del razonamiento lógico o causal, tan importante en el ser humano. (9)

Pizarra interactiva

La Pizarra Interactiva, también denominada Pizarra Digital Interactiva (PDi), consiste en un ordenador conectado a un video-proyector, que proyecta la imagen de la pantalla sobre una superficie, desde la que se puede controlar el ordenador. (10)

Los principales beneficios de las pizarras interactivas en el aula son los siguientes (11):

- Hace que sea más fácil para el profesor estructurar las lecciones.
- Hace que el proceso de aprendizaje sea más divertido.
- Aumenta el nivel de compromiso entre profesores y estudiantes.
- Hace que la revisión de las lecciones sea más conveniente.
- Permite que los estudiantes con diversas discapacidades aprendan mejor.
- Aporta comodidad y flexibilidad en el aprendizaje.

XAUCE ATcnea

Es un software desarrollado por el Centro de Tecnologías para la Formación (FORTES) de la Universidad de las Ciencias Informáticas (UCI), mediante el cual los estudiantes y profesores pueden interactuar a través de los medios tecnológicos como tabletas, pizarras tecnológicas y laptop. Entre las principales características con las que cuenta el sistema se tiene la autenticación de los usuarios, la creación de grupos dentro de la clase para poder realizar atención diferenciada a los estudiantes y crear actividades grupales, la creación de preguntas con varias tipologías y exámenes.

Además, se puede configurar a los tipos de evaluaciones de las entidades, registrar asistencia, compartir audio y video, chat, enviar archivos, el profesor puede controlar la terminal de los estudiantes, además brinda reportes de asistencia y de evaluaciones. Con este software se busca que la tecnología sea la base para la creación de un ambiente colaborativo y dinámico donde se enriquezca el contenido académico de las asignaturas y además disponer de un producto basado en tecnologías libres y actuales que no representa un impedimento para la soberanía tecnológica que se lleva a cabo en el país. El software XAUCE ATcnea cuenta con tres aplicaciones, la primera es la correspondiente al profesor que se instalará en la laptop con sistema operativo Nova 2015 o superior o Windows 10, la segunda aplicación es la del estudiante que se instalará en cada uno de los *tablets* con sistema operativo NovaDroid o Android 5.0 o superior y la tercera aplicación es la correspondiente al estudiante que se instalará en la laptop con sistema operativo Nova 2015 o superior o Windows 10.

El **módulo Pizarra** v1.0 del software XAUCE-ATcnea engloba las funcionalidades relacionadas con la pizarra virtual del ordenador del profesor. Este permite al profesor compartir la pizarra con los estudiantes que se encuentren conectados a la clase o dejar de compartir la pizarra. También permite dibujar, insertar formas geométricas como: círculo, cuadrado y línea, insertar texto, limpiar la pizarra, goma de borrar, insertar figura, insertar imagen, seleccionar el color de relleno y color de texto.

1.3. Análisis de los módulos pizarra para los software de aulas tecnológicas

OpenBoard 1.5

OpenBoard es un software libre y abierto para pizarras digitales interactivas. OpenBoard es una derivación del proyecto de software abierto Open-Sankoré 2.0. Open-Sankoré está basado en software de Uniboard originalmente desarrollado por la Universidad de Lausanne, Suiza. El software empezó a ser desarrollado en 2003 y fue inicialmente utilizado por los profesores de la Universidad en octubre 2003. El proyecto se cedió más tarde a una nueva compañía llamada Mnemis SA. Posteriormente fue vendido a la Agrupación Francesa de Interés Público para Educación Digital en África (GIP ENA), la cual compró la propiedad intelectual del software para convertirlo en un proyecto de código abierto bajo la licencia GNU (LGPL). (12)

OpenBoard es un nuevo software basado en el código fuente de Open-Sankoré que respeta los derechos de autor originales. Esta bifurcación se creó para volver a enfocar el software en sus funcionalidades y valores originales, que es el trabajo de un maestro en un aula, privilegiando la facilidad de uso. La evolución de este software, por lo tanto, tendrá lugar de acuerdo con estos principios básicos. Este software combina la simplicidad de las herramientas de enseñanza tradicionales con los beneficios que aportan las TIC en la educación. Openboard es una herramienta de compilación de medios reales. Permite beneficiarse de la contribución esencial de la escritura a mano al tiempo que agrega elementos visuales, imágenes, gráficos, videos o navega por Internet. Todos estos medios se pueden anotar y completar, los pasajes resaltados o comentados con un lápiz interactivo. (12)

Las principales características del módulo pizarra de este software son (12):

- Disponible para los Sistemas Operativos macOS, Windows y Ubuntu.
- Herramientas de dibujo (pluma, resaltador).
- Permite insertar diferentes elementos en la pizarra como: cuadros de texto, imágenes, videos, música y *widgets.*

- Fácil administración de todos estos elementos a través de un panel de "biblioteca" incorporado.
- Modo de escritorio, para anotar fácilmente cualquier cosa fuera de OpenBoard.
- Navegador web integrado.

El principal inconveniente para el uso de este software es que no permite el trabajo con ecuaciones matemáticas.

MimioStudio 11.5

El software de aula MimioStudio permite a los educadores crear lecciones de pizarra interactivas, así como colaborativas y actividades de equipo, y realizar evaluaciones formativas en tiempo real con soporte para bolígrafos, toque único, multitáctil y gestos. El software MimioStudio lleva el aprendizaje aún más lejos con la aplicación MimioMobile, que permite el aprendizaje en grupo y la colaboración en casi cualquier dispositivo. (13)

Los maestros pueden crear evaluaciones con estudiantes de respuestas cortas, ensayos breves y números. Incluye el libro de calificaciones de MimioStudio, que registra automáticamente las opciones múltiples, numéricas y de respuesta corta a las preguntas de evaluación, y elimina la necesidad de que los maestros hagan las pruebas a mano. Permite importar archivos desde SMART, Promethean, PowerPoint, Formato de archivo común (IWB) y video. *ActivityWizard* ofrece un motor de conocimiento incorporado para ayudar a los maestros a generar una educación sólida y participar en actividades docentes en minutos. (13)

Este software es multiplataforma y se encuentra disponible para los Sistemas Operativos: Windows, Macintosh y Ubuntu, permite el uso simultáneo de teléfonos y tabletas Apple y Android, computadoras portátiles y *Chromebooks* para evaluación y colaboración, permite que hasta 50 dispositivos de estudiantes trabajen simultáneamente en una actividad de MimioStudio y soporta 33 idiomas. (13)

El módulo pizarra de este software comprende las siguientes funcionalidades (13):

- Multi-página.
- Trabajo con objetos.
- Opciones de color, lápiz, estilo de línea, formas.
- Opciones de presentación.
- Trabajo con archivos de audio, video y flash.

- Exportación a HTML, PDF y formatos de imágenes.

El principal inconveniente para el uso de este software es que no cuenta con una funcionalidad que permita el trabajo con ecuaciones matemáticas, además, es privativo por lo tanto no cumple con las políticas de migración de Cuba al uso del software libre, y no es posible integrarlo a XAUCE ATcnea.

Mythware Classroom Management Software

Optimizado para proporcionar una solución amigable y colaborativa en el aula, el software Mythware Classroom Management tiene como misión promover la enseñanza y el aprendizaje en un estilo interactivo con un mejor rendimiento y disponibilidad en una amplia gama de idiomas, Mythware está diseñado con una interfaz amigable y fácil de operar, se integra fácilmente en el sistema de educación. También está equipado con un alto nivel de compatibilidad y estabilidad, lo que garantiza la capacidad de trabajar con otro software y permite un largo proceso de enseñanza. Se desempeña bien tanto en entornos cableados como inalámbricos. (14)

Sirve como una plataforma de gestión para la enseñanza interactiva multimedia en aulas de informática, este software de gestión de aula permite al profesor controlar y gestionar la clase de manera eficaz, supervisar las actividades de los alumnos y mantener un buen orden en la clase, y los alumnos pueden aprender, comunicarse y colaborar con cada uno. Este software está disponible para 5 sistemas operativos: Windows, Android, iOS, Mac OS y Linux. (14)

El módulo pizarra de este software cuenta con una variedad de funcionalidades como (14):

- Insertar imagen.
- Insertar figura.
- Captura de pantalla.
- Importar documento.
- Compartir pizarra.
- Dibujar.
- Insertar formas.
- Editar pizarra.
- Insertar, modificar tipo de letra, tamaño, color y estilo del texto.
- Borrador.
- Guardar pizarra.

El principal inconveniente para el uso de este software es que no cuenta con una funcionalidad para el trabajo con ecuaciones matemáticas, además es privativo por lo tanto no cumple con las políticas de migración de Cuba al uso del software libre y no es posible integrarlo a XAUCE ATcnea.

WhiteBoard_4.50

El software WhiteBoard_4.50 está diseñado para ser una herramienta interactiva fácil de usar como una herramienta poderosa para la educación y los negocios. Tiene como principales características (15):

- Funciones de página.
- Funciones de operación con la página.
- Funciones de lápiz.
- Funciones de figura.
- Funciones de borrador.
- Funciones de inserción de objetos.
- Funciones de grabado y reproducción.
- Herramientas auxiliares.
- Guardar y cargar pizarra.
- Word, Excel, Notas.
- Configuración de color, ancho de línea y transparencia.
- Trabajo con ecuaciones.
- Importar y exportar archivos.

WhiteBoard_4.50 permite el trabajo con ecuaciones matemáticas mediante un software editor de ecuaciones que trae incorporado, el cual es externo y privativo. Al usuario presionar la herramienta de ecuaciones el editor de ecuaciones se abre permitiéndole al usuario crear ecuaciones, editarlas y guardarlas en formato de imagen. Este software también permite guardar y cargar la pizarra.

El principal inconveniente para el uso de este software es que es privativo por lo tanto no cumple con las políticas de migración de Cuba al uso del software libre, y además no es posible integrarlo con XAUCE ATcnea.

Resultados del estudio

En el estudio realizado se pudo observar que los módulos pizarra para software de aulas tecnológicas existentes constituyen una potente herramienta para la enseñanza interactiva y los negocios ya que cuentan con una serie de herramientas y funcionalidades que facilitan el proceso de enseñanza-aprendizaje, pero han sido desarrollados con tecnología privativa y no es posible su integración con XAUCE ATcnea. Es válido destacar que, aunque ninguno satisface en su totalidad los requerimientos necesarios para su utilización sí aportaron ideas novedosas y de utilidad a la presente investigación. Fueron útiles para proponer nuevas funcionalidades que enriquecieron al software desde el punto de vista funcional como: listar, insertar y mover ecuaciones matemáticas, guardar la pizarra e importar la pizarra.

1.4. Entorno de desarrollo

Para darle solución al problema planteado en la investigación, se hizo necesario el estudio de los elementos que van a conformar el entorno de desarrollo de la misma. A la hora de realizar este estudio se tuvo en cuenta que la solución informática que se pretende construir va a formar parte del software XAUCE ATcnea, por lo que se estudió el entorno de desarrollo que se utiliza en el proyecto ATcnea y a partir de ahí se seleccionaron todos los elementos necesarios para desarrollar la propuesta de solución. A continuación, se presentan todas las tecnologías, herramientas, y lenguajes, con sus principales características:

1.4.1. Lenguajes de desarrollo

Java 8

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Java es rápido, seguro y fiable. Desde portátiles hasta centros de datos, desde consolas para juegos hasta súper computadoras, desde teléfonos móviles hasta Internet. Java 8 incluye nuevas características, mejoras y correcciones de bugs para mejorar la eficacia en el desarrollo y la ejecución de programas Java dentro de las que se encuentran (16):

- Método forEach () en interfaz iterable.
- Métodos predeterminados y estáticos en interfaces.
- Interfaces funcionales y expresiones Lambda.
- API de Java Stream para operaciones de datos masivos en colecciones.

- API de Java Time.
- Mejoras en la API de colección.
- Mejoras en la API de concurrencia.
- Mejoras de Java IO.

FXML 8

FXML es un lenguaje de marcado de interfaz de usuario basado en XML creado por Oracle Corporation para definir la interfaz de usuario de una aplicación JavaFX. Proporciona una alternativa conveniente para construir tales gráficos en código de procedimiento, y es ideal para definir la interfaz de usuario de una aplicación JavaFX, ya que la estructura jerárquica de un documento XML se asemeja mucho a la estructura del gráfico de escena JavaFX. (17)

CSS 2

El nombre hojas de estilo en cascada viene del inglés *Cascading Style Sheets*, del que toma sus siglas. CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (*World Wide Web Consortium*) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores. La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación. (18)

XML 1.0

XML, siglas en inglés de *eXtensible Markup Language*, traducido como "Lenguaje de Marcado Extensible" o "Lenguaje de Marcas Extensible", es un meta-lenguaje que permite definir lenguajes de marcas desarrollado por el *World Wide Web Consortium* (W3C) utilizado para almacenar datos en forma legible. Permite definir la gramática de lenguajes específicos para estructurar documentos grandes. A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información. (19)

UML

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. (20)

1.4.2. Herramientas de desarrollo

JavaFX 8

JavaFX es un conjunto de paquetes de gráficos y medios que permite a los desarrolladores diseñar, crear, probar, depurar y desplegar aplicaciones de cliente enriquecidas que operan de manera consistente en diversas plataformas. Las aplicaciones JavaFX compiladas para JDK 7 y posteriores también se ejecutan en todas las plataformas de escritorio principales (Windows, Mac OS X y Linux), la compatibilidad multiplataforma permite una experiencia de tiempo de ejecución consistente para los desarrolladores y usuarios de aplicaciones JavaFX. (21)

Android Software Development Kit (SDK) 26

SDK, que es el conjunto de herramientas de software necesarias para desarrollar programas que interactúen con otro software mediante una API. El Android SDK (kit de desarrollo de software) es un conjunto de herramientas de desarrollo utilizadas para desarrollar aplicaciones para la plataforma Android. El SDK de Android incluye lo siguiente (22):

- Bibliotecas requeridas.
- Depurador.
- Un emulador.
- Documentación relevante para las interfaces del programa de aplicación de Android (API).
- Código fuente de muestra.
- Tutoriales para el sistema operativo Android.

Java Development Kit (JDK) 8

El JDK es un entorno de desarrollo para crear aplicaciones, *applets* y componentes utilizando el lenguaje de programación Java. El JDK incluye herramientas útiles para desarrollar y probar programas escritos en lenguaje de programación Java y que se ejecutan en la plataforma Java. (23)

Git 2.8.0

Git es un sistema de control de versiones que tiene como objetivo controlar los cambios en el desarrollo de cualquier tipo de software, permitiendo conocer el estado actual de un proyecto, de su código, los cambios que se le han realizado a cualquiera de sus piezas, así como las personas que intervinieron en ellos. (24)

Maven 3

Maven es una herramienta de software para la gestión y construcción de proyectos Java creada por Jason van Zyl, de Sonatype, en 2002. Maven utiliza un *Project Object Model* (POM) para describir el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Una característica clave de Maven es que está listo para usar en red. El motor incluido en su núcleo puede dinámicamente descargar *plugins* de un repositorio, el mismo repositorio que provee acceso a muchas versiones de diferentes proyectos *Open Source* en Java, de Apache y otras organizaciones y desarrolladores. (25)

Gradle 4

Gradle, es una herramienta que permite la automatización de compilación de código abierto, la cual se encuentra centrada en la flexibilidad y el rendimiento. Los *scripts* de compilación de Gradle se escriben utilizando Groovy o Kotlin DSL (*Domain Specific Language*). Además, es el sistema de compilación oficial para Android y cuenta con soporte para diversas tecnologías y lenguajes. (26)

JavaFX Scene Builder 2.0

Es una herramienta de diseño para la plataforma JavaFX 8 Early Access y que ayuda con el diseño visual de la interfaz de usuario de una aplicación que está vinculada a la lógica de la aplicación. Simplemente se arrastra los componentes de la interfaz gráfica de usuario (GUI) en una escena JavaFX y el código fuente FXML para su diseño se genera automáticamente. (27)

IntelliJ IDEA 2018.1.3

IntelliJ IDEA es un entorno de desarrollo integrado (IDE) que permite escritura de código sin complicaciones. Gracias a su profunda comprensión de los lenguajes y tecnologías, crea un entorno adecuado en el que todos los miembros del equipo pueden trabajar juntos de manera

eficiente. Integración transparente con una amplia variedad de sistemas de control de versiones. Permite a los miembros del equipo permanecer en sincronía con los cambios de otros, asegurando que todas las contribuciones sean productivas. El IDE constantemente valida la calidad del código y ofrece soluciones inmediatas para los problemas encontrados en todos los niveles, desde la instrucción individual para arquitectura global, utilizando las inspecciones de código avanzado y análisis de matriz de dependencia. (28)

Principales características (28):

- Completamiento inteligente.
- Completamiento de la cadena.
- Completamiento de miembros estáticos.
- Análisis de flujo de datos.
- Inyección de lenguaje.
- Detectando duplicados.
- Inspecciones y reparaciones rápidas.
- Control de versiones.
- Herramientas de construcción.
- Herramientas de base de datos.

Android Studio 3.0.1

Android Studio es el Entorno de Desarrollo Integrado (IDE) oficial para la plataforma Android. Está basado en el software IntelliJ IDEA de JetBrains, y es publicado de forma gratuita a través de la Licencia Apache 2.0. Está disponible para las plataformas Microsoft Windows, Mac OS X y GNU/Linux. Android Studio está diseñado específicamente para el desarrollo de aplicaciones para el Sistema Operativo Android. (29)

Principales características (29):

- Nuevo diseño del editor con soporte para la edición de temas.
- Nueva interfaz específica para el desarrollo en Android.
- Permite la importación de proyectos realizados en el entorno Eclipse.
- Posibilita el control de versiones accediendo a un repositorio desde el que poder descargar Mercurial, Git, Github o Subversión.
- Alertas en tiempo real de errores sintácticos, compatibilidad o rendimiento antes de compilar la aplicación.

- Vista previa en diferentes dispositivos y resoluciones.

Visual Paradigm 8.0

Visual Paradigm for UML (VP-UML) es una herramienta CASE que propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación; es capaz de soportar el modelado mediante UML y proporciona asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del ciclo de vida de desarrollo de un software. (30)

1.5. Metodología de desarrollo de software

La metodología de desarrollo de software Variación de AUP para la UCI es una variante realizada por la Universidad de las Ciencias Informáticas a la metodología ágil AUP (Proceso Ágil Unificado) y está definida por la universidad como la metodología rectora de la actividad productiva. La metodología Variación de AUP para la UCI está formada por tres fases, (Inicio, Ejecución y Cierre) para el ciclo de vida de los proyectos de la universidad. (31)

Inicio

Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto. (31)

Ejecución

En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, se obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto. Durante esta fase el software es transferido al ambiente de los usuarios finales o entregado al cliente junto con la documentación. Además, en esta transición se capacita a los usuarios finales sobre la utilización de la aplicación. (31)

Cierre

En el cierre se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto. (31)

La metodología Variación de AUP para la UCI cuenta con 7 disciplinas (Modelado de negocio, Requisitos, Análisis y diseño, Implementación, Pruebas interna, Pruebas de liberación, Pruebas de aceptación) para el ciclo de vida de los proyectos. (31)

Modelado de negocio

El Modelado del Negocio es la disciplina destinada a comprender los procesos de negocio de una organización. Se comprende cómo funciona el negocio que se desea informatizar para tener garantías de que el software desarrollado va a cumplir su propósito. Para modelar el negocio se proponen las siguientes variantes (31):

1. Casos de Uso del Negocio (CUN).
2. Descripción de Proceso de Negocio (DPN).
3. Modelo Conceptual (MC).

Requisitos

El esfuerzo principal en la disciplina Requisitos es desarrollar un modelo del sistema que se va a construir. Esta disciplina comprende la administración y gestión de los requisitos funcionales y no funcionales del producto. Existen tres formas de encapsular los requisitos [Casos de Uso del Sistema (CUS), Historias de usuario (HU) y Descripción de requisitos por proceso (DRP)], agrupados en cuatro escenarios condicionados por el Modelado de negocio. (31)

Análisis y diseño

En esta disciplina, si se considera necesario, los requisitos pueden ser refinados y estructurados para conseguir una comprensión más precisa de estos, y una descripción que sea fácil de mantener y ayude a la estructuración del sistema (incluyendo su arquitectura). Además, en esta disciplina se modela el sistema y su forma (incluida su arquitectura) para que soporte todos los requisitos, incluyendo los requisitos no funcionales. (31)

Implementación

En la implementación, a partir de los resultados del Análisis y Diseño se construye el sistema. (31)

Pruebas interna

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. Se deben desarrollar artefactos de prueba como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de prueba ejecutables para automatizar las pruebas. (31)

Pruebas de liberación

Pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación. (31)

Pruebas de Aceptación

Es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido. (31)

A partir de que el Modelado de negocio propone tres variantes a utilizar en los proyectos (CUN, DPN o MC) y existen tres formas de encapsular los requisitos (CUS, HU, DRP), surgen cuatro escenarios para modelar el sistema en los proyectos (31):

Escenario No 1: Proyectos que modelen el negocio con CUN solo pueden modelar el sistema con CUS.

Escenario No 2: Proyectos que modelen el negocio con MC solo pueden modelar el sistema con CUS.

Escenario No 3: Proyectos que modelen el negocio con DPN solo pueden modelar el sistema con DRP.

Escenario No 4: Proyectos que no modelen negocio solo pueden modelar el sistema con HU.

Se seleccionó para el trabajo en esta investigación el escenario No 4 ya que el proyecto ha evaluado el negocio a informatizar y como resultado se ha obtenido un negocio 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, el proyecto no es muy extenso y, además, es el escenario utilizado por el proyecto al cual tributa este Trabajo de Diploma.

1.6. Conclusiones del capítulo

En el presente capítulo se realizó un estudio de los principales conceptos asociados al dominio del problema y se analizaron los diferentes módulos de pizarra para software de aulas tecnológicas en el ámbito internacional y nacional, lo cual permitió conocer las diferentes características que presentan los mismos, y se identificó que los sistemas privativos estudiados no se pueden utilizar en la propuesta de solución, debido a que no cumplen con las políticas de migración de Cuba al uso del software libre y no se pueden integrar a XAUCE ATcnea. Además, se llevó a cabo un estudio de las herramientas y tecnologías seleccionadas para el desarrollo del módulo y se definió la metodología Variación de AUP para la UCI en su escenario No 4 para guiar el proceso de desarrollo de software dando cumplimiento a las buenas prácticas que define CMMI-DEV v1.3.

CAPÍTULO 2: PROPUESTA DE SOLUCIÓN

2.1. Introducción

Después de elaborar el marco teórico, seleccionar las herramientas y la metodología de desarrollo de software a emplear, se han creado las condiciones para realizar una propuesta de solución que satisfaga el problema a resolver. Este capítulo está enmarcado en el análisis y diseño para la futura implementación del componente y se tiene como objetivo describir la propuesta que se plantea para establecer una solución, realizar la especificación de los requisitos del software y establecer los patrones que ayuden a modelar y entender los objetos del sistema.

Teniendo en cuenta que la metodología de desarrollo de software Variación de AUP para la UCI es una metodología iterativa, se decidió realizar 3 iteraciones en el proceso de desarrollo del software, donde la 1ra iteración correspondería a la aplicación ATcnea-Teacher-PC, la 2da iteración a la aplicación ATcnea-Student-Android y la 3ra iteración a la aplicación ATcnea-Student-PC.

2.2. Disciplina Requisitos

La obtención y análisis de los requisitos constituye una de las etapas fundamentales en el proceso de ingeniería de requisitos. En esta actividad los ingenieros de software trabajan en conjunto con los clientes y usuarios finales, para determinar el dominio de la aplicación, los servicios que debe proporcionar, el rendimiento requerido para el sistema, así como las restricciones del hardware. Estos requisitos que no son más que una especificación del diseño del software, es decir una descripción abstracta de este, que es la base para un diseño e implementación detallada. (32)

2.2.1. Descripción de la propuesta de solución

El módulo Pizarra 1.1 para el software XAUCE ATcnea permite que el usuario pueda trabajar con ecuaciones matemáticas, donde se le presentarán al usuario una lista de ecuaciones matemáticas de las cuales este podrá seleccionar la que desee, posteriormente modificarla e insertarla en la pizarra. También el usuario podrá guardar el estado de la pizarra y cargar una pizarra guardada.

2.2.2. Requisitos funcionales

Los requisitos funcionales (RF) son declaraciones de los servicios que debe proporcionar el sistema, de la manera en que este debe reaccionar a entradas particulares y de cómo se debe comportar ante situaciones particulares. En algunos casos, los requisitos funcionales pueden declarar también explícitamente lo que el sistema no debe hacer. (32)

1ra iteración: aplicación ATcnea-Teacher-PC.

- **RF1. Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Teacher-PC:** El sistema debe permitir al profesor listar las ecuaciones en la pizarra.
- **RF2. Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Teacher-PC:** El sistema debe permitir al profesor insertar una ecuación en la pizarra.
- **RF3. Mover ecuación en el módulo Pizarra de la aplicación ATcnea-Teacher-PC:** El sistema debe permitir al profesor mover una ecuación después de ser insertada en la pizarra.
- **RF4. Guardar pizarra en el módulo Pizarra de la aplicación ATcnea-Teacher-PC:** El sistema debe permitir al profesor guardar la pizarra.
- **RF5. Cargar pizarra en el módulo Pizarra de la aplicación ATcnea-Teacher-PC:** El sistema debe permitir al profesor cargar la pizarra.

2da iteración: aplicación ATcnea-Student-Android.

- **RF6. Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Student-Android:** El sistema debe permitir al estudiante listar las ecuaciones en la pizarra.
- **RF7. Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Student-Android:** El sistema debe permitir al estudiante insertar una ecuación en la pizarra.
- **RF8. Mover ecuación en el módulo Pizarra de la aplicación ATcnea-Student-Android:** El sistema debe permitir al estudiante mover una ecuación después de ser insertada en la pizarra.
- **RF9. Guardar pizarra en el módulo Pizarra de la aplicación ATcnea-Student-Android:** El sistema debe permitir al estudiante guardar la pizarra.
- **RF10. Cargar pizarra en el módulo Pizarra de la aplicación ATcnea-Student-Android:** El sistema debe permitir al estudiante cargar la pizarra.

3ra iteración: aplicación ATcnea-Student-PC.

- **RF11. Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Student-PC:** El sistema debe permitir al estudiante listar las ecuaciones en la pizarra.
- **RF12. Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Student-PC:** El sistema debe permitir al estudiante insertar una ecuación en la pizarra.
- **RF13. Guardar pizarra en el módulo Pizarra de la aplicación ATcnea-Student-PC:** El sistema debe permitir al estudiante guardar la pizarra.
- **RF14. Cargar pizarra en el módulo Pizarra de la aplicación ATcnea-Student-PC:** El sistema debe permitir al estudiante cargar la pizarra.

2.2.3. Requisitos no funcionales

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y normas o estándares. Además, no se aplican regularmente a rasgos o servicios individuales del sistema. (32)

A continuación, se describen los requisitos no funcionales utilizados en la implementación del módulo:

RNF 1. Requisitos de Usabilidad:

RNF 1.1. Las interfaces del componente deben cumplir con las pautas de diseño establecidas en el proyecto ATcnea.

RNF 2. Requisitos de Portabilidad:

RNF 2.1. Tableta con Sistema Operativo NovaDroid o Android 5.0 o superior.

RNF 2.2. Tableta con 1 GB de RAM o superior.

RNF 2.3. Tableta con 7 pulgadas o superior de pantalla.

RNF 2.4. Tableta con CPU/GPU Dual Core 1.0GHz o superior.

RNF 2.5. PC con el Sistema Operativo Nova 2015 o superior o Windows 10.

RNF 2.6. PC con 4GB de RAM o superior.

RNF 2.7. PC con CPU Core 2 E6300 o superior.

RNF 2.8. PC con 1GB de Tarjeta de video o superior.

2.2.4. Historias de Usuario

La historia de usuario es una técnica utilizada para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (33)

Se realizaron las historias de usuarios correspondiente a cada requisito por cada una de las 3 iteraciones definidas anteriormente para el proceso de desarrollo del software. A continuación, se presenta la historia de usuario perteneciente al requisito funcional Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Teacher-PC, el resto de los artefactos de este tipo se encuentran en los anexos A.

Tabla 1. Elaboración propia. Historia de Usuario del Requisito funcional Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Teacher-PC

Número: 1	**Nombre del requisito:** Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Teacher-PC
Programador: Brian Mena Gómez	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 120 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 110 horas
Objetivo: Permitir listar ecuaciones al profesor. **Acciones para lograr el objetivo (precondiciones y datos):** Para listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Teacher-PC se tiene que: - El usuario debe de estar autenticado en el sistema con el rol Profesor.	

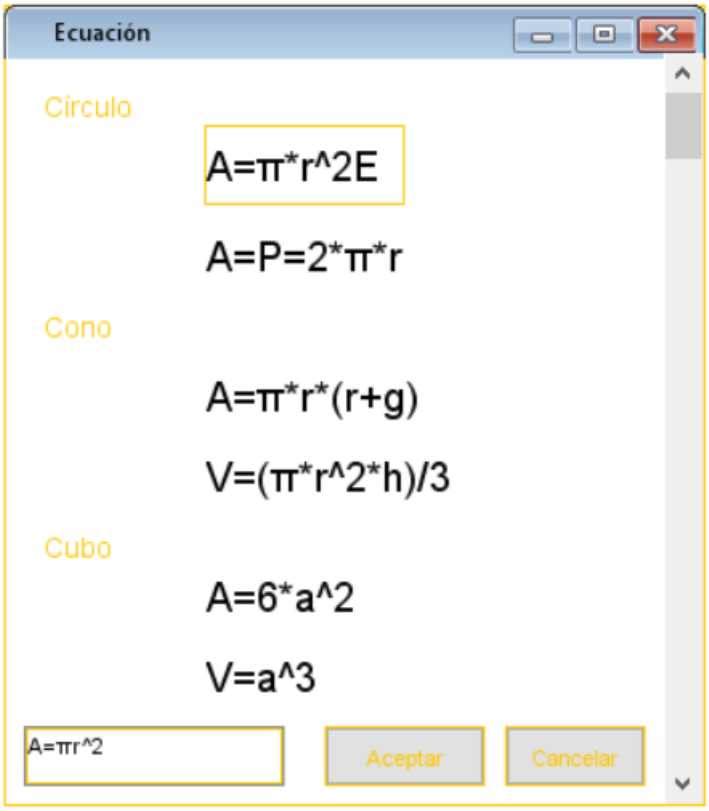
- El usuario debe haber creado una clase.

- El usuario debe haber seleccionado la opción Pizarra.

3- Flujo de la acción a realizar:

Una vez en la pizarra el profesor puede seleccionar la opción Ecuación, con la cual se le desplegará una lista con 21 ecuaciones matemáticas donde este puede seleccionar la que desee.

Prototipo elemental de interfaz gráfica de usuario:



2.3. Disciplina Análisis y Diseño

En esta etapa 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 integrantes del equipo de desarrollo. (34)

2.3.1. Patrones arquitectónicos

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. (35)

Patrón Modelo-Vista-Controlador

El *Model-View-Controller* o Modelo-Vista-Controlador (MVC) es un patrón arquitectónico de software, que sirve para clasificar la información**,** la lógica del sistema y la interfaz que se le presenta al usuario. En este tipo de arquitectura existe un sistema central o controlador que gestiona las entradas y la salida del sistema, uno o varios modelos que se encargan de buscar los datos e información necesaria y una interfaz que muestra los resultados al usuario final. (36)

Las aplicaciones ATcnea-Teacher-PC y ATcnea-Student-PC fueron desarrolladas en JavaFx, y en este tipo de aplicaciones, el patrón arquitectónico MVC es el que se utiliza para trabajar. La aplicación ATcnea-Student-Android desarrollada para el Sistema Operativo Android también fue implementada utilizando el patrón MVC. Teniendo en cuenta lo antes expuesto se decide utilizar el patrón arquitectónico Modelo-Vista-Controlador en la solución propuesta ya que esta va a formar parte de las aplicaciones ATcnea-Teacher-PC, ATcnea-Student-PC y ATcnea-Student-Android.

Patrón Cliente-Servidor

Este patrón consiste en dos partes; un servidor y múltiples clientes. El componente del servidor proporcionará servicios a múltiples componentes del cliente. Los clientes solicitan servicios del servidor y el servidor proporciona servicios relevantes a esos clientes. Además, el servidor sigue escuchando las solicitudes de los clientes. (37)

Las aplicaciones ATcnea-Teacher-PC, ATcnea-Student-PC y ATcnea-Student-Android utilizan el patrón arquitectónico Cliente-Servidor donde la aplicación ATcnea-Teacher-PC funciona como servidor y las aplicaciones ATcnea-Student-PC y ATcnea-Student-Android funcionan como clientes, permitiéndoles a los estudiantes conectarse desde su computadora personal o tableta a la clase creada desde la computadora personal del profesor. Teniendo en cuenta lo antes expuesto se decide utilizar el patrón arquitectónico Cliente-Servidor en la solución propuesta ya que esta va a formar parte de las aplicaciones ATcnea-Teacher-PC, ATcnea-Student-PC y ATcnea-Student-Android.

Por lo tanto, en la implementación del módulo se utiliza el patrón Modelo-Vista-Controlador para el desarrollo de la solución propuesta en las aplicaciones ATcnea-Teacher-PC, ATcnea-Student-Android y ATcnea-Student-PC, y el patrón Cliente-Servidor para permitir la comunicación entre estas aplicaciones. A continuación, se muestra la siguiente figura para una mejor comprensión:

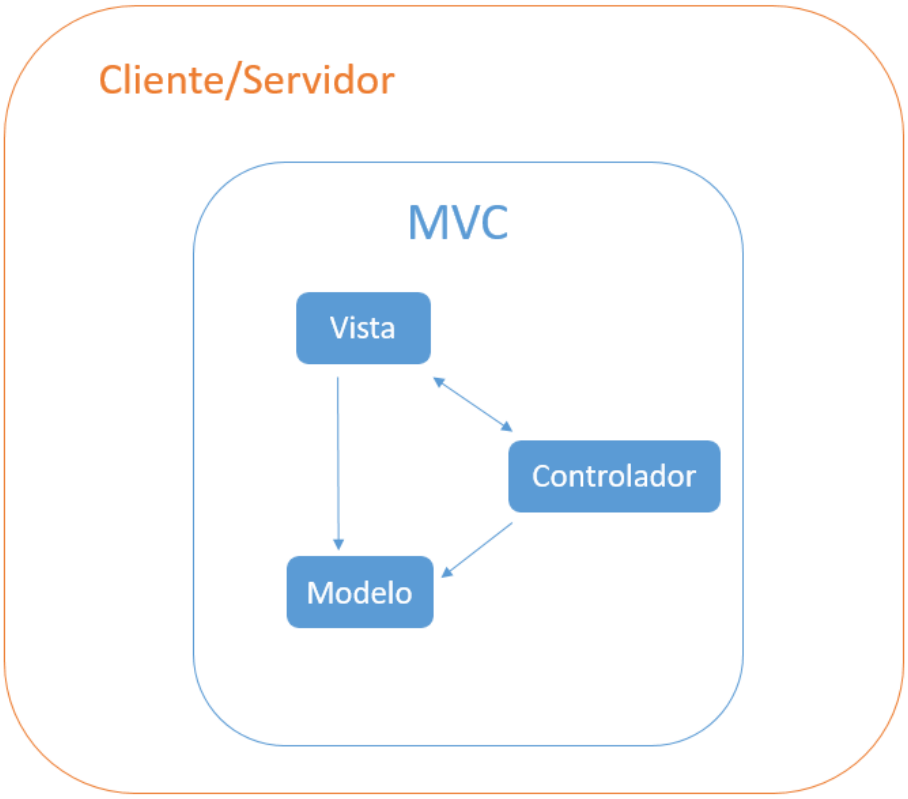


Figura 1. Elaboración propia. Arquitectura del módulo Pizarra

2.3.2. Patrones del diseño

Un patrón de diseño describe una estructura que resuelve un problema de diseño en particular dentro de un contexto específico y en medio de fuerzas que pueden tener un impacto en la manera en que se aplica y utiliza el patrón. (35)

Patrones GRASP

Los patrones generales de software para asignar responsabilidad (GRASP por sus siglas en inglés *General Responsibility Asignment Software Patterns*) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de

patrones (38). Dentro de los patrones GRASP utilizados para el diseño del módulo se encuentran los siguientes:

Experto: El experto en la información es la clase que tiene la información necesaria para cumplir la responsabilidad (38). En la solución propuesta este patrón se evidencia en la clase BGraphic.

Creador: Este patrón permite crear objetos de una clase determinada. Es utilizado en la mayoría de las clases controladoras para crear instancias a formularios y entidades, para la vista del usuario (38). En la solución propuesta este patrón se evidencia en la clase BGraphic.

Controlador: Este se basa en asignar la responsabilidad de todos los eventos realizados a una clase específica que constituye un único punto de entrada para cada evento (38). En la solución propuesta este patrón se evidencia en la clase MainScreenBlackboardController y EquationDialog.

Patrones GOF

Los patrones GOF (por sus siglas en inglés *The Gang of Four*) son aquellos que describen soluciones simples y elegantes a problemas específicos en el diseño de software orientado a objetos. Estos se pueden clasificar según su propósito en (39):

- **Patrones de creación:** conciernen al proceso de creación de objetos.
- **Patrones estructurales:** tratan la composición de clases y/u objetos.
- **Patrones de comportamiento:** caracterizan las formas en las que interactúan y reparten responsabilidades las distintas clases u objetos.

***Singleton* (Instancia Única):** El patrón de diseño *Singleton* es un patrón GOF de creación, recibe su nombre debido a que sólo se puede tener una única instancia para toda la aplicación de una determinada clase, esto se logra restringiendo la libre creación de instancias de esta clase mediante el operador *new* e imponiendo un constructor privado y un método estático para poder obtener la instancia. La intención de este patrón es garantizar que solamente pueda existir una única instancia de una determinada clase y que exista una referencia global en toda la aplicación (40). En la solución propuesta este patrón se evidencia en la clase ATcneaSinglenton.

***Compositive*:** El patrón de diseño *Compositive* es un patrón GOF de tipo estructural, este patrón permite crear y manejar estructuras de objetos en forma de árbol, en las que un objeto puede contener a otro(s) (41). En la solución propuesta este patrón se evidencia en la clase BGraphic.

2.3.3. 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 (34). Como parte del modelo de implementación se obtienen los diagramas de componentes que se muestran en el presente epígrafe.

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. (34)

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 (34).

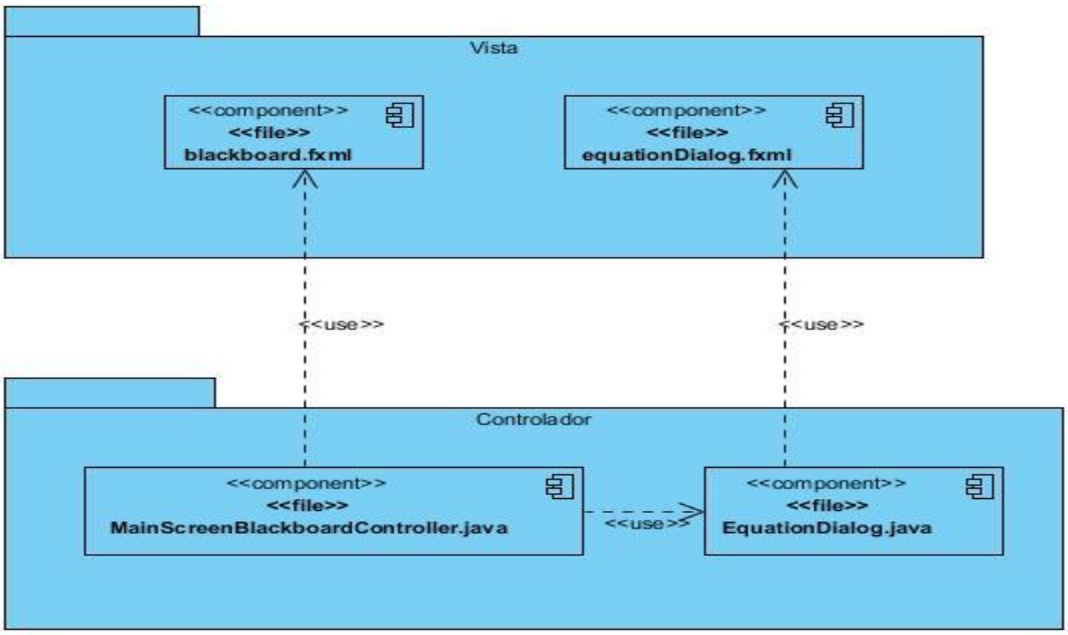


Figura 2. Elaboración propia. Diagrama de componentes de la aplicación ATcnea-Teacher-PC y ATcnea-Student-PC

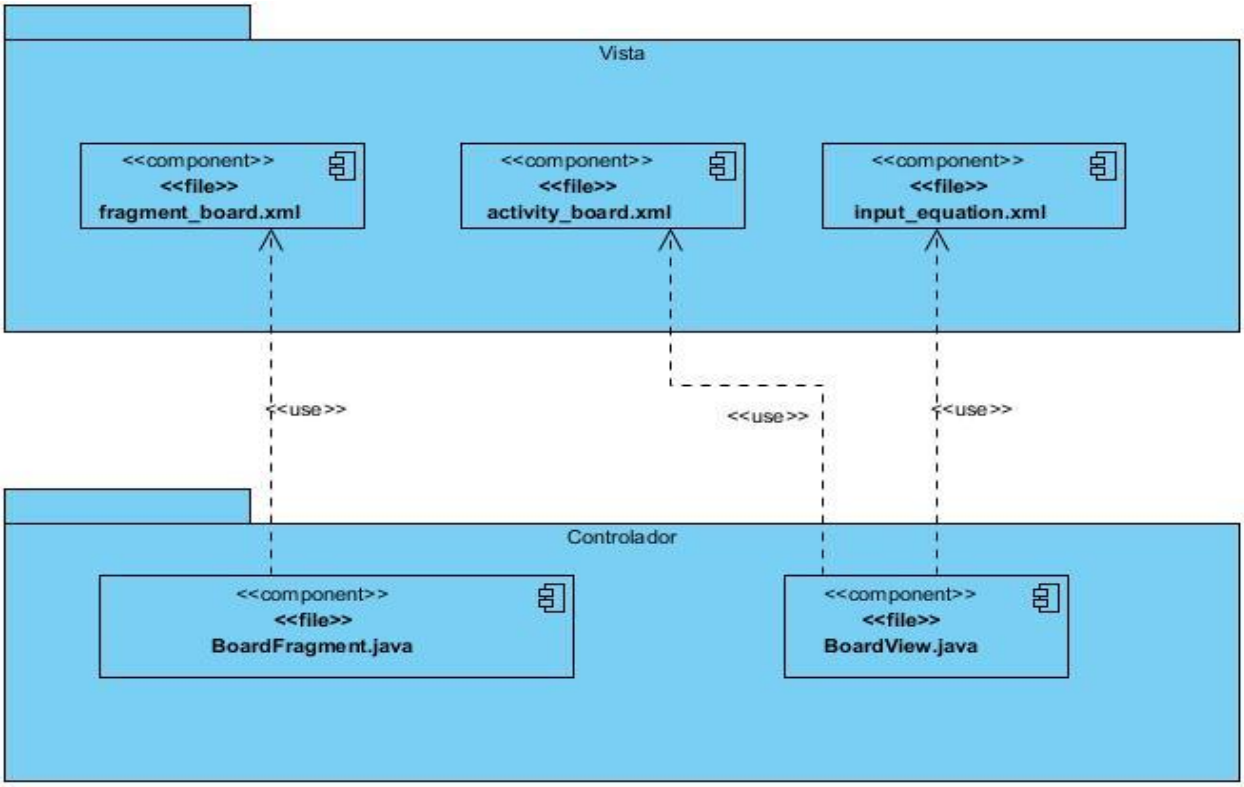


Figura 3. Elaboración propia. Diagrama de componentes de la aplicación ATcnea-Student-Android

2.3.4. Estándares de codificación

Los estándares de codificación son buenas prácticas de la programación que hacen posible un estilo de programación homogéneo en un proyecto, lo que permite que todos los participantes puedan entender en menos tiempo el código del proyecto y que el código en consecuencia sea mantenible. (42)

Inicialización: intentar inicializar las variables locales al ser declaradas. La única razón para no inicializar una variable donde se declara es si el valor inicial depende de algunos cálculos que deben ocurrir.

Colocación: poner las declaraciones solo al principio de los bloques (un bloque es cualquier código encerrado por llaves "{" y "}".) No esperar al primer uso para declararlas; puede confundir a programadores no preavisados y limitar la portabilidad del código dentro de su ámbito de visibilidad.

void myMethod() {
int int1 = 0; // comienzo del bloque del método

```
if (condición) {

int int2 = 0; // comienzo del bloque del "if"

…

}

}
```

La excepción de la regla son los índices de bucles for, que en Java se pueden declarar en la sentencia for.

```
for (int i = 0; i < maximoVueltas; i++) {}
```

Evitar las declaraciones locales que ocultan declaraciones de niveles superiores, por ejemplo, no declarar la misma variable en un bloque interno:

```
int cuenta;

…

miMetodo () {

if (condición) {

int cuenta = 0; // evitar esto

…

}

…

}
```

Declaraciones de clases e interfaces: al codificar clases e interfaces de Java, se siguen las siguientes reglas de formato:

- Ningún espacio en blanco entre el nombre de un método y el paréntesis "(" que abre su lista de parámetros.
- La llave de apertura "{" aparece al final de la misma línea de la sentencia declaración.

- La llave de cierre "}" empieza una nueva línea para ajustarse a su sentencia de apertura correspondiente, excepto si no existen sentencias entre ambas, que debe aparecer inmediatamente después de la de apertura "{".

```
class Ejemplo extends Object {

int ivar1;

int ivar2;

Ejemplo (int i, int j) {

ivar1 = i;

ivar2 = j;

}

ivar2 = j;

int metodoVacio () {}

…

}
```

Sentencias if, if-else, if else-if else: la clase de sentencias if-else debe tener la siguiente forma:

```
if (

condición

) {

sentencias;

}

if (

condición

) {
```

sentencias;

} else {

sentencias;

}

if (

condición

) {

sentencia;

} else if (

condición

) {

sentencia;

} else {

sentencia;

}

Nota: las sentencias if usan siempre llaves {}, evitar la siguiente forma, propensa a errores:

if (condición)

sentencia; //evitar esto

Sentencias for: una sentencia for debe tener la siguiente forma:

for (inicialización; condición; actualización) {

sentencias;

}

Variables: Excepto las constantes, todas las instancias y variables de clase o método empezarán con minúscula. Las palabras internas que lo forman (si son compuestas) empiezan con su primera letra en mayúsculas. Los nombres de variables no deben empezar con los caracteres subguión "_" o signo del dólar "$", aunque ambos están permitidos por el lenguaje. Los nombres de las variables deben ser cortos, pero con significado. Los nombres de variables de un solo carácter se deben evitar, excepto para variables índices temporales.

2.3.5. 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. (35)

A continuación, se presenta el diagrama de despliegue propuesto para el sistema. En el mismo se muestra que la computadora del profesor se comunica con la tableta del estudiante y la computadora del estudiante mediante los protocolos:

- UDP (Protocolo de datagrama de usuario) que es un protocolo no orientado a conexión de la capa de transporte del modelo TCP/IP. (43)
- TCP (Protocolo de Control de Transmisión) que es uno de los principales protocolos de la capa de transporte del modelo TCP/IP. Haciendo uso de este, las aplicaciones pueden comunicarse en forma segura (gracias al sistema de acuse de recibo del protocolo TCP) independientemente de las capas inferiores. Esto significa que los *routers* (que funcionan en la capa de Internet) sólo tienen que enviar los datos en forma de datagramas, sin preocuparse con el monitoreo de datos porque esta función la cumple la capa de transporte (o más específicamente el protocolo TCP). (44)

UDP es utilizado para el proceso de encontrar las clases activas y TCP para el resto de la comunicación entre estos dispositivos.

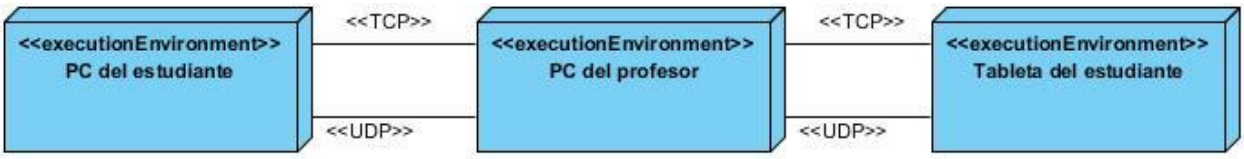


Figura 4. Elaboración propia. Diagrama de despliegue

2.3.6. Modelo de diseño

En esta etapa 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 (34).

Diagrama de clases del diseño

Contiene debido a la adaptación del modelo de diseño a la implementación, mayor cantidad de detalles que el diagrama de análisis entre los que podemos citar: clases, atributos, operaciones, subsistemas y relaciones (34).

Se diseñaron los diagramas de clases del diseño correspondientes a cada requisito de cada una de las 3 iteraciones definidas anteriormente para el proceso de desarrollo del software. A continuación, se muestra el diagrama de clase correspondiente al requisito Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Teacher-PC perteneciente a la 1ra iteración, el resto de los artefactos de este tipo se encuentran en los anexos C.

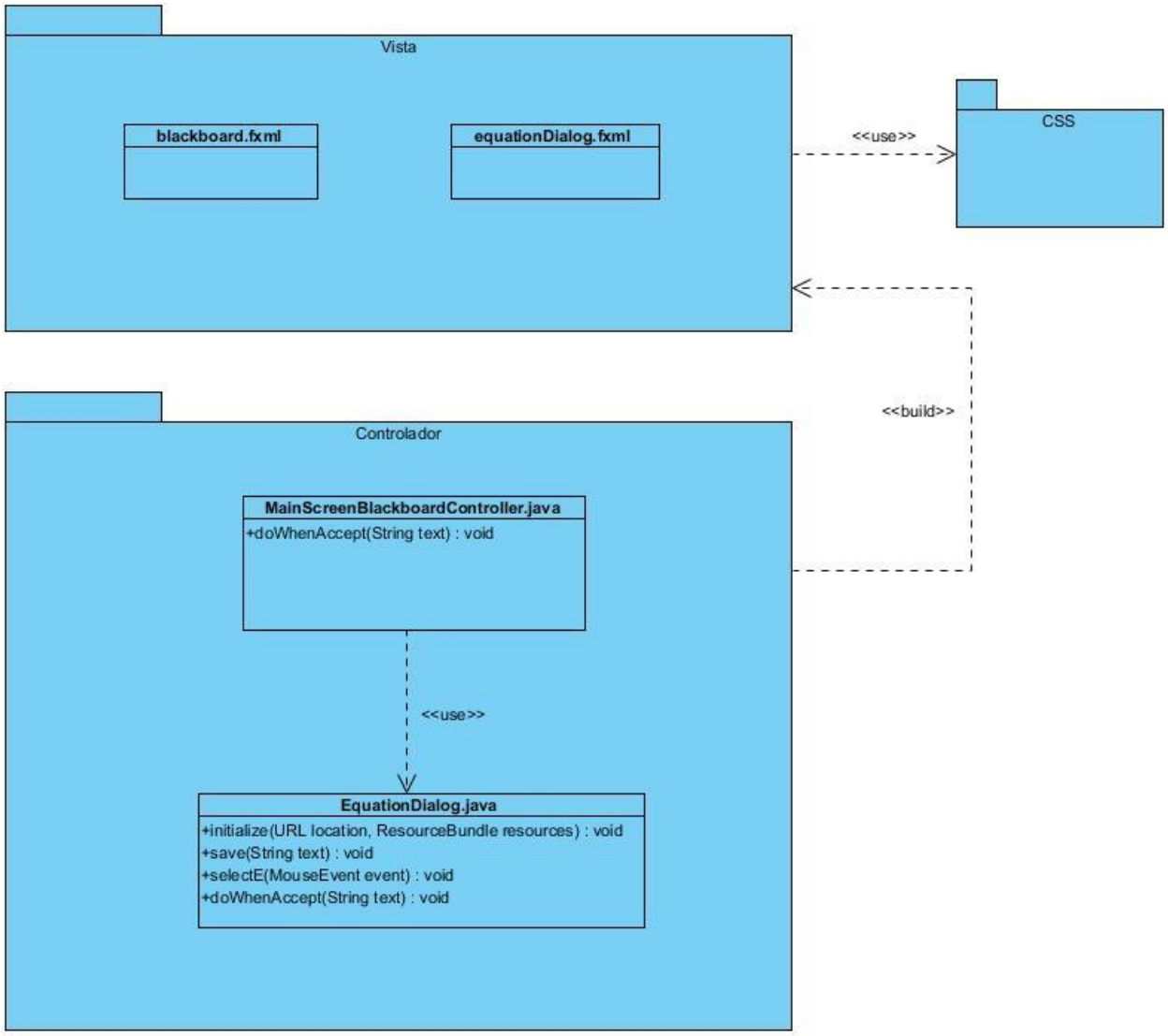


Figura 5. Elaboración propia. Diagrama de clases correspondiente al requisito funcional Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Teacher-PC

2.3.7. Diseños de casos de prueba

Un Diseño de Casos de Prueba (DCP) es un artefacto generado en la disciplina Análisis y Diseño por la metodología de desarrollo de software seleccionada y utilizado en las disciplinas Implementación, Pruebas internas, Pruebas de Liberación y Pruebas de Aceptación para llevar a cabo las pruebas funcionales. Tiene como objetivo comprobar el correcto funcionamiento del sistema, en estos se incluyen las entradas, resultados y condiciones con la que se ha de verificar, constituyendo la guía principal para el probador. (32)

Para el diseño de estos casos de pruebas se tuvo en cuenta el método de prueba Caja negra haciendo uso de la técnica Partición de equivalencia. Esta técnica se basa en evaluar las clases de equivalencia para una condición de entrada, una condición de entrada es un valor numérico específico, un rango de valores, un miembro de un conjunto de valores o lógica y una clase de equivalencia representa un conjunto de estados válidos y no válidos para una condición de entrada. (45)

Se diseñaron los casos de pruebas por cada requisito de cada una de las 3 iteraciones definidas anteriormente para el proceso de desarrollo del software. A continuación, se presenta el DCP perteneciente al requisito Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Teacher-PC correspondiente a la 1ra iteración, el resto de los artefactos de este tipo se encuentran en los anexos B.

Tabla 2. Elaboración propia. DCP correspondiente al requisito funcional Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Teacher-PC

Escenario	Descripción	Ecuación	Respuesta del sistema	Flujo central
EC 1.1 Opción Ecuación	El usuario selecciona la opción Ecuación.	N/A	El sistema muestra un listado de las ecuaciones agrupadas por figuras para que el usuario seleccione una ecuación. También muestra el campo modificar ecuación y las opciones: *Aceptar *Cancelar	Clase X /Pizarra/Ecuación
EC 1.2 Opción de cancelar	El usuario selecciona la opción de Cancelar.	N/A	El sistema no inserta la ecuación matemática en la pizarra y cierra la ventana.	Clase X /Pizarra/Ecuación/Cancelar
EC 1.3 Opción de aceptar	El usuario selecciona la opción Aceptar.	V	El sistema inserta la ecuación matemática seleccionada en la pizarra. ***Ver DCP: "Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Teacher-PC".***	Clase X /Pizarra/Ecuación/Aceptar

2.4. Implementación

En la implementación a partir de los resultados del análisis y diseño se construyó el sistema. Se decidió la realización de 3 iteraciones donde cada iteración representa la agrupación de requisitos por aplicación con el objetivo de obtener resultados incrementales:

1ra iteración: aplicación ATcnea-Teacher-PC.

2da iteración: aplicación ATcnea-Student-Android.

3ra iteración: aplicación ATcnea-Student-PC.

En cada iteración se implementaron las historias de usuarios definidas para cada uno de los requisitos especificados en el análisis y diseño obteniendo como resultado el módulo Pizarra 1.1 donde se añadió el trabajo con ecuaciones matemáticas, así como el guardar y el cargar la pizarra. A continuación, se muestran imágenes de los resultados obtenidos.

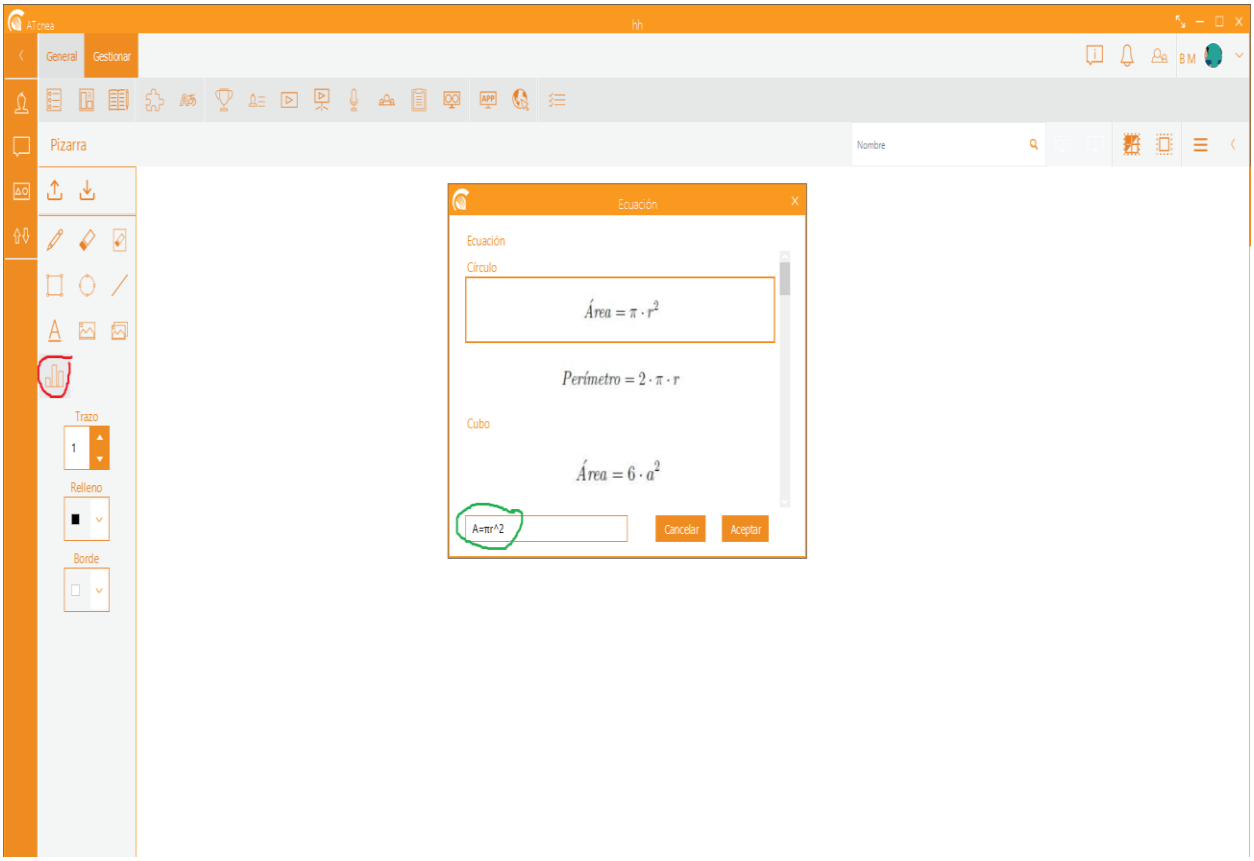


Figura 6. Elaboración propia. Ecuaciones matemáticas en las aplicaciones ATcnea-Teacher-PC y ATcnea-Student-PC

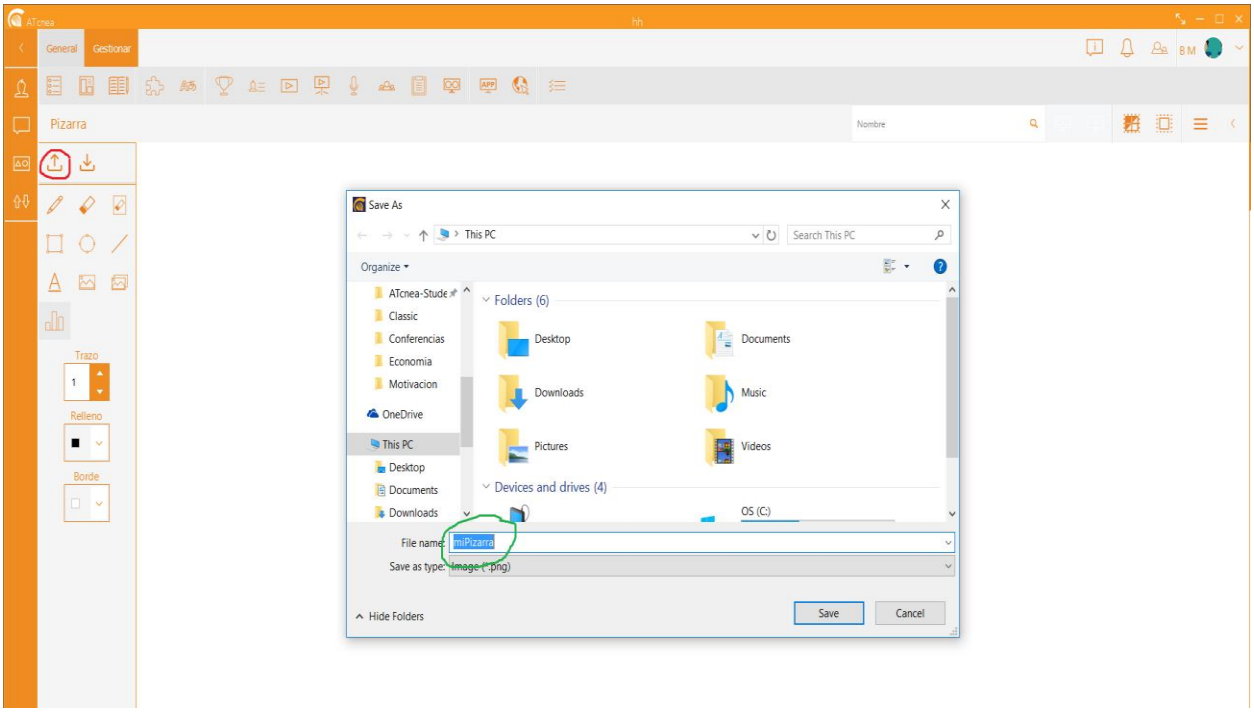


Figura 7. Elaboración propia. Guardar pizarra en las aplicaciones ATcnea-Teacher-PC y ATcnea-Student-PC

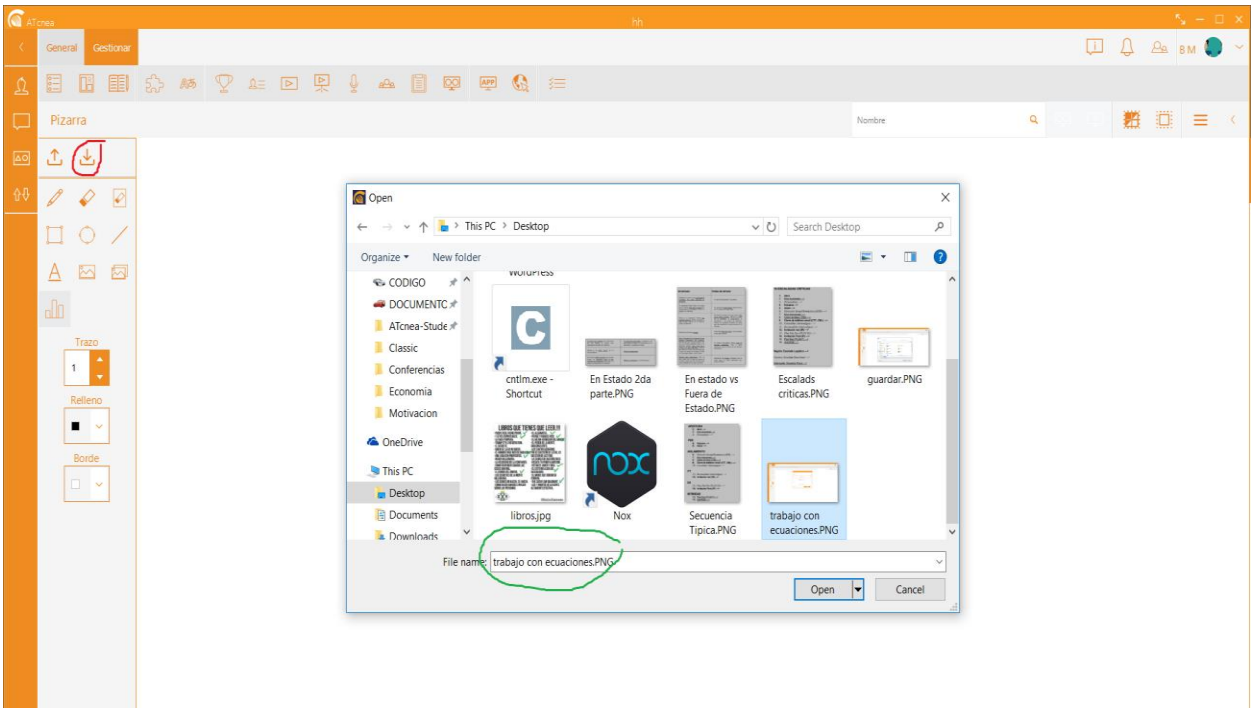


Figura 8. Elaboración propia. Cargar pizarra en las aplicaciones ATcnea-Teacher-PC y ATcnea-Student-PC

2.5. Conclusiones del capítulo

La especificación de los 14 requisitos funcionales y 9 no funcionales brindó un enfoque de la versión 1.1 del módulo Pizarra a implementar. Igualmente, la confección de los artefactos definidos en la disciplina Análisis y Diseño de la metodología AUP en su variante UCI, permitió definir las bases necesarias para la implementación del módulo. En la disciplina Implementación se construyó el módulo Pizarra 1.1 propuesto el cual permite el trabajo con ecuaciones matemáticas, así como guardar y cargar la pizarra.

CAPÍTULO 3: PRUEBAS

3.1. Introducción

En este capítulo se exponen las distintas pruebas realizadas al software y se recoge los resultados de las mismas, con el objetivo de comprobar que el sistema cumpla con los requerimientos establecidos.

3.2. Pruebas de calidad de software

Calidad de software ha sido definida como la concordancia con los requisitos funcionales y de rendimiento explícitamente establecidos con los estándares de desarrollo documentados y con las características implícitas que se esperan de todo software desarrollado profesionalmente (35). La calidad de software es sinónimo de eficiencia, flexibilidad, corrección, confiabilidad, mantenibilidad, portabilidad, usabilidad, seguridad e integridad. De acuerdo con el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) es el grado con el que un sistema, componente o proceso cumple los requerimientos especificados y las necesidades o expectativas del cliente o usuario. (46)

En el desarrollo de software, la calidad de diseño acompaña a la calidad de los requisitos y especificaciones. La concordancia es un aspecto centrado principalmente en la implementación, la cual si sigue al diseño y el sistema resultante cumple con los objetivos de requisitos y rendimiento, la calidad de concordancia es alta. Entonces a grandes rasgos la calidad de software es el grado con el cual el usuario percibe que el software cumple con sus expectativas. (47)

Durante el ciclo de desarrollo del software se hace necesario darle seguimiento a una serie de parámetros para verificar que el mismo se encuentre libre de errores. En todos los equipos de desarrollo deben existir personas encargadas de ello, las cuales se guían por un plan de pruebas que permite velar por la calidad del software. (47)

Las pruebas es una actividad en la cual un sistema o componente es ejecutado bajo unas condiciones o requerimientos específicos, los resultados son observados y registrados, y una evaluación es hecha de algún aspecto del sistema o componente. (48)

La prueba de un sistema se define como el proceso de ejercitar o evaluar el sistema, por medios manuales o automáticos, para verificar que satisface los requerimientos o, para identificar diferencias entre los resultados esperados y los que producen el sistema. (49)

3.2.1. Niveles de prueba

Las pruebas se encuentran definidas dentro de niveles específicos, donde se figuran los métodos de cada uno de ellos y sus objetivos como tal. A continuación, se mencionan los niveles de pruebas que existen:

Pruebas de componente: Es un nivel de prueba de software donde se prueban unidades / componentes individuales de un software. El propósito es validar que cada unidad del software funciona según lo diseñado. Una unidad es la parte comprobable más pequeña de cualquier software. Por lo general, tiene una o unas pocas entradas y por lo general una sola salida. En la programación de procedimientos, una unidad puede ser un programa individual, una función, un procedimiento, etc. En la programación orientada a objetos, la unidad más pequeña es un método, que puede pertenecer a una base / superclase, una clase abstracta o una clase derivada / secundaria. Los marcos de prueba de la unidad, controladores, talones y objetos simulados / falsos se usan para ayudar en la prueba de la unidad. (50)

Prueba de integración: Es un nivel de prueba de software donde las unidades individuales se combinan y se prueban como un grupo. El propósito de este nivel de prueba es exponer fallas en la interacción entre unidades integradas. Es ejecutada para garantizar que en el modelo de implementación los componentes operen correctamente cuando son combinadas para ejecutar un requisito funcional. Para realizar pruebas de integración a un sistema es necesario establecer una estrategia de integración. A continuación, se muestran dichas estrategias (51):

- 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.

Pruebas 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. (35)

Pruebas de aceptación: Es un nivel de prueba de software donde un sistema se prueba para su aceptabilidad. El propósito de esta prueba es evaluar el cumplimiento del sistema con los requisitos del negocio y evaluar si es aceptable para la entrega. (52)

3.2.2. Tipos de prueba

Los tipos de pruebas de software que se le realizaron a la propuesta de solución fueron:

Pruebas funcionales: Es un tipo de prueba de software por el cual el sistema se prueba contra los requisitos / especificaciones funcionales. Las funciones (o características) se prueban al alimentarlas con entrada y examinar la salida. Las pruebas funcionales garantizan que la aplicación cumpla correctamente con los requisitos. Este tipo de prueba no se refiere a cómo se produce el procesamiento, sino a los resultados del procesamiento. Simula el uso real del sistema, pero no hace suposiciones de la estructura del sistema. Durante las pruebas funcionales, se utiliza la técnica de prueba de caja negra en la que el probador desconoce la lógica interna del sistema que se está probando. (53)

Pruebas de usabilidad: Son técnicas formales que tienen como objetivo estudiar la usabilidad de una aplicación en un entorno real con usuarios reales (ISO 9241, 2006). Usabilidad se define como 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. Son una forma de medir el número de errores que puede una persona presentar al usar un objeto hecho por el hombre, como puede ser una página web, una interfaz de usuario, un documento o un dispositivo. Existen varias técnicas para medir la usabilidad del software, entre las que se encuentran (54):

- **Entrevistas individuales:** las entrevistas individuales se refieren normalmente a hablar con un usuario a la vez cara a cara, por teléfono, o con la mensajería instantánea u otros medios asistidos por ordenador. Estas entrevistas no tienen que ver con mirar una obra de usuario. Por lo tanto, esto es diferente de entrevistar a los usuarios en una sesión de pruebas de usabilidad o la realización de entrevistas contextuales.
- **Grupo focal:** un grupo focal es una reunión entre 8 y 12 usuarios o posibles usuarios de un sitio, que tienen un debate moderado sobre el mismo. Normalmente dura unas dos horas y cubre una amplia gama de temas programados de antemano. Es una técnica de

investigación de mercados tradicionales, por lo que los departamentos de *marketing* suelen estar más familiarizados con los grupos de enfoque que con las pruebas de usabilidad o entrevistas contextuales. En un grupo de enfoque típico, los participantes hablan acerca de su trabajo. En este tipo de técnica se trabaja en la búsqueda de información acerca de las actitudes del usuario, las creencias, los deseos y sus reacciones al mostrarles diferentes ideas o prototipos.

- **Evaluación heurística:** una evaluación heurística es un método de inspección de usabilidad de software que ayuda a identificar problemas en la interfaz de usuario. En concreto, supone que los evaluadores examinan y juzgan su conformidad con los principios reconocidos los cuales son llamados también heurísticas de usabilidad. Las evaluaciones heurísticas generalmente se llevan a cabo por un pequeño grupo (de uno a tres) de probadores, los cuales de forma independiente examinan una interfaz de usuario y juzgan el cumplimiento de un conjunto de principios de usabilidad. El resultado de este análisis es una lista de problemas.
- **Cuestionarios:** existen varios tipos de cuestionarios, actualmente los más usados con fines de usabilidad son los cuestionarios en línea para los cuales se estructuran las entrevistas con los usuarios, donde se muestran una lista de preguntas y respuestas.
- **Análisis de tareas:** el análisis de tareas consiste en aprender acerca de las metas de los usuarios, lo que quiere hacer en un sitio web, cómo funcionan a través de las tareas específicas que hacen para alcanzar sus objetivos y qué medidas toman para llevarlas a cabo. El análisis de tareas complementa la comprensión de usuario, ya que se pueden ver directamente las tareas generales que están tratando de lograr o cómo las realizan actualmente.

Pruebas de confirmación: Se utiliza para probar un producto de software nuevamente, para validar la precisión de los resultados de sus pruebas y para garantizar la rectificación de todos los errores anteriores encontrados en el sistema o sus componentes. En este tipo de pruebas, los casos de prueba que antes eran efectivos en la detección de defectos en el producto de software, se ejecutan nuevamente una vez que se solucionan los errores en el producto. Esto se hace para garantizar que el producto de software que no pudo pasar los casos de prueba ahora puede pasar estos casos de prueba después de aplicar las correcciones en el producto de software para eliminar errores. (55)

Pruebas de regresión: Es un tipo de prueba de software que pretende garantizar que los cambios (mejoras o correcciones de defectos) en el software no lo hayan afectado

negativamente. La probabilidad de que algún cambio en el código afecte a las funcionalidades que no están directamente asociadas con el código siempre está ahí y es esencial que se realicen pruebas de regresión para asegurarse de que arreglar una cosa no haya roto otra. Durante las pruebas de regresión, los nuevos casos de prueba no se crean, pero los casos de prueba creados previamente se vuelven a ejecutar. (56)

No se realizarán pruebas de portabilidad ya que este tipo de pruebas se le aplican al sistema en su conjunto y no a una parte de él, por lo que, a pesar de que la propuesta de solución tiene requisitos de portabilidad, no se le harán estas pruebas.

3.2.3. Métodos y técnicas de prueba

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 (35).

Pruebas de caja blanca: También conocido como Prueba de caja clara, Prueba de caja abierta, Prueba de caja de vidrio, Prueba de caja transparente, Prueba basada en código o Prueba estructural es un método de prueba de software en el que el probador conoce la estructura / diseño / implementación interna del elemento que se está probando. El probador elige entradas para realizar recorridos a través del código y determina las salidas apropiadas. El conocimiento de programación y el conocimiento de implementación son esenciales. La prueba de caja blanca se realiza para probar más allá de la interfaz del usuario (57). Dentro de las pruebas de caja blanca se incluyen las técnicas de prueba que serán descritas a continuación (58):

- **Flujo de control:** en estas pruebas se quiere encontrar errores en la parte lógica del programa, utilizando las condiciones simples operador-relación o con condiciones complejas.
- **Flujo de datos:** por medio de esta técnica se hace la selección más adecuada del flujo de datos, para llegar a una resolución correcta, esto para probar las variables y definiciones en el programa.
- **Ruta básica:** esta prueba lo que demuestra es el conjunto de pasos base del programa, lo que quiere lograr es que cada sentencia de código se ejecute mínimo una vez.

- **Bifurcación (*branch testing*):** como lo dice su título, es ligado a una bifurcación o para bucles. Con ella podemos definir si algún bucle esta correctamente implementado y si las líneas de código donde exista una condición es la mejor opción o si debería ser cambiada.

Pruebas de caja negra: También conocido como Pruebas de comportamiento es un método de prueba de software en el que el probador no conoce la estructura interna /diseño/ implementación del elemento que se está probando. Estas pruebas pueden ser funcionales o no funcionales, aunque generalmente son funcionales (59). Dentro de las pruebas de caja negra se incluyen las técnicas de prueba que serán descritas a continuación (35):

- **Partición de Equivalencia:** divide el dominio de entrada de un programa en un número finito de variables de equivalencia. Se definen dos tipos de variables de equivalencia, las válidas, que representan entradas válidas al programa, y las no válidas, que representan valores de entrada erróneos, aunque pueden existir valores no relevantes a los que no sea necesario proporcionar un valor real de dato.
- **Análisis de valores límites:** prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- **Grafos Causa-Efecto:** permite validar complejos conjuntos de acciones y condiciones.

3.3. Pruebas en la disciplina de implementación

En la implementación con los resultados del análisis y diseño se desarrolla el sistema en término de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares. En este flujo de trabajo se deben hacer las pruebas de componente, cada desarrollador es responsable de probar las unidades que produzca. El resultado final de este flujo de trabajo es un sistema ejecutable. (60)

3.3.1. Ejecución del Nivel de pruebas de componente

Para realizar las pruebas unitarias en java se utiliza jUnit que es un conjunto de bibliotecas que permite la realización de la ejecución de clases de manera controlada, para poder comprobar que los métodos realizan su cometido de forma correcta. (61)

Para la realización de las pruebas unitarias al módulo se implementó una clase java en el subdirectorio *test/* del proyecto por cada caso de prueba y se siguió el siguiente orden para cada requisito funcional de cada una de las iteraciones del proceso de desarrollo del software:

1. Implementar casos de prueba unitaria para el requisito funcional "X"

2. Implementar requisito funcional "X"

3. Ejecutar casos de prueba unitaria para el requisito funcional "X"

4. Resolver no conformidades detectadas

5. Ejecutar pruebas de confirmación

6. Ejecutar pruebas de regresión

En este proceso se implementaron un total de 14 casos de prueba. Estos últimos, hicieron posible que se pudieran detectar 10 no conformidades. Estas fueron resueltas satisfactoriamente, según los resultados arrojados por las pruebas de confirmación que fueron aplicadas al corregir las no conformidades identificadas en cada requisito. También se realizaron pruebas de regresión para verificar que los cambios realizados no provocaron nuevos errores en los requisitos implementados con anterioridad.

3.4. Disciplina Pruebas Internas

En esta disciplina se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones finales a ser liberadas. (31)

3.4.1. Ejecución del Nivel de Pruebas de integración

El nivel de pruebas de integración se ejecutó dentro de la disciplina Pruebas Internas, donde se utilizó la estrategia de integración Big Bang. Esta estrategia fue seleccionada después de analizar las siguientes ventajas (62):

- 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.

Y se le realizaron al software 3 iteraciones de pruebas funcionales por cada una de las iteraciones del proceso de desarrollo del software definidas anteriormente:

1ra iteración: aplicación ATcnea-Teacher-PC.

2da iteración: aplicación ATcnea-Student-Android.

3ra iteración: aplicación ATcnea-Student-PC.

Para realizar las pruebas funcionales a la solución propuesta se utilizó el método de prueba caja negra haciendo uso de la técnica partición de equivalencia, para las cuales se utilizaron los casos de pruebas diseñados con anterioridad en el Capítulo 2.

La realización de las pruebas funcionales en el Nivel de pruebas de integración permitió detectar un conjunto de no conformidades en cada una de las 3 iteraciones definidas para el desarrollo del módulo, estas fueron agrupadas en las siguientes tablas siendo evaluadas en un rango comprendido entre: Alta, Media, Baja y No procede.

Tabla 3. Elaboración propia. Resultados de las pruebas funcionales en la 1ra iteración

Iteraciones	No conformidades				
	Alta	Media	Baja	No procede	Total
1	2	4	2	-	8
2	1	2	2	-	5
3	-	-	-	-	-
	3	6	4	-	13

En la 1ra iteración se detectaron en total 13 no conformidades: 1 de tipo excepción, 1 de validación, 7 de funcionalidad y 4 de opciones que no funcionan.

Se resolvieron satisfactoriamente todas las no conformidades, al realizar las pruebas de confirmación todo estaba correcto y las pruebas de regresión mostraron que los cambios realizados no habían causado problemas en otras partes del sistema.

Tabla 4. Elaboración propia. Resultados de las pruebas funcionales en la 2da iteración

Iteraciones	No conformidades				
	Alta	Media	Baja	No procede	Total
1	1	2	4	-	7
2	-	1	2	-	3
3	-	-	-	-	-
	1	3	6	-	10

En la 2da iteración se detectaron en total 10 no conformidades: 2 de tipo excepción, 1 de validación, 5 de funcionalidad y 2 de opciones que no funcionan.

Se resolvieron satisfactoriamente todas las no conformidades, al realizar las pruebas de confirmación todo estaba correcto y las pruebas de regresión mostraron que los cambios realizados no habían causado problemas en otras partes del sistema.

Tabla 5. Elaboración propia. Resultados de las pruebas funcionales en la 3ra iteración

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

En la 3ra iteración se detectaron en total 5 no conformidades: 1 de tipo excepción, 3 de funcionalidad y 1 de opciones que no funcionan.

Se resolvieron satisfactoriamente todas las no conformidades, al realizar las pruebas de confirmación todo estaba correcto y las pruebas de regresión mostraron que los cambios realizados no habían causado problemas en otras partes del sistema.

A continuación, se muestra un gráfico donde se representa el total de no conformidades identificadas en el nivel de pruebas de integración, en el que se puede apreciar que fueron resueltas cada una de las no conformidades hasta llegar a la total corrección de las mismas.

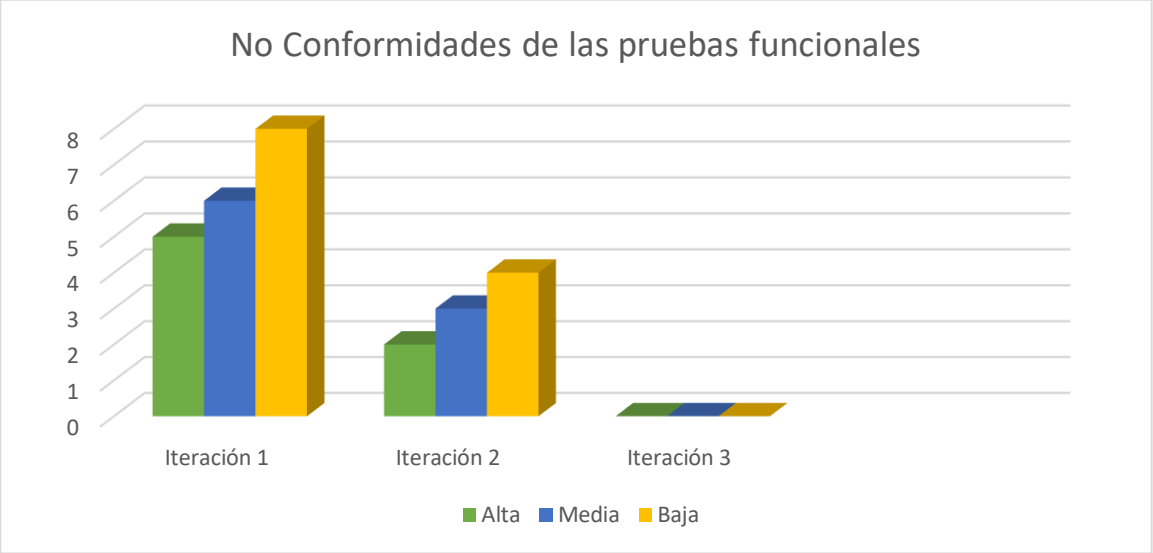


Figura 9. Elaboración Propia. Cantidad de No Conformidades por iteración de las pruebas funcionales

3.4.2. Ejecución del Nivel de Pruebas de sistema

El nivel de pruebas de sistema también se ejecutó dentro de la disciplina Pruebas Internas donde se le realizaron al software 3 iteraciones de pruebas usabilidad por cada una de las iteraciones del proceso de desarrollo del software definidas anteriormente:

1ra iteración: aplicación ATcnea-Teacher-PC.

2da iteración: aplicación ATcnea-Student-Android.

3ra iteración: aplicación ATcnea-Student-PC.

Para realizar las pruebas de usabilidad a la solución propuesta se utilizó la técnica Grupo focal descrita anteriormente la que se aplicó a un grupo de 6 estudiantes de 5to año de la carrera Ingeniería en Ciencias Informáticas pertenecientes a la Facultad 4 de la Universidad de las Ciencias Informáticas, al jefe del proyecto ATcnea y al diseñador del proyecto ATcnea.

Las No Conformidades detectadas en cada una de las 3 iteraciones definidas para el desarrollo del módulo fueron agrupadas en las siguientes tablas siendo evaluadas en un rango comprendido entre: Alta, Media, Baja y No procede.

Tabla 6. Elaboración Propia. Resultados de las pruebas de usabilidad en la 1ra iteración

Iteraciones	No conformidades				
	Alta	Media	Baja	No procede	Total
1	1	2	2	-	5
2	-	1	2	-	3
3	-	-	-	-	-
	1	3	4	-	8

Se resolvieron satisfactoriamente todas las no conformidades, al realizar las pruebas de confirmación todo estaba correcto y las pruebas de regresión mostraron que los cambios realizados no habían causado problemas en otras partes del sistema.

Tabla 7. Elaboración Propia. Resultados de las pruebas de usabilidad en la 2da iteración

Iteraciones	No conformidades				
	Alta	Media	Baja	No procede	Total
1	3	3	2	-	8
2	1	1	1	-	3
3	-	-	-	-	-
	4	4	3	-	11

Se resolvieron satisfactoriamente todas las no conformidades, al realizar las pruebas de confirmación todo estaba correcto y las pruebas de regresión mostraron que los cambios realizados no habían causado problemas en otras partes del sistema.

Tabla 8. Elaboración Propia. Resultados de las pruebas de usabilidad en la 1ra iteración

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

Se resolvieron satisfactoriamente todas las no conformidades, al realizar las pruebas de confirmación todo estaba correcto y las pruebas de regresión mostraron que los cambios realizados no habían causado problemas en otras partes del sistema.

A continuación, se muestra un gráfico donde se representa el total de no conformidades identificadas en el nivel de pruebas de sistema, en el que se puede apreciar que fueron resueltas cada una de las no conformidades identificadas, hasta llegar a la total corrección de las mismas.

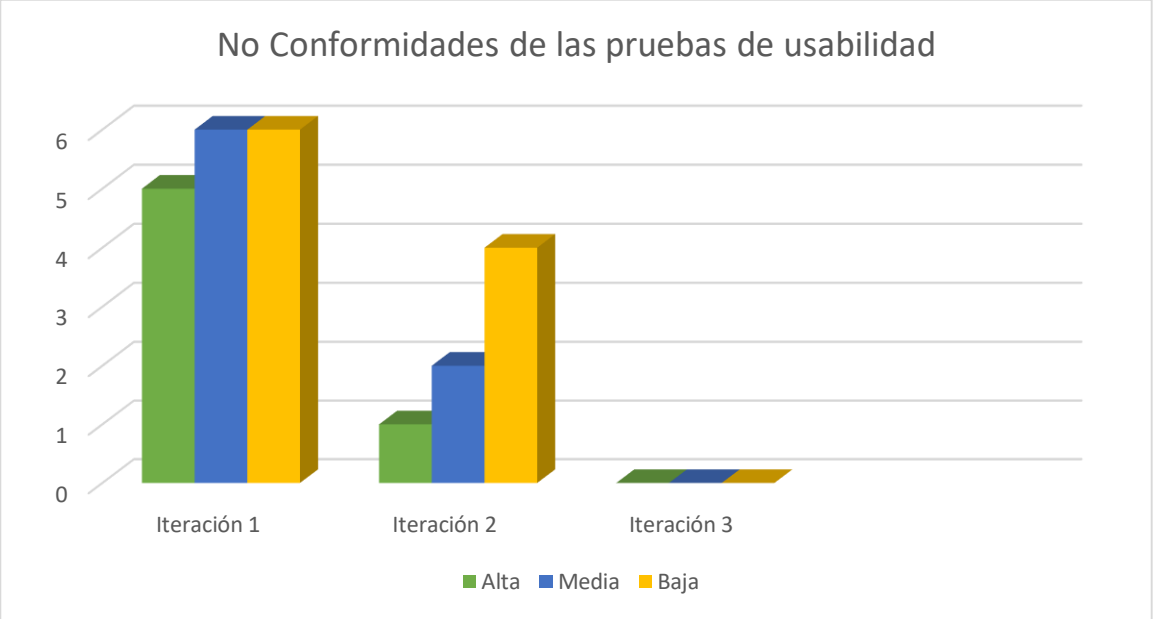


Figura 10. Elaboración Propia. Cantidad de No Conformidades por iteración de las pruebas de usabilidad

3.5. Disciplina Pruebas de Aceptación

El objetivo principal de las pruebas de aceptación es verificar si el producto de software desarrollado cumple con las normas de aceptación definidas en función de los requisitos del usuario y de la empresa, a fin de declararlo aceptable o no para su uso por parte de los usuarios. Las pruebas de aceptación son uno de los últimos tipos de pruebas de software realizadas en un software o aplicación. Lo lleva a cabo un grupo de usuarios específicos para garantizar la preparación y la calidad del sistema desde la perspectiva del usuario, lo que permite al equipo satisfacer sus necesidades y expectativas. (63)

3.5.1. Ejecución del Nivel de Pruebas de aceptación

El Nivel de pruebas de aceptación se ejecutó dentro de la Disciplina Pruebas de Aceptación donde se le realizaron al software 2 iteraciones de pruebas funcionales por cada una de las iteraciones del proceso de desarrollo del software definidas anteriormente:

1ra iteración: aplicación ATcnea-Teacher-PC.

2da iteración: aplicación ATcnea-Student-Android.

3ra iteración: aplicación ATcnea-Student-PC.

Para realizar las pruebas funcionales a la solución propuesta se utilizó el método de prueba caja negra haciendo uso de la técnica partición de equivalencia, para las cuales se utilizaron los casos de pruebas diseñados con anterioridad en el Capítulo 2. A continuación, se muestran en las siguientes tablas los resultados obtenidos.

Tabla 9. Elaboración Propia. Resultados de las pruebas funcionales en la 1ra iteración

Iteraciones	No conformidades				
	Alta	Media	Baja	No procede	Total
1	-	-	1	-	1
2	-	-	-	-	-
	-	-	1	-	1

En la 1ra iteración se detectó 1 no conformidad de tipo validación relacionada con el negocio.

Se resolvieron satisfactoriamente todas las no conformidades, al realizar las pruebas de confirmación todo estaba correcto y las pruebas de regresión mostraron que los cambios realizados no habían causado problemas en otras partes del sistema.

Tabla 10. Elaboración Propia. Resultados de las pruebas funcionales en la 2da iteración

Iteraciones	No conformidades				
	Alta	Media	Baja	No procede	Total
1	-	-	1	-	1
2	-	-	-	-	-
	-	-	1	-	1

En la 2da iteración se detectó 1 no conformidad de tipo validación relacionada con el negocio.

Se resolvieron satisfactoriamente todas las no conformidades, al realizar las pruebas de confirmación todo estaba correcto y las pruebas de regresión mostraron que los cambios realizados no habían causado problemas en otras partes del sistema.

Tabla 11. Elaboración Propia. Resultados de las pruebas funcionales en la 3ra iteración

Iteraciones	No conformidades				
	Alta	Media	Baja	No procede	Total
1	-	-	1	-	1
2	-	-	-	-	-
	-	-	1	-	1

En la 3ra iteración se detectó 1 no conformidad de tipo validación relacionada con el negocio.

Se resolvieron satisfactoriamente todas las no conformidades, al realizar las pruebas de confirmación todo estaba correcto y las pruebas de regresión mostraron que los cambios realizados no habían causado problemas en otras partes del sistema.

A continuación, se muestra un gráfico donde se representa el total de no conformidades identificadas en el nivel de pruebas de aceptación, en el que se puede apreciar que fueron resueltas cada una de las no conformidades identificadas, hasta llegar a la total corrección de las mismas.

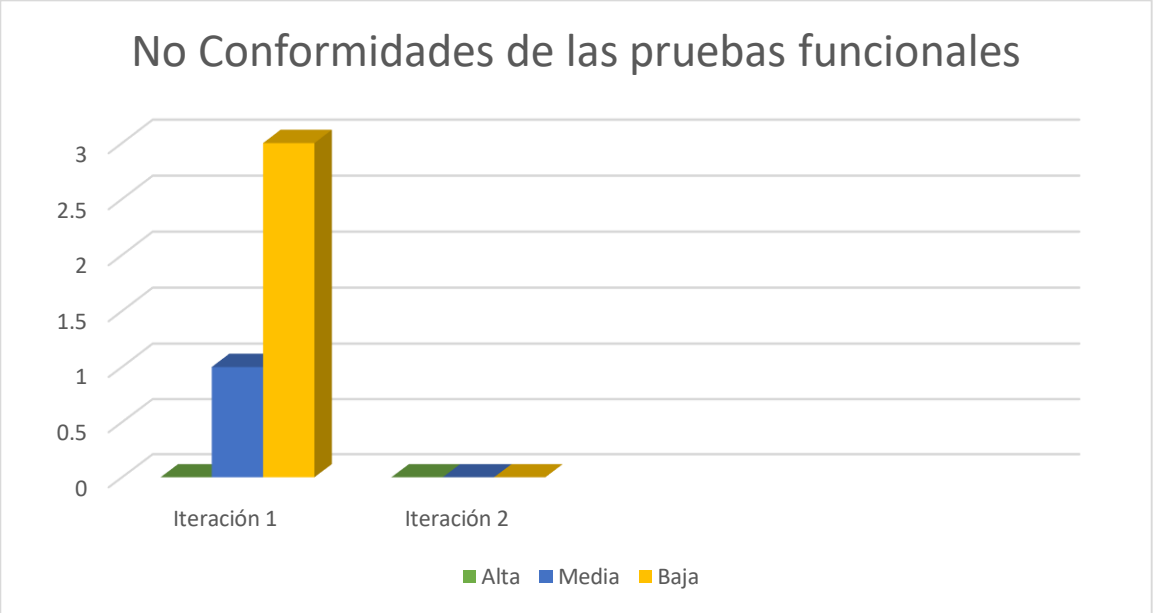


Figura 11. Elaboración Propia. Cantidad de No Conformidades por iteración de las pruebas funcionales

Una vez realizadas las pruebas funcionales en el nivel de pruebas de aceptación, se generó el Acta de aceptación del producto como constancia de la conformidad por parte del cliente de la solución propuesta, dando de esta manera culminación al proceso de pruebas de software.

3.6. Conclusiones del capítulo

En el presente capítulo se le realizaron al módulo Pizarra 1.1 pruebas en los niveles de integración, sistema y de aceptación, y se expusieron los resultados obtenidos, resolviendo de manera incremental las no conformidades detectadas en cada iteración de prueba con el objetivo de obtener un producto final que cumpla con los requerimientos planteados al inicio de la investigación.

CONCLUSIONES GENERALES

Con la culminación del presente trabajo de diploma se ha dado cumplimiento a los objetivos trazados en la investigación, obteniéndose como resultado principal, el módulo Pizarra 1.1 del software XAUCE ATcnea para dar soporte a las ecuaciones matemáticas. A continuación, se exponen las conclusiones generales:

- De los módulos pizarra estudiados, algunos no permiten trabajar con ecuaciones matemáticas y el resto son privativos, no se tiene acceso al código fuente y forman parte intrínseca de un software específico.
- El módulo Pizarra 1.1 permite el trabajo con ecuaciones matemáticas en el Aula tecnológica XAUCE ATcnea.
- El módulo Pizarra 1.1 cumple con los requerimientos definidos, lo cual se expresa en los resultados de las pruebas de calidad de software realizadas.

REFERENCIAS BIBLIOGRÁFICAS

1. Olivar, Anderson y Daza, Alfredo. Las Tecnologías de la Información y Comunicación (TIC) y su Impacto en la Educación del Siglo XXI; Julio 2007. 22 p.
2. Sitio oficial de la Universidad de las Ciencias Informáticas [en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: http://www.uci.cu/universidad/mision
3. Sitio oficial de la Universidad de las Ciencias Informáticas [en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: https://www.uci.cu/investigacion-y desarrollo/productos/xauce/atcnea-10
4. Marshall, Grettel y Aguilera, Osvaldo. Módulo Complejo Residencial v2.0 para el Sistema de Gestión de Residencia; junio 2013. 5 p.
5. GEDEME [en línea] [fecha de consulta: 4 febrero 2019]. Disponible en: http://www.gedeme.cu/es/tecnologia-de-la-informacion/aula-tecnologica
6. Tecnologia [en línea] [fecha de consulta: 20 marzo 2019]. Disponible en: https://tecnologiaintegrada.com.mx/aula-interactiva/
7. Definiciones ABC [en línea] [fecha de consulta: 18 enero 2019]. Disponible en: https://www.definicionabc.com/general/ecuacion.php
8. Guillen, Michael. Cinco ecuaciones que cambiaron el mundo: El poder y la oculta belleza de las matemáticas, 1995
9. Ecuaciones matemáticas [en línea] [fecha de consulta: 20 marzo 2019]. Disponible en: https://ecuacionesmatematicas.wordpress.com/2010/09/25/la-importancia-de-las ecuaciones-matematicas-en-la-educacion
10. Educarm [en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: http://servicios.educarm.es/admin/webForm.php?aplicacion=PIZARRA_DIGITAL&mode =visualizaAplicacionWeb&web=37&ar=332&liferay=1&zona=EDUCARM
11. Eztalks [en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: https://www.eztalks.com/whiteboard/benefits-of-interactive-whiteboards-in-the-classroom.html
12. OpenBoard 1.5 User's Guide
13. BOXLIGHT en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: boxlight.com/mimiostudio
14. Mythware [en línea] [fecha de consulta:13 octubre 2018]. Disponible en: http://www.mythware.com/classroom-management-software
15. Interactive whiteboard v4.50, Software User Manual.

16. Java [en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: https://www.java.com/es/download/faq/whatis_java.xml
17. Wikipedia [en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: https://en.wikipedia.org/wiki/FXML
18. Ecured [en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: https://www.ecured.cu/CSS3
19. Silberschatz, Abraham. McGRAW-HILL, ed. Fundamentos de bases de datos.
20. Cáceres González, Abdiel. Lenguajes de Programación. Instituto tecnológico de Monterrey, México. 2015
21. Oracle [en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm
22. Techopedia [en línea] [fecha de consulta: 26 abril 2019]. Disponible en: https://www.techopedia.com/definition/4220/android-sdk
23. Oracle [en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: https://www.oracle.com/technetwork/java/javase/ downloads/jdk8-downloads 2133151.html
24. DesarrolloWeb [en línea] [fecha de consulta: 15 octubre 2018]. Disponible en: https://desarrolloweb.com/articulos/introduccion-git-github.html
25. Wikipedia [en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: https://es.wikipedia.org/wiki/Maven
26. OpenWebinars [en línea] [fecha de consulta: 27 noviembre 2018]. Disponible en: https://openwebinars.net/blog/que-es-gradle/
27. Oracle [en línea] [fecha de consulta: 22 noviembre 2018]. Disponible en: https://docs.oracle.com/javafx/scenebuilder/2/ release_notes_2-0/jsbpub-release_notes_2-0.htm
28. JetBrains [en línea] [fecha de consulta: 2 febrero 2019]. Disponible en: https://www.jetbrains.com/idea/features/
29. Academia Android. [en línea] [fecha de consulta: 18 enero 2019]. Disponible en: https://academiaandroid.com/android-studio-v1-caracteristicas-comparativa eclipse/.
30. PARADIGM V. Visual paradigm for uml. Visual Paradigm for UML-UML tool for software application development. 2013 p. 72.
31. Rodríguez Sánchez T. Metodología de desarrollo para la Actividad productiva de la UCI. La Habana, Cuba: Universidad de las Ciencias Informáticas; 2015.
32. Sommerville I. Software Engineering. 8va ed. Pearson Education Limited; 2006. 824 p.

33. Canós JH, Letelier P, Penadés MC. Metodologías Ágiles en el Desarrollo de Software. Valencia, España: Universidad Politécnica de Valencia; p. 8.
34. Jacobson, IvarBooch, et al. El proceso unificado de desarrollo de software/The unified software development process. s.l.: Pearson Educación, 2000.
35. Pressman RS. Ingeniería de software: Un enfoque práctico. 6ta ed. Nueva York, EUA: McGrawHill; 2007.
36. Coding or not [en línea] [fecha de consulta: 28 febrero 2019]. Disponible en: https://codingornot.com/mvc-modelo-vista-controlador-que-es-y-para-que-sirve
37. Medium [en línea] [fecha de consulta: 28 febrero 2019]. Disponible en: https://medium.com/@maniakhitoccori/los-10-patrones-comunes-de-arquitectura-de-software-d8b9047edf0b
38. Larman, Craig. UML y Patrones. 2ª Edición. 2003.
39. Guerrero CA, Suárez JM, Gutiérrez LE. Patrones de Diseño GOF (The Gang of Four) en el contexto de Procesos de Desarrollo de Aplicaciones Orientadas a la Web. Santander, Colombia; 2009 p. 12.
40. Reactiveprogramming [en línea] [fecha de consulta: 28 febrero 2019]. Disponible en: https://reactiveprogramming.io/books/design-patterns/es/catalog/singleton
41. Informaticapc [en línea] [fecha de consulta: 28 febrero 2019]. Disponible en: https://informaticapc.com/patrones-de-diseno/composite.php
42. Arias, Manuel. Carmen. Los estándares de codificación, 2014.
43. Sities Google [en línea] [fecha de consulta: 28 marzo 2019]. Disponible en: https://sites.google.com/site/investigacionesitlm/unidad-2/2-2-4-protocolos-udp
44. CCM [en línea] [fecha de consulta: 28 marzo 2019]. Disponible en: https://es.ccm.net/contents/281-protocolo-tcp
45. Scribd [en línea] [fecha de consulta: 12 abril 2019]. Disponible en: https://es.scribd.com/document/348969126/03-Casos-de-Prueba-Particion-Equivalente
46. Montañez, Luis. Diseño de un modelo para el control de la calidad de los proyectos de desarrollo de software en las organizaciones de los estados venezolanos, 2006.
47. Vega, Prieto y Rodríguez, Zaylí. Procedimiento para realizar pruebas de usabilidad. Universidad de las Ciencias Informáticas, 2010.
48. Prado, Elena. Casi todas las pruebas del software, 2007.
49. Rojas, Johanna y Barrios, Emilio. Investigación sobre estado del arte en diseño y aplicación de pruebas de software, 2007.

50. Software Testing [en línea] [fecha de consulta: 20 mayo 2019]. Disponible en: http://softwaretestingfundamentals.com/unit-testing/
51. Software Testing [en línea] [fecha de consulta: 20 mayo 2019]. Disponible en: http://softwaretestingfundamentals.com/integration-testing/
52. Software Testing [en línea] [fecha de consulta: 17 abril 2019]. Disponible en: softwaretestingfundamentals.com/acceptance-testing/
53. Software Testing [en línea] [fecha de consulta: 27 mayo 2019]. Disponible en: http://softwaretestingfundamentals.com/functional-testing/
54. USABIITY [en línea] [fecha de consulta: 22 abril 2019]. Disponible en: [http://Usability.gov.]
55. ProfessionalQA.com [en línea] [fecha de consulta: 25 mayo 2019]. Disponible en: http://www.professionalqa.com/confirmation-testing
56. Software Testing [en línea] [fecha de consulta: 20 mayo 2019]. Disponible en: http://softwaretestingfundamentals.com/regression-testing/
57. Software Testing [en línea] [fecha de consulta: 12 febrero 2019]. Disponible en: http://softwaretestingfundamentals.com/white-box-testing/
58. Pérez, Cesar. Fundamentos de pruebas de software. Diseño de caso de prueba de la caja blanca y del camino básico. Universidad Estatal del Sur de Manabí. Ecuador, 2001.
59. Software Testing [en línea] [fecha de consulta: 14 abril 2019]. Disponible en: http://softwaretestingfundamentals.com/black-box-testing/
60. Cidecame [en línea] [fecha de consulta: 26 mayo 2019]. Disponible en: http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro10/354_implementacin__y_pruebas.html
61. tutorialspoint [en línea] [fecha de consulta: 25 mayo 2019]. Disponible en: https://www.tutorialspoint.com/junit/
62. Pruebas [en línea] [fecha de consulta: 8 abril 2019]. Disponible en: https://modelosdepruebasw.wordpress.com/
63. ProfessionalQA.com [en línea] [fecha de consulta: 25 mayo 2019]. Disponible en: http://www.professionalqa.com/acceptance-testing

ANEXOS

Anexo A: Historias de usuario

Tabla 12. Elaboración propia. Historia de Usuario del Requisito funcional Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Student-PC

Número: 2	**Nombre del requisito:** Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Student-PC
Programador: Brian Mena Gómez	**Iteración Asignada:** 3ra
Prioridad: Alta	**Tiempo Estimado:** 120 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 110 horas
Objetivo: Permitir listar ecuaciones al estudiante. **Acciones para lograr el objetivo (precondiciones y datos):** Para listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Student-PC se tiene que: - El usuario debe de estar autenticado en el sistema con el rol Estudiante. - El usuario debe haber seleccionado la opción buscar clase. - El usuario debe haber seleccionado una clase ya creada. - El usuario debe haber seleccionado la opción pizarra. **3- Flujo de la acción a realizar:** Una vez en la pizarra el profesor puede seleccionar la opción Ecuación, con la cual se le desplegará una lista con 21 ecuaciones matemáticas donde este puede seleccionar la que desee.	
Prototipo elemental de interfaz gráfica de usuario:	

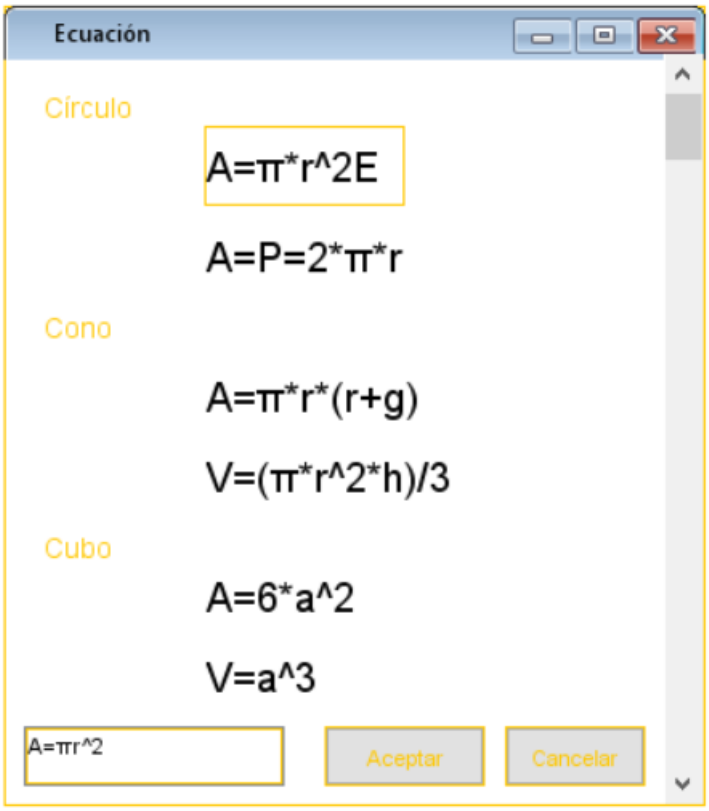

Ecuación
Círculo
A=π*r^2E
A=P=2*π*r
Cono
A=π*r*(r+g)
V=(π*r^2*h)/3
Cubo
A=6*a^2
V=a^3
A=πr^2
Aceptar
Cancelar

Tabla 13. Elaboración propia. Historia de Usuario del Requisito funcional Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Student-Android

Número: 3	**Nombre del requisito:** Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Student-Android
Programador: Brian Mena Gómez	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:** 120 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 110 horas

Objetivo:

Permitir listar ecuaciones al estudiante.

Acciones para lograr el objetivo (precondiciones y datos):

Para listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Student-Android se tiene que:

- El usuario debe de estar autenticado en el sistema con el rol Estudiante.

- El usuario debe haber seleccionado la opción buscar clase.

- El usuario debe haber seleccionado una clase ya creada.

- El usuario debe haber seleccionado la opción pizarra.

3- Flujo de la acción a realizar:

Una vez en la pizarra el usuario presiona el botón herramientas de dibujo, al presionarlo se le despliega una serie de herramientas, dentro de estas herramientas selecciona la opción Ecuación, con la cual se le desplegará una lista con 21 ecuaciones matemáticas donde este puede seleccionar la que desee.

Prototipo elemental de interfaz gráfica de usuario:

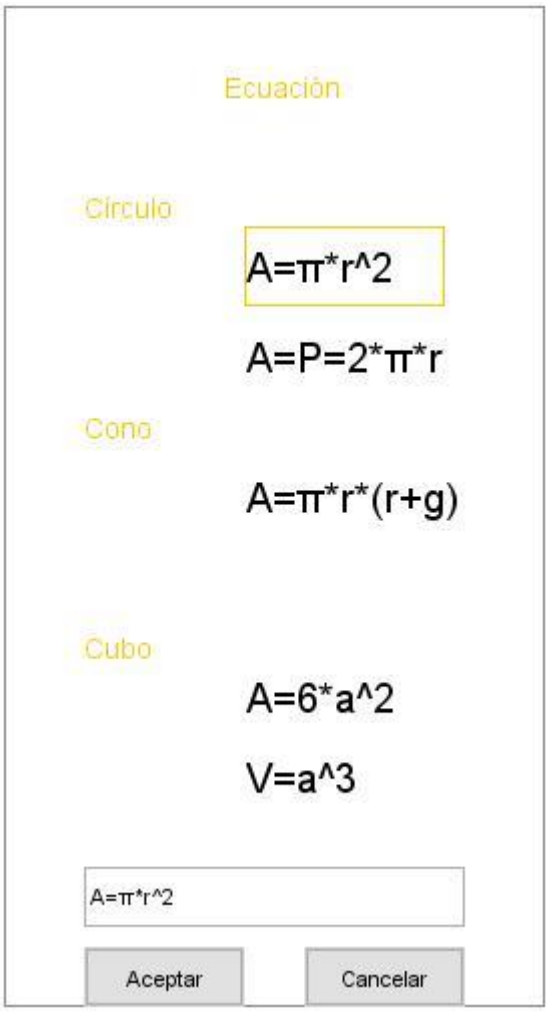

Ecuación
Círculo
A=π*r^2
A=P=2*π*r
Cono
A=π*r*(r+g)
Cubo
A=6*a^2
V=a^3
A=π*r^2
Aceptar
Cancelar

Tabla 14. Elaboración propia. Historia de Usuario del Requisito funcional Insertar ecuación en el módulo Pizarra de la aplicación ATcnea- Teacher-PC

Número: 4	**Nombre del requisito:** Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Teacher-PC
Programador: Brian Mena Gómez	**Iteración Asignada:** 3ra
Prioridad: Alta	**Tiempo Estimado:** 120 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 110 horas
Objetivo: Permitir insertar ecuaciones al profesor. **Acciones para lograr el objetivo (precondiciones y datos):** Para insertar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Teacher-PC se tiene que: - El usuario debe de estar autenticado en el sistema con el rol Profesor. - El usuario debe haber creado una clase. - El usuario debe haber seleccionado la opción Pizarra. **3- Flujo de la acción a realizar:** Una vez en la pizarra el profesor puede seleccionar la opción Ecuación, con la cual se le desplegará una lista con 21 ecuaciones matemáticas donde este puede seleccionar la que desee e insertarla en la pizarra.	
Prototipo elemental de interfaz gráfica de usuario:	

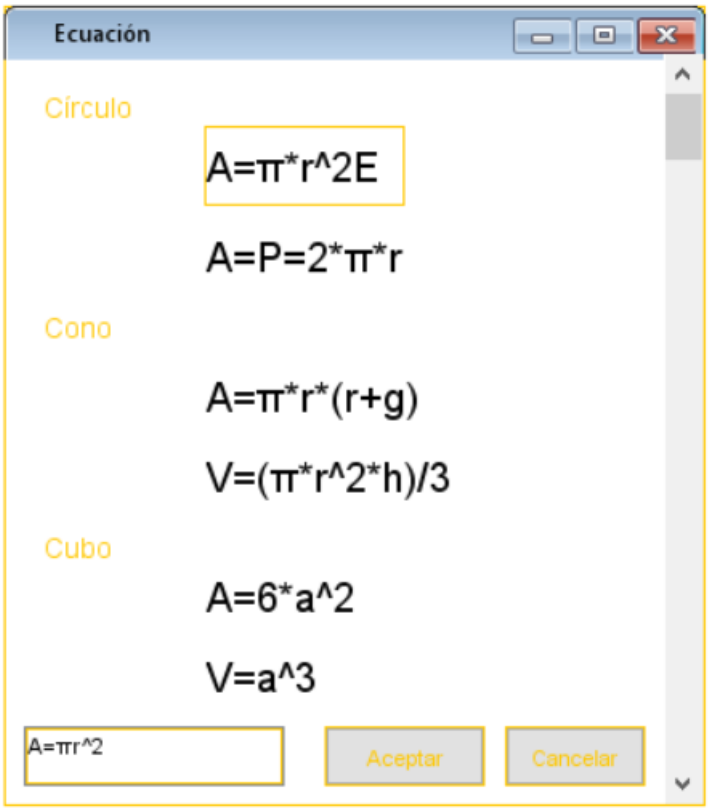

Ecuación
Círculo
A=π*r^2E
A=P=2*π*r
Cono
A=π*r*(r+g)
V=(π*r^2*h)/3
Cubo
A=6*a^2
V=a^3
A=πr^2
Aceptar
Cancelar

Tabla 15. Elaboración propia. Historia de Usuario del Requisito funcional Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Student-Android

Número: 5	**Nombre del requisito:** Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Student-Android
Programador: Brian Mena Gómez	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:** 120 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 110 horas

Objetivo:

Permitir insertar ecuaciones al estudiante.

Acciones para lograr el objetivo (precondiciones y datos):

Para insertar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Student-Android se tiene que:

- El usuario debe de estar autenticado en el sistema con el rol Estudiante.

- El usuario debe haber seleccionado la opción buscar clase.

- El usuario debe haber seleccionado una clase ya creada.

- El usuario debe haber seleccionado la opción pizarra.

3- Flujo de la acción a realizar:

Una vez en la pizarra el usuario presiona el botón herramientas de dibujo, al presionarlo se le despliega una serie de herramientas, dentro de estas herramientas selecciona la opción Ecuación, con la cual se le desplegará una lista con 21 ecuaciones matemáticas donde este puede seleccionar la que desee e insertarla en la pizarra.

Prototipo elemental de interfaz gráfica de usuario:

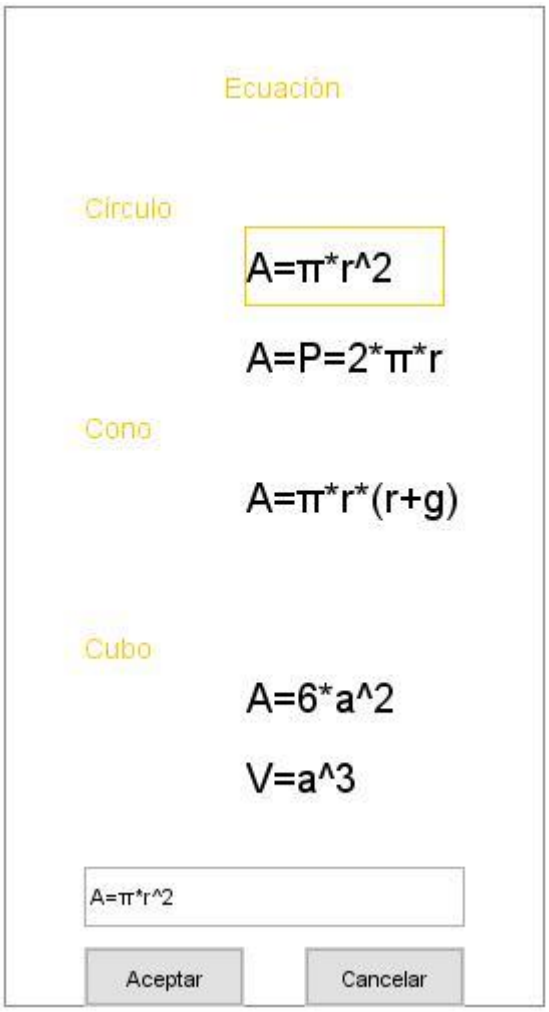

Ecuación
Círculo
A=π*r^2
A=P=2*π*r
Cono
A=π*r*(r+g)
Cubo
A=6*a^2
V=a^3
A=π*r^2
Aceptar
Cancelar

Tabla 16. Elaboración propia. Historia de Usuario del Requisito funcional Guardar pizarra en el módulo Pizarra de la aplicación ATcnea-Teacher-PC

Número: 4	**Nombre del requisito:** Guardar pizarra en el módulo Pizarra de la aplicación ATcnea-Teacher-PC
Programador: Brian Mena Gómez	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 120 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 110 horas
Objetivo: Permitir guardar la pizarra al profesor. **Acciones para lograr el objetivo (precondiciones y datos):** Para guardar la pizarra en el módulo Pizarra de la aplicación ATcnea-Teacher-PC se tiene que: - El usuario debe de estar autenticado en el sistema con el rol Profesor. - El usuario debe haber creado una clase. - El usuario debe haber seleccionado la opción pizarra. **3- Flujo de la acción a realizar:** Una vez en la pizarra el profesor presiona el botón Guardar pizarra y se le muestra una ventana donde puede asignarle un nombre a la pizarra que guardará y escoger la locación para guardar la pizarra.	
Prototipo elemental de interfaz gráfica de usuario:	

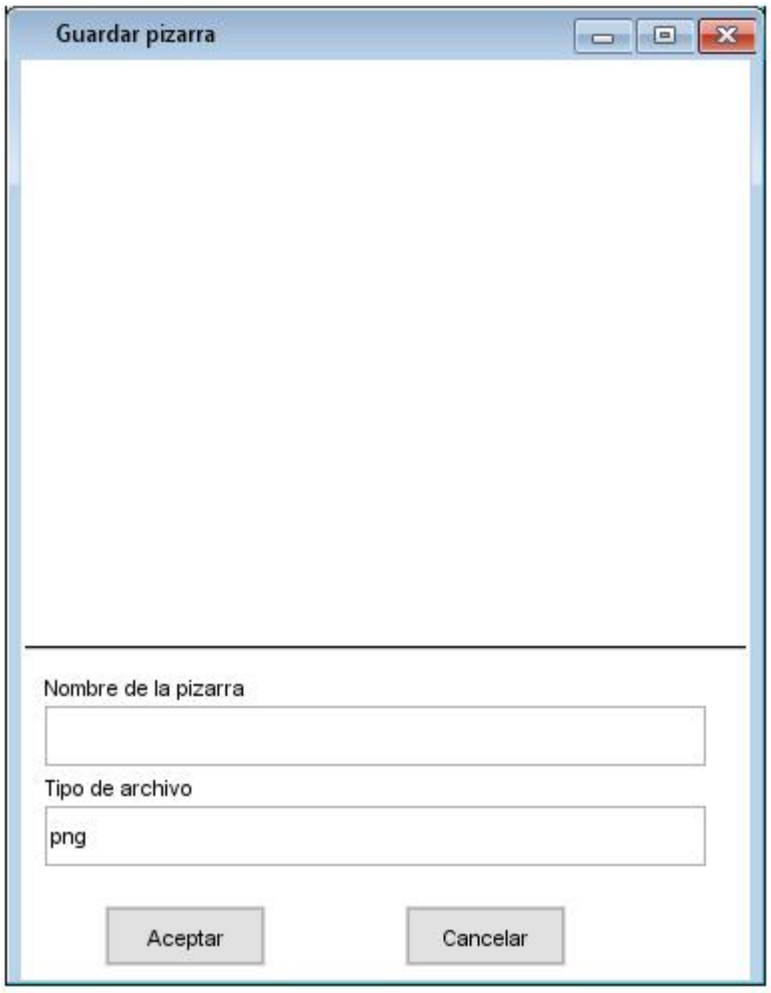

Guardar pizarra
Nombre de la pizarra
Tipo de archivo
png
Aceptar
Cancelar

Tabla 17. Elaboración propia. Historia de Usuario del Requisito funcional Guardar pizarra en el módulo Pizarra de la aplicación ATcnea-Student-Android

Número: 5	**Nombre del requisito:** Guardar pizarra en el módulo Pizarra de la aplicación ATcnea-Student-Android
Programador: Brian Mena Gómez	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:** 120 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 110 horas
Objetivo: Permitir guardar la pizarra al estudiante. **Acciones para lograr el objetivo (precondiciones y datos):** Para guardar la pizarra en el módulo Pizarra de la aplicación ATcnea-Student-Android se tiene que: - El usuario debe de estar autenticado en el sistema con el rol Estudiante. - El usuario debe haber seleccionado la opción buscar clase. - El usuario debe haber seleccionado una clase ya creada. - El usuario debe haber seleccionado la opción pizarra. **3- Flujo de la acción a realizar:** Una vez en la pizarra el estudiante presiona el botón Guardar pizarra y se le muestra una ventana donde puede asignarle un nombre a la pizarra que guardará y escoger la locación para guardar la pizarra.	
Prototipo elemental de interfaz gráfica de usuario:	

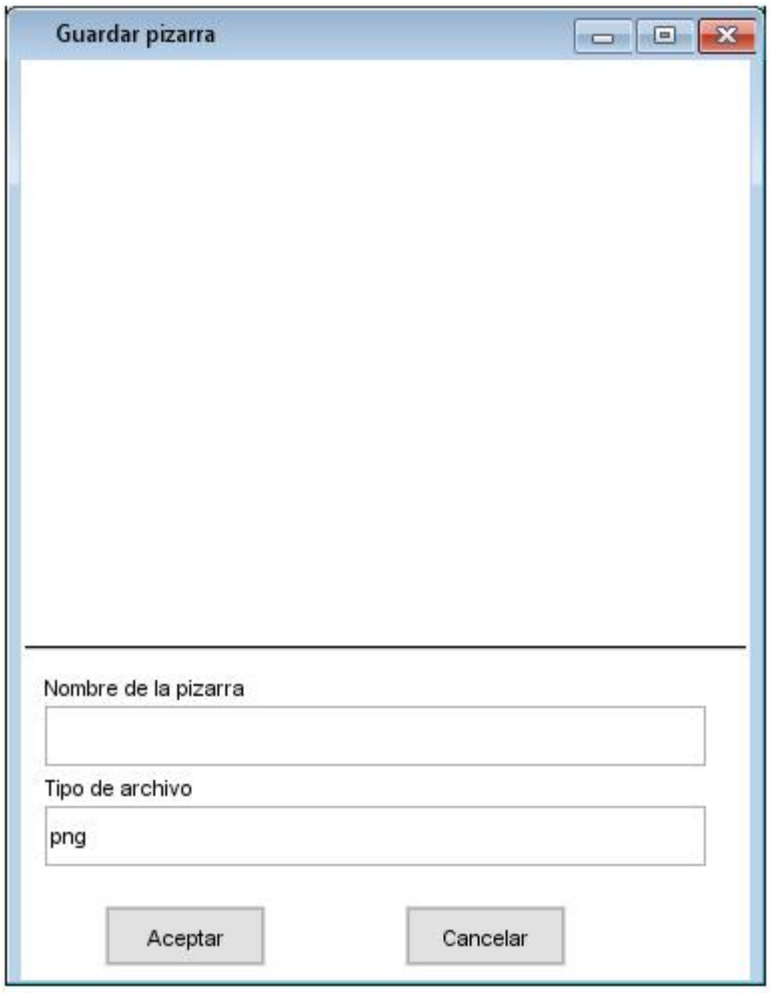

Guardar pizarra
Nombre de la pizarra
Tipo de archivo
png
Aceptar
Cancelar

Tabla 18. Elaboración propia. Historia de Usuario del Requisito funcional Cargar pizarra en el módulo Pizarra de la aplicación ATcnea-Teacher-PC

Número: 4	**Nombre del requisito:** Cargar pizarra en el módulo Pizarra de la aplicación ATcnea-Teacher-PC
Programador: Brian Mena Gómez	**Iteración Asignada:** 1ra
Prioridad: Alta	**Tiempo Estimado:** 120 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 110 horas
Objetivo: Permitir cargar la pizarra al profesor. **Acciones para lograr el objetivo (precondiciones y datos):** Para cargar la pizarra en el módulo Pizarra de la aplicación ATcnea-Teacher-PC se tiene que: - El usuario debe de estar autenticado en el sistema con el rol Profesor. - El usuario debe haber creado una clase. - El usuario debe haber seleccionado la opción pizarra. **3- Flujo de la acción a realizar:** Una vez en la pizarra el profesor presiona el botón Cargar pizarra y se le muestra una ventana donde puede buscar la locación de la pizarra guardada anteriormente y seleccionar esta.	
Prototipo elemental de interfaz gráfica de usuario:	


Cargar pizarra
Archvo seleccionado
miPIzarra
Aceptar
Cancelar

Tabla 19. Elaboración propia. Historia de Usuario del Requisito funcional Cargar pizarra en el módulo Pizarra de la aplicación ATcnea-Student-Android

Número: 6	**Nombre del requisito:** Cargar pizarra en el módulo Pizarra de la aplicación ATcnea-Student-Android
Programador: Brian Mena Gómez	**Iteración Asignada:** 2da
Prioridad: Alta	**Tiempo Estimado:** 120 horas
Riesgo en Desarrollo: Alto	**Tiempo Real:** 110 horas

Objetivo:

Permitir cargar la pizarra al estudiante.

Acciones para lograr el objetivo (precondiciones y datos):

Para cargar la pizarra en el módulo Pizarra de la aplicación ATcnea-Student-Android se tiene que:

- El usuario debe de estar autenticado en el sistema con el rol Estudiante.

- El usuario debe haber seleccionado la opción buscar clase.

- El usuario debe haber seleccionado una clase ya creada.

- El usuario debe haber seleccionado la opción pizarra.

3- Flujo de la acción a realizar:

Una vez en la pizarra el estudiante presiona el botón Cargar pizarra y se le muestra una ventana donde puede buscar la locación de la pizarra guardada anteriormente y seleccionar esta.

Prototipo elemental de interfaz gráfica de usuario:


Cargar pizarra
Archvo seleccionado
miPIzarra
Aceptar
Cancelar

Anexo B: Diseño de Casos de Prueba (DCP)

Tabla 20. Elaboración propia. DCP correspondiente al requisito funcional Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Student-PC

Escenario	Descripción	Ecuación	Respuesta del sistema	Flujo central
EC 1.1 Opción Ecuación	El usuario selecciona la opción Ecuación.	N/A	El sistema muestra un listado de las ecuaciones agrupadas por figuras para que el usuario seleccione una ecuación. También muestra el campo modificar ecuación y las opciones: *Aceptar *Cancelar	Clase X /Pizarra/Ecuación
EC 1.2 Opción de cancelar	El usuario selecciona la opción de Cancelar.	N/A	El sistema no inserta la ecuación matemática en la pizarra y cierra la ventana.	Clase X /Pizarra/Ecuación/Cancelar
EC 1.3 Opción de aceptar	El usuario selecciona la opción Aceptar.	V	El sistema inserta la ecuación matemática seleccionada en la pizarra. ***Ver DCP: "Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Student-PC".***	Clase X /Pizarra/Ecuación/Aceptar

Tabla 21. Elaboración propia. DCP correspondiente al requisito funcional Listar ecuaciones en el módulo Pizarra de la aplicación ATcnea-Student-Android

Escenario	Descripción	Ecuación	Respuesta del sistema	Flujo central
EC 1.1 Opción Ecuación	El usuario selecciona la opción Ecuación.	N/A	El sistema muestra un listado de las ecuaciones agrupadas por figuras para que el usuario seleccione una ecuación. También muestra el campo modificar ecuación y las opciones: *Aceptar *Cancelar	Clase X /Pizarra/Herramientas de dibujo/Ecuación
EC 1.2 Opción de cancelar	El usuario selecciona la opción de Cancelar.	N/A	El sistema no inserta la ecuación matemática en la pizarra y cierra la ventana.	Clase X /Pizarra/Herramientas de dibujo/Ecuación/Cancelar
EC 1.3 Opción de aceptar	El usuario selecciona la opción Aceptar.	V	El sistema inserta la ecuación matemática seleccionada en la pizarra. ***Ver DCP: "Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Student-Android".***	Clase X /Pizarra/Herramientas de dibujo/Ecuación/Aceptar

Tabla 22. Elaboración propia. DCP correspondiente al requisito funcional Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Teacher-PC

Escenario	Descripción	Ecuación	Respuesta del sistema	Flujo central
EC 1.1 Opción Ecuación	El usuario selecciona la opción Ecuación.	N/A	El sistema muestra un listado de las ecuaciones agrupadas por figuras para que el usuario seleccione una ecuación. También muestra el campo modificar ecuación y las opciones: *Aceptar *Cancelar	Clase X /Pizarra/Ecuación
EC 1.2 Opción de cancelar	El usuario selecciona la opción de Cancelar.	N/A	El sistema no inserta la ecuación matemática en la pizarra y cierra la ventana.	Clase X /Pizarra/Ecuación/Cancelar
EC 1.3 Opción de aceptar	El usuario selecciona la opción Aceptar.	V	Valida los datos. Inserta la ecuación seleccionada en la pizarra.	Clase X /Pizarra/Ecuación/Aceptar
EC 1.4 Datos incompletos	Existen datos incompletos.	I	Muestra un mensaje de información. Regresa al EC 1.2.	Clase X /Pizarra/Ecuación/Aceptar
		V		
EC 1.5 Datos incorrectos	Existen datos incorrectos.	I	Muestra un mensaje de información. Regresa al EC 1.2.	Clase X /Pizarra/Ecuación/Aceptar

		V		

Tabla 23. Elaboración propia. DCP correspondiente al requisito funcional Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Student-Android

Escenario	Descripción	Ecuación	Respuesta del sistema	Flujo central
EC 1.1 Opción Ecuación	El usuario selecciona la opción Ecuación.	N/A	El sistema muestra un listado de las ecuaciones agrupadas por figuras para que el usuario seleccione una ecuación. También muestra el campo modificar ecuación y las opciones: *Aceptar *Cancelar	Clase X /Pizarra/Herramientas de dibujo/Ecuación
EC 1.2 Opción de cancelar	El usuario selecciona la opción de Cancelar.	N/A	El sistema no inserta la ecuación matemática en la pizarra y cierra la ventana.	Clase X /Pizarra/ Herramientas de dibujo/Ecuación/Cancelar
EC 1.3 Opción de aceptar	El usuario selecciona la opción Aceptar.	V	Valida los datos. Inserta la ecuación seleccionada en la pizarra.	Clase X /Pizarra/ Herramientas de dibujo/Ecuación/Aceptar
EC 1.4 Datos incompletos	Existen datos incompletos.	I	Muestra un mensaje de información. Regresa al EC 1.2.	Clase X /Pizarra/ Herramientas de dibujo/Ecuación/Aceptar
		V		

EC 1.5 Datos incorrectos	Existen datos incorrectos.	I	Muestra un mensaje de información. Regresa al EC 1.2.	Clase X /Pizarra/ Herramientas de dibujo/Ecuación/Aceptar

Tabla 24. Elaboración propia. DCP correspondiente al requisito funcional Guardar pizarra en el módulo Pizarra de la aplicación ATcnea-Teacher-PC

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción Guardar pizarra	El usuario selecciona la opción Guardar pizarra	El sistema permite al usuario guardar la pizarra en la locación escogida y con el nombre elegido.	Clase X/ Pizarra / Guardar pizarra
EC 1.2 Opción de aceptar	El usuario selecciona la opción Aceptar.	El sistema guarda la pizarra en la locación escogida y con el nombre elegido.	Clase X/ Pizarra / Guardar pizarra/Aceptar
EC 1.3 Opción de cancelar	El usuario selecciona la opción Cancelar.	El sistema no guarda la pizarra en la locación escogida y cierra la ventana.	Clase X/ Pizarra / Guardar pizarra/Cancelar

Tabla 25. Elaboración propia. DCP correspondiente al requisito funcional Guardar pizarra en el módulo Pizarra de la aplicación ATcnea-Student-Android

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción Guardar pizarra	El usuario selecciona la opción Guardar pizarra	El sistema permite al usuario guardar la pizarra en la locación escogida y con el nombre elegido.	Clase X/ Pizarra / Herramientas de guardado/ Guardar pizarra
EC 1.2 Opción de aceptar	El usuario selecciona la opción Aceptar.	El sistema guarda la pizarra en la locación escogida y con el nombre elegido.	Clase X/ Pizarra / Herramientas de guardado/ Guardar pizarra/Aceptar
EC 1.3 Opción de cancelar	El usuario selecciona la opción Cancelar.	El sistema no guarda la pizarra en la locación escogida y cierra la ventana.	Clase X/ Pizarra / Herramientas de guardado/ Guardar pizarra/Cancelar

Tabla 26. Elaboración propia. DCP correspondiente al requisito funcional Cargar pizarra en el módulo Pizarra de la aplicación ATcnea-Teacher-PC

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción Cargar pizarra	El usuario selecciona la opción Cargar pizarra	El sistema permite al usuario cargar la pizarra guardada desde su locación.	Clase X/ Pizarra / Cargar pizarra
EC 1.2 Opción de aceptar	El usuario selecciona la opción Aceptar.	El sistema carga la pizarra desde locación escogida.	Clase X/ Pizarra / Cargar pizarra/Aceptar

EC 1.3 Opción de cancelar	El usuario selecciona la opción Cancelar.	El sistema no carga la pizarra y cierra la ventana.	Clase X/ Pizarra / Cargar pizarra/Cancelar

Tabla 27. Elaboración propia. DCP correspondiente al requisito funcional Cargar pizarra en el módulo Pizarra de la aplicación ATcnea-Student-Android

Escenario	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Opción Cargar pizarra	El usuario selecciona la opción Cargar pizarra	El sistema permite al usuario cargar la pizarra guardada desde su locación.	Clase X/ Pizarra / Herramientas de guardado/Cargar pizarra
EC 1.2 Opción de aceptar	El usuario selecciona la opción Aceptar.	El sistema carga la pizarra desde locación escogida.	Clase X/ Pizarra / Herramientas de guardado/Cargar pizarra/Aceptar
EC 1.3 Opción de cancelar	El usuario selecciona la opción Cancelar.	El sistema no carga la pizarra y cierra la ventana.	Clase X/ Pizarra / Herramientas de guardado/Cargar pizarra/Cancelar

Anexo C: Diagrama de clases del diseño

Figura 12. Elaboración propia. RF Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Teacher-PC y la aplicación ATcnea-Student-PC

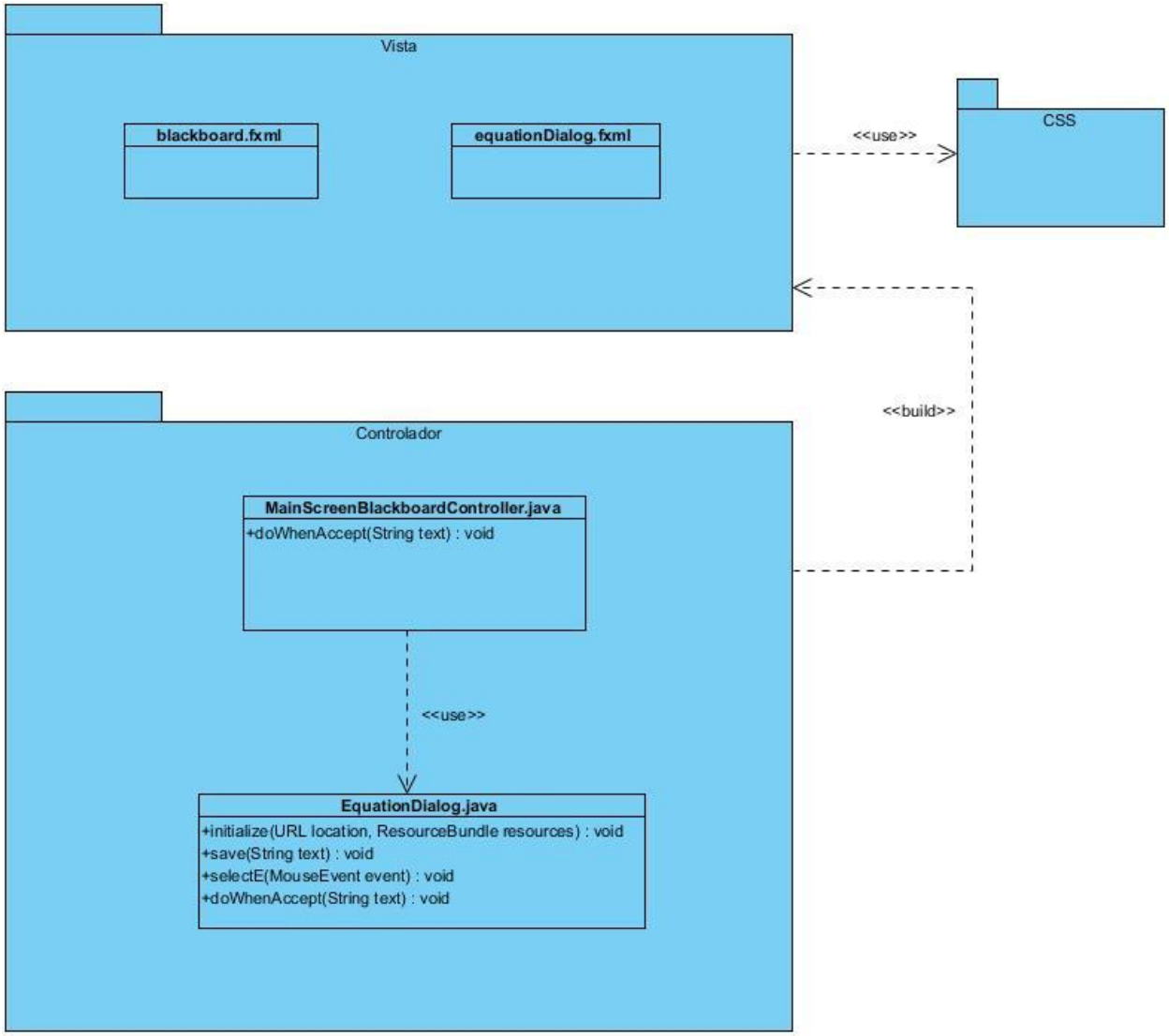


Figura 13. Elaboración propia. RF Insertar ecuación en el módulo Pizarra de la aplicación ATcnea-Student-Android

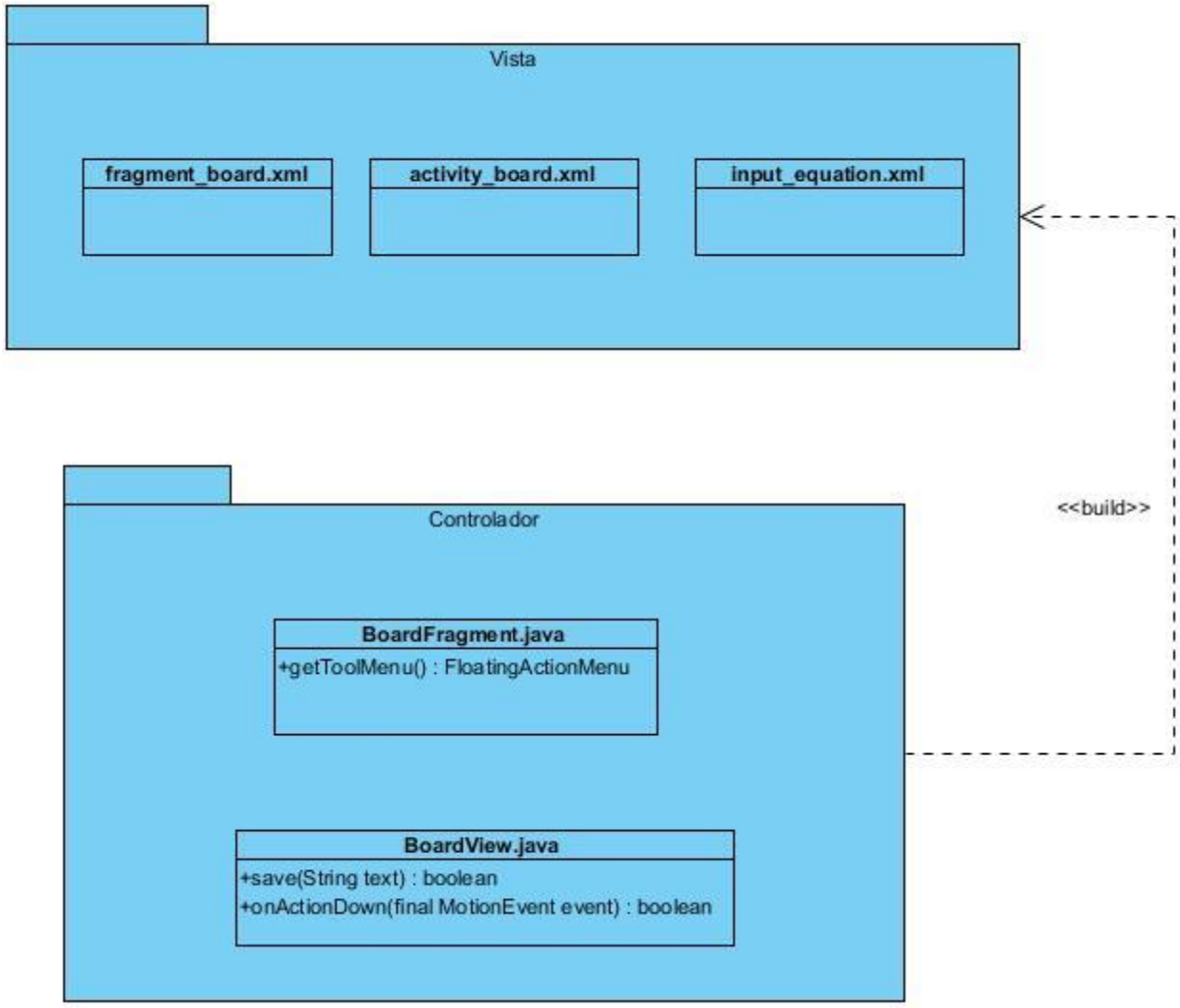


Figura 14. Elaboración propia. RF Guardar pizarra en el módulo Pizarra de la aplicación ATcnea-Teacher-PC y la aplicación ATcnea-Student-PC

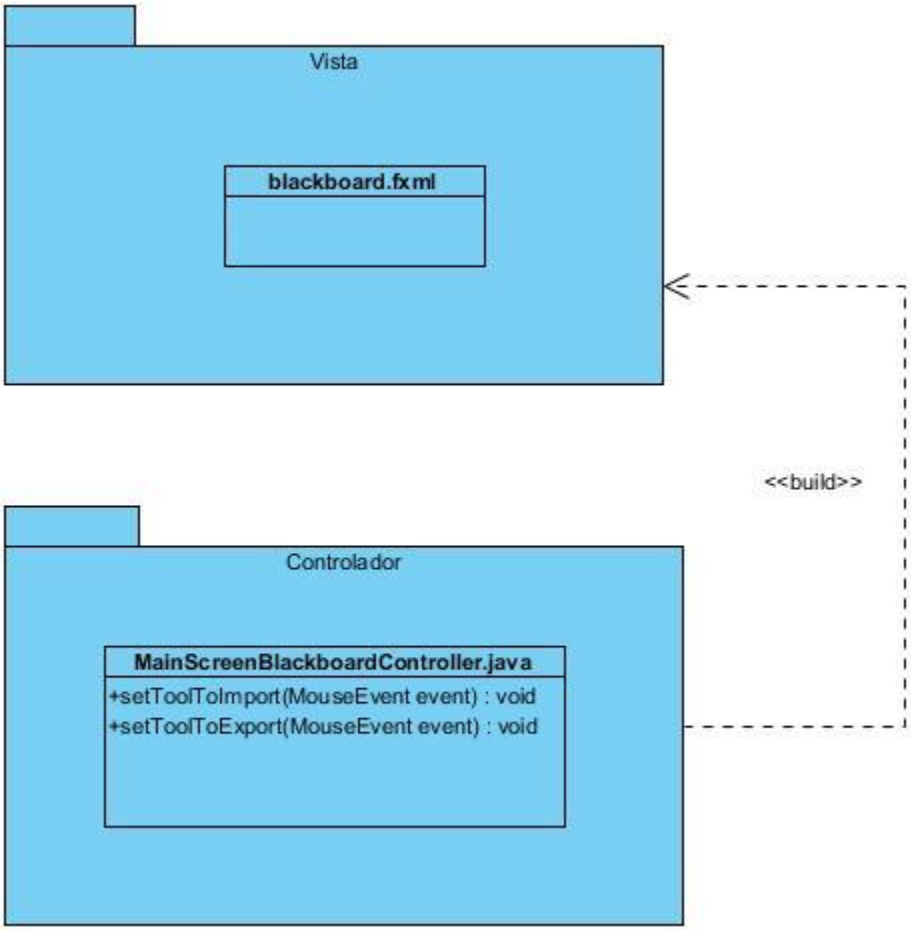


Figura 15. Elaboración propia. RF Guardar pizarra en el módulo Pizarra de la aplicación ATcnea-Student-Android