Universidad de las Ciencias Informáticas

Facultad 5

Título: Sistema de entrenamiento de Rayos X Aduaneros (SAEX)

Trabajo de diploma para optar por el Título de Ingeniero en Ciencias Informáticas.

Autor:

Eduardo Hernández Anzardo

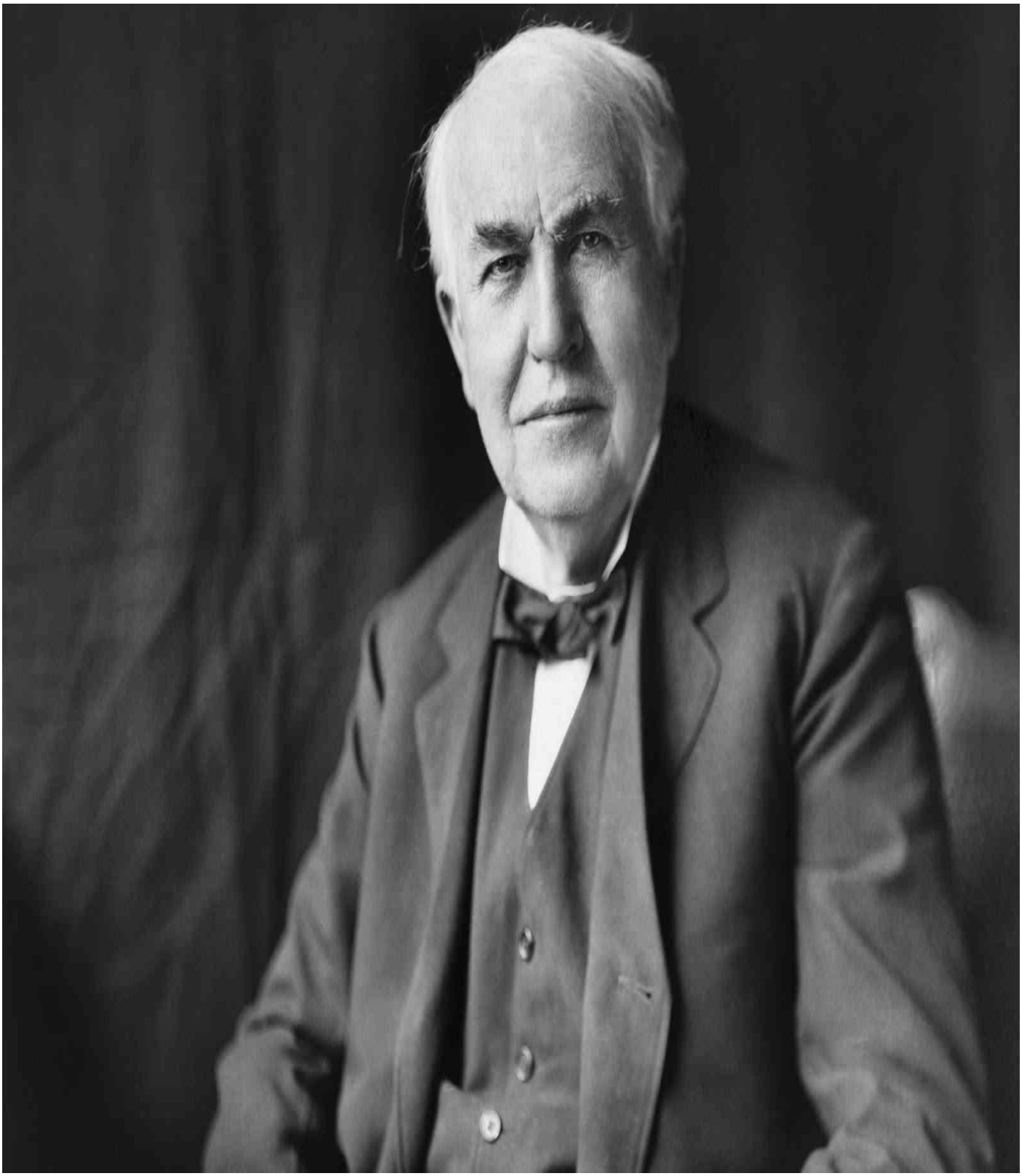
Tutores:

Ing. Juan Miguel Rodríguez Sillero

Ing. Nallía Iris Ramírez Castro

“Año 56 de la revolución”

Julio 2015



"Tu valor consiste en lo que eres y no en lo que tienes."

Thomas Alva Edison

Declaración de autoría

Declaro ser autor único de la presente tesis y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

_____

Eduardo Hernández Anzardo

Autor

_____

Ing. Nallía Iris Ramírez Castro

Tutor

_____

Ing. Juan Miguel Rodríguez Sillero

Tutor

Datos de contacto

Tutor: Ing. Nallía Iris Ramírez Castro.

Edad: 25.

Ciudadanía: cubana.

Institución: Universidad de las Ciencias Informáticas.

Título: Ingeniera en Ciencias Informáticas.

E-mail: niramirez@uci.cu

Graduado en: Universidad de las Ciencias Informáticas.

Tutor: Ing. Juan Miguel Rodríguez Sillero.

Edad: 27.

Ciudadanía: cubano.

Institución: Universidad de las Ciencias Informáticas.

Título: Ingeniero en Ciencias Informáticas.

E-mail: jmsillero@uci.cu

Graduado en: Universidad de las Ciencias Informáticas.

Dedicatoria

A mi familia.

Que creyeron en mí en todo momento, incluso cuando yo no me tenía confianza, por estar ahí siempre que necesité de ellos, a todos los quiero mucho.

A mi padre.

Que más que padre, es un amigo, un hermano, un compañero. Para tí papá un abrazo, un te quiero y un gran beso.

A mi madre.

La mujer más maravillosa del mundo, doy gracias una y mil veces por tener una madre como tú, siempre a mi lado y haciendo hasta lo imposible para apoyarme en todo, logrando que siempre cumpla mi objetivo. Mi ángel guardián, ha llegado el momento de decirte, has pasado 23 años cuidando de mí y dándome lo que he necesitado, ahora es el momento de intercambiar los papeles, te amo mamá.

Agradecimientos

Agradezco a todas aquellas personas que de una forma u otra me apoyaron para que pudiese llegar a ser un profesional.

A mis tutores Nallia y Juan Miguel, que me dieron su apoyo incondicional, estuvieron ahí atentos y siempre dispuestos a ayudarme siempre que lo necesité, a ti juan te diré que más que un amigo, te considero un hermano.

A mis amigos, el antiguo piquete del 87, que nos reorganizaron, algunos fueron para el 88 y otros para el 91, gracias a ustedes por compartir tantos momentos lindos en estos 5 años, nunca olvidaré la forma en que celebrábamos los cumpleaños, que se volvió de una manera más formal desde el año pasado, todo parece indicar que ya nos estamos poniendo viejos, no mencionaré nombres pues me sentiría muy triste si se me quedara alguno por mencionar, a todos ustedes los quiero, siempre podrán contar conmigo para cualquier cosa que necesiten, gracias amigos.

A mi grupo, el cual fue una familia más que me ayudó a ser mejor persona.

A mis vecinos Alí, Silvia, Pochy, que siempre estaban atentos a los resultados de mis exámenes y preguntándome que como me sentía, que todo saldría bien.

A mi familia por apoyarme en todas y cada una de las decisiones que tomé, a mis primo Lenier, Liena, mis abuelos y abuelas, Ricardo, Rodolfo, Nachy, Elena. A mis tíos y tías; Fillo, Chuchita, Callito, Annia, Ualmi, Icha, Eva, Ever, que más que tíos y tías los considero madres y padres. A toda mi familia, gracias.

A mi novia, por recalcarme la frase, "cuando se quiere se puede", gracias por darme consejos los cuales me fueron de gran ayuda, gracias por todo, te quiero mucho.

A mi papá, aunque no pudo venir sé que está orgulloso de mi y que puedo contar contigo para lo que necesite, para ti papá gracias por todo.

Y por último a la mujer que más quiero en este mundo, mi madre, por ser la luz que ilumina mi camino, por darme su apoyo en todo momento, por estar siempre corrigiendo mis faltas y celebrando mis triunfos, una vida no me alcanza para demostrarte todo el amor que siento por ti, gracias mamá.

Resumen

En las terminales aduaneras, los especialistas de rayos X juegan un papel importante en la lucha contra la corrupción. Anteriormente estos no contaban con una formación basada en la práctica sino que se enfrentaban directamente a un escenario real, por lo que fue necesario el desarrollo del Sistema de Entrenamiento de Rayos X (SERX) con el objetivo de minimizar los posibles errores cometidos ante la revisión de bultos o equipajes.

SERX tuvo gran aceptación, pero el producto no pudo ser comercializado con otros clientes (fue desarrollado solo para la Aduana General de la República). Además, presenta soluciones no óptimas, lo que penaliza el rendimiento de la aplicación. Por lo que era necesario un nuevo producto, que incluyera también nuevas funcionalidades y redujera las limitaciones que fueron detectadas en SERX.

Se utiliza la base técnica del centro VERTEX para el desarrollo del sistema, lo que permite brindarle un mejor soporte. Entre las nuevas funcionalidades implementadas, se encuentra la opción de generar un reporte, en forma de documento, para un posterior análisis de los resultados obtenidos por el usuario. Además, se permite añadir o quitar funciones realce y marcajes sin necesidad de cambiar el código fuente de la aplicación. Con este trabajo de diploma se obtiene un nuevo Sistema de Entrenamiento de Rayos X Aduaneros (SAEX, nombre comercial del producto), que además de poderse comercializar, presenta mejor rendimiento y flexibilidad que el anterior.

Palabras claves: aduana, entrenamiento, rayos X, sistema.

Índice de contenidos

Índice de tablas

Índice de figuras

Índice de gráficos

Introducción

En la actualidad, el desarrollo de la informática y las telecomunicaciones ha tenido un crecimiento gradual, extendiéndose a diversas esferas de la vida del hombre. Entre estas esferas se encuentra la lucha contra el delito, donde las fronteras aduaneras utilizan sistemas y equipos con el objetivo de detectar tráficos de armas, drogas, explosivos y otros fraudes.

La Aduana General de la República (AGR) de Cuba lleva a cabo una ardua lucha por evitar estos fraudes, siendo la mayoría detectados por los especialistas de rayos X. Para la capacitación, dichos especialistas reciben una formación teórica y la práctica la realizan en el Sistema de Entrenamiento de Rayos X (SERX), software desarrollado por el centro VERTEX de la Universidad de las Ciencias Informáticas (UCI).

SERX simula el equipo original de rayos X utilizado por las aduanas en las terminales (aéreas, navales y terrestres) y dispone de controles similares que familiarizan al usuario con el software. Para simular los equipajes se utilizan imágenes extraídas de los equipos reales de rayos X, las cuales pueden tener varias maneras de visualizarse, es decir, la imagen en su estado normal que es aquella que no tiene ninguna modificación y la imagen con alguna función realce (filtro) aplicada, las mismas son funciones con el objetivo de resaltar elementos u objetos de interés dentro de la imagen.

Este sistema es capaz de asimilar nuevos conjuntos de imágenes con el objetivo de lograr la variabilidad de los equipajes en los entrenamientos, la incorporación de nuevas imágenes se realiza de manera estática, o sea, se tiene que incluir al sistema la imagen en su estado normal, además una serie de imágenes resultado de haber aplicado varias funciones realce a la misma. SERX cuenta con una colección de 7 funciones realce, por lo que por cada equipaje que se quiera incorporar al sistema es necesario agregarle al mismo un total de 8 imágenes, lo que provoca para la realización de un entrenamiento con un número elevado de imágenes, una carga excesiva de la memoria RAM (*Random-Access Memory)* de la computadora, además de demoras al iniciarlo.

Otras de las funcionalidades de SERX es que provee una serie de marcajes con los que se pueden identificar las infracciones detectadas por el usuario a las imágenes con las que realiza el entrenamiento, para incorporar o retirar dichos marcajes es necesario la modificación del código fuente del sistema, de igual manera sucede con las funciones realce, lo que puede traer como consecuencia introducción de errores al sistema, así como dificultades en las tareas de soporte a dicha aplicación.

El Sistema de Entrenamiento de rayos X no permite exportar un reporte del entrenamiento realizado con el objetivo de un posterior análisis de los resultados obtenidos, además de que es un sistema desarrollado solamente para la AGR, dificultando así su comercialización a nuevos clientes.

Dado el planteamiento anterior se tiene como **problema de investigación:** ¿Cómo reducir las limitaciones de la primera versión del sistema de entrenamiento de rayos X?

Definiéndose como **objetivo general:** Desarrollar un Sistema de Entrenamiento de Rayos X Aduanero.

El **objeto de estudio** son los sistemas de rayos X aduaneros y el **campo de acción** se enmarca en los sistemas de rayos X utilizados en las terminales aéreas, navales y terrestres para la lucha contra el delito.

Con la realización de las siguientes tareas de la investigación, se pretende materializar el objetivo antes planteado:

- Elaboración del marco teórico para conocer acerca de los conceptos fundamentales de la investigación.
- Análisis de la SDK del centro para acoplar correctamente el sistema desarrollado.
- Selección de la metodología y las herramientas a utilizar para el desarrollo del software.
- Diseño de la aplicación.
- Implementación de la solución propuesta.
- Ejecución de pruebas para comprobar el cumplimiento de todas las funcionalidades requeridas por el cliente.

Para el desarrollo de la siguiente investigación se utilizarán los siguientes métodos de investigación:

Analítico-sintético: Permitió recopilar toda la información necesaria para el desarrollo del presente trabajo, además permitió realizar una profunda búsqueda de las tecnologías, herramientas y metodologías posibles a utilizar en el desarrollo de la aplicación.

Entrevista: Método empírico que permite obtener información proveniente de los clientes acerca de interfaz de usuario, y los requisitos no funcionales que estos desean que tenga el sistema de entrenamiento de rayos X.

Consultas de las fuentes de información: Es un método empírico que va a permitir la búsqueda de información existente en el mundo actualmente sobre los sistemas de entrenamientos de rayos X.

El presente trabajo está estructurado de la siguiente manera: resumen, introducción, tres capítulos de contenido, conclusiones, recomendaciones, bibliografía, glosario de términos y anexos. A continuación se muestra un breve resumen del contenido que aborda cada uno de estos capítulos.

Capítulo1. "Fundamentación teórica": en este capítulo se hace referencia a los conceptos relacionados con la investigación, el estado del arte, así como las tendencias, tecnologías y metodologías en las que se apoya para darle solución al problema planteado. Se tratan los elementos teóricos que sustentan y fundamentan los objetivos del trabajo.

Capítulo 2. "Solución Propuesta": aquí se muestran las características de la propuesta de solución mediante una descripción detallada de la misma. Se dan a conocer los requisitos no funcionales y funcionales de la aplicación mediante la realización de las historias de usuario, mostrando además el número de iteraciones necesarias para implementar cada historia de usuario. Además se relacionan las tarjetas CRC para conocer las clases que conforman el sistema y la forma en que las mismas interactúan, el patrón arquitectónico y los patrones de diseño son otros de los aspectos reflejados en este capítulo.

Capítulo 3. "Implementación y Prueba": en este capítulo se realiza la planificación y duración de la implementación de cada historia de usuario mediante la creación de las tareas de ingeniería, se muestra además el conjunto de pruebas realizadas a la aplicación con el objetivo de detectar cualquier error cometido mediante el desarrollo de la misma.

Capítulo #1. Fundamentación teórica

1.1-Tecnología de rayos X y su uso en las aduanas

Rayo es un concepto que tiene su origen en *radĭus*, un vocablo latino. El concepto se usa para nombrar a la línea que nace en el espacio donde se genera una cierta clase de energía y que se prolonga en la misma dirección hacia donde la energía en cuestión se propaga (1).

La noción de rayos X, en este sentido, se refiere a las ondas de tipo electromagnético que son emitidas por los electrones internos de un átomo. Por sus características, los rayos X están en condiciones de atravesar diferentes cuerpos y de lograr una impresión fotográfica. Cuentan con una energía capaz de ionizar los átomos de la materia, algo que permite que sean utilizados con diferentes fines (1).

Actualmente, existen equipos capaces de inspeccionar cuerpos u objetos y mostrar la impresión fotográfica obtenida con rayos X en un monitor o pantalla integrado al equipo, todo en tiempo real. Estos aparatos son muy utilizados en los hospitales y terminales aduaneras por solo mencionar algunos. (Ver Ilustración 1).

Ilustración 1. Equipo de rayos X aduanero.

En las terminales aduaneras estos equipos o aparatos de rayos X juegan un papel fundamental, ya que permiten detectar cualquier anomalía (tráfico de armas, drogas entre otras) dentro de los equipajes o paquetes de los pasajeros. El proceso de inspección de equipajes comienza cuando el equipo escanea el equipaje, mostrando su contenido en una imagen proyectada en el monitor del equipo. Entonces, el especialista revisa detalladamente la imagen en busca de anomalías y en caso de existir alguna reporta la situación a las autoridades aduaneras, sino entrega el equipaje a

la persona y comienza de nuevo el ciclo con el siguiente paquete.

1.2-Proceso de capacitación de los especialistas aduaneros de rayos X en Cuba

Además de la teoría que reciben, los especialistas aduaneros deben entrenarse en la actividad que realizan, pero ¿qué se define como entrenamiento?

El entrenamiento es una actividad que surge para abarcar al efecto de entrenar. Se trata de un procedimiento pensado para obtener conocimientos, habilidades y capacidades ya sean físicas o mentales (2).

Una vez definido el concepto de entrenamiento surge la siguiente interrogante: ¿Por qué es necesario entrenar a los especialistas?

El entrenamiento con el equipo mejora la capacidad de los especialistas aduaneros a la hora de detectar delitos en las inspecciones de equipajes, pero estos equipos son costosos y muy escasos en las aduanas, generalmente para cubrir puntos de afluencia de pasajeros, por lo que no cubren la demanda del personal que se necesita capacitar. Además, los especialistas deben demostrar ciertas habilidades antes de enfrentarse a una inspección de un equipaje real debido a la importancia de este proceso. En respuesta a esta necesidad se utilizan software que simulan los equipos de rayos X, donde solo es necesario la utilización de una *Personal Computer* o PC.

1.3-Entrenadores de rayos X existentes en el mercado

Simfox

Es un simulador de sistemas de inspección por rayos X que se utiliza para formar y evaluar las habilidades de detección en los operadores de equipos de rayos X. Se ha implementado con éxito por los aeropuertos y empresas de transporte en toda Europa con el fin de cumplir con el Reglamento [UE] nº 185/10 (3) que indica la obligatoriedad de que los operadores de equipos de rayos X realicen un entrenamiento de 6 horas en un simulador de sistemas de inspección por rayos X cada 6 meses. Simfox (Ver Ilustración 2) se adapta a las necesidades del cliente y es utilizado por las Aduanas, la Policía, el Ejército, las Prisiones, los Hospitales e Instituciones Gubernamentales de más de 40 países en todo el mundo. El cliente puede acceder a la herramienta mediante Internet o instalarlo en una red local. Se puede adquirir mediante diferentes tipos de licencias desarrolladas para satisfacer las necesidades tanto de organizaciones grandes como de las pequeñas (4).

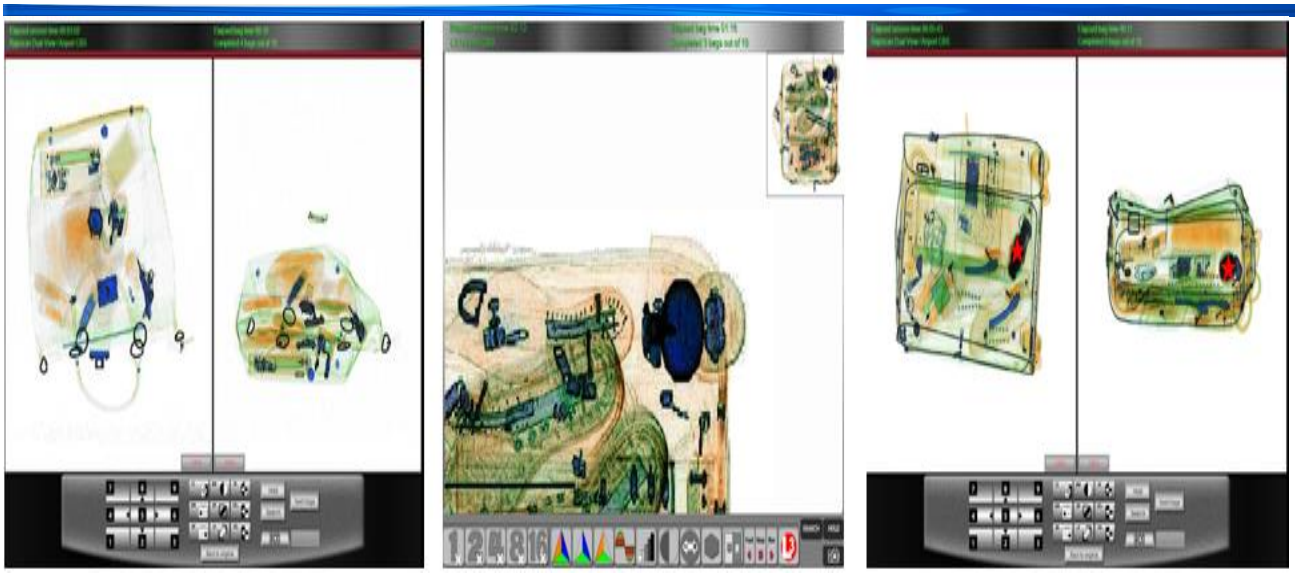


Ilustración 2. SIMFOX.

Este simulador está enfocado a examinar desde bolsas pequeñas hasta grandes *conteiner* o automóviles. Algunas de las ventajas que presenta son:

- Imágenes de rayos X de alta calidad.
- Botones digitales para la manipulación de imágenes.
- Revisión minuciosa del paquete.

  Después de cada sesión con el simulador de rayos X, el usuario es dirigido a una pantalla donde se puede revisar cada bolsa estudiada, aislar cada elemento dentro de ella y verlos en detalle. Esto asegura que los operarios puedan aprender rápida y fácilmente de sus errores sin la ayuda inmediata de un profesor.

- La revisión de la sesión se guarda en el sistema y puede ser revisada posteriormente.

  Se crean y exportan informes rápidamente con el objetivo de ayudar a los profesores a detectar qué operadores no cumplen con las habilidades necesarias.

- Posibilidad de crear los equipajes.

Otra de las cosas que se pueden hacer en este simulador es bolsa para el test, o sea se puede elegir una bolsa vacía e incorporarle a esta cualquiera de los más de 2000 artículos disponibles. Esto proporciona un flujo constante de nuevos contenidos y asegura que los operarios utilicen sus habilidades de detección en lugar de su memoria para detectar amenazas.

Entre sus desventajas se encuentran que es privativo, hay que pagar licencia por su utilización, y la revisión se realiza de manera *online,* es decir, el operario realiza el entrenamiento y su resultado es revisado por un administrador o especialista conectado al mismo tiempo en que el operario realiza el entrenamiento.

SERX

El SERX (Ver Ilustración 3) es una aplicación de escritorio que simula los tradicionales equipos de rayos X y fue desarrollada en la Universidad de las Ciencias Informáticas (UCI). Actualmente está siendo utilizado en todas las aduanas del país con el objetivo de entrenar a los especialistas (5).

Ilustración 3. SERX

Dentro de sus funcionalidades se encuentran:

- Posee una opción de identificación donde el usuario podrá crear/eliminar un perfil que lo identifique cada vez que desee entrenar. Además, en dicho perfil se almacenan sus resultados.
- Cuenta con los controles *play/pause/stop* que permiten reproducir, pausar y detener la transición de equipajes.
- Le permite al usuario aplicar diferentes filtros a los equipajes, lo cual equivale a aplicar las funciones de realce con las que cuenta el equipo de rayos X original.
- Cuenta con la opción de marcar el equipaje según las etiquetas dadas (drogas, armas,

explosivos, tabaco, patrimonio, valores, medios técnicos, duda, contrabando y comercial).

- Tiene la opción *zoom* que da la posibilidad de acercar y alejar la visión de los equipajes.

- Le permite al usuario obtener un resumen de su desempeño a través de un reporte dado por la aplicación donde se visualiza la cantidad de equipajes marcados de bien y mal y un por ciento de efectividad.

- La pantalla principal de la aplicación consta de tres paneles fundamentales: Superior, Intermedio e Inferior.

En el panel superior se encuentran los marcajes de infracción, como: tabaco, drogas, explosivos, etc. En el intermedio es donde se simula la estera por la que pasan las imágenes de los equipajes a las cuales les realizan los marcajes y se le aplican los filtros correspondientes. Y por último en el panel inferior es donde se encuentran los filtros que se aplican a las imágenes de los equipajes (blanco y negro, normal, etc.). También están los controles para iniciar, pausar y parar el entrenamiento (*play*, *pause*, *stop*).

A las limitaciones enunciadas anteriormente, se agrega además que la herramienta no es extensible ya que no permite simular otros equipos de rayos X utilizados en las aduanas, debido a que las funciones realce no son las mismas.

1.4- Metodologías de desarrollo

Las metodologías de desarrollo de software son un conjunto de procedimientos, técnicas y ayudas a la documentación para el desarrollo de productos software. Las mismas van indicando paso a paso todas las actividades a realizar para lograr el producto informático deseado, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla (6).

A continuación se presentan algunas características de estas metodologías.

1.4.1-Tradicional o pesada

Son aquellas con mayor énfasis en la planificación y control del proyecto, en especificación precisa de requisitos y modelado, reciben el apelativo de Metodologías Tradicionales o Pesadas. Estas metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo del software, con el fin de conseguir un software más eficiente. Para ello, se hace énfasis en la

planificación total de todo el trabajo a realizar y una vez que está todo detallado, comienza el ciclo de desarrollo del software. Se centran especialmente en el control del proceso, mediante una rigurosa definición de roles, actividades, artefactos, herramientas y notaciones para el modelado y documentación detallada. Además, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar.

Entre estas metodologías tradicionales se pueden nombrar:

Rational Unified Proccess (RUP): es un conjunto de metodologías adaptables al contexto y necesidades de cada organización. Constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. RUP es un proceso de desarrollo de software que se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).También es una herramienta determinada por ciclos y fases para el proceso del modelado, estas son: Fase de Inicio, Fase de Elaboración, Fase de Construcción y Fase de Transición. Además brinda la facilidad de utilizar UML de forma práctica, como apoyo para realizar muchos procesos que existen para modelar o documentar el sistema de una organización.

Microsoft Solutions Framework (MSF): es una metodología desarrollada por *Microsoft Consulting Services* que define un marco de trabajo de referencia para construir e implantar sistemas empresariales distribuidos basados en herramientas y tecnologías de Microsoft para cualquier plataforma (Linux, Citrix, Microsoft, Unix). MSF provee un conjunto de principios, modelos, disciplinas, conceptos y lineamientos para la entrega de tecnología de la información (TI) utilizando soluciones Microsoft. MSF no se limita sólo al desarrollo de aplicaciones, también es aplicable a otros proyectos de TI como, proyectos de implementación de redes o infraestructura, no obliga al desarrollador a utilizar una determinada metodología, pero les permite decidir qué método utilizar.

1.4.2-Ágiles

Un modelo de desarrollo ágil, generalmente es un proceso incremental, (entregas pequeñas con ciclos rápidos), también cooperativo (clientes y desarrolladores trabajan constantemente con una comunicación muy estrecha y constante), sencillo (el método es fácil de aprender y modificar para

el equipo, es bien documentado por medio de libros o la Web) y finalmente adaptativo (capaz de permitir cambios de último momento). Las metodologías ágiles proporcionan una serie de pautas y principios junto a técnicas pragmáticas que puede que no curen todos los males pero harán la entrega del proyecto menos complicada y más satisfactoria tanto para los clientes como para los equipos de entrega.

A continuación se muestran algunas de estas metodologías más representativas con sus respectivas características.

SCRUM: desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada *sprint* es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto. Éstas son las verdaderas protagonistas, especialmente la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración.

***eXtreme Programing* (XP)**: la programación extrema o XP es una metodología de desarrollo de la ingeniería de software formulada por Kent Beck, autor del primer libro sobre la materia, *Extreme Programming Explained: Embrace Change* (1999). Es el más destacado de los procesos ágiles de desarrollo de software. Al igual que éstos, la programación extrema se diferencia de las metodologías tradicionales principalmente en que pone más énfasis en la adaptabilidad que en la previsibilidad. Los defensores de la XP consideran que los cambios de requisitos sobre la marcha son un aspecto natural, inevitable e incluso deseable del desarrollo de proyectos. Creen que ser capaz de adaptarse a los cambios de requisitos en cualquier punto de la vida del proyecto es una aproximación mejor y más realista que intentar definir todos los requisitos al comienzo del proyecto e invertir esfuerzos después en controlar los cambios en los requisitos.

1.4.3-Valoración de la selección de la metodología

Después de haber hecho un análisis entre las características de las metodologías tradicionales y ágiles, se selecciona una metodología ágil debido a que se corresponde con el proceso de desarrollo de software a seguir, con el equipo de desarrollo, en cuanto a organización y metas a seguir, con la propuesta de un proceso adaptativo para garantizar la retroalimentación de la información y de los requisitos.

Se selecciona XP como metodología de desarrollo teniendo en cuenta algunas características como:

- Está dirigida a equipos pequeños o medianos donde el entorno físico debe ser un ambiente que permita la comunicación y colaboración entre todos durante todo el tiempo, al estar conformado el equipo por un integrante permite cumplir con dichos elementos.
- El cliente está vinculado al proyecto permitiendo así el intercambio continuo con el desarrollador, con el objetivo de llegar a acuerdos para una mejor implementación de dicho sistema.
- Ofrece una solución factible para proyectos con requisitos muy cambiantes, lo que se manifiesta en el desarrollo de la aplicación pues los requisitos pueden cambiar en dependencia de las necesidades del cliente.

1.5-Herramientas CASE y lenguaje de modelado

Las herramientas para el modelado son aplicaciones informáticas usadas para aumentar la productividad en el desarrollo de software, reduciendo considerablemente el tiempo. Estas aumentan la calidad del software; mejoran la planificación del proyecto; apoyan la reutilización, portabilidad y estandarización de la documentación del software; facilitan a su vez el uso de las distintas metodologías de desarrollo de software; permiten modelar un software en su totalidad antes de codificarlo, mediante diagramas que permiten modelar cada etapa del ciclo de vida del software (7).

Visual Paradigm (8.0) es una herramienta CASE (*computer aided software engineering*, ingeniería de software asistida por computadora), que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. Ayuda a una más rápida construcción de aplicaciones de calidad, con un menor costo. Permite construir todos los tipos de diagramas necesarios para la documentación del software, así como importar proyectos realizados en *Rational Rose*. Además, presenta un diseño enfocado al negocio que genera un software de calidad. Permite transformar diagramas Entidad-Relación en esquemas de base de datos y viceversa. Permite realizar ingeniería directa e inversa, obtener el código a partir de diagramas, así como diagramas a partir de un código previamente escrito (8).

1.5.1-Lenguaje de modelado

Lenguaje Unificado de Modelado (UML, por sus siglas en inglés, *Unified Modeling Language*): es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está

respaldado por el OMG (*Object Management Group*). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema. UML ofrece un estándar para describir un "plano" del sistema (modelo), incluyendo aspectos conceptuales tales como procesos de negocio, funciones del sistema, y aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y compuestos reciclados.

Dentro de sus principales beneficios de UML están (9):

- Modelar sistemas (y no sólo de software) utilizando conceptos orientados a objetos.
- Establecer conceptos y artefactos ejecutables.
- Encaminar el desarrollo del escalamiento en sistemas complejos de misión crítica.
- Crear un lenguaje de modelado utilizado tanto por humanos como por máquinas.
- Mejor soporte a la planeación y al control de proyectos.

1.6-Herramientas para el desarrollo y lenguaje de programación

Qt Creator es un IDE (Entorno de Desarrollo Integrado) multiplataforma que se ajusta a las necesidades de los desarrolladores. Se centra en proporcionar características que ayudan a los nuevos usuarios a aprender y comenzar a desarrollar rápidamente, también aumenta la productividad de los desarrolladores con experiencia en Qt (10).

A continuación se muestran algunas de sus características:

- Editor de código con soporte para C++, QML y ECMAscript.
- Herramientas para la rápida navegación del código.
- Resaltado de sintaxis y auto-completado de código.
- Control estático de código y estilo a medida que se escribe.
- Soporte para *refactoring* de código.
- Ayuda sensitiva al contexto.
- Paréntesis coincidentes y modos de selección.

1.6.1-Lenguaje de programación

El lenguaje de programación C++ hace uso eficiente del paradigma de Programación Orientada a Objetos. Permite un excelente control de la memoria y una buena administración de los recursos de la computadora (11). Dentro de las principales ventajas que presenta el lenguaje C++ se encuentran:

Difusión: al ser uno de los lenguajes más empleados en la actualidad, posee gran número de usuarios y tiene una excelente bibliografía.

Versatilidad: C++ es un lenguaje de propósito general, se puede emplear para resolver cualquier tipo de problema.

Portabilidad: se encuentra estandarizado, por tanto, el mismo código fuente puede ser compilado en diferentes plataformas.

Eficiencia: C++ es uno de los lenguajes más rápidos en tiempo de ejecución.

Herramientas: existe gran cantidad de compiladores, depuradores y bibliotecas de clases basadas en este lenguaje.

1.7- Base técnica del centro VERTEX

La base técnica del centro VERTEX o como también se nombra VirtualLabSDK es una arquitectura que combina el estilo basado en capas con el estilo basado en componentes, la combinación de los mismos permite un amplio nivel de reutilización y facilidad de desarrollo. Combina los procesos más comunes de todos los laboratorios virtuales en componentes. Así, es posible ensamblar un cierto número de estos y lograr la funcionalidad básica de un laboratorio virtual a la vez que se reducen los costos de desarrollo. La misma posibilita la creación de módulos con lógicas independientes, provee además un conjunto de componentes los cuales pueden ser reutilizados para el desarrollo de aplicaciones (12).

Entre los componentes que brinda el VirtualLabSDK se encuentran:

PanelTools o panel de herramientas, el cual es un panel que puede contener varios elementos organizados por categorías para ser utilizados por la aplicación que se vaya a desarrollar. Posee un filtro que permite buscar dentro de sus elementos algunos en específico con una mayor facilidad, además puede ser acoplado en ambos laterales de la aplicación.

DigitalClock es otro de los componentes de esta arquitectura el cual puede ser utilizado como reloj o cronómetro en dependencia de como se quiera utilizar en el desarrollo de una determinada aplicación.

ReportPDF posibilita la realización de un reporte en un documento externo a la aplicación con el objetivo de guardar datos de interés para posteriores análisis de dichos datos.

ReadEsceneForInformation, este componente se utiliza como su nombre lo indica para leer información de ficheros xml (*eXtensible Markup Language*) con una estructura determinada por el VirtualLabSDK.

Conclusiones parciales

En este capítulo se dieron a conocer los conceptos fundamentales que van a ser tratados durante el desarrollo del trabajo de diploma para un mejor entendimiento de este. Con la realización del estudio en el mercado de los simuladores existentes se tomaron las buenas prácticas como por ejemplo la distribución de los controles dentro de la GUI, así como la utilización de máquinas de estado para controlar el flujo básico del sistema desde que inicia hasta que el usuario finaliza el entrenamiento y se tuvieron en cuenta algunas funcionalidades como la de exportar el resumen del entrenamiento realizado en un documento. Teniendo en cuenta además las características del equipo de desarrollo y luego del estudio de las metodologías ágiles y tradicionales, se selecciona la metodología XP para guiar el proceso de desarrollo del presente trabajo.

Capítulo #2. Solución propuesta

Esta investigación tiene como propósito general, lograr la creación de una aplicación que mejore las limitaciones detectadas en SERX, este nuevo Sistema de Entrenamiento de rayos X aduanero recibe el nombre de SAEX, el cual fue definido por el centro VERTEX. SAEX sirve como medio de enseñanza a los operarios de rayos X en las terminales aduaneras, con el fin de entrenarlos o evaluarlos, antes de que se enfrenten a un escenario real. La incorporación de este a la base técnica propició la utilización de los componentes DigitalClock (para tener un control de la duración del entrenamiento) y el ReportPDF (para la realización de un reporte fuera de la aplicación en un documento de determinado entrenamiento, para un posterior análisis de los resultados obtenidos). Además de esto se diseñó una interfaz de usuario lo más acorde posible con los equipos reales utilizados en nuestras fronteras aduaneras, teniendo en cuenta elementos para diseños de interfaz gráfica de usuarios, tales como **Ley de Fitt** y el **uso de metáforas** para los controles, los cuales fueron organizados según el flujo lógico de las acciones del usuario.

Ley de Fitt: tanto en las UI (Interfaces de Usuario) para aplicaciones Web, como para aplicaciones de escritorio. El tiempo en que un usuario se demora en alcanzar el objetivo está en función de la distancia que tiene que recorrer para alcanzarlo y el tamaño de dicho objetivo. Por lo que las opciones importantes en una UI deben establecerse utilizando objetos grandes (13).

Uso de metáforas: las metáforas son figuras que mentalmente son muy fáciles de recordar. Una UI puede contener objetos que le recuerden al usuario de una manera visual, con sonido u otra característica fácil de percibir por el usuario algún elemento conceptual (13).

2.1-Descripción de la propuesta de solución

En la descripción de la propuesta de solución se tomará al **especialista de rayos X aduanero** como **usuario**.

Se propone la siguiente solución la cual sigue el siguiente flujo (Ver Ilustración 4).

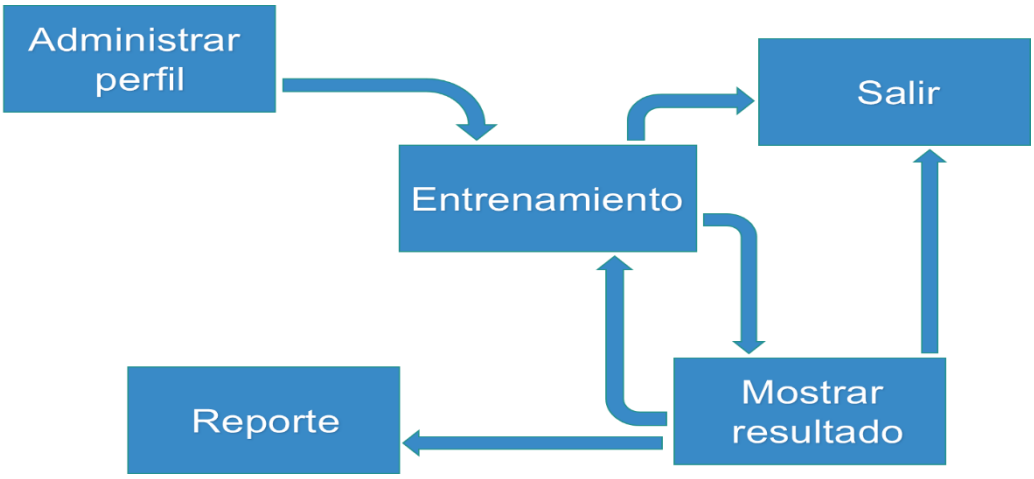


Ilustración 4. Flujo Básico de la solución propuesta.

Primeramente se tiene lo relacionado con los datos del perfil (eliminar, crear o seleccionar perfil) (Ver Ilustración 5 e Ilustración 6).

Ilustración 5. Administración del perfil (seleccionar, eliminar).

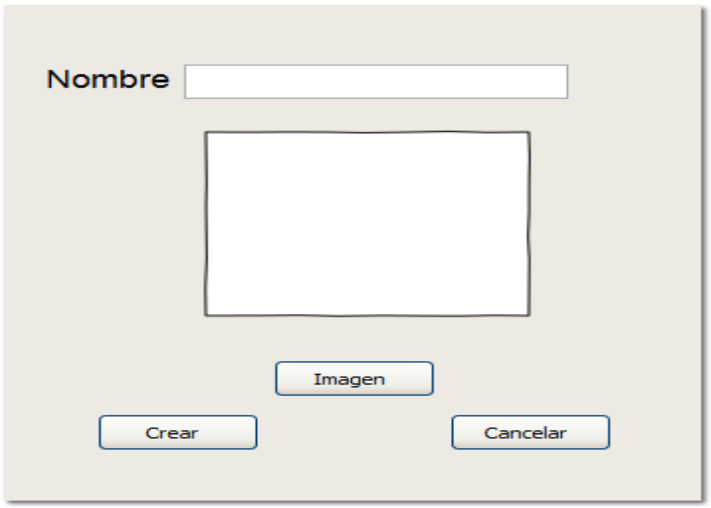


Ilustración 6. Administración del perfil (crear).

En estas pantallas los usuarios tendrán la posibilidad de introducir sus datos para el posterior entrenamiento o de seleccionar un perfil en caso de que ya se tenga creado. Una vez creado el perfil se procede a la realización del entrenamiento dirigiéndose así a la pantalla principal (Ver Ilustración 7).

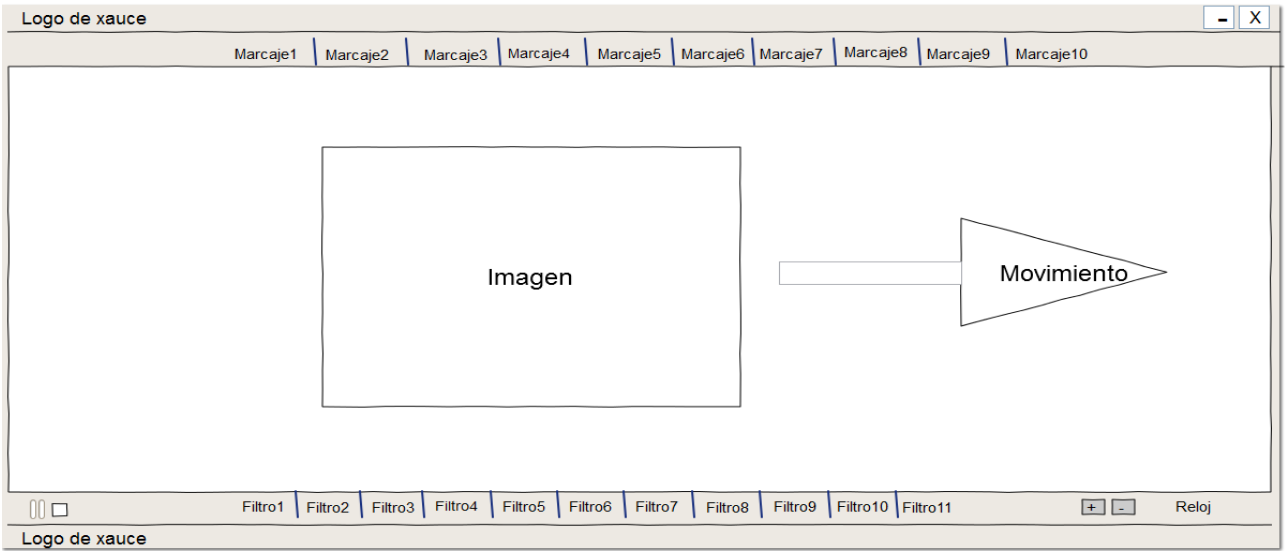


Ilustración 7. Pantalla principal de SAEX.

Esta pantalla es la encargada de cargar todos los filtros, marcajes, y de ubicar las funcionalidades de inicio, pausa y parada del entrenamiento, así como la funcionalidad de agrandar o disminuir las imágenes y la del reloj que mostrará el tiempo de duración del entrenamiento. En esta pantalla el técnico rayos X aduanero tiene la posibilidad de detectar la infracción que considere que existe en las imágenes que pasan por la pantalla, al igual que la posibilidad de aplicarle el filtro deseado a dichas imágenes, seleccionando la imagen con un clic y seleccionando el filtro o marcaje de esta

imagen de la misma forma (dándole clic encima). Luego de terminado el entrenamiento ya sea por decisión de usuario o porque se mostraron todas la imágenes que se tenían, se pasa a la siguiente pantalla (Ver Ilustración 8),

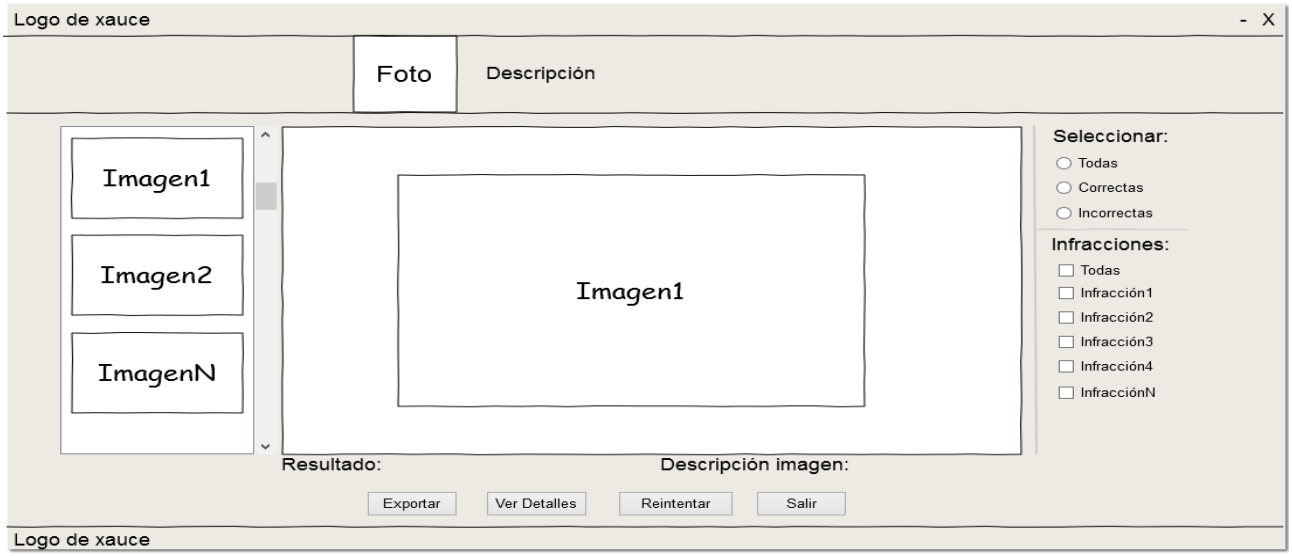


Ilustración 8. Resumen del entrenamiento realizado.

Aquí se muestra el resultado del entrenamiento antes realizado, mostrándose detalles como foto del perfil, cantidad de imágenes, fecha, un panel en la parte derecha en el cual el usuario tendrá la opción de ver todas las imágenes o solo ver las correctas o las incorrectas, al igual que podrá seleccionar si quiere ver todas las infracciones o elegir cuál de ellas desea ver .Ya mostrado el resultado , el usuario tendrá la posibilidad de realizar un nuevo entrenamiento, ver los detalles de los entrenamientos antes realizados(Ver Ilustración 9) por él, cerrar la aplicación o la posibilidad de realizar un reporte en formato pdf el cual contiene las imágenes mostradas en el entrenamiento y una breve descripción de las mismas.

Ilustración 9. Resultados históricos de los entrenamientos realizados.

2.2-Levantamiento de requisitos

Los requisitos de software son las capacidades o condiciones que debe tener un sistema y están encaminados a cumplir con las exigencias que quedan plasmadas en un documento formal o contrato. A continuación se relacionan los requisitos funcionales y no funcionales de SAEX.

2.2.1-Requisitos funcionales

Los requerimientos funcionales de un sistema describen lo que el sistema debe hacer. Estos requerimientos dependen del tipo de software que se desarrolle, de los posibles usuarios del software y del enfoque general tomado por la organización al redactar requerimientos. Cuando se expresan como requerimientos del usuario, habitualmente se describen de una forma bastante abstracta. Sin embargo los requerimientos funcionales del sistema describen con detalle la función de éste, sus entradas y salidas, excepciones, etcétera (14).

Los requisitos funcionales de la aplicación se ven reflejados en la descripción de las **HU (historias de usuario).**

2.2.2-Requisitos no funcionales

Los requisitos no funcionales son las cualidades que debe tener el producto. Debe pensarse en estas propiedades como las características que hacen al producto usable y rápido. Normalmente están vinculados a requisitos funcionales, es decir una vez que se conoce lo que el sistema debe hacer es posible determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser (15).

Dentro de los requisitos no funcionales se encuentran algunos como los de software, donde debe mencionarse el software que se debe disponer después de implementado el sistema; los de hardware son otros de los requerimientos, en estos de debe enunciar los elementos de hardware que se necesitan para que el software cumpla sus funcionalidades.

A continuación se muestran los requisitos no funcionales de la aplicación.

Requerimientos Software:

- El sistema debe mostrarse en forma de aplicación de escritorio.
- Sistema operativo Ubuntu 14.04 o superior.

Requerimientos Hardware:

El sistema necesitará como requisitos de hardware mínimo.

- 512 GB de RAM.
- 40 GB de disco duro.

2.3-Exploración

La metodología XP, comienza en su ciclo de vida con la fase de exploración, proponiendo definir durante esta etapa el alcance general del proyecto; además, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo se realiza una familiarización con las herramientas, tecnologías y prácticas que se emplearán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. Las estimaciones realizadas en esta fase son primarias, debido a que están basadas en datos de alto nivel y podrían variar cuando se analicen con mayor detalle en cada iteración. Para esta fase se sugiere una extensión de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

2.3.1-Historias de usuario

Las Historias de Usuario (HU) son una técnica para sustituir a los documentos de especificación funcional y a los casos de uso. Estas HU son escritas por el cliente en su propio lenguaje como descripciones cortas de lo que el sistema debe realizar. El tratamiento de las HU es muy dinámico y flexible, permite que en cualquier momento se puedan romper, reemplazar por otras más

específicas o generales, añadirse nuevas o ser modificadas. Para ser implementadas las HU, el cliente y los desarrolladores se reúnen para detallar las funcionalidades de cada una. El tiempo de desarrollo ideal para una HU varía entre 1 y 3 semanas (16).

2.3.2-Descripción de los requisitos funcionales

A continuación se describirán los requisitos funcionales, lo cuales cuentan con características como, número, usuario, registro de desarrollo, observaciones entre otros.

Historia de usuario	
Número: 1	**Nombre de la HU:** Administrar perfil
Fecha: 18/3/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Media
Puntos estimados: 0.8	**Iteración asignada:** 2
Programador responsable: Eduardo Hernández Anzardo	
Descripción: El usuario puede crear, seleccionar o eliminar un perfil.	
Prototipo: 	

Tabla 1. HU#1. Administrar perfil.

Historia de usuario	
Número: 2	**Nombre de la HU:** Salir
Fecha: 18/3/2015	**Usuario:** Especialista

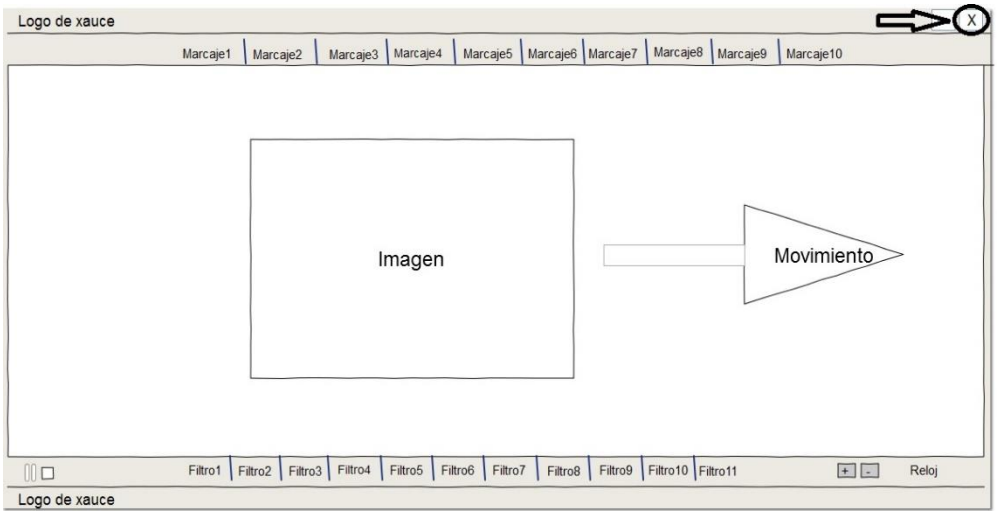
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.4	**Iteración asignada:** 2
Programador responsable: Eduardo Hernández Anzardo	
Descripción: Permite la salida de la aplicación o del entrenamiento.	
Prototipo: 	

Tabla 2. HU#2. Salir.

Historia de usuario	
Número: 3	**Nombre de la HU:** Entrenar
Fecha: 18/3/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1.4	**Iteración asignada:** 1
Programador responsable: Eduardo Hernández Anzardo	
Descripción: Permite iniciar entrenamiento	
Prototipo:	

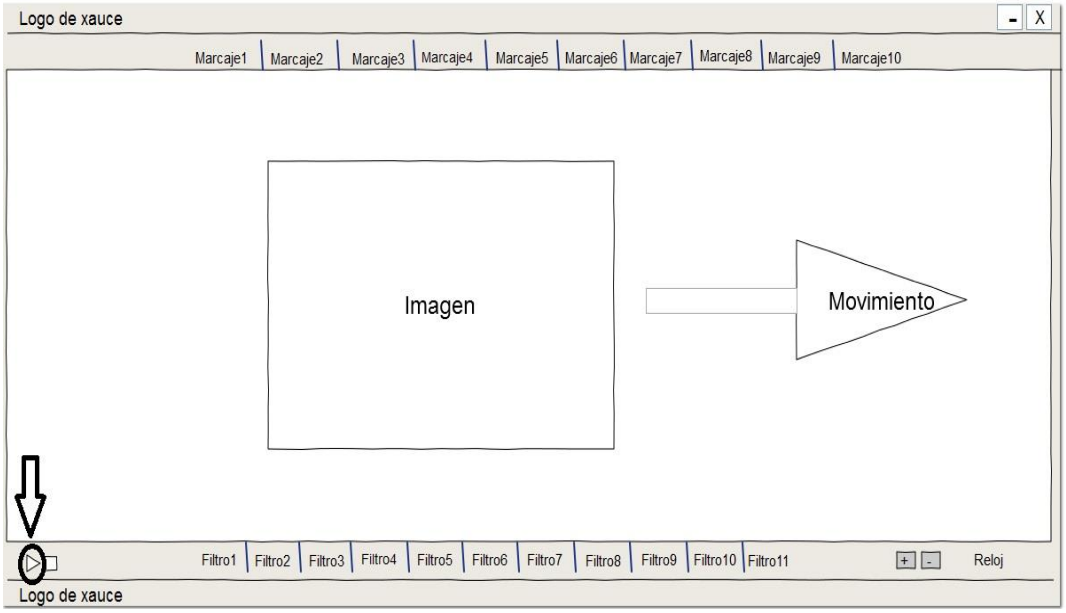


Tabla 3. HU#3. Entrenar.

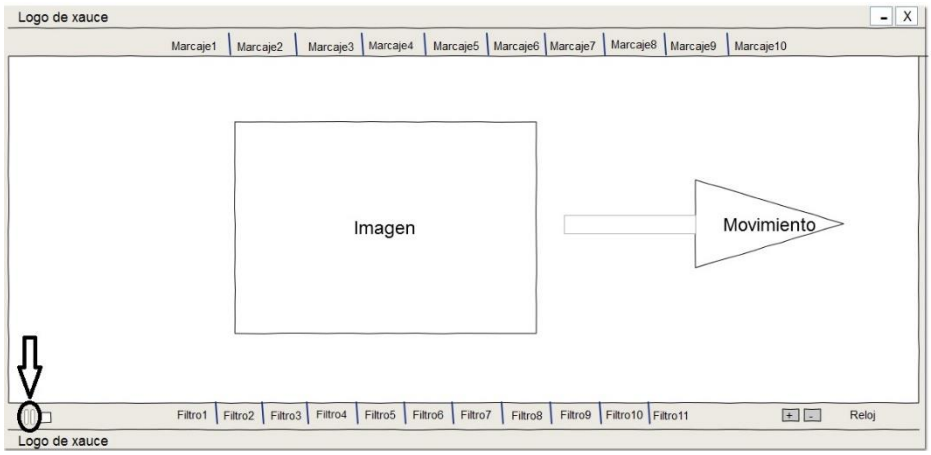
Historia de usuario	
Número: 4	**Nombre de la HU:** Pausar
Fecha: 18/3/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.6	**Iteración asignada:** 1
Programador responsable: Eduardo Hernández Anzardo	
Descripción: Permite pausar el entrenamiento (por 15 segundos).	
Prototipo: 	

Tabla 4. HU#4. Pausar.

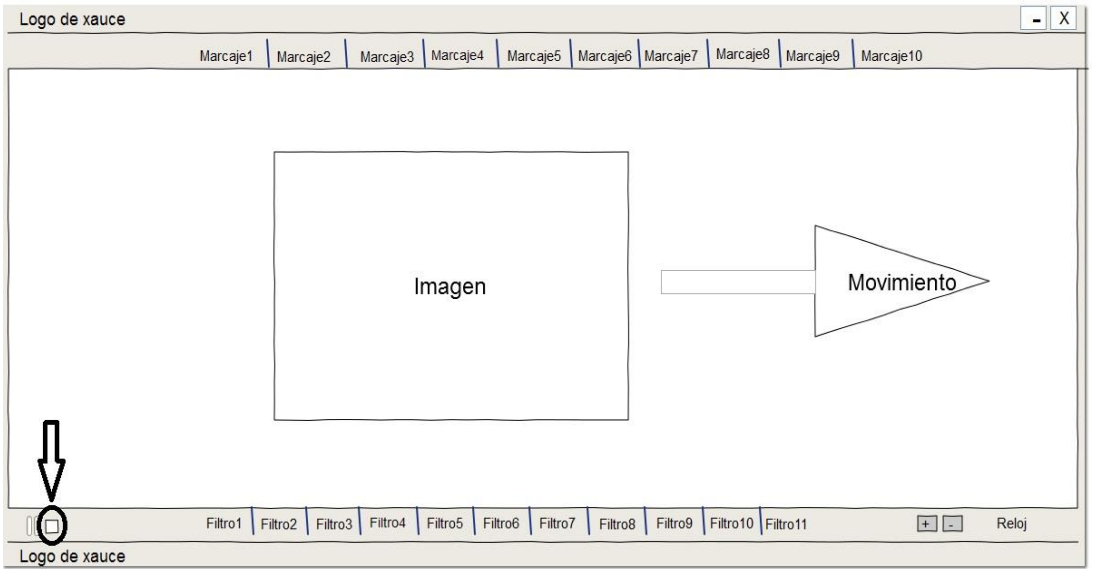
Historia de usuario	
Número: 5	**Nombre de la HU:** Parar
Fecha: 18/3/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.6	**Iteración asignada:** 1
Programador responsable: Eduardo Hernández Anzardo	
Descripción: Permite parar el entrenamiento.	
Prototipo: 	

Tabla 5. HU#5. Parar.

Historia de usuario	
Número: 6	**Nombre de la HU:** Mostrar Imágenes
Fecha: 18/3/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1.2	**Iteración asignada:** 1
Programador responsable: Eduardo Hernández Anzardo	

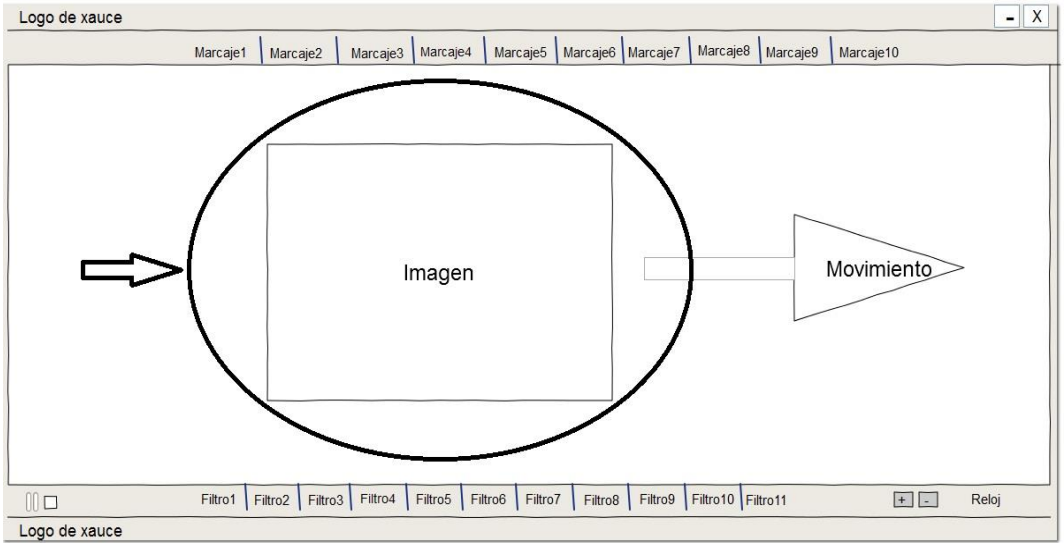
Descripción: Muestra las imágenes a las cuales se les aplican lo filtros y se les realiza el marcaje.
Prototipo: 

Tabla 6. HU#6. Mostrar imágenes.

Historia de usuario	
Número: 7	**Nombre de la HU:** Aplicar marcajes a las imágenes
Fecha: 18/3/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 2
Programador responsable: Eduardo Hernández Anzardo	
Descripción: Permite etiquetar una imagen con las siguientes infracciones (duda, tabaco, armas, patrimonio, droga)	
Prototipo:	

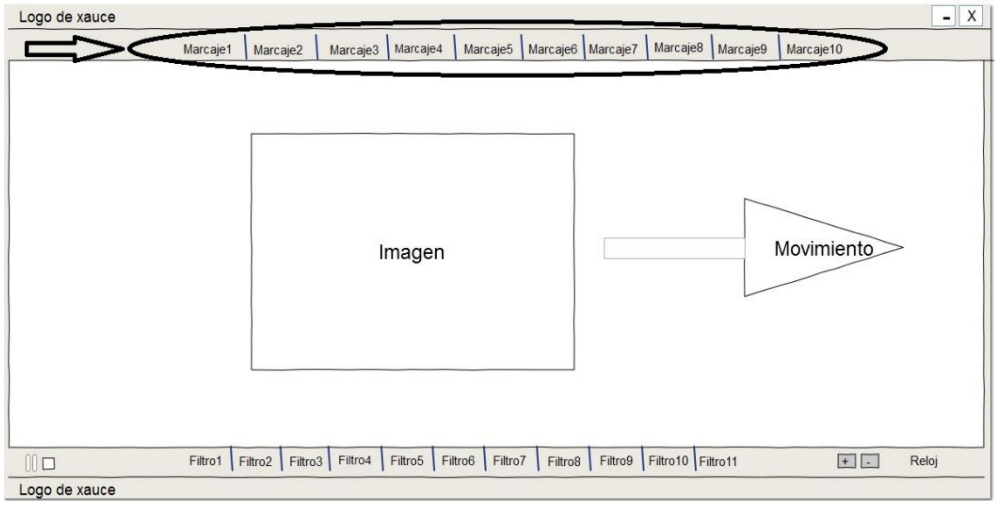


Tabla 7. HU#7. Aplicar marcajes a las imágenes.

Historia de usuario	
Número: 8	**Nombre de la HU:** Aplicar filtros a las imágenes
Fecha: 18/3/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.8	**Iteración asignada:** 3
Programador responsable: Eduardo Hernández Anzardo	
Descripción: Permite aplicarle filtros a las imágenes en movimiento que pasan por la estera.	
Prototipo:	

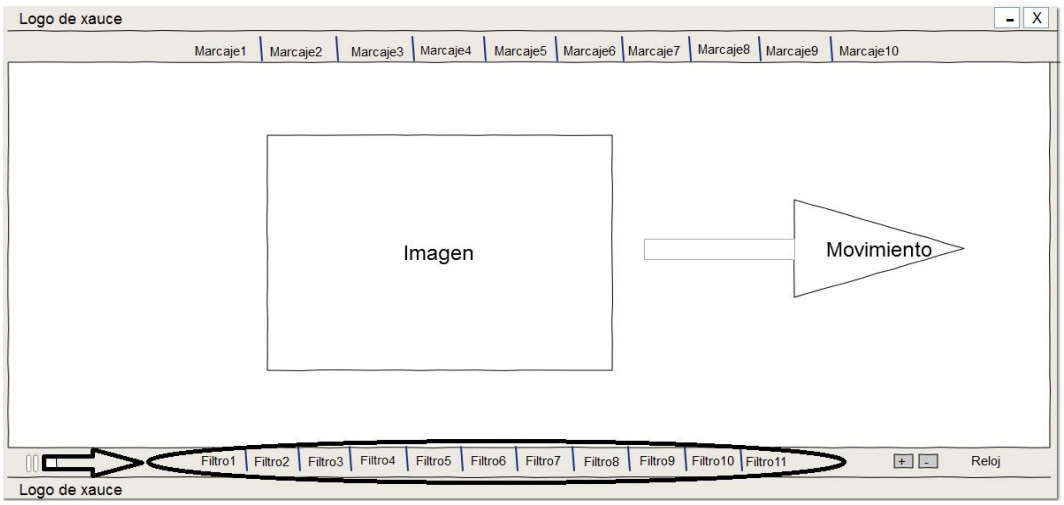


Tabla 8. HU#8. Aplicar filtros a las imágenes.

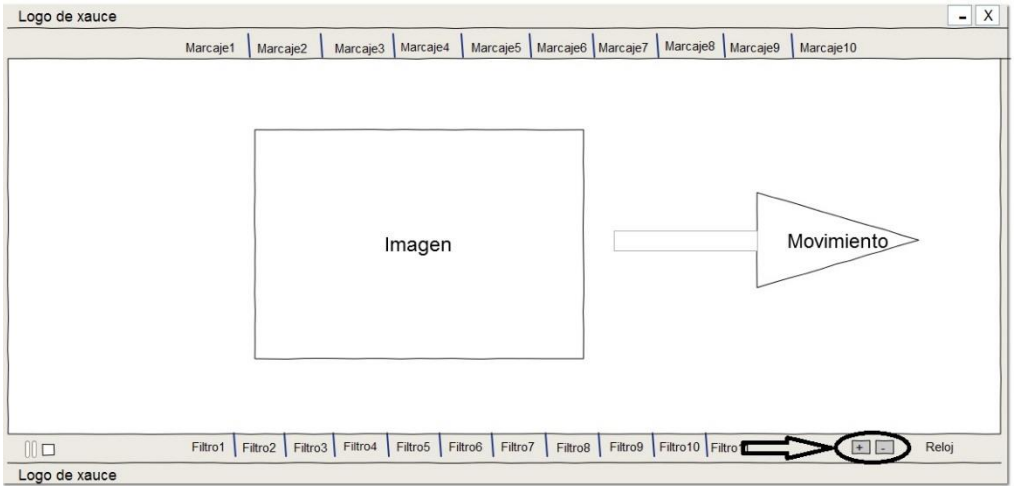
Historia de usuario	
Número: 9	**Nombre de la HU:** Realizar *zoom*
Fecha: 18/3/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.4	**Iteración asignada:** 3
Programador responsable: Eduardo Hernández Anzardo	
Descripción: Permite agrandar o disminuir el tamaño de las imágenes en dependencia de botón *zoom* presionado (*zoomIn*, *zoomOut*).	
Prototipo: 	

Tabla 9. HU#9. Realizar *Zoom.*

Historia de usuario	
Número: 10	**Nombre de la HU:** Ver detalles del entrenamiento
Fecha: 18/3/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 0.8	**Iteración asignada:** 3
Programador responsable: Eduardo Hernández Anzardo	
Descripción: Permite ver el resumen del entrenamiento mostrando los datos: nombre, total de	

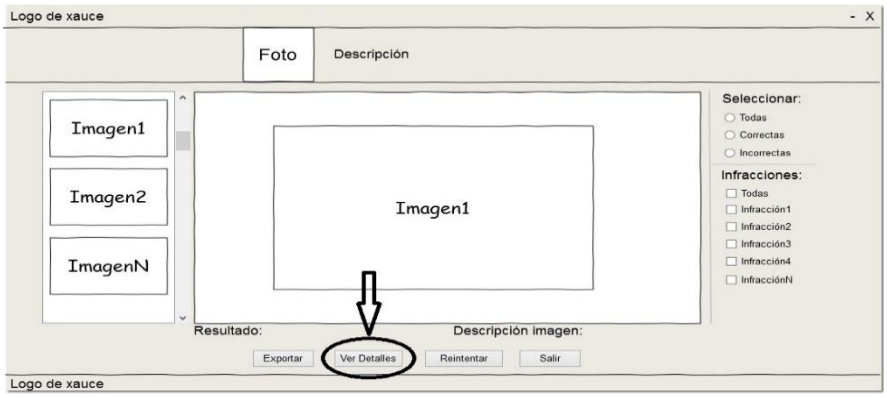
imágenes (correctas e incorrectas), efectividad, duración, fecha y hora.
Prototipo: 

Tabla 10. HU#10. Ver detalles del entrenamiento.

Historia de usuario	
Número: 11	**Nombre de la HU:** Reintentar
Fecha: 18/3/2015	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: 1	**Iteración asignada:** 3
Programador responsable: Eduardo Hernández Anzardo	
Descripción: Permite realizar un nuevo entrenamiento.	
Prototipo: 	

Tabla 11. HU#11. Reintentar.

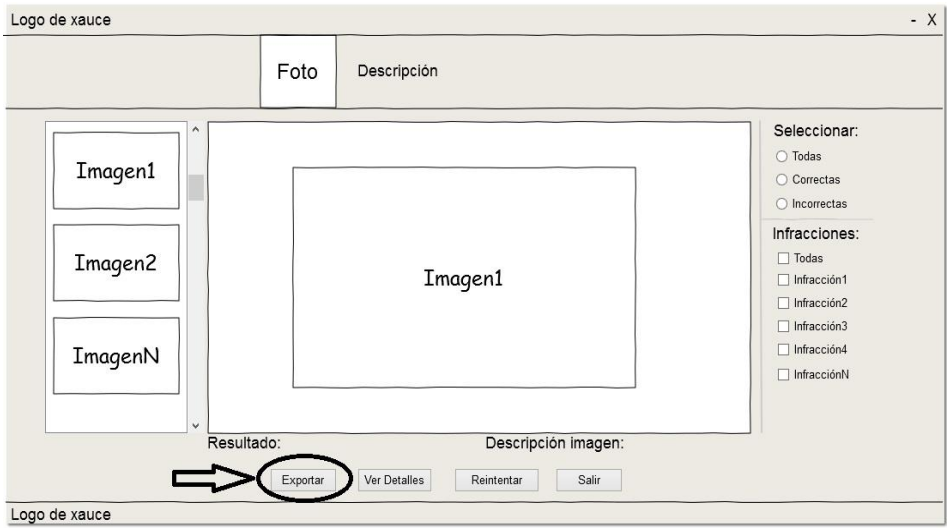
Historia de usuario	
Número: 12	**Nombre de la HU:** Exportar
Fecha: definir	**Usuario:** Especialista
Prioridad en negocio: Alta	**Riesgo en desarrollo:** Alta
Puntos estimados: definir	**Iteración asignada:** definir
Programador responsable: Eduardo Hernández Anzardo	
Descripción: Permite exportar el resultado del entrenamiento realizado en un documento fuera de la aplicación para posteriores análisis de los resultados obtenidos.	
Prototipo: 	

Tabla 12. HU#12. Exportar.

2.4- Planificación

La planificación es una fase donde el cliente establece la prioridad de cada HU y los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Típicamente esta fase consiste en una o varias reuniones grupales de planificación y el resultado es un plan de entregas. Una vez terminado esto, se procede a organizarlas en las iteraciones correspondientes, teniéndose en cuenta la prioridad especificada por el cliente y del tiempo de desarrollo de cada una. Las estimaciones de esfuerzo asociadas a la implementación de las historias la establecen los programadores utilizando como medida el punto, lo que equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. La planificación se puede

realizar basándose en el tiempo o el alcance.

2.4.1-Plan de estimación

El Plan de estimación decide en qué iteración será implementada una historia de usuario, teniendo en cuenta que se implementan primero las funcionalidades indispensables para el desarrollo de la aplicación.

Se muestra a continuación una tabla de la planificación de las historias de usuario quedando reflejada en que iteración serán implementadas según su prioridad.

No	Historia de Usuario	Prioridad	Esfuerzo Estimado
1	Administrar perfil	Alta	0.8
2	Salir	Alta	0.4
3	Entrenar	Alta	1.4
4	Pausar	Alta	0.6
5	Parar	Alta	0.6
6	Mostrar Imágenes	Alta	1.0
7	Aplicarle marcajes a las imágenes	Alta	0.8
8	Aplicarle filtros a las imágenes	Alta	0.8
9	Realizar zoom	Alta	0.4
10	Ver detalles del entrenamiento	Alta	0.8
11	Reintentar	Alta	1.0
12	Exportar	Alta	0.4

Tabla 13. Planificación de las historias de usuario.

2.4.2-Plan de iteraciones

La metodología XP se basa en iteraciones relativamente cortas. Aquí las HU son divididas en tareas que son asignadas a los programadores, creándose así el plan de iteraciones donde se especifica la duración de cada una y las HU que se implementarán en cada iteración.

Para el desarrollo de SAEX se decide realizar 3 iteraciones, las cuales se describen a continuación.

Iteraciones	Orden de las HU a implementar

1	La primera iteración del desarrollo del sistema tendrá como objetivo la implementación de algunas de las funciones básicas para la puesta en marcha de la aplicación. Para esto se tendrán en cuenta las HU 2 ,3 ,4 y 5. Al finalizar esta iteración se habrán podido cumplir las funcionalidades de mostrar imágenes en movimiento, ponerlas en pausa, parar el entrenamiento y salir de la aplicación.
2	En la segunda iteración del desarrollo del desarrollo del sistema se implementaran las HU 1 ,6 ,7 y 12. Al finalizar esta iteración el sistema será capaz de realizar las funcionalidades de crear perfil, eliminarlo o seleccionarlo, así como ver las imágenes a las cuales se les realizara el marcaje y de generar un reporte fuera de la aplicación del entrenamiento realizado.
3	La tercera iteración del desarrollo tiene como objetivo implementar las historias de usuario 8, 9, 10, 11. El usuario puede una vez terminada esta iteración aplicar diferentes filtros a las imágenes, agrandar o disminuir dichas imágenes, realizar un nuevo entrenamiento y ver los detalles del entrenamiento realizado o de entrenamientos anteriores. Al finalizar esta iteración ya se tiene la versión 2 del sistema de entrenamiento de rayos X, el cual se somete a un conjunto de pruebas para evaluar su desempeño.

Tabla 14. Plan de iteraciones.

2.4.3-Plan de entrega

Luego de una planificación se genera un plan de entrega, el cual genera las fechas de entrega de

cada una de las iteraciones planificadas.

Historia de Usuario	Final Iteración 1 (5 de febrero)	Final Iteración 2 (20 de marzo)	Final Iteración 3 (5 de abril)
Salir	V1.0		
Entrenar	V1.0		
Pausar	V1.0		
Parar	V1.0		
Administrar perfil		V1.1	
Mostrar imágenes		V1.1	
Aplicar marcajes a las imágenes		V1.1	
Exportar		V1.1	
Aplicar filtros a las imágenes			V1.2
Realizar *Zoom*			V1.2
Ver detalles del entrenamiento			V1.2
Reintentar			V1.2

Tabla 15. Plan de entrega.

2.5-Arquitectura y diseño

2.5.1-Tarjetas CRC

Fueron propuestas por Ward Cunningham and Kend Beck. Estas involucran las Clases, Responsabilidades de esa clase y los Colaboradores. El objetivo de las mismas es hacer, mediante tarjetas, un inventario de las clases que se van a necesitar para implementar el sistema y la forma en que van a interactuar, de esta forma se pretende facilitar el análisis y discusión de las mismas por parte de varios actores del equipo de proyecto con el objetivo de que el diseño sea

lo más simple posible verificando las especificaciones del sistema (17).

A continuación se muestran algunos ejemplos de las tarjetas CRC.

Tarjeta CRC	
Nombre de la clase: serx-game-ui	
Responsabilidades	**Colaborador**
Se encargada de configurar la pantalla principal del entrenamiento, carga los filtros, los marcajes, configura la escena.	• serx-scene • serx-filter-interface • serx-mark-interface • serx-plugins-manager • filteredimage

Tabla 16. Tarjeta CRC serx-game-ui.

Tarjeta CRC	
Nombre de la clase: serx-scene	
Responsabilidades	**Colaborador**
Configura las animaciones de las imágenes; inicia, pausa o detiene el entrenamiento.	• serx-resources-manager • serx-group-image • serx-animation-manager • filteredimage

Tabla 17. Tarjeta CRC serx-scene.

2.5.2- Carga de recursos, filtros y marcajes

Para un mejor entendimiento de la manera en que SAEX realiza sus principales funcionalidades o sea, la carga dinámica de recursos, filtros y marcajes, asi como la aplicación de las dos últimas a las imágenes que transitan por la estera, se muestra a continuación un diagrama de clases en el cuál están reflejadas las clases que intervienen en la implementación de las mismas.

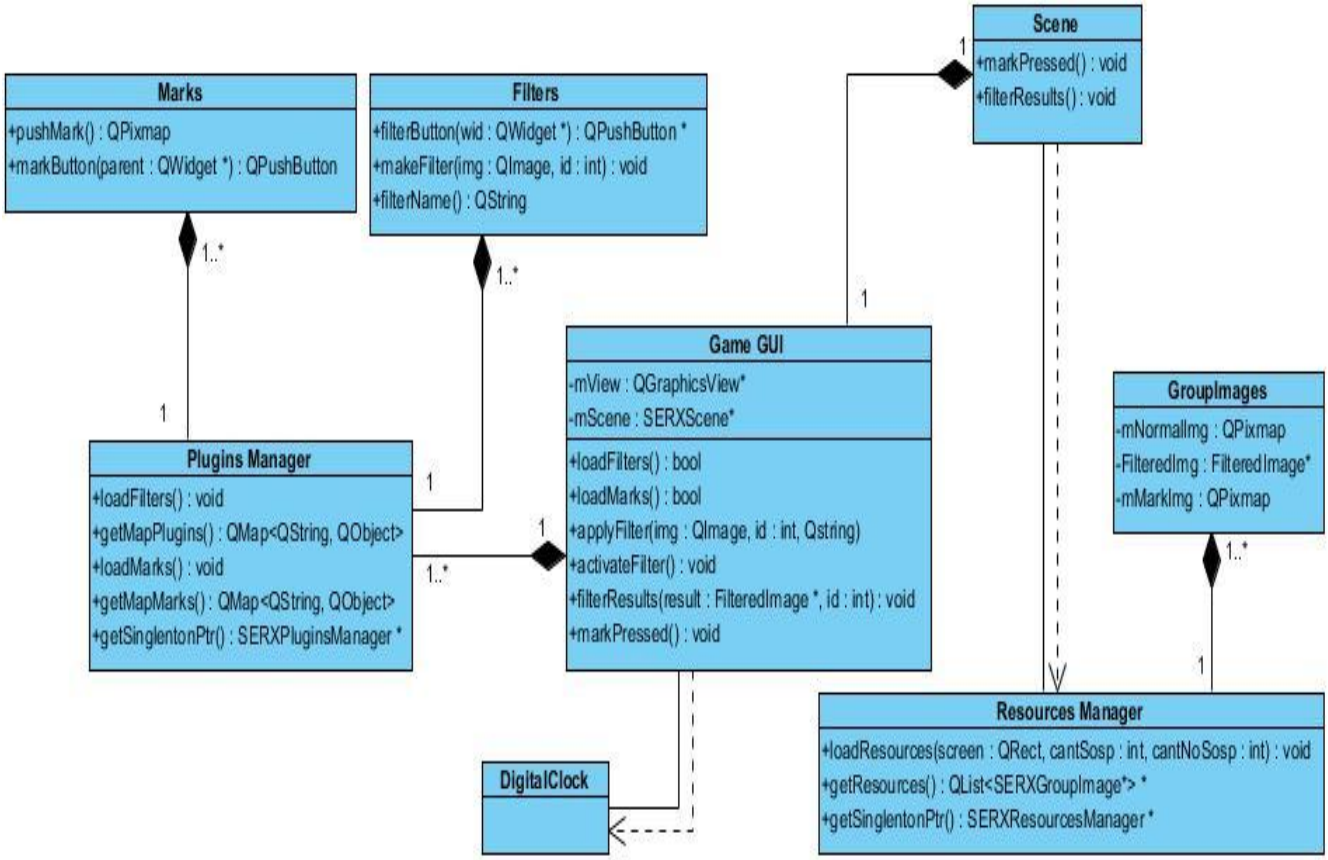


Ilustración 10. Diagrama de clases de las funcionalidades de cargar recursos, filtros y marcajes.

2.5.3- Aplicar filtros a las imágenes

Para aplicar un filtro o función realce a las imágenes que transitan por la estera, tienen participación varias clases de SAEX, las cuales se muestran en la **Ilustración 10**. Las imágenes transitan de manera lineal por la estera de izquierda a derecha, pasando por 3 estados diferentes, el primero cuando estas se encuentran antes de entrar en el área visible de la pantalla, luego cuando están siendo visualizadas por el usuario, y por ultimo al salir de la pantalla. Cuando el usuario activa uno de los filtros que se muestran en la Interfaz Gráfica de usuario (GUI), o sea, hace clic sobre un filtro, se determina a través del nombre de este, cual es el filtro que se activa y el sistema selecciona cuales son las imágenes que en ese momento estén siendo mostradas en la pantalla, luego cada una de estas imágenes por separadas se le transfieren al filtro

correspondiente, para que el mismo realice a través de algoritmos matemáticos la transformación visual correspondiente, una vez obtenido un resultado, este pasa a través de una señal hacia la escena, y esta se encarga de mostrar la nueva imagen, ya filtrada.

Luego que el filtro esté activo, las imágenes que van entrando secuencialmente a la parte visible de la pantalla, se le aplica el mismo procedimiento explicado anteriormente, ya que estas al cambiar de estado emiten una señal correspondiente a su cambio de estado.

Una vez que las imágenes dejen de transitar por la pantalla y pasen al último estado, se libera la imagen filtrada de la memoria de la PC, con el objetivo de disminuir el consumo de recursos en el ordenador.

2.5.4-Patrón arquitectónico

- **Basado en componentes**

Una arquitectura basada en componentes describe una aproximación de ingeniería de software al diseño y desarrollo de un sistema. Esta arquitectura se enfoca en la descomposición del diseño en componentes funcionales o lógicos que expongan interfaces de comunicación bien definidas. Esto provee un nivel de abstracción mayor que los principios de orientación por objetos y no se enfoca en asuntos específicos de los objetos como los protocolos de comunicación y la forma como se comparte el estado (18).

Un componente es una unidad de composición de aplicaciones, que posee un conjunto de interfaces especificadas contractualmente y dependencias del contexto explícitas. Que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio (18).

El estilo de arquitectura basado en componentes tiene como principales características:

- Ser un estilo de diseño para aplicaciones compuestas de componentes Individuales.
- Pone énfasis en la descomposición del sistema en componentes lógicos o funcionales que tienen interfaces bien definidas. Define una aproximación de diseño que usa componentes discretos, los que se comunican a través de interfaces que contienen métodos, eventos y propiedades.

Ventajas:

Facilidad de Instalación: cuando una nueva versión esté disponible, se podrá remplazar la versión existente sin impacto en otros componentes o el sistema como un todo.

Costos reducidos: el uso de componentes de terceros permite distribuir el costo del desarrollo y del mantenimiento.

Facilidad de desarrollo: los componentes implementan una interface bien definida para proveer la funcionalidad permitiendo el desarrollo sin impactar otras partes del sistema.

Reusable: el uso de componentes reutilizables significa que ellos pueden ser usados para distribuir el desarrollo y el mantenimiento entre múltiples aplicaciones y sistemas.

Facilidad de prueba: el uso de compontes en la aplicación permite probarlos a cada uno como una unidad independiente facilitando más tarde las tareas de pruebas (18).

2.6-Patrones de diseño

Un patrón describe un problema que ocurre una y otra vez en el entorno de desarrollo y describe también el núcleo de la solución al problema, de forma que puede utilizarse un millón de veces sin tener que hacer dos veces lo mismo.

Un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño general en un contexto particular.

2.6.1-Patrón Banda de los cuatro (GoF)

Los patrones GoF (Gang of Four, en español Pandilla de los Cuatro, formada por Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides) se clasifican en 3 categorías basadas en su propósito: creacionales, estructurales y de comportamiento (19).

Creacionales: los patrones creacionales abstraen el proceso de creación de instancias y ocultan los detalles de cómo los objetos son creados o inicializados.

Estructurales: los patrones estructurales se ocupan de cómo las clases y objetos se combinan para formar grandes estructuras y proporcionar nuevas funcionalidades.

Comportamiento: los patrones de comportamiento están relacionados con los algoritmos y la asignación de responsabilidades entre los objetos. Son utilizados para organizar, manejar y combinar comportamientos.

A continuación se muestra un ejemplo de estos patrones GOF.

- **Patrón de diseño singleton:**

El patrón singleton se implementa creando en la clase un método que crea una instancia del objeto sólo si todavía no existe alguna. Para asegurar que la clase no puede ser instanciada nuevamente se regula el alcance del constructor (con atributos como protegido o privado). Garantiza que una clase solo tenga una instancia y proporciona un punto de acceso global a ella (20).

Es el que más se utiliza en la aplicación, este contribuye a que durante el entrenamiento se trabaje con un solo perfil y un mismo conjunto de recursos. Ejemplo de esto es su utilización en la clase *serx-animation-manager*, la cual controla la animación de las imágenes que fueron cargadas en la aplicación.

2.6.2-Patrón de Asignación de Responsabilidades (GRASP)

Los patrones GRASP representan los principios básicos de la asignación de responsabilidades a objetos, expresados en forma de patrones. GRASP es el acrónimo para General Responsibility Assignment Software Patterns (Patrones Generales de Software para Asignar Responsabilidades) (19).

Para el desarrollo del SAEX se tuvieron en cuenta los siguientes patrones GRASP:

- **Patrón Experto:**

Consiste en la asignación de responsabilidades a la clase que contiene la información necesaria para el cumplimiento de las mismas. Si estas tareas se asignan de forma adecuada los sistemas tienden a ser mucho más fáciles de entender y se puede aprovechar la oportunidad de reutilizar componentes en futuras aplicaciones (19).

Este se ve reflejado en la clase serx-game-ui la cual es la encargada, entre otras cosas, de la configuración de los paneles donde se cargarán los filtros, marcajes, y donde se ubicarán los controles para iniciar, pausar o detener el entrenamiento.

- **Patrón Bajo Acoplamiento:**

Este patrón tiene un fácil entendimiento debido a que se encarga de mantener las clases poco ligadas con el fin de que las modificaciones en cualquier clase no repercutan en las restantes (19).

Se ve reflejado en el código fuente de la aplicación en la clase serx-Profile-Model, esta contiene un modelo de datos que enumera una serie de perfiles de usuario que hayan realizado algún entrenamiento con anterioridad. Por su parte la clase serx-select-Profile-ui visualiza los datos que contiene la clase serx-Profile-Model, de tal manera que cualquier modificación que ocurra en el modelo no requiera de cambios significativos dentro del código de la clase serx-select-Profile-ui.

Patrón Alta Cohesión:

Se caracteriza por asignar a las clases responsabilidades estrechamente relacionadas que no realicen un trabajo enorme, simplifica el mantenimiento y las mejoras de funcionalidad (19).

Este está reflejado en el código fuente de la aplicación en la clase serx-profile-manager la cual es a encargada de todo lo relacionado con los perfiles del usuario, o sea guardar los perfiles en un fichero xml, cargarlos, eliminarlos por solo mencionar algunos. Esta clase aunque delega algunas responsabilidades a la clase xml-profile-loader, mantiene su alta cohesión.

- **Patrón Controlador:**

Un controlador es un objeto de interfaz no destinada al usuario que debe delegar a otros objetos el trabajo que ha de realizarse mientras coordina la actividad (19).

Se ve reflejado en la clase serx-scene la cual es la encargada de configurar la escena principal del entrenamiento, dándole las animaciones a las imágenes así como controlar el inicio o parada del entrenamiento.

Conclusiones parciales

Con la solución propuesta se pretende brindar mayor preparación a los especialistas aduaneros en su etapa de capacitación. Quedaron definidas además las HU y las iteraciones en la que se realizan cada una para que posteriormente se especifiquen las tareas correspondientes a cada HU.

Capítulo #3. Implementación y Prueba.

El contenido que se encuentra en el presente capítulo trata sobre la descripción de la implementación del sistema. Se realiza además la descripción de las tareas de ingeniería para cada HU y se especifican las pruebas a las que fue sometida la aplicación para su correcto funcionamiento.

3.1- Fase de implementación

En esta fase se realiza la implementación de las HU (historias de usuario) que corresponden a cada una de las iteraciones antes descritas, se crean además las tareas de programación (tareas de ingeniería) para la implementación exitosa de cada HU.

Número	HU	Tareas por HU
1	Administrar perfil	**1.** Implementar añadir perfil. **2.** Implementar seleccionar o eliminar un perfil. **3.** Implementar los métodos que procesarán la información del perfil. **4.** Insertar los valores del perfil.
2	Salir	**1.** Implementar salir de la aplicación.
3	Entrenar	**1.** Diseñar el botón. **2.** Cargar el botón. **3.** Implementar el método que controlara el inicio del entrenamiento.
4	Pausar	**1.** Diseñar el botón. **2.** Cargar el botón. **3.** Implementar el método que controlara la pausa del entrenamiento.
5	Parar	**1.** Diseñar el botón. **2.** Cargar el botón. **3.** Implementar el método que controlara la parada del entrenamiento.

6	Mostrar imágenes	**1.** Cargar imágenes. **2.** Implementar el método para las animaciones. **3.** Implementar los métodos para mostrar imágenes.
7	Aplicarle filtros a las imágenes	**1.** Crear panel. **2.** Cargar filtros. **3.** Implementar métodos para aplicarles filtros a las imágenes.
8	Aplicarle marcajes a las imágenes	**1.** Crear panel. **2.** Cargar marcajes. **3.** Implementar métodos para realizarle marcajes a las imágenes.
9	Realizar zoom	**1.** Diseñar botones. **2.** Cargar botones. **3.** Implementar métodos para agrandar o disminuir las imágenes.
10	Ver detalles del entrenamiento	**1.** Crear interfaz. **2.** Implementar métodos para mostrar detalles del entrenamiento.
11	Reintentar	**1.** Diseñar botón. **2.** Implementar métodos para realizar un nuevo entrenamiento.
12	Exportar	**1.** Diseñar botón. **2.** Implementar métodos para generar reporte.

Tabla 18. Tareas de ingeniería por cada historia de usuario.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 1
Nombre de la tarea: Implementar añadir perfil.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 20/12/2014	**Fecha Fin:** 20/12/2014
Descripción: Se diseña e implementa la interfaz gráfica de usuario con los campos correspondientes a los datos que deberá entrar el especialista aduanero para crear el perfil.	

Tabla 19. Tarea de ingeniería. Implementar añadir perfil.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 1
Nombre de la tarea: Implementar seleccionar o eliminar un perfil.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 20/12/2014	**Fecha Fin:** 20/12/2014
Descripción: Se diseña e implementa la interfaz gráfica de usuario con los campos correspondientes para que el usuario (especialista aduanero en adiestramiento) pueda seleccionar o eliminar un perfil ya existente.	

Tabla 20. Tarea de ingeniería. Implementar seleccionar o eliminar un perfil.

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de Usuario:** 1
Nombre de la tarea: Implementar los métodos que procesarán la información del perfil.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 23/12/2014	**Fecha Fin:** 24/01/2015
Descripción: Se implementa en la clase serx-add-profile los métodos necesarios para añadir el perfil, esta se encarga además de validar que el perfil introducido no esté ya en el sistema. En la clase serx-select-profile se realiza la implementación de los métodos para la selección y la eliminación del perfil seleccionado.	

Tabla 21. Tarea de ingeniería. Implementar los métodos que procesarán la información del perfil.

Tarea de Ingeniería	
Número de tarea: 4	**Número de Historia de Usuario:** 1
Nombre de la tarea: Insertar los valores del perfil.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 24/12/2015	**Fecha Fin:**24/12/2015
Descripción: Se insertan los datos del perfil para luego ser procesados y rectificados por los métodos pertinentes.	

Tabla 22. Tarea de ingeniería. Insertar los valores del perfil.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 2
Nombre de la tarea: Implementar salir de la aplicación.	

Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 20/01/2015	**Fecha Fin:** 25/01/2015
Descripción: Se implementan las posibles formas de salir de la aplicación.	

Tabla 23. Tarea de ingeniería. Implementar salir de la aplicación.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 3
Nombre de la tarea: Diseñar el botón.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 26/01/2015	**Fecha Fin:** 26/01/2015
Descripción: Se realizó el diseño del botón que será presionado para iniciar el entrenamiento en el photoshop.	

Tabla 24. Tarea de ingeniería. Diseñar el botón.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 3
Nombre de la tarea: Cargar el botón.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 2701/2015	**Fecha Fin:** 27/01/2015
Descripción: Se realiza la implementación de los métodos para cargar el botón y ubicarlo en la aplicación en la clase serx-game-ui.	

Tabla 25. Tarea de ingeniería. Cargar el botón.

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de Usuario:** 3
Nombre de la tarea: Implementar el método que controlara el inicio del entrenamiento.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 1
Fecha Inicio: 28/01/2015	**Fecha Fin:** 02/02/2015
Descripción: Se realiza la implementación de método en la clase serx-scene, la cual es la encargada de darle animaciones a las imágenes.	

Tabla 26. Tarea de ingeniería. Implementar el método que controlará el inicio del entrenamiento.

Tarea de Ingeniería

Número de tarea: 1	**Número de Historia de Usuario:** 4
Nombre de la tarea: Diseñar el botón.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 03/02/2015	**Fecha Fin:** 03/02/2015
Descripción: Se realiza el diseño del botón que será presionado para darle pausa al entrenamiento.	

Tabla 27. Tarea de ingeniería. Diseñar el botón.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 4
Nombre de la tarea: Cargar el botón.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 03/02/2015	**Fecha Fin:** 03/02/2015
Descripción: Se realiza la implementación de la carga del botón para ubicarlo en la aplicación en la clase serx-game-ui. La misma es la encargada de configurar la interfaz principal de la aplicación.	

Tabla 28. Tarea de ingeniería. Cargar el botón.

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de Usuario:** 4
Nombre de la tarea: Implementar el método que controlara la pausa del entrenamiento.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 04/02/2015	**Fecha Fin:** 04/02/2015
Descripción: Se realiza la implementación de método en la clase serx-scene, la cual es la encargada de darle animaciones a las imágenes y de configurar la pantalla principal.	

Tabla 29. Tarea de ingeniería. Implementar el método que controlará la pausa del entrenamiento.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 5
Nombre de la tarea: Diseñar el botón.	

Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 05/02/2015	**Fecha Fin:** 05/02/2015
Descripción: Se realiza el diseño del botón que será presionado para parar el entrenamiento en el photoshop.	

Tabla 30. Tarea de ingeniería. Diseñar el botón.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 5
Nombre de la tarea: Cargar el botón.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 06/02/2015	**Fecha Fin:** 06/02/2015
Descripción: Se realiza la implementación de la carga del botón para ubicarlo en la aplicación en la clase serx-game-ui.	

Tabla 31. Tarea de ingeniería. Cargar el botón.

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de Usuario:** 5
Nombre de la tarea: Implementar el método que controlará la parada del entrenamiento.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 07/02/2015	**Fecha Fin:** 07/02/2015
Descripción: Se realiza la implementación del método que controla la parada del entrenamiento en la clase serx-game-ui.	

Tabla 32. Tarea de ingeniería. Implementar el método que controlará la parada del entrenamiento.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 6
Nombre de la tarea: Cargar imágenes.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 08/02/2015	**Fecha Fin:** 08/02/2015
Descripción: Se implementan los métodos para cargar las imágenes en la clase xml-resources-loader.	

Tabla 33. Tarea de ingeniería. Cargar imágenes.

Tarea de Ingeniería

Número de tarea: 2	**Número de Historia de Usuario:** 6
Nombre de la tarea: Implementar el método para las animaciones.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.6
Fecha Inicio: 09/02/2015	**Fecha Fin:** 11/02/2015
Descripción: Se implementa el método para darle animaciones a las imágenes en la clase serx-scene.	

Tabla 34. Tarea de ingeniería. Implementar el método para darle animaciones a las imágenes.

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de Usuario:** 6
Nombre de la tarea: Implementar los métodos para mostrar imágenes.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 12/02/2015	**Fecha Fin:** 13/02/2015
Descripción: Se implementa el método en la clase serx-game-ui.	

Tabla 35. Tarea de ingeniería. Implementar los métodos para mostrar imágenes.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 7
Nombre de la tarea: Crear panel.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 16/02/2014	**Fecha Fin:** 16/02/2014
Descripción: Se implementa la creación del panel donde se cargarán los filtros en la clase serx-game-ui.	

Tabla 36. Tarea de ingeniería. Crear panel.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 7
Nombre de la tarea: Cargar filtros.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 17/02/2015	**Fecha Fin:** 17/02/2015
Descripción: Se implementa el método encargado de cargar los filtros que se les aplicarán a las imágenes en la clase serx-plugin-manager.	

Tabla 37. Tarea de ingeniería. Cargar filtros.

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de Usuario:** 7
Nombre de la tarea: Implementar métodos para aplicarles filtros a las imágenes.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.6
Fecha Inicio: 18/02/2015	**Fecha Fin:** 20/02/2015
Descripción: Se implementan los métodos para aplicarle el filtro seleccionado a la imagen seleccionada en la clase serx-game-ui.	

Tabla 38. Tarea de ingeniería. Implementar métodos para aplicar filtros a las imágenes.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 8
Nombre de la tarea: Crear panel.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 23/02/2014	**Fecha Fin:** 23/02/2014
Descripción: Se implementa la creación del panel donde se ubican los marcajes en la clase serx-game-ui.	

Tabla 39. Tarea de ingeniería. Crear panel.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 8
Nombre de la tarea: Cargar marcajes.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 24/02/2015	**Fecha Fin:** 24/02/2015
Descripción: Se implementa el método encargado de cargar los marcajes que se les realizan a las imágenes en la clase serx-plugin-manager.	

Tabla 40. Tarea de ingeniería. Cargar marcajes.

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de Usuario:** 8
Nombre de la tarea: Implementar métodos para realizarle marcajes a las imágenes.	

Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.4
Fecha Inicio: 25/02/2015	**Fecha Fin:** 25/02/2015
Descripción: Se implementa el método para aplicarle el marcaje seleccionado, a la imagen seleccionada, en la clase serx-game-ui.	

Tabla 41. Tarea de ingeniería. Implementar métodos para realizar marcajes a las imágenes.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 9
Nombre de la tarea: Diseñar botones.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 26/02/2015	**Fecha Fin:** 26/02/2015
Descripción: El diseño de los botones que serán presionados para agrandar o disminuir las imágenes se realizó con la herramienta photoshop.	

Tabla 42. Tarea de ingeniería. Diseñar botones.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 9
Nombre de la tarea: Cargar los botones.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 27/02/2015	**Fecha Fin:** 27/02/2015
Descripción: Se realiza la implementación de la carga dinámica de los botones en la clase serx-game-ui.	

Tabla 43. Tarea de ingeniería. Cargar los botones.

Tarea de Ingeniería	
Número de tarea: 3	**Número de Historia de Usuario:** 9
Nombre de la tarea: Implementar métodos para agrandar o disminuir las imágenes.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 28/02/2015	**Fecha Fin:** 28/02/2015
Descripción: Se implementan los métodos en la clase serx-game-ui, los cuales permiten agrandar o disminuir las imágenes según el botón presionado para realizar dicha acción.	

Tabla 44. Tarea de ingeniería. Implementar métodos para agrandar o disminuir las imágenes.

Tarea de Ingeniería

Número de tarea: 1	**Número de Historia de Usuario:** 10
Nombre de la tarea: Crear interfaz.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 01/03/2015	**Fecha Fin:** 01/03/2015
Descripción: Se crea la interfaz con los campos que mostrarán los detalles del entrenamiento realizado en la clase serx-profile-details-ui.	

Tabla 45. Tarea de ingeniería. Crear interfaz.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 10
Nombre de la tarea: Implementar métodos para mostrar detalles del entrenamiento.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.6
Fecha Inicio: 02/03/2015	**Fecha Fin:** 04/03/2015
Descripción: Se implementan los métodos necesarios para mostrar los detalles del entrenamiento en la clase serx-profile-details.	

Tabla 46. Tarea de ingeniería. Implementar métodos para mostrar detalles del entrenamiento.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 11
Nombre de la tarea: Diseñar botón.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 01/04/2015	**Fecha Fin:** 01/04/2015
Descripción: Se diseña el botón que será presionado para realizar un nuevo entrenamiento, en el qt creator.	

Tabla 47. Tarea de ingeniería. Diseñar botón.

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 11
Nombre de la tarea: Implementar métodos para realizar un nuevo entrenamiento.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.8
Fecha Inicio: 02/04/2015	**Fecha Fin:** 05/04/2015
Descripción: Se implementan los métodos necesarios para que el usuario pueda realizar un nuevo entrenamiento en la clase serx-profile-details-ui.	

Tabla 48. Tarea de ingeniería. Implementar métodos para realizar un nuevo entrenamiento.

Tarea de Ingeniería	
Número de tarea: 1	**Número de Historia de Usuario:** 12
Nombre de la tarea: Diseñar el botón.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 21/02/2015	**Fecha Fin:** 21/02/2015
Descripción: Se realizó el diseño del botón que será presionado para realizar el reporte del entrenamiento en el qt creator.	

Tabla 49. Tarea de ingeniería. Diseñar el botón

Tarea de Ingeniería	
Número de tarea: 2	**Número de Historia de Usuario:** 12
Nombre de la tarea: Implementar los métodos necesarios para la generación del reporte.	
Tipo de tarea: Desarrollo	**Puntos Estimados:** 0.2
Fecha Inicio: 22/02/2015	**Fecha Fin:** 22/02/2015
Descripción: Se implementan los métodos para la generación del reporte en la clase serx-report-ui.	

Tabla 50. Tarea de ingeniería. Implementar los métodos necesarios para la generación del reporte.

3.2-Pruebas del Sistema

Al concluir la fase de implementación se pasa a las pruebas del sistema en la cual tanto el cliente como el desarrollador analizan si la aplicación tiene la calidad requerida y los errores que puede arrojar la misma. De los diversos tipos de pruebas existentes se decide realizar los tipos siguientes tipos de pruebas.

3.2.1-Pruebas de caja blanca

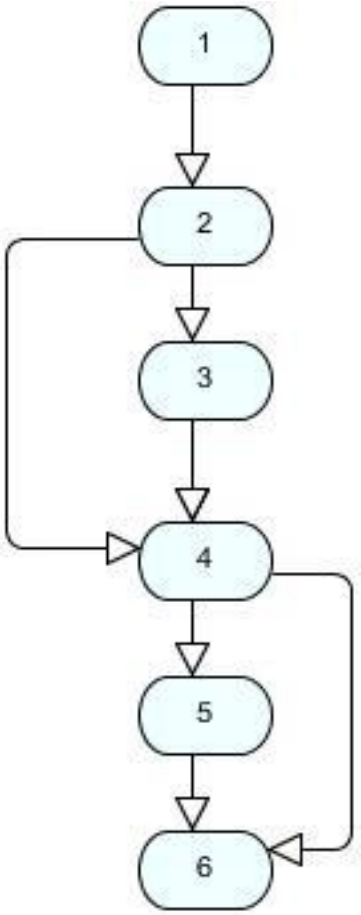
Pruebas que se enfocan en los mecanismos internos de un sistema o componente o sea en la parte del código. Estas están basadas en la estructura de un programa en el cual se requiere representar la ejecución de este mediante grafos de flujo. Estos permiten analizar los posibles caminos que pueden tener un método, determinada función o algoritmo, luego es probado para ver si el resultado obtenido es el esperado.

Método: marcaje presionado

```
SERXGroupImage * selected = 0;
 if (this->selectedItems().size())
 selected =  (SERXGroupImage*)(this->selectedItems().at(0));

 if (selected)
 {
 selected->setMMarkName(MarkName);
 selected->setMMarkImg(markImage);
 selected->update();
 }
```

Grafo resultante:



Complejidad Ciclomática:

V (G)= # de regiones = 3

V (G)=A-N+2 = 7-6+2 = 3

V (G)= P+1= 2+1 = 3

Caminos básicos:

{1-2-3-4-5-6}

{1-2-3-4-6}

{1-2-3-4-5-6}

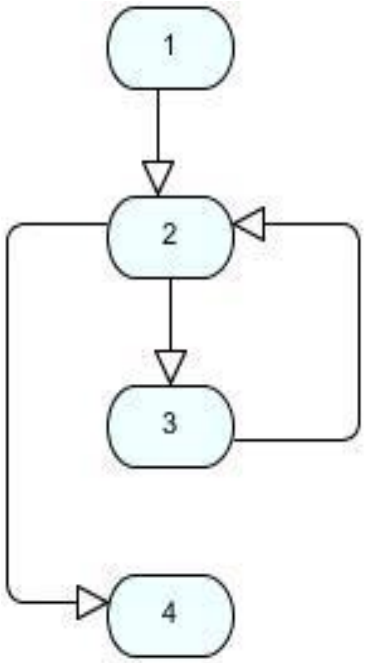
{1-2-4-6}

Tabla 51. Prueba de caja blanca#2. Marcaje presionado.

Método: configurar escena

```
setupAnimation();
SERXGroupImage *imgG;
QListIterator <SERXGroupImage*> _itr(*mGroupImages);
 while (_itr.hasNext())
 {
 imgG = _itr.next();
 imgG->run();
 }
 mFilterActive="NORMAL";
 isPlaying = false;
```

Grafo resultante:



Complejidad Ciclomática:

V (G)= # de regiones = 2

V (G)=A-N+2 = 4-4+2 = 2

V (G)= 1+1= +1 = 2

Caminos básicos:

{1-2-3-2-4}

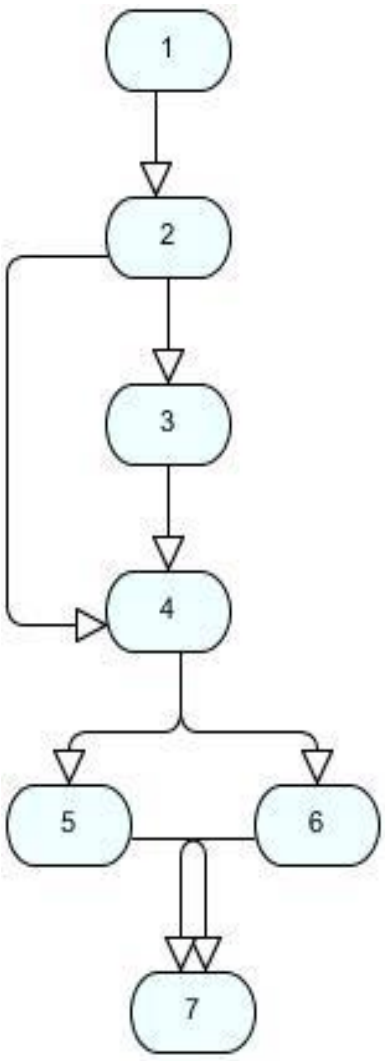
{1-2-4}

Tabla 52. Prueba de caja blanca#2. Configurar escena.

Método: entrenar

```
bool isPlaying = mScene->getIsPlaying();
 if (isFirst)
 {
clock->reset();
clock->start();
isFirst = false;
 }
 if (!isPlaying)
 {
mScene->play();
qDebug() << "#### Play Pressed: " << btPlay->text();
btPlay->setIcon(QIcon(":/Gui/pause"));
 }
 else
 {
mScene->pause();
btPlay->setIcon(QIcon(":/Gui/play"));
 pauseTimer->start(15000);
 }
```

Grafo resultante:



Complejidad Ciclomática:

V (G)= # de regiones = 4

V (G)=A-N+2 = 9-7+2 =4

V (G)= P+1= 2+1 = 2

Caminos básicos:

{1-2-3-4-5-6-8}

{1-2-3-4-5-7-8}

Tabla 53. Prueba de caja blanca#3. Entrenar.

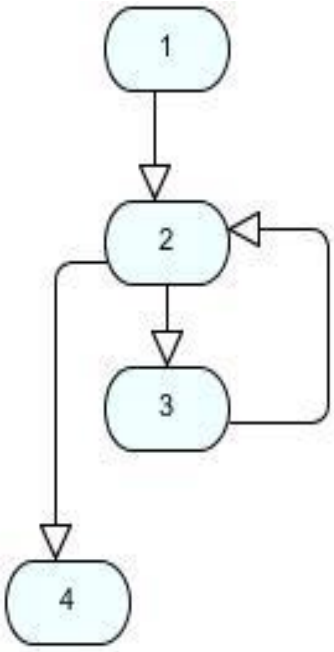
Método: conectar imágenes

```
mGroupImages = SERXResourcesManager::getSinglentonPtr()-
>getResources();

 int count = mGroupImages->size();
 SERXGroupImage *_tempGroup;

 for (int i = 0; i < count; i++)
 {
  _tempGroup = mGroupImages->at(i);
  connect(_tempGroup, SIGNAL(applyActiveFilter(QImage,int,
QString)), this, SIGNAL(applyActiveFilter(QImage,int, QString)));
}
```

Grafo resultante:



Complejidad Ciclomática:

V (G)= # de regiones = 2

V (G)=A-N+2 = 4-4+2 = 2

V (G)= P+1= 1+1 = 2

Caminos básicos:

{1-2-4}

{1-2-3-2-4}

Tabla 54. Prueba de caja blanca#4. Conectar imágenes.

En la ejecución de estas pruebas de caja blanca se probaron las principales funcionalidades del sistema. Se realizaron dos iteraciones de pruebas de caja blanca con un total de 12 casos de prueba, donde en la primera iteración 8 resultaron satisfactorios y 4 fallidos, estos casos fallidos fueron corregidos en la segunda iteración.

3.2.2-Pruebas de aceptación

Son básicamente pruebas funcionales sobre el sistema completo y buscan una cobertura de la especificación de requisitos y del manual del usuario, las mismas permiten al cliente dar el visto bueno al sistema desarrollado. A continuación se muestran las pruebas realizadas a las principales funcionalidades.

Caso de Prueba de Aceptación	
Código: H1_P1	**Historia de Usuario:** 1
Nombre: Crear perfil.	
Descripción: El sistema permite al usuario crear un perfil.	
Condiciones de ejecución: No presenta ninguna condición de ejecución.	
Entrada/Pasos de ejecución: 1. Clic en el botón “Crear”. 2. Introducir el texto en el campo nombre. 3. Clic en el botón “Imagen”. 4. Buscar y seleccionar la imagen que se desea para el perfil. 5. Clic en el botón “Añadir”.	
Resultado esperado: El sistema debe comprobar si el usuario que se introduce no está creado, en caso de no estarlo, se crea el perfil.	
Resultado obtenido: Se creó el perfil .	
Evaluación de la prueba: Satisfactoria.	

Tabla 55. Prueba de aceptación#1. HU-1.

Caso de Prueba de Aceptación	
Código: H2_P1	**Historia de Usuario:** 2
Nombre: Salir	

Descripción: El sistema permite al usuario salir del entrenamiento
Condiciones de ejecución: No presenta ninguna condición de ejecución.
Entrada/Pasos de ejecución: 1. Clic en el botón salir 2. Clic en el botón “Si” en la ventana de confirmación de salida que aparece luego de del clic en el botón salir.
Resultado esperado: El sistema deberá cerrar la aplicación.
Resultado obtenido: Se cierra el sistema.
Evaluación de la prueba: Satisfactoria.

Tabla 56. Prueba de aceptación#2. HU-2.

Caso de Prueba de Aceptación	
Código: H3_P1	**Historia de Usuario:** 3
Nombre: Entrenar	
Descripción: El sistema permite darle inicio al entrenamiento	
Condiciones de ejecución: El usuario debe haberse creado un perfil.	
Entrada/Pasos de ejecución: 1. Clic en el botón “Nueva partida” 2. Clic en el botón “*Play*”	
Resultado esperado: El sistema carga todas las imágenes que serán utilizadas durante el entrenamiento, dará animaciones a dichas imágenes, mostrará las imágenes en movimiento por el medio de la pantalla, cargará los filtros y marcajes conjuntamente con los botones iniciar, pausar, detener, agrandar.	
Resultado obtenido: Se cargaron todas la imágenes, se le dieron animaciones a todas las imágenes lo cual permitió que se muestre en movimiento una tras otra en la pantalla, cargándose conjuntamente todos los filtros y marcajes.	
Evaluación de la prueba: Satisfactoria.	

Tabla 57. Prueba de aceptación#3. HU-3.

Caso de Prueba de Aceptación	
Código: H10_P1	**Historia de Usuario:** 10
Nombre: Ver Detalles del entrenamiento.	
Descripción: El sistema permite ver lo detalles del entrenamiento, brindando así información como cantidad de entrenamientos, efectividad, fecha, por solo mencionar algunos. También permite ver los resultados anteriores del cada perfil.	
Condiciones de ejecución: El usuario debe tener un perfil creado.	
Entrada/Pasos de ejecución: 1. Clic en el botón “Ver Detalles”	
Resultado esperado: El sistema debe cargar todos los entrenamientos anteriores realizados por el usuario con el actual perfil y mostrar detalles de dichos entrenamientos.	
Resultado obtenido: Se cargaron todos los entrenamientos antes realizado por el usuario y se mostraron los detalles de cada partida realizada.	
Evaluación de la prueba: Satisfactoria.	

Tabla 58. Prueba de aceptación#4. HU-10.

Se realizaron dos iteraciones de pruebas para eliminar las deficiencias encontradas ya que al realizar una primera de iteración de 12 casos de pruebas aplicados, 10 fueron satisfactorios, los restantes fueron corregidos en una segunda iteración.

3.3- Comparación entre SAEX y SERX

Una vez terminada la implementación se realizaron pruebas para comparar SAEX y SERX en cuanto a consumo de la memoria RAM de la computadora y tiempo en que demora cada uno en cargar las imágenes de los equipajes.

Estas pruebas se realizaron en un pc con las siguientes prestaciones:

Procesador: Intel(R) Core (TM) i3-4030U CPU 1.90GHz (gigahercio).
Memoria: 4096 MB.
Sistema operativo: Ubuntu 14.04.

Con un total de 52 imágenes SAEX utilizó 90 MB de la memoria RAM de la computadora mientras que el SERX 400 MB (ver Gráfico 1), ya que este último además de cargar la imagen normal

también carga las imágenes de su respectivo filtro (Ejemplo: filtro BW (negro y blanco), se carga en la memoria la imagen normal más la imagen en blanco y negro).

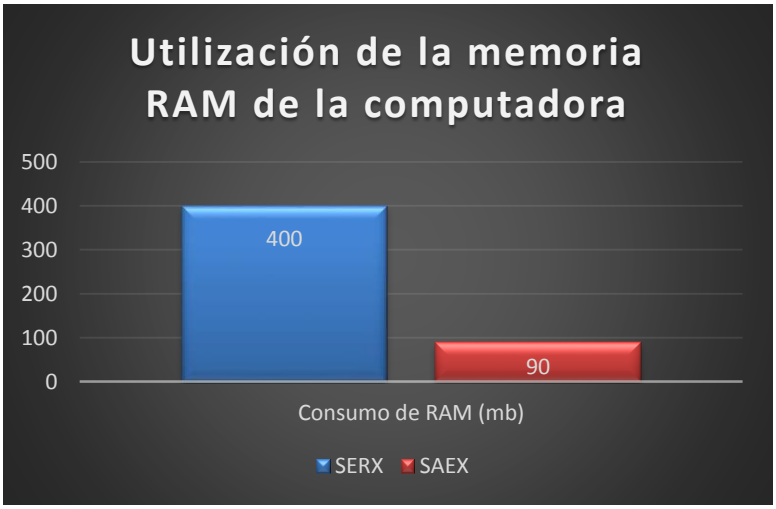


Gráfico 1. Utilización de la memoria RAM de la computadora.

En cuanto al tiempo de demora en cargar las colecciones de imágenes se obtuvo el siguiente resultado (ver Gráfico 2):

SAEX demoró 3 segundos pues para la aplicación de los filtros solo necesita cargar las imágenes de los equipajes en su vista normal; mientras que SERX demoró 24 segundos ya que se multiplica el conjunto de imágenes en su vista normal por 8 que son las vistas de las imágenes con los filtros aplicados.

Gráfico 2. Tiempo de carga de los recursos.

Conclusiones parciales

En este capítulo se realizaron las tareas de implementación del sistema de entrenamiento de rayos X aduanero, se analizaron además los resultados de las distintas pruebas realizadas para comprobar si se logró el objetivo deseado, brindando este análisis un resultado positivo.

Conclusiones generales

Con la realización de este trabajo se cumple el objetivo planteado pues se obtiene un nuevo sistema de entrenamiento de rayos X aduaneros que corrige las limitaciones de la primera versión. Es un sistema que puede ser comercializado ya que no está sujeto a ningún cliente como el caso de SERX que fue creado solamente para la AGR, SAEX fue desarrollado sobre la base técnica del centro, lo que permite brindarle un mejor soporte.

Al incluir nuevas funcionalidades como la de generar un reporte en un documento fuera de aplicación, brinda la posibilidad de tener constancia del entrenamiento realizado para un posterior análisis de los resultados obtenidos.

La utilización de componentes para realizar los marcajes y funciones realce le proporciona al sistema extensibilidad sin realizar modificaciones a su código fuente, y así obtener un producto personalizado dependiendo del equipo de rayos X que se desee simular. También se utilizan componentes para cargar las imágenes que son utilizadas durante los entrenamientos y los datos de los usuarios, de esta manera el sistema puede ser capaz de incorporarlos desde diferentes orígenes, ya sea ficheros o bases de datos, en este último caso estos datos pueden ser centralizados, dependiendo del ambiente donde se vaya a utilizar el sistema, ya sea en red o estaciones de trabajo sin conexión.

Al realizar el filtrado de las imágenes mediante algoritmos se reduce considerablemente el uso de la memoria RAM del ordenador, y también el tiempo con que el sistema carga la colección de imágenes para cada entrenamiento que se desee realizar.

Recomendaciones

A continuación se muestran algunas recomendaciones para futuros trabajos con la aplicación:

- Centralizar los datos de los usuarios en una base de datos para que sean manejados desde un servidor, permitiendo así que los usuarios puedan obtener un registro de sus entrenamientos anteriores desde cualquier estación de trabajo, en caso de que sea desplegado en un ambiente conectado en red.
- Centralizar las imágenes en una base de datos para que sean manejadas desde un servidor y permitir así la incorporación de nuevos conjuntos de imágenes de manera simultánea para cada estación de trabajo donde se realizan los entrenamientos.
- Incorporar a la escena principal la opción de darle reversa a la estera por la cual pasan las imágenes.

Referencias bibliográficas

1. Definicion de rayos X. [En línea] http://definicion.de/rayos x..

2. Definicion de entrenamiento. [En línea] http://definicion.de/entrenamiento/.

3. Reglamentos. [En línea] 4 de Marzo de 2010. [Citado el: 20 de Mayo de 2015.] http://www.aena.es/csee/ccurl/811/666/Reglamento%20%28UE%29%20185_2010%20esp.pdf.

4. Renful. [En línea] http://www.es.renful.com/x-ray-cbt/simfox#descripcin.

5. *Sistema de entrenamiento de rayos X.* Habana : s.n., 2010.

6. Metodologías tradicionales, ágiles, para juegos y para aplicaciones móviles. [En línea] [Citado el: 2 de Febrero de 2015.] http://tallerinf281.wikispaces.com/file/view/METODOLOG%C3%8DAS+TRADICIONALES.pdf.

7. Camejo Días, Lianet. *Diseño de una base de datos para el sistema de captura y catalogación de medidas.* Habana : s.n., 2010.

8. Fernández Sanchez, Mayté. *Análisis y diseño del componente alertas y avisos del Marco de Trabajo Sauxe.* Habana : s.n., 2011.

9. Olvera, Carlos. ¿Que es un UML? [En línea] 2010. [Citado el: 5 de 3 de 2015.] http://profesores.fi-b.unam.mx/carlos/aydoo/uml.html. .

10. Category:Tools::QtCreator Spanish. [En línea] [Citado el: 10 de Enero de 2015.] http://wiki.qt.io/Category:Tools::QtCreator_Spanish.

11. Silva Rojas, Luis Guillermo. *VISUALIZACIÓN DIRECTA PARA ENDOSCOPIAS VIRTUALES.* Habana : s.n., 2011.

12. Merzeau Martínez, Alejandro David. *Arquitectura de software para los Laboratorios Virtuales.* Habana : s.n., 2012.

13. Rodriguez Sillero, Juan Miguel. *Interfaz de Usuario para los laboratorios de informática.* Habana : s.n., 2011.

14. Olivera Sosa, Ángel Gabriel. Scribd. [En línea] 6 de Septiembre de 2010. [Citado el: 2 de Junio de 2015.] http://es.scribd.com/doc/37187866/Requerimientos-funcionales-y-no-funcionales#scribd.

15. Granada, Universidad de. Especificación de requerimientos. [En línea] [Citado el: 2 de Marzo de 2015.] http://elvex.ugr.es/idbis/db/docs/design/2-requirements.pdf.

16. Extreme-Programming-Explained-Embrace-Edition. [En línea] [Citado el: 20 de Enero de 2015.] http://www.amazon.com/Extreme-Programming-Explained-Embrace-Edition/dp/0321278658. Extreme Programming Explained: Embrace Change.

17. Desarrollo del software.Targetas CRC. [En línea] [Citado el: 10 de Marzo de 2015.] https://jummp.wordpress.com/2012/01/10/desarrollo-de-software-tarjetas-crc/.

18. Merzeau Martínez, Alejandro David. *Arquitectura de software para los laboratorios virtuales.* Habana : s.n., 2012.

19. EcuRed. *Patrones de diseño y arquitectura.* [En línea] [Citado el: 25 de Febrero de 2015.] http://www.ecured.cu/index.php/Patrones_de_dise%C3%B1o_y_arquitectura#Patrones_de_dise.C3.B1o_GoF.

20. El patron singlenton. Developer Network. [En línea] [Citado el: 25 de Abril de 2015.] http://msdn.microsoft.com/es-es/library/bb972272.aspx..

Glosario de términos

Simulador: Un simulador es una máquina que reproduce el comportamiento de un sistema en ciertas condiciones, lo que permite que la persona que debe manejar dicho sistema pueda entrenarse. Los simuladores suelen combinar partes mecánicas o electrónicas y partes virtuales que le ayudan a generar una reproducción precisa de la realidad.

SDK: Kit de desarrollo de software o SDK (siglas en inglés de *software development kit*) es generalmente un conjunto de herramientas de desarrollo que le permite a un programador crear aplicaciones para un sistema concreto, por ejemplo ciertos paquetes de software, *frameworks*, plataformas de *hardware*, computadoras, videoconsolas, sistemas operativos, etc.

Container: Contenedor, que procede del inglés *container*, es un recipiente que se utiliza para depositar residuos o un embalaje grande, de dimensiones y tipos normalizados internacionalmente, que se utiliza para el traslado de mercancías.

Funciones realce o filtros: Algoritmo matemático que se le aplica a una imagen para cambiar su apariencia con el objetivo de resaltar determinados elementos.

Anexos

Anexo 1: Imágenes de la aplicación.

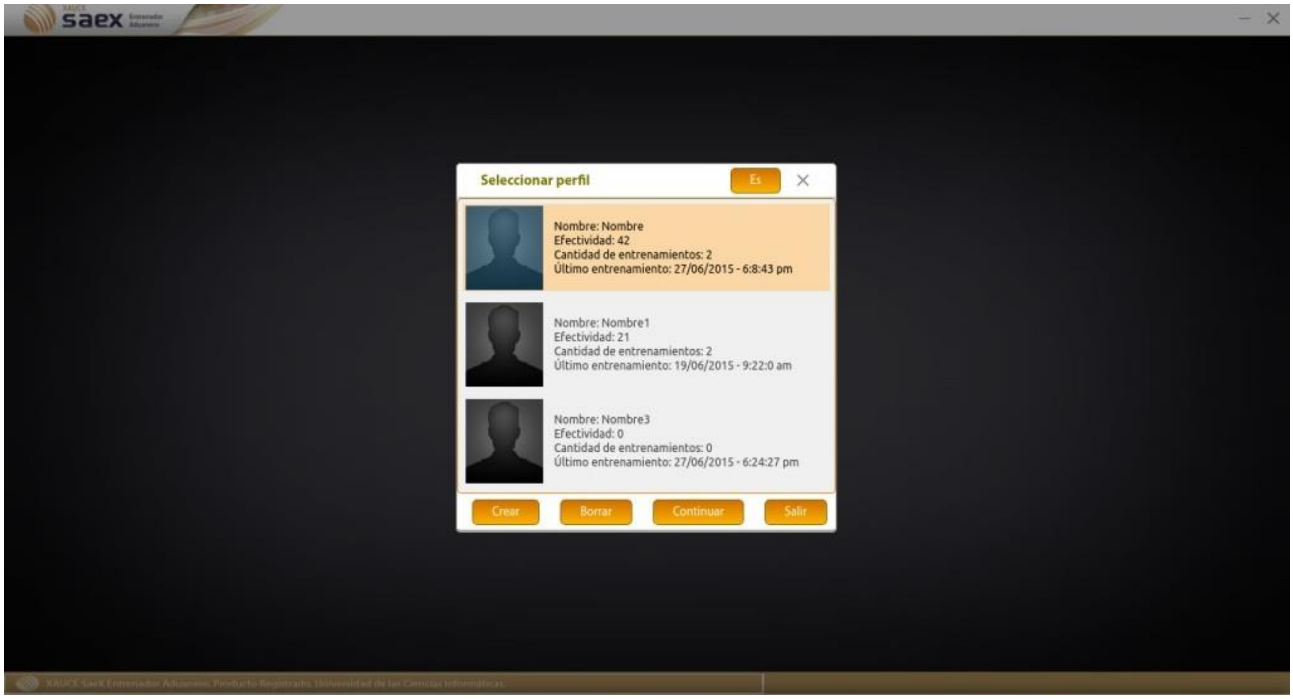


Ilustración 11. Administración del perfil (seleccionar, borrar).

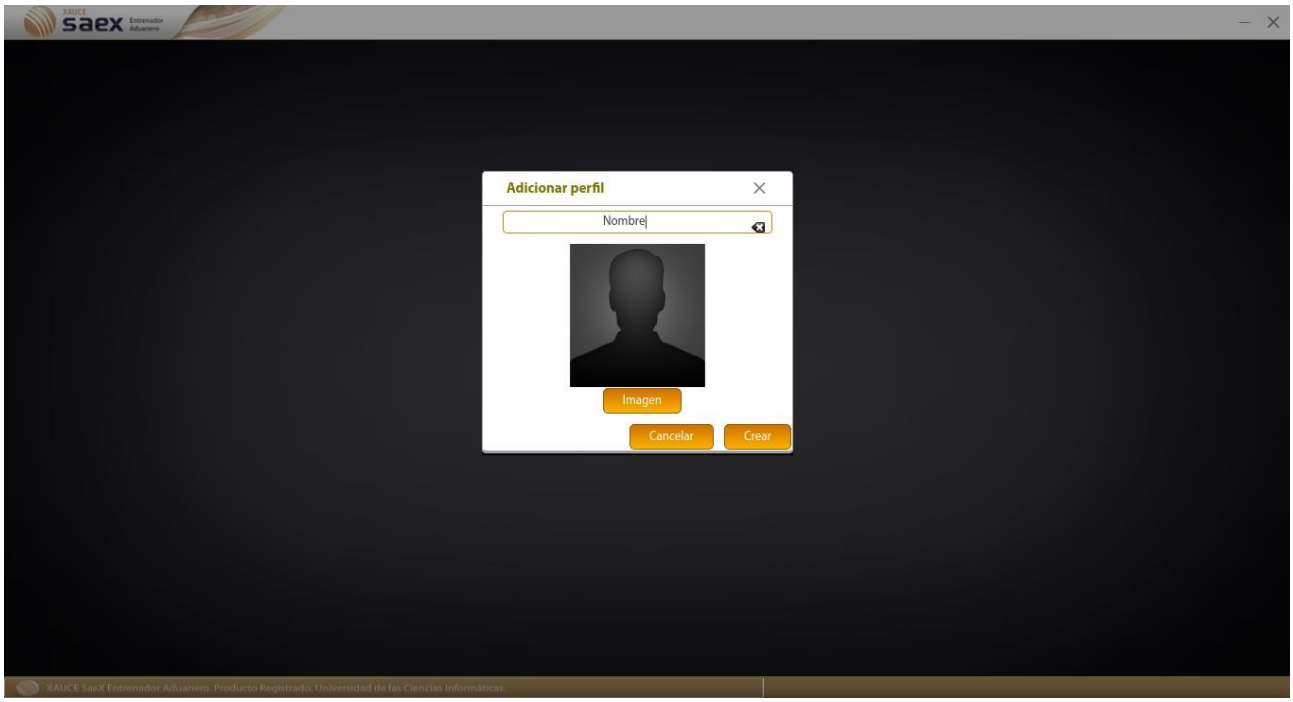


Ilustración 12. Administración del perfil (añadir).

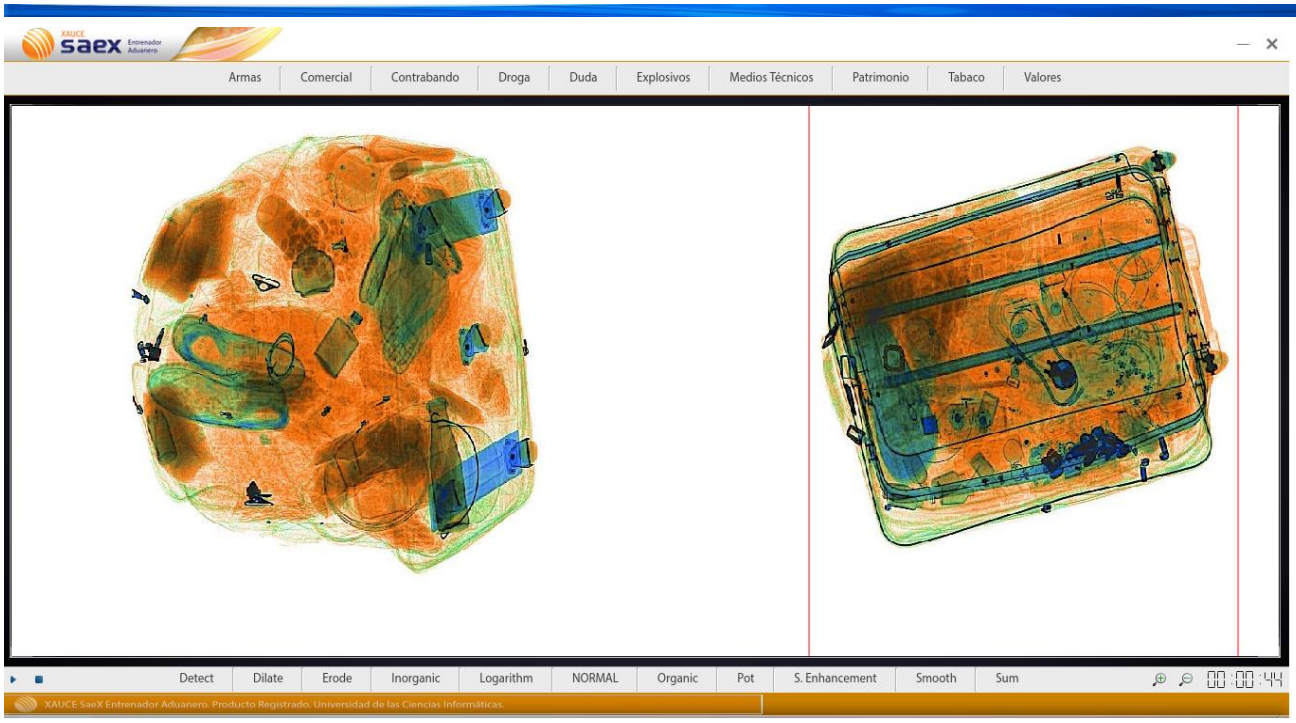


Ilustración 13. Pantalla principal del entrenamiento.

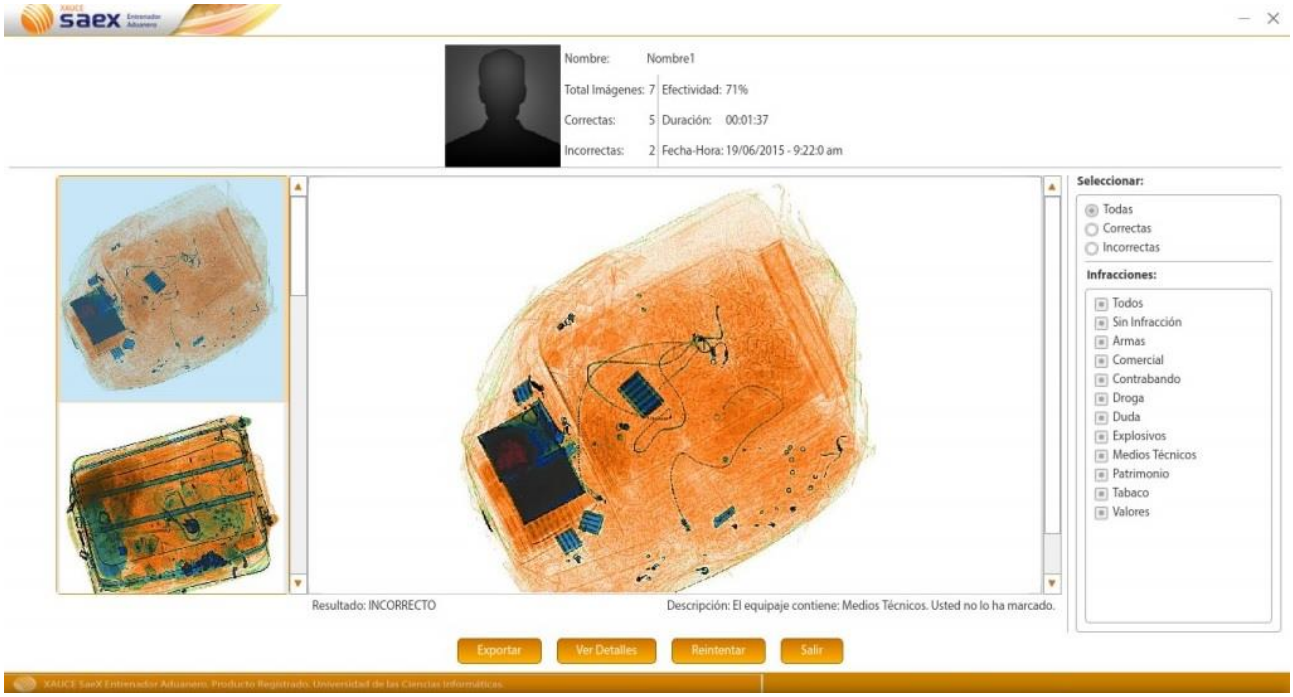


Ilustración 14. Resultado del entrenamiento realizado.

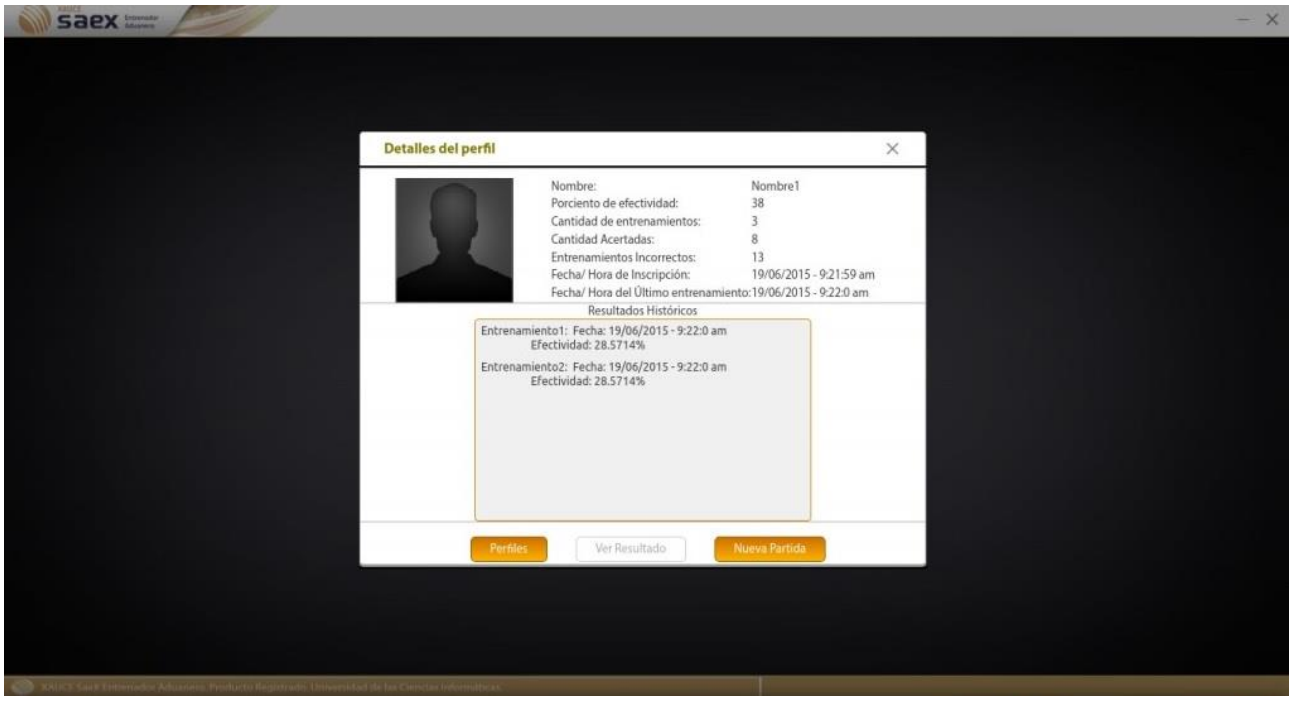


Ilustración 15. Detalles de los entrenamientos realizados.

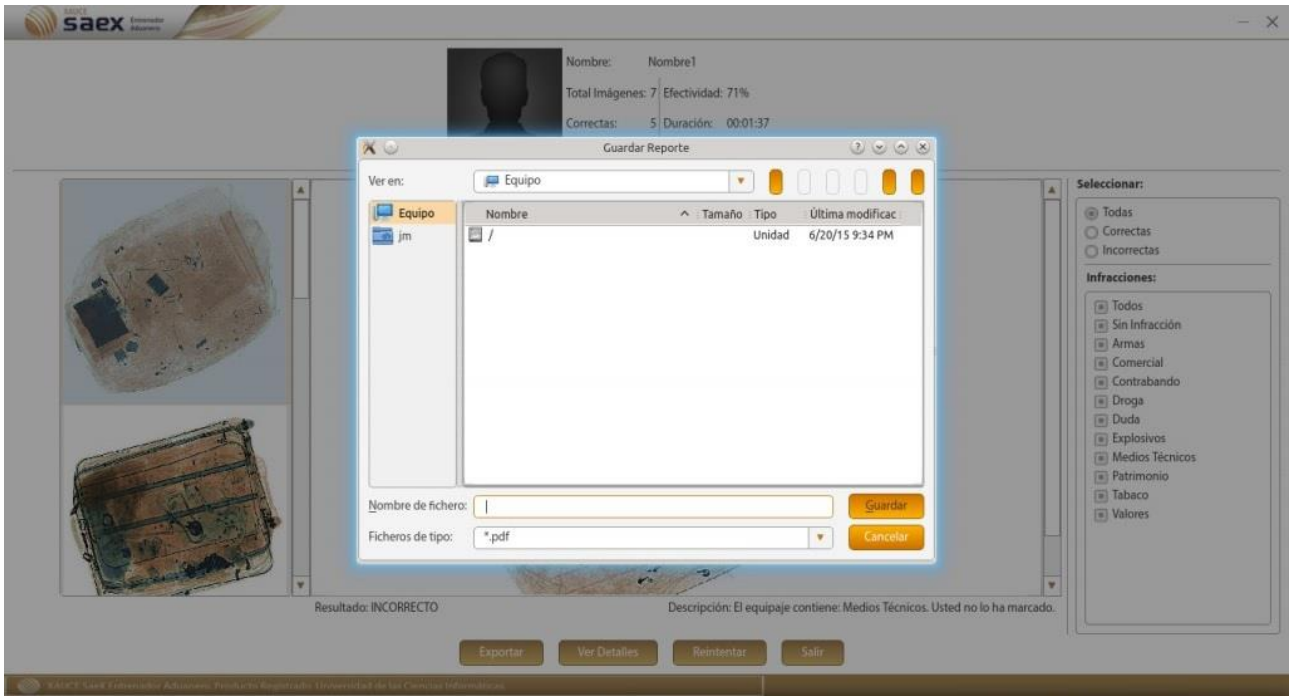


Ilustración 16. Realizar reporte.