DISEÑO E IMPLEMENTACIÓN DE UN SOFTWARE UTILIZANDO TECNOLOGÍA KINECT COMO APOYO AL PROCESO DE REHABILITACIÓN FÍSICA DE PERSONAS CON PROBLEMAS POSTURALES (RFPOST)

Leonor María Genes Prada

Jesús David Ramírez González

Asesor:
Mario Macea Anaya
Ingeniero de Sistemas

UNIVERSIDAD DE CÓRDOBA
FACULTAD DE INGENIERÍAS
PROGRAMA INGENIERÍA DE SISTEMAS
LORICA – CÓRDOBA
2014

DISEÑO E IMPLEMENTACIÓN DE UN SOFTWARE UTILIZANDO TECNOLOGÍA KINECT COMO APOYO AL PROCESO DE REHABILITACIÓN FÍSICA DE PERSONAS CON PROBLEMAS POSTURALES (RFPOST)

ÁREA DE INVESTIGACIÓN TECNOLÓGICA APLICADA

Leonor María Genes Prada

Jesús David Ramírez González

Asesor:
Mario Macea Anaya
Ingeniero de Sistemas

TRABAJO DE GRADO PARA OBTENER EL TITULO DE INGENIERÍA DE SISTEMAS

UNIVERSIDAD DE CÓRDOBA
FACULTAD DE INGENIERÍAS
PROGRAMA INGENIERÍA DE SISTEMAS
LORICA - CÓRDOBA
2014

HOJA DE ACEPTACIÓN

_____

_____

_____

_____

_____

_____

FIRMA PRESIDENTE DEL JURADO

_____

FIRMA DEL JURADO

_____

FIRMA DEL JURADO

AGRADECIMIENTOS

Nos gustaría que estas líneas sirvieran para expresar nuestro más profundo y sincero agradecimiento a todas aquellas personas que con su ayuda han colaborado en la realización del presente trabajo.

Le damos gracias

A Dios, por llenarnos de muchas bendiciones y fortaleza en nuestro diario vivir.

A nuestros padres que nos brindaron su apoyo en todo momento.

A los ingenieros Mario Macea Anaya por la orientación, el seguimiento y la supervisión continúa de la misma, pero sobre todo por la motivación y el apoyo recibido a lo largo de estos años, Pedro Guevara por enseñarnos las bases necesarias de programación y a Ruben Baena por compartir su conocimiento de bases de datos y asi realizar este proyecto que hoy es no da la oportunidad de subir un peldaño más en nuestras vidas.

A todos los docentes de la universidad de Córdoba que nos orientaron día a día en nuestra formación académica.

A los compañeros de universidad por su amistad y colaboración.

Para triunfar en la vida, no es importante llegar primero. Para triunfar simplemente hay que llegar, levantándose cada vez que se cae en el camino.

DEDICATORIA

Al finalizar mi carrera profesional quiero darles gracias de manera especial a las personas que me apoyaron, con todo respeto y amor dedico este triunfo:

A TI MI DIOS

"porque hiciste realidad este sueño". Por estar junto a mí en cada instante de mi vida, por darme fortaleza para seguir adelante y por haber puesto en mi camino a aquellas personas que han sido mi apoyo y compañía.

A MIS PADRES

ARNEDIS ANTONIO GENES MÓRELO Y PILAR PRADA VELÁSQUEZ ***"son mi vida entera"*** Porque creyeron en mí, porque me sacaron adelante dándome ejemplos dignos de superación y entrega, por su apoyo, sus consejos, su amor, por darme una carrera para mi futuro. Han sido mi inspiración y mi fuerza todos los días de mi vida, me demostraron que el amor de padres es un amor incondicional, gracias por todos los sacrificios que hicieron para que pudiera cumplir mi meta. ***Esta tesis es para ustedes y por ustedes.....***

A MI TÍA

ILUMINADA LÓPEZ gracias por tus oraciones por tu cariño y preocupación y por ser una mujer excepcional a la que quiero muchísimo.

A MI FAMILIA

Hermanos, primos, tíos y abuelos quisiera nombrarlos a todos, pero son muchos, mis palabras no bastarían para agradecerles su cariño y palabras de aliento. ***"Gracias por estar siempre presentes".***

A MIS AMIGOS

Tanto a los que conozco desde antes , como a los que tuve la oportunidad de conocer en la universidad les agradezco a todos el haber llegado a mi vida, por ayudarme cuando lo necesite, por compartir conmigo momentos inolvidables, que quedaran en mi mente y corazón para siempre, en cada uno de ustedes hay alguien especial para mí.

A MI COMPAÑERO DE TESIS

JESÚS RAMÍREZ por brindarme su amistad, y compartir dificultades y triunfos tanto en el proceso de desarrollo de nuestro trabajo de tesis, como en toda la carrera. ***Muchas gracias por tu paciencia y ayuda...***

Leonor María Genes Prada

DEDICATORIA

Este trabajo realizado con mucho esfuerzo pero con mucha pasión, ganas, dedicación, entusiasmo lo dedico primeramente a Dios, quien me ha dado la sabiduría e inteligencia necesaria para desarrollar un trabajo tan arduo como este, y no solo porque me dio la sabiduría si no también la fuerza para aguantar todos esos trasnochos cuando solo sentía su compañía y que escuchaba su voz susurrándome al oído "si puedes, vamos si puedes" porque yo estoy contigo a ese Dios se lo dedico. También se lo dedico a mis padres que me aguantaron todos esos momentos de aburrimiento estrés donde no quería que nadie me interrumpiera porque estaba estudiando para un parcial, haciendo un trabajo de esos bien largos, preparando exposiciones, haciendo reportes de laboratorios entre otras cosas que se me escapan pero algo claro tenia y es que de una forma u otra siempre me apoyaron y me acompañaron en este sueño que hoy se hace realidad, nunca me dijeron no puedes, siempre me brindaron sus mejores herramientas y sus mejores deseos siempre estuvieron conmigo, por eso hoy quiero dedicarle este título, esta meta y demostrarle que no fue en vano todos esfuerzos, cuando se asomaban de madrugada y me veían el computador y colocaban esa cara como de "todavía estudiando" y en mi mente "así es, aun estudiando" a ellos hoy se los dedico.

Por otra parte quiero agradecerles a todos esos compañeros, amigos, profesores que me ayudaron a realizar este trabajo, que me brindaron su apoyo, cada uno haciendo lo que más sabe hacer, luciéndose en su labor, haciéndola en un buen nivel simplemente para ayudar a que este proyecto se hiciera realidad, simplemente gracias por su colaboración.

Por ultimo a todas muchas gracias muchísimas gracias, espero poder ayudarles también en lo que necesiten. GRACIAS!

Jesús David Ramírez González

RESUMEN

La rehabilitación es un proceso sanitario de gran importancia para la correcta recuperación de personas cuya salud lo requiera. La utilización de nuevas tecnologías engloba la posibilidad de obtener nuevos mecanismos que ayuden en este proceso. Concretamente, el uso de interacción en movimiento por medio de los sensores que la habilitan, ofrece grandes ventajas a la hora de mejorar ciertos procesos de rehabilitación. El proyecto tiene como objetivo ayudar en la rehabilitación física de personas con problemas posturales, esto a través de un software (RFPOST), utilizando tecnología Kinect como herramienta de apoyo para dicha rehabilitación, es decir el software proporciona una rutina de ejercicios por cada problema postural (cifosis, escoliosis, rodillas varas, cuello) donde la persona realizara dicha rutina y el software dará un porcentaje de rendimiento de acuerdo al ejercicio o movimiento realizado, también notificara si el paciente está o no haciendo el ejercicio, movimiento o postura correcta. Además le permitirá al médico o fisioterapeuta agregar los datos del paciente con su respectiva historia clínica en una base de datos.

El proyecto va dirigido a los fisioterapeutas para que lo usen como herramienta de apoyo para atender a pacientes que tengan defectos de postura, y a los pacientes para ayudarlos a su recuperación física de una manera más interesante haciendo que el proceso de rehabilitación sea más divertido, provocando en el usuario un nivel de satisfacción mayor que hace que se adhiera mejor a su terapia.

Palabras claves: Defectos posturales, rehabilitación, terapia, ejercicio, movimiento, Kinect.

ABSTRACT

Rehabilitation is a medical process of great importance for the successful recovery of people whose health requires. The use of new technologies includes the possibility of new mechanisms to assist in this process. Specifically, the use of interaction through motion sensors that enable it offers great advantages in improving certain processes of rehabilitation. The project aims to assist in the physical rehabilitation of people with postural problems , this through a software (RFPOST) using Kinect technology as a support tool for such rehabilitation , ie the software provide a workout for each postural problem (kyphosis , scoliosis, rods knees, neck) where the person undertake such routine and give you a percentage of software performance based on the exercise or movement made also notified if the patient is or is not doing the exercise , proper posture or movement . Also allows the physician or physical therapist will add patient data with their respective medical records in a database.

The project is aimed at physiotherapists to use it as a support tool to assist patients with postural defects, and patients to assist them with their physical recovery in a more interesting way making the rehabilitation process more fun, causing the user a higher level of satisfaction that makes better adhere to their therapy.

Keywords: Postural Defects, rehabilitation, therapy, exercise, movement, Kinect.

TABLA DE CONTENIDO

Lista de tablas

Lista de figuras

1. OBJETIVOS

1.1 Objetivo General

Diseñar e implementar un software utilizando tecnología Kinect como apoyo al proceso de rehabilitación física de personas con problemas posturales.

.

1.2 Objetivos Específicos

- Identificar las principales alternativas de la terapéutica de rehabilitación en las personas con problemas posturales.

- Diseñar una aplicación en visual studio 2010 (c#) que interactúe con kinect y permita realizar rutina de ejercicio para corregir o mejorar problemas posturales.

- Crear una Base de Datos que se comunique con aplicación para gestionar los datos del paciente, medico e historia clínica.

2. INTRODUCCIÓN

2.1 Ambientación

La rehabilitación es el tratamiento para recuperar una función del organismo disminuida o pérdida a consecuencia de una lesión o enfermedad, por ejemplo, por medio de masajes y ejercicios.

La mala postura produce un mayor gasto de energía del cuerpo, ya sea cuando éste se encuentra en actividad o en reposo, provocando cansancio y/o dolor. Las personas al tratar de restablecer el equilibrio de sus cuerpos, adoptan nuevas posiciones, ocasionando mayores deformidades, en vez de apaciguar los efectos de una mala postura. Estas deformidades pueden ser incapacitantes desde el punto de vista estético y de orden funcional. Para ello se hace necesario el uso de la rehabilitación física mediante ejercicios que ayuden a mejorar su higiene postural ya que es el medio más efectivo para poder vivir con más plenitud.

Haciendo uso de interacción basada en movimiento es posible crear sistemas que permitan a los pacientes que lo requieran, la posibilidad de llevar a cabo en su hogar el proceso de rehabilitación indicado por el personal médico cualificado. La capacidad de detectar movimiento permite hacer un seguimiento de la actividad del paciente, la cual puede ser analizada. El resultado del análisis permite llevar a cabo acciones tanto de corrección como de recopilación de datos. La corrección se obtiene al comparar cómo se ha de realizar el ejercicio en cuestión con el que realmente ha llevado a cabo el paciente. Si el resultado es correcto el sistema

debe comunicárselo al paciente pero si por el contrario es incorrecto, ha de corregirlo realizando las indicaciones y aclaraciones pertinentes.

Hoy por hoy la tecnología y el uso de ella se encuentra inmersa en todos nuestras actividades, y para este caso no es una excepción, el uso del KINECT como tecnología se convertido en una herramienta poderosa en el campo de la fisioterapia ayudando a las persona en su recuperación física, es decir una rehabilitación de una mala postura causada por diferentes factores, permitiéndole al paciente recuperarse de una forma más atractiva e interesante.

2.2. Problemática

Cada vez más en el área de fisioterapia surgen muchos pacientes con problemas o lesiones en la mecánica postural en diferentes edades, géneros y en profesiones u oficios, esto debido a la adopción de malas posturas al realizar diferentes actividades de la vida diaria, estudiantil y laboral. Existen muchas causas que se asocian a defectos de postura como son: defectos de visión, ropa y calzado impropio que producen presiones o tracciones, dolor que produce actitudes de defensa, pero principalmente los hábitos de postura defectuosa. A diario los individuos se encuentran con diferentes actividades que necesitan mantener el cuerpo en diferentes posiciones estáticas o con carga de peso, y para que estas posiciones no impliquen ningún riesgo, se deben adoptar tomando en cuenta ciertos consejos de higiene de columna para mantener la funcionabilidad del sistema músculo esquelético y de esta manera que el individuo se pueda desempeñar lo mejor posible sin desencadenar padecimientos.

Villacorta y Morales (2010) dicen que:

> Los problemas posturales traen consigo efectos directos sobre la cotidianidad de las personas ya que no permite desarrollar las actividades de la mejor manera inclusive pueden llegar a incapacitar por largo tiempo. Esto conduce a que el biotipo ideal de la persona que es buena postura se aleje cada día más de la realidad.

De modo que lo que se busca con este software es brindar una herramienta de apoyo utilizando tecnología Kinect para la rehabilitación física de las personas con

defectos de postura del municipio de Lorica a través de ejercicios que le ayuden a corregir dicho problema. Por consiguiente a raíz de esta problemática surge la iniciativa de:

DISEÑAR E IMPLEMENTAR UN SOFTWARE UTILIZANDO TECNOLOGÍA KINECT COMO APOYO AL PROCESO DE REHABILITACIÓN FÍSICA DE PERSONAS CON PROBLEMAS POSTURALES.

2.3 Formulación Del Problema

Con base en las situaciones antes mencionadas se procede a plantear el siguiente interrogante: ¿cómo el diseño e implementación de un software utilizando tecnología Kinect puede apoyar al proceso rehabilitación física de personas con problemas posturales de la comunidad de Lorica?

2.4 Antecedentes

Rosario (2005) “afirma que actualmente las Tecnologías de la Información y la Comunicación TICs están sufriendo un desarrollo vertiginoso, esto está afectando a prácticamente todos los campos de nuestra sociedad, y la educación no es una excepción” (p. 100-103). Esas tecnologías se presentan cada vez más como una necesidad en el contexto de sociedad donde los rápidos cambios, el aumento de los conocimientos y las demandas de una educación de alto nivel constantemente actualizada se convierten en una exigencia permanente.

2.4.1 Contexto Internacional

En 2013 Venegas y Alviare desarrollaron en Valparaíso chile una solución que combina “Kinectsiology es el nombre de Kinect, Windows Azure y Skydrive, y que ayuda a médicos kinesiólogos y sus pacientes a hacer el proceso de rehabilitación más interesante, fácil de completar, divertido y preciso” (p. 67).

Zorrilla (2012) desarrollo una aplicación titulada:

Rehabilit-AR elaborada en burgo España. Es una aplicación para mejorar los ejercicios físicos de rehabilitación e informar al paciente sobre su evolución. El programa emplea la tecnología de realidad aumentada, que incorpora datos informáticos a la visión real del entorno. A través de una cámara web se captan los movimientos, generalmente de las extremidades, y el programa comprueba si el movimiento de un brazo, por ejemplo, llega al extremo superior y al extremo inferior que se han marcado previamente como puntos clave para ejecutar un determinado movimiento. El sistema almacena esta información y la puede transformar en gráficos que, con el paso de los días, ofrecen datos sobre la evolución de la persona que lleva a cabo la rehabilitación (p.61).

CTS (1997), desarrollo en Brasil un software llamado:

Fisimetrix. Es un software de Fisioterapia especializado cuyo objetivo es facilitarle al terapeuta su práctica profesional ofreciendo Evaluación Postural - Ortopédica a través de imágenes fotográficas, usando medidas y ángulos y asistiendo en el seguimiento fisioterapéutico. El producto genera informes de exámenes físicos (diagramas de dolor, dermatomos, perimetría, rango de movimiento…), sugerencias y orientaciones posturales y permite grabar un CD o DVD individualizado (en la misma computadora donde está instalado) para que cada usuario practique los ejercicios usando el vídeo en casa (p.75).

2.4.2 Contexto Nacional

En 2013 Henao y López realizaron un proyecto llamado:

Sistema de Rehabilitación basado en el Uso de Análisis Biomecánico y Videojuegos mediante el Sensor Kinect en el instituto tecnológico de Pereira. Es un novedoso sistema para la rehabilitación física de pacientes con múltiples patologías, a través de dinámicas con videojuegos de ejercicio (exergames) y el análisis de los movimientos de los pacientes usando un software desarrollado. Este sistema está basado en el uso del sensor Kinect para ambos fines: divertir al paciente en su terapia a través de exergames y proporcionarle al especialista una herramienta para el registro y análisis de datos de captura de movimiento (MoCap) tomados a través del sensor Kinect y procesados utilizando análisis biomecánico mediante la transformación angular de Euler. Todo el sistema interactivo se encuentra instalado en un centro de rehabilitación y actualmente se realizan investigaciones con diferentes patologías (stroke, IMOC, trauma craneoencefálico, entre otros), los pacientes realizan sus sesiones con el sistema interactivo mientras el especialista registra los datos para un posterior análisis, el cual se realiza en un software creado para dicho fin. El software arroja gráficas de movimiento en los planos sagital, frontal y rotacional de 20 puntos distribuidos en el cuerpo. El sistema final es portable, no-invasivo, económico, de interacción natural con el paciente y de fácil implementación por parte del personal médico (p. 53).

Muñoz (2012) presento un trabajo titulado:

Creación de videojuegos serios para la salud en la Clínica del dolor de Pereira. Atraves de este trabajo, se conocieron sus últimos avances en la hibridación de dos tecnologías: interfaces cerebro-computador y reconocimiento de gestos a través del neurocascoEmotiv EPOC y el popular sensor Kinect. Con ellos consiguen realizar terapias de rehabilitación mediante videojuegos en pacientes con enfermedades neuromotoras, como el accidente cerebro vascular, el síndrome Piramidal y el Parkinson (p. 5).

EAFIT (2010) desarrolla lo sigueinte:

Una aplicación de videojuegos para rehabilitación de pacientes que han sufrido accidentes cerebro-vasculares. La propuesta es proponer un modelo matemático, y su implementación en una aplicación computacional, que permita caracterizar el grado de recuperación que el paciente realiza en su proceso de rehabilitación. Para medir la recuperación se utilizan las medidas que se toman, hasta 100 veces por segundo, utilizando dispositivos como "trackers" electromagnéticos, que permiten detectar la posición y orientación relacionadas con el tratamiento.

2.4.3. Contexto Regional

En el contexto regional no se ha llevado a cabo ningún trabajo que vaya enfocado a la corrección de postura usando la tecnología Kinect.

2.5 Justificación

En la actualidad la mayoría de personas en el mundo laboral y estudiantil realizan diferentes actividades como: escribir, leer, dibujar, etc. con hábitos de postura defectuosos, que desarrollan un desequilibrio músculo esquelético; muchos de estos llegan al punto del dolor agudo y si no son tratados se producen deformidades o desviaciones debido a las posiciones que adoptan las personas para evitar molestias, estas tienen repercusiones a largo plazo en cada individuo afectando la funcionabilidad respiratoria y músculo esquelética**.** Hay que recordar que el concepto salud no solo se refiere al bienestar físico sino que también a un bienestar integral que incluye en mente, espíritu, cuerpo, alma así como la parte social y afectiva, es decir un equilibrio del interior y el exterior del ser humano**.** El presente proyecto será elaborado con la finalidad de ayudar en la rehabilitación física a las personas con defectos de postura del municipio de Lorica a través de un software (RFPOST) con uso de tecnología Kinect que permite realizar ejercicios que le ayuden a mejorar dicho problema. Es de mencionar que no se han realizado estudios sobre este tema en el municipio y así que con este proyecto se mejorara las condiciones físicas de cada persona y se tendrá una buena funcionabilidad del sistema muscular y esquelético de la columna vertebral en equilibrio con el resto de estructuras anatómicas del cuerpo humano durante. La factibilidad de la investigación se basa en que se cuenta con el recurso humano y económico para poder llevarla a cabo.

3. MARCO TEÓRICO

3.1. Postura

Tohen (1970) “dice que a postura es la relación de las posiciones de todas las articulaciones del cuerpo y su correlación entre la situación de las extremidades con respecto al tronco y viceversa” (p. 163). O sea, es la posición del cuerpo con respecto al espacio que le rodea y como se relaciona el sujeto con ella y está influenciada por factores: culturales, hereditarios, profesionales, hábitos (pautas de comportamiento), modas, psicológicos, fuerza, flexibilidad. En pocas palabras la postura es la posición que nuestro cuerpo adopta habitualmente. Cuando estamos sentados, de pie o corriendo adoptamos posturas determinadas.

3.2. Clasificación de la postura

Gattoronchieri (2005) “la postura se ha clasificado en tres grupos” (p.101):

Primer grupo o de buena postura

Es aquel en que las relaciones anatómicas y fisiológicas están dentro de los límites normales.

El segundo grupo o de postura mediana

Solo existen ligeros defectos de postura.

Tercer grupo o mala postura

Es aquel en que hay gran número defectos posturales ligeros, o un defecto postural extremo que puede llegar a la estructuración.

3.3. Defectos de postura

Chávez (2012) “le llama defecto postural a la mala alineación o postura que no puede ser corregida por la voluntad de la persona y es debida a una patología concreta”. La principal causa son las malas posturas, que poco a poco van desviando los huesos de su posición natural.

3.4. Causas de defectos de postura:

1. Hábitos de postura defectuosa en las ocupaciones diarias como escribir, leer, dibujar, tocar piano, el violín, por imitación etc.

2. Mesas, sillas mal adaptadas que no permiten tomar una posición correcta.

3. Defectos de la visión, de la audición o iluminación durante las ocupaciones.

4. Desnutrición y astenia que den debilidad del aparato músculo esquelético.

5. Ropa y calzado impropio que produzcan presiones o tracciones defectuosas.

3.5. Defectos posturales más comunes

Los defectos posturales más comunes son:

Kendall (2006) dice que:

La cifosis es la curvatura de la columna que produce un arqueamiento o redondeo de la espalda, llevando a que se presente una postura jorobada o agachada. Cifosis postural el tipo más común, normalmente atribuida a encorvarse pueden ocurrir tanto en los viejos y los jóvenes. En los jóvenes, se le puede llamar 'encorvarse y es reversible mediante la corrección de los

desequilibrios musculares. En el antiguo, puede ser llamado "hipercifosis" o "joroba de viuda".

.

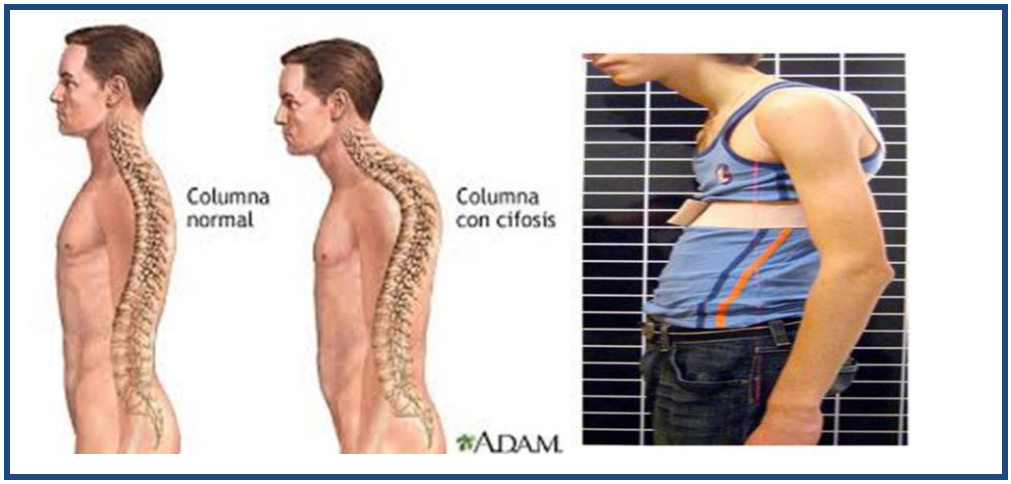


Figura 1. Columna con cifosis.
Fuente: Carambula, P. (2013).*Curvatura en la región dorsal de la columna* [imagen de paciente con cifosis]. Recuperado de http://www.sanar.org/enfermedades/cifosis

La causa de la cifosis postural es fácil de entender: una mala postura repetida puede conllevar una curvatura excesiva de la espalda alta (columna dorsal), dando lugar a lo que comúnmente se denomina como “joroba” o “chepa”. La cifosis postural puede corregirse simplemente con realizar un esfuerzo consciente y sostenido de mantener la espalda erguida y aguantar una buena postura. Por eso se dice que este tipo de cifosis es *flexible;* porque su enderezamiento puede obtenerse a través del simple esfuerzo voluntario de la persona que lo padece. En este caso no existen deformaciones en los huesos de la columna.

La Escoliosis es la deformación de la columna vertebral, que toma una forma como de número 5, hacia la derecha o hacia la izquierda. Este problema se

manifiesta principalmente durante la adolescencia y con mayor incidencia en las mujeres. Las escoliosis tienen su origen durante la infancia por malas posturas y afecta sobre a todo a niños y jóvenes que tienen una marcada debilidad de músculos y ligamentos, de aquí que, con frecuencia, esta afección vaya acompañada de otras debilidades ligamentosas, como el pie plano. Cuando una persona cojea tiene el riego de desarrollar más fácilmente escoliosis.

Figura 2. Columna con escoliosis.

Fuente: [imagen de persona con escoliosis]. Recuperado de http://www.medical-exercise.com/patologias/escoliosis.php

Tobillos se tocan, las rodillas están separadas entre sí, o como si las piernas estuvieran arqueadas hacia afuera.

Figura 3. Rodillas varas (genu varum)

Fuente: [imagen de paciente con rodillas varas]. Recuperado de http://tintito.blogspot.com/2010/02/rodillas-varas.html

Desviación anterior del cuello normalmente hay una ligera desviación hacia adelante del cuello, correspondiendo esto a la parte superior de la curvatura dorsal normal (p.180).

Figura 4. Desviación anterior del cuello.
Fuente: Montoya, M. (2010). *Ergonomía y biomecánica* [imagen de paciente con desviación de cuello]. Recuperado de http://www.slideshare.net/pantufli/tema-8-ergonomia-y-biomecanica

3.6. Cuidados de postura

Postura al sentarse: "Los glúteos deben estar perfectamente reposados al fondo del asiento. La espalda recta y unida al respaldo, que debe ser alto. Los pies apoyados al suelo. Las rodillas un poco más bajas que las caderas" (Areli, 2008)

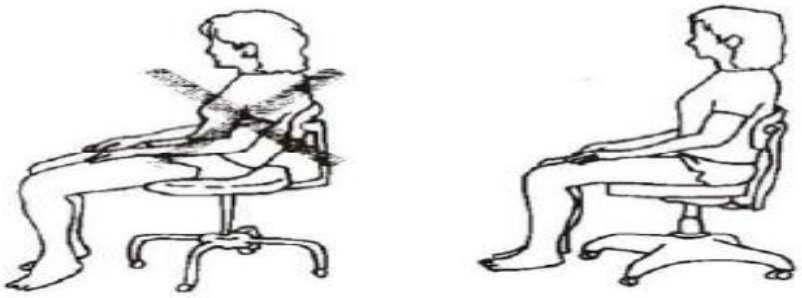


Figura 5. Postura al sentarse
Fuente: Zona hospitalaria de medicina y salud [imagen cuidados de postura al sentarse]. (2008).Recuperado de www.zonahospitalaria.com

Para Guerrero (2013)

La postura al escribir sobre una mesa, desde la posición de sentado, procurar que el tronco, al flexionarlo, se apoye con el borde de ésta, no cruzar las piernas cuando se está sentado, en su lugar, se puede cruzar los pies por los tobillos, no sentarse dando la espalda al aparato de aire acondicionado o a la calefacción.

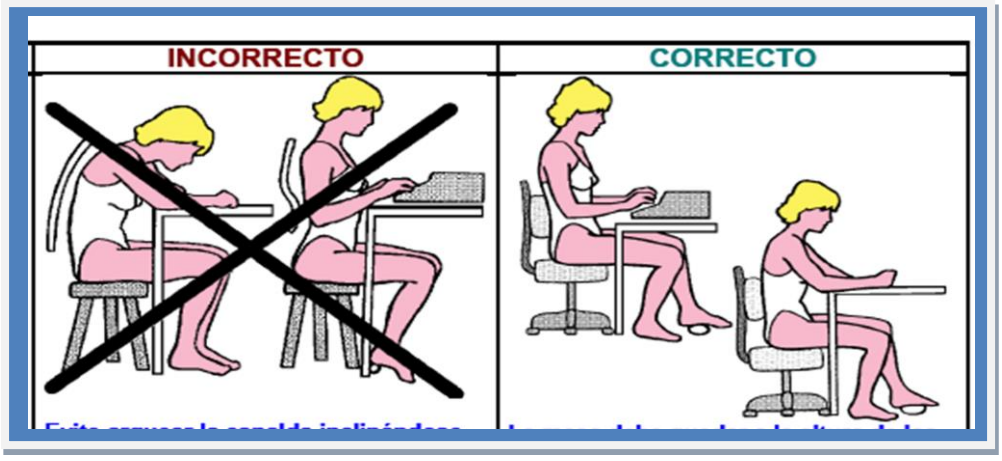


Figura 6. Postura al escribir.
Fuente: Zona hospitalaria de medicina y salud [imagen cuidados de postura al escribir]. (2008).Recuperado de www.zonahospitalaria.com

Para Duran (2008)

No es recomendable conducir con la espalda muy inclinada hacia atrás, en extensión, porque se obliga a llevar el cuello flexionado). Es conveniente llevar un apoyo para la cabeza para evitar el efecto del "latigazo" que puede ocasionar fracturas en vértebras de la columna cervical, e incluso lesiones medulares. El latigazo es muy frecuente en accidentes de coche o frenazos, pues al frenar, el cuerpo, por energía cinética se va hacia delante y luego hacia detrás. En contraposición, tampoco conducir pegado al volante. Para entrar en un coche, primero, sentarse y luego meter las piernas. Y para salir se ha de realizar la operación contraria.

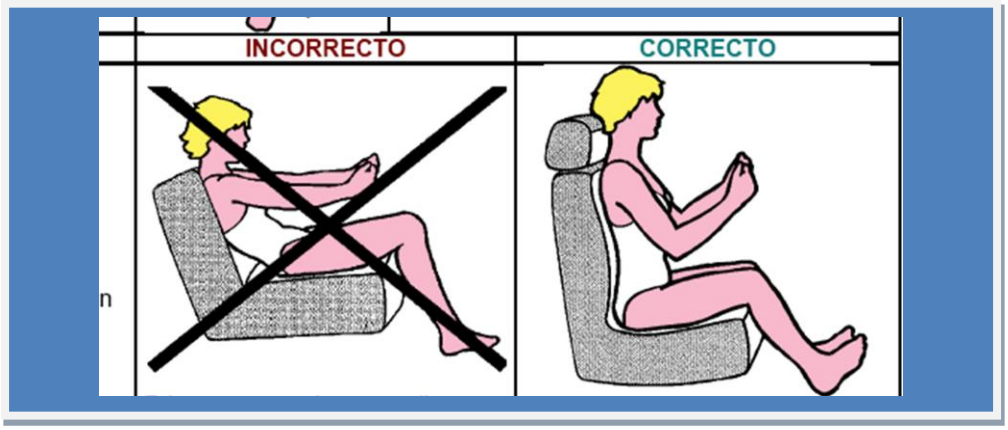


Figura 7. Postura al conducir
Fuente: Zona hospitalaria de medicina y salud [imagen cuidados de postura al conducir]. (2008).Recuperado de www.zonahospitalaria.com

Para Valencia (2011)

Al elevar un peso desde el suelo se debe, flexionar las piernas y el tronco recto. Así, el esfuerzo se realiza con las piernas y los brazos.

Evite cargar pesos de un solo lado, procure distribuirlos en dos volúmenes, uno de cada lado. Nunca cargar objetos más pesados de los que la persona pueda cargar. Evitar movimientos bruscos. Girar todo el cuerpo para alcanzar un objeto que se encuentre a un lado o atrás de las personas.

Figura 8. Postura al levantar pesos
Fuente: Zona hospitalaria de medicina y salud [imagen cuidados de postura al conducir]. (2008).Recuperado de www.zonahospitalaria.com

3.7. Actividad física para la rehabilitación de los defectos de postura

García (2012) Declara:

Actividad física es la realizada espontáneamente o bajo dirección especializada, en el tiempo libre o en el tiempo profesional, dirigida a la formación y desarrollo de hábitos y habilidades de movimiento corporal de variados niveles de complejidad y el desarrollo de capacidades físicas, coordinativas y mentales que contribuyen al desarrollo individual y social; la formación integral, la satisfacción espiritual y la salud de las personas que la realizan.

En el proceso de rehabilitación de personas con deformidades posturales pone de manifiesto que mediante el ejercicio físico sistemático y dosificado, atendiendo a las limitaciones individuales de cada uno, se producen cambios que pueden

restablecer y reincorporar al hombre a la sociedad. Además de prevenir cualquier alteración que afecte tanto la salud como a la estética del cuerpo humano.

La realización de las actividades con ejercicios físicos terapéuticos, influyen positivamente en la rehabilitación física de las deformidades posturales escoliosis, logrando corregir dichos padecimientos.

3.8. Rol de las tecnologías en la medicina

La informática médica nació en 1959 con un artículo científico en la revista *Science*. En 1963 se instalaron las primeras computadoras para uso clínico en centros médicos y laboratorios. En los años setenta se implantaron los sistemas diagnósticos, y en el año 2003, la secuenciación del genoma humano inició la unión de ambas ramas médicas: la información clínica con la información genómica. El resultado fue la medicina traslacional.

El papel del futuro de las TIC en medicina todavía está por definir, y dependerá del uso y del valor que se les dé. “La tecnología a través de comunidades virtuales favorece la comunicación, la interrelación profesional y el intercambio de conocimientos” (Saluspost, 2013). Además, los sistemas de información integran diversos datos clínicos en todas sus dimensiones, mejorando así la eficiencia del sistema. De hecho, gracias a las TIC, el sistema de salud puede reducir costes, como por ejemplo en los pacientes crónicos.

4. MARCO CONCEPTUAL

4.1. Sensor Kinect

Kinect (originalmente conocido por el nombre en clave «Project Natal»), es «un controlador de juego libre y entretenimiento» creado por Alex Kipman, desarrollado por Microsoft para la videoconsola Xbox 360, y desde junio del 2011 para PC a través de Windows 7 y Windows 8. Kinect permite a los usuarios

controlar e interactuar con la consola sin necesidad de tener contacto físico con un controlador de videojuegos tradicional, mediante una interfaz natural de usuario que reconoce gestos, comandos de voz, y objetos e imágenes.

“Kinect es un dispositivo, inicialmente pensado como un simple controlador de juego, que gracias a los componentes que lo integran: sensor de profundidad, cámara RGB, array de micrófonos y sensor de infrarrojos (emisor y receptor), es capaz de capturar el esqueleto humano, reconocerlo y posicionarlo en el plano” (Murillo, 2012).

Figura 9. Sensor Kinect
Fuente: Rehabilitación con Kinect [imagen controlador kinect]. (2012). Recuperado de http://www.vidaextra.com/juegos/accesorios/kinect

La comunidad “Kinesis” nos provee con un SDK, el cual es una librería Javascript, que nos permite desarrollar aplicaciones de escritorio o Web basadas en gestos gracias al Kinect, esta funciona con Xbox y Windows Kinect.

Componentes del Kinect

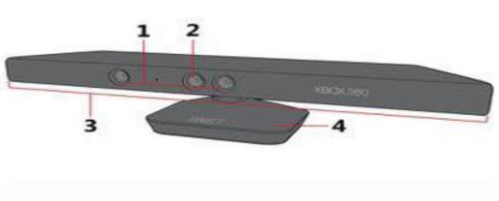


Figura 10. Componentes del Kinect.
Fuente: Roriguez, A. (2013). *Sistema de entrenamiento con kinect*[imagen componentes del kinect]. Recuperado de http://upcommons.upc.edu/pfc/bitstream/2099.1/18626/1/85859.pdf

1. Sensores de profundidad.

Son el elemento clave del funcionamiento de la Kinect, y son los encargados de hacer el seguimiento del cuerpo. Se basan en dos cámaras de infrarrojos que construyen un mapa de profundidad.

2. Cámara RGB.

La cámara RGB (rojo, verde, azul) de la Kinect. Es del tipo CMOS, trabaja por defecto de **640x480 a 30fps**.

3. Micrófono multi matriz

En la parte delantera del sensor kinect hay una serie de micrófonos que se usan para el reconocimiento de órdenes y charla. No son necesarios de cara al proyecto.

4. Inclinación motorizada

Una unidad mecánica en la base del sensor de kinect lo inclina automáticamente hacia arriba o hacia abajo según sea necesario. Esta característica, aunque útil, tampoco es vital para el desarrollo del proyecto.

4.2. Skeletal Tracking

“Es la funcionalidad estrella del sensor Kinect, significa seguimiento de esqueleto y se basa en un algoritmo que logra identificar partes del cuerpo de las personas que están en el campo de visión del sensor” (esmsdn, 2011). Por medio de este algoritmo podemos obtener puntos que hacen referencia a las partes del cuerpo de una persona y hacer un seguimiento de éstos identificando gestos y/o posturas.

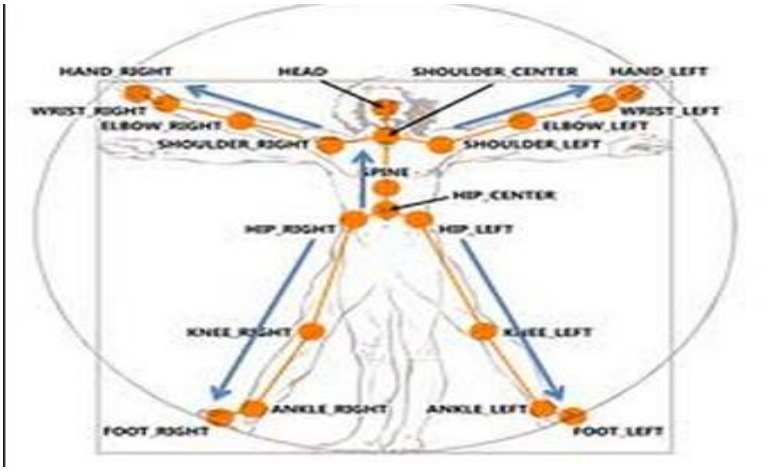


Figura 11. Skeletal tracking
Fuente: Kigo el robot clasificador de residuos [imagen algoritmo skeletal tracking]. (2011). Recuperado de http://lsedkigo.blogspot.com/2012/05/integracion-de-kinect-parte-i.html

4.3. Kinect para windows sdk beta 2

El SDK permite a una creciente comunidad de desarrolladores, investigadores académicos y entusiastas crear nuevas experiencias que incluyen detección de profundidad, monitoreo de movimientos humanos y reconocimiento de voz y objetos con el uso de la tecnología Kinect en Windows 7.

"El SDK de Kinect para Windows, que funciona con Windows 7, incluye controladores, APIs integrales para flujos de sensor no procesados, interfaces de usuario naturales, documentos de instalación y materiales de recursos" (Ruiz, 2010). El SDK brinda capacidades Kinect a los desarrolladores que generan aplicaciones con C++, C# o Visual Basic con el uso de Microsoft Visual Studio 2010.

4.4. Dynamic Time Warping (DTW)

Esta técnica permite reconocer patrones lineales en el tiempo que se produzcan a velocidades diferentes a la velocidad en la que fueron capturados. Estos sistemas tienen la principal ventaja de que permiten un porcentaje de acierto muy alto aun variando la velocidad de realización del gesto. Usando esta técnica se pueden

definir en tiempo de ejecución las posturas y los gestos, grabándose a uno mismo y después realizando la validación al realizar el gesto directamente sobre el software.

4.4.1 Kinect Sdk Dynamic Time Warping (Dtw) Gesture Recognition

“Este proyecto permite un reconocimiento de gestos en Microsoft Kinect SDK proyectos de C #. Utiliza seguimiento esquelético y actualmente soporta vectores 2D” (codeplex, 2011).

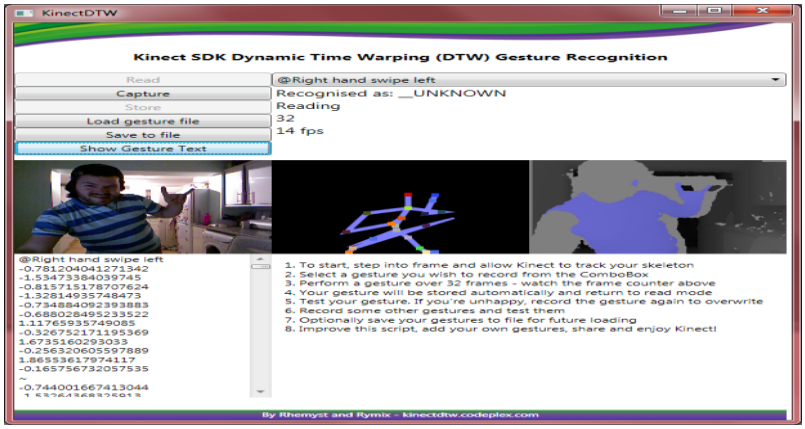


Figura 12. Kinect sdk DTW
Fuente: Rymix, A. (2011).*Gesture Recognition Dynamic Time Warping Kinect* [imagen interfaz SDK dtw].Recuperado de http://kinectdtw.codeplex.com/

4.5. Visual Studio 2010

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta multiples lenguajes de programación tales como C++, C#, Visual Basic .NET, F#, Java, Phyton, Ruby, php; al igual que entornos de desarrollo web como ASP.NET MVC, Django, etc.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos, consolas (la xbox 360 y xboxone), etc (Wikipedia, 2013).

4.6. Lenguaje De Programación C#

C# (pronunciado *si sharp* en inglés) es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. Su sintaxis básica deriva de C/C++ y utiliza el modelo de objetos de la plataforma .NET, similar al de Java, aunque incluye mejoras derivadas de otros lenguajes.

Aunque C# forma parte de la plataforma .NET, ésta es una API, mientras que C# es un lenguaje de programación independiente diseñado para generar programas sobre dicha plataforma. Ya existe un compilador implementado que provee el marco Mono -DotGNU, el cual genera programas para distintas plataformas como Windows, Unix,Android, iOS, Windows Phone, Mac OS y GNU/Linux (Wikipedia, 2012).

4.7. Windows Presentation Foundation (Wpf)

Windows Presentation Foundation (WPF) permite el desarrollo de interfaces en Windows con los lenguajes de programación .NET. Ofrece una amplia

infraestructura y potencia gráfica con la que es posible desarrollar aplicaciones visualmente atractivas. Permite integrar elementos de animación, vídeo, imágenes y audio de manera sencilla, cosa que viene bien para el desarrollo de este proyecto.

WPF utiliza el lenguaje XAML. La ventaja específica que XAML lleva a WPF es que XAML es un lenguaje completamente declarativo. En un lenguaje de programación declarativa, el desarrollador (o diseñador) describe el comportamiento y la integración de los componentes sin utilizar programación procedural. Aunque es raro que una aplicación completa se construya totalmente en XAML, la introducción de XAML permite a los diseñadores de aplicaciones contribuir más eficazmente al ciclo de desarrollo de aplicaciones (Wikipedia, 2013). Aspectos legales (Ver anexo B).

4.8. Factores influyentes en la base de datos

xampp es un servidor independiente de plataforma, software libre, que consiste principalmente en la base de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script: PHP y Perl, Mysql-connector-net 6.8.3, permite añadir un archivo .dll en visual studio para la establecer la conexión a mysql. itexSharp, permite exportar o crear un pdf desde el código c#.

5. METODOLOGÍA

5.1 Fases del proyecto

El proceso de la investigación de este proyecto, se realizó teniendo en cuenta la secuencia de las siguientes fases de desarrollo.

Fase I: Investigación

Actividades

- Recolección de información relacionada con los principales temas de investigación como: Tecnología kinect, Windows sdk para kinect, reconocedor de gestos y herramientas de uso de las mismas.

- Organización y clasificación de proyectos e investigaciones en diversos campos de aplicación recolectada a nivel nacional e internacional.

- Análisis de ejercicios que tienen como objetivo mejorar o corregir los problemas posturales.

Fase II: Elaboración

Actividades

- Especificación de requisitos.
- Diseño del prototipo inicial del dispositivo.
- Diseño de la arquitectura del sistema.
- Realización de diagramas UML correspondientes al sistema.
- Recopilar los distintos ejercicios para cada problema postural.
- Guardar los distintos ejercicios para cada problema postural en un archivo de texto.

Fase Ill: Diseño Y Desarrollo Del Sistema

Actividades

- Diseño de clases.
- Utilización del dispositivo Kinect.
- creación de archivos de persistencias (.txt).
- Programación del software
- Integración del código.
- Prueba inicial del software.

Fase IV: Transición

- Realización de pruebas finales, que permitan evaluar el rendimiento y óptimo funcionamiento del dispositivo.
- Elaboración de encuestas para verificar la aceptación del sistema propuesto.
- Documentación del proyecto.
- Entrega Final.

5.2 Metodología de desarrollo del producto

Figura 13. Metodología RUP

Fuente: [imagen de metodología RUP]. Recuperado de http://www.jlaya.com/?p=204

Para el desarrollo del producto se utilizó la metodología RUP(*Rational Unified Process*) “es un proceso de desarrollo de software que junto con el Lenguaje Unificado de Modelado UML, constituye la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos” (Wikipedia).

Inicio:

5.2.1. Resultados de encuesta aplicada a la población de estudio.

Se realizó una encuesta inicial, a un total de 20 personas que presentaban algún tipo de problema postural.

Esta encuesta permitió dar respuesta al primer objetivo específico planteado en esta investigación. De acuerdo a la información obtenida se generaron las siguientes conclusiones descritas de forma gráfica. (Ver anexo C)

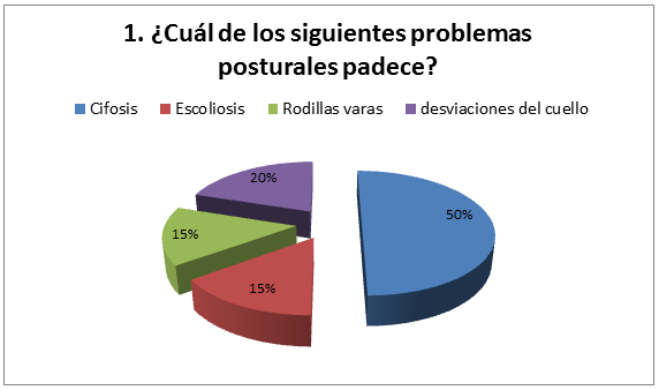


Figura 14. Gráfica de resultado a la pregunta 1 de la encuesta inicial.

El 50% de las personas encuestadas padecen cifosis, lo que quiere decir que es el defecto postural más predomínate en la población, un 15% tiene escoliosis, un 15% sufre de rodillas varas y un 20% presenta desviaciones del cuello.

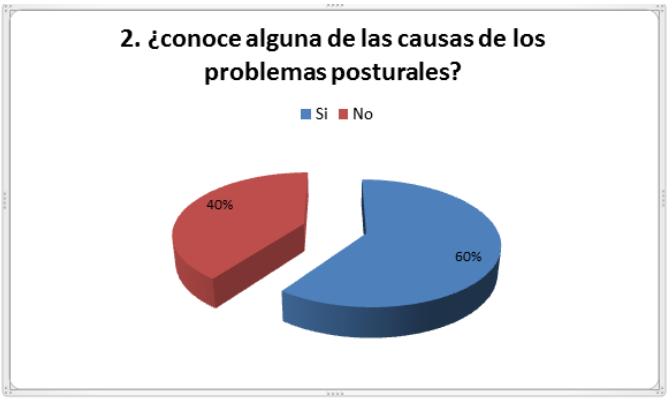


Figura 15. Gráfica de resultado a la pregunta 2 de la encuesta inicial.

Se puede evidenciar que la mayoría de las personas encuestadas conocen las causas que provocan los problemas posturales.

Elaboración:

5.2.2. Especificación de requisitos

5.2.2.1. Funcionales

A continuación se detallan algunos requisitos funcionales para los casos de uso que se describirán más adelante:

Tabla 3. Requisitos Funcionales

REFERENCIA	REQUERIMIENTOS	PRIORIDAD		
		Alta	media	baja
REC 001	El sistema debe permitir al usuario ingresar digitando un respectivo usuario y contraseña.	**X**		
REC 002	El sistema debe habilitar las opciones de los problemas posturales.			
REC 003	El sistema debe permitir al usuario seleccionar el problema postural a tratar.	**X**		
REC004	El sistema debe mostrar las opciones con los ejercicios correspondientes al problema postural.	**X**		
REC 005	El sistema debe permitir al usuario escoger el ejercicio realizar.	**X**		
REC 006	El sistema debe mostrar a través de imágenes las indicaciones de cómo se realiza el ejercicio.	**X**		
REC007	El sistema debe permitir digitar manualmente el tiempo y numero de rutinas del ejercicio en seg.	**X**		
REC 008	El sistema debe reconocer si el usuario esa realizando el ejercicio	**X**		
REC 009	El sistema debe mostrar al usuario un progreso del ejercicio realizado en porcentaje.	**X**		
REC 010	El sistema debe permitir ingresar un paciente.		**X**	
REC 011	El sistema debe permitir actualizar los datos un paciente.		**X**	
REC 012	El sistema debe permitir eliminar los datos un paciente.		**X**	
REC 013	El sistema debe permitir consultar los datos un paciente.		**X**	
REC 014	El sistema debe permitir ingresar una historia clínica.		**X**	
REC 015	El sistema debe permitir actualizar una historia clínica.		**X**	

REC 016	El sistema debe permitir eliminar una historia clínica.		**X**	
REC 017	El sistema debe permitir consultar una historia clínica.		**X**	
REC 018	El sistema debe permitir generar un reporte de una historia clínica		**X**	
REC 019	El sistema debe permitir ingresar los datos de un médico.		**X**	
REC 020	El sistema debe permitir actualizar los datos de un médico.		**X**	
REC 021	El sistema debe permitir eliminar los datos de un médico.		**X**	
REC 022	El sistema debe permitir consultar los datos de un médico.		**X**	

5.2.2.2. No funcionales

Se refieren a todos los requisitos que ni describen información a guardar, ni funciones a realizar. Son a menudo llamados las cualidades de un sistema.

- Laptops
- sistemas operativos Windows 7.
- Consola X-box 360 (Kinect)
- Un espacio físico no reducido, debe ser un área mínimo de 4m2
- Xammp
- Visual Studio

5.2.3. Arquitectura del sistema.

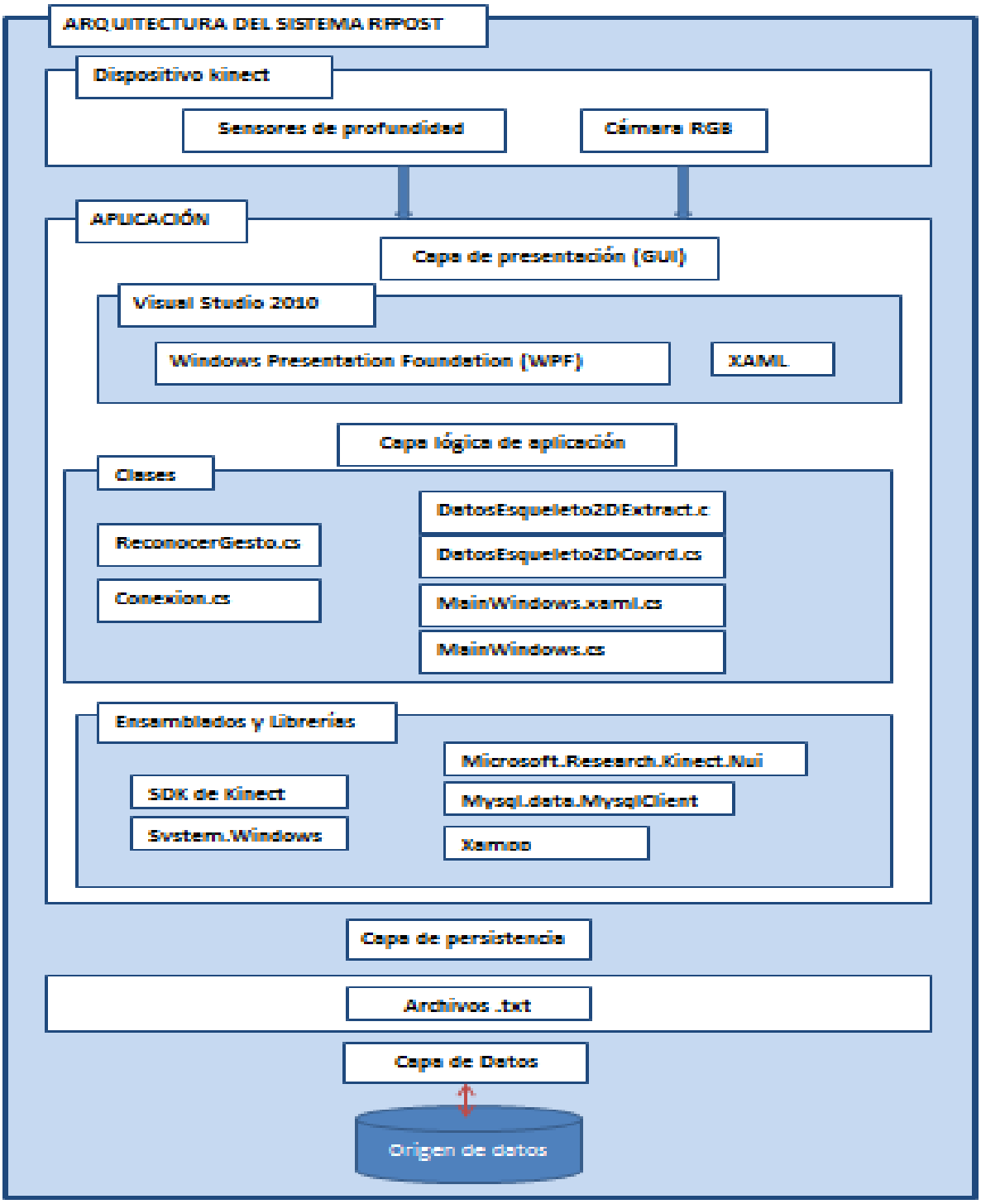


Figura 23. Arquitectura del sistema

La figura 23 muestra la arquitectura del sistema RFPOST, especificando en primera medida los componentes que hacen parte del dispositivo que se comunicará con la aplicación. En éste, se encuentra los sensores de profundidad que son los encargados de hacer el seguimiento del cuerpo, se basan en dos cámaras de infrarrojos que construyen un mapa de profundidad y la cámara RGB (rojo, verde, azul) de la Kinect es del tipo CMOS, trabaja por defecto de **640x480 a 30fps.** A su vez, se muestra el desarrollo de la aplicación basada en un modelo de arquitectura de capas a razón de establecer independencia entre cada uno de los componentes de la aplicación, representados a través de 3 capas:

Capa de presentación (GUI)

Capa lógica

Capa de persistencia

La primera capa llamada presentación, corresponde a la interfaz de usuario construida especialmente en visual studio 2010 (C#) usando que WPF ofrece una amplia infraestructura y potencia gráfica con la que es posible desarrollar aplicaciones visualmente atractivas, con facilidades de interacción que incluyen animación, vídeo, audio, documentos, navegación o gráficos 3D. Separa, con el lenguaje declarativo XAML .

La segunda capa concierne a la capa lógica de la aplicación, en ella se define cada una de las clases desarrolladas para el manejo de reconocer gestos, extraer datos de las coordenadas de las distintas partes del cuerpo y de la conexión con la base de datos. Además de los sdk de Kinect y los ensamblados que se usaron.

La capa de persistencia, se enfoca en almacenar cada uno de las de rutas de los diferentes archivos .txt que se utilizaran en la aplicación.

De igual forma es necesario aclarar la necesidad de capa de datos requerida para el almacenamiento de datos pertinentes al paciente, medico e historia clínica.

5.2.3.1 Funcionamiento del sistema

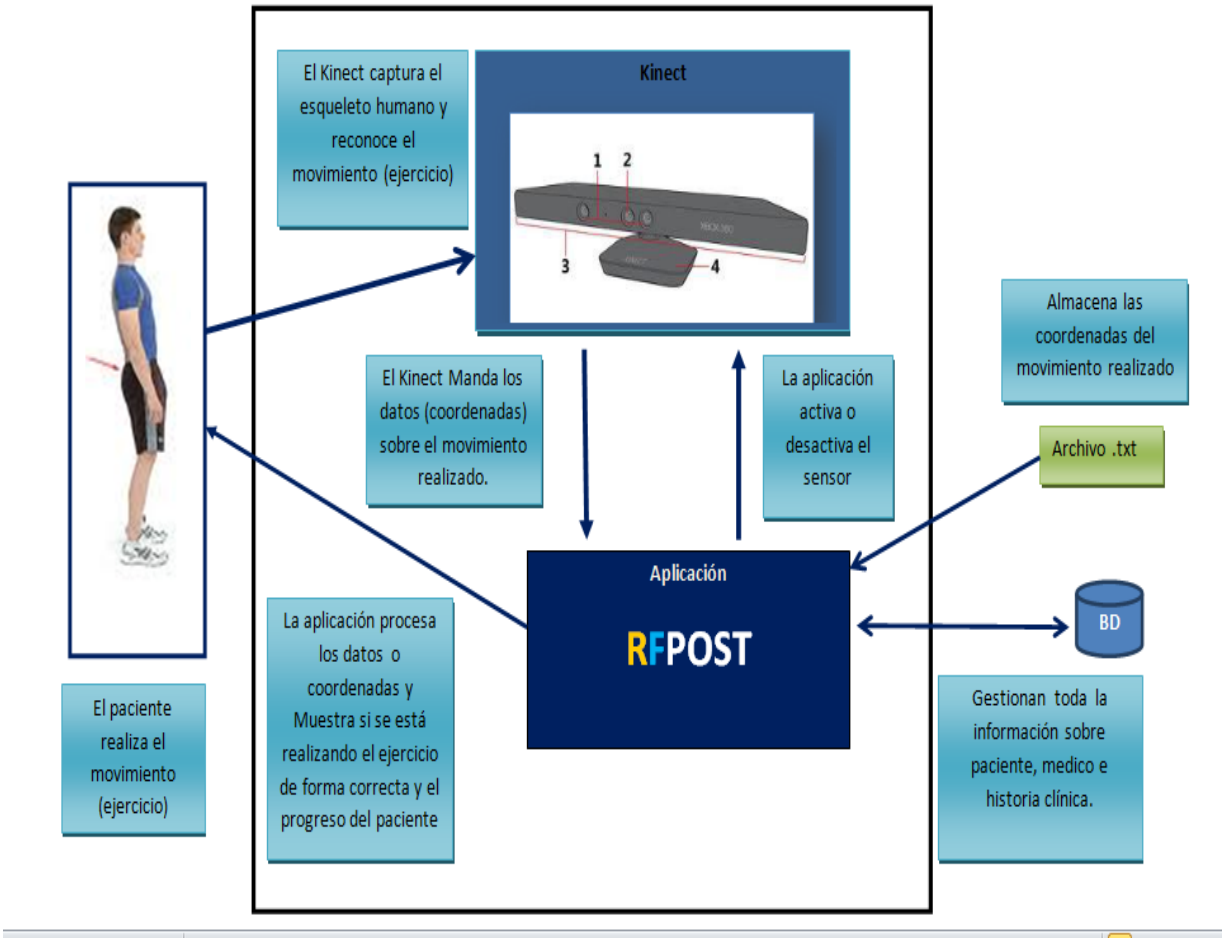


Figura 24. Funcionamiento del sistema

5.2.4. Diagramas del sistema

Para entender mejor todas las funciones que se pueden realizar con este sistema de información, los elementos y objetos de los cuales está compuesta se han realizado los diagramas UML. A continuación se detallan brevemente las características de la aplicación en base a estos diagramas.

Descripciones de los casos de uso (Ver anexo D).

5.2.4.1. Diagrama de casos de uso

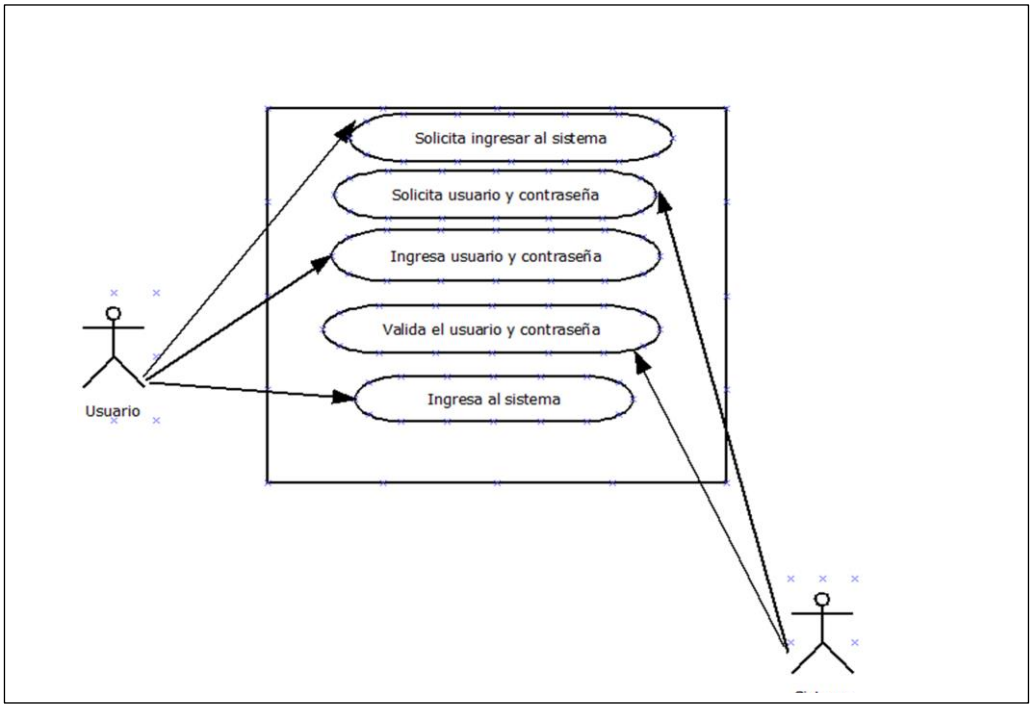


Figura 25. Caso de uso ingresar al sistema

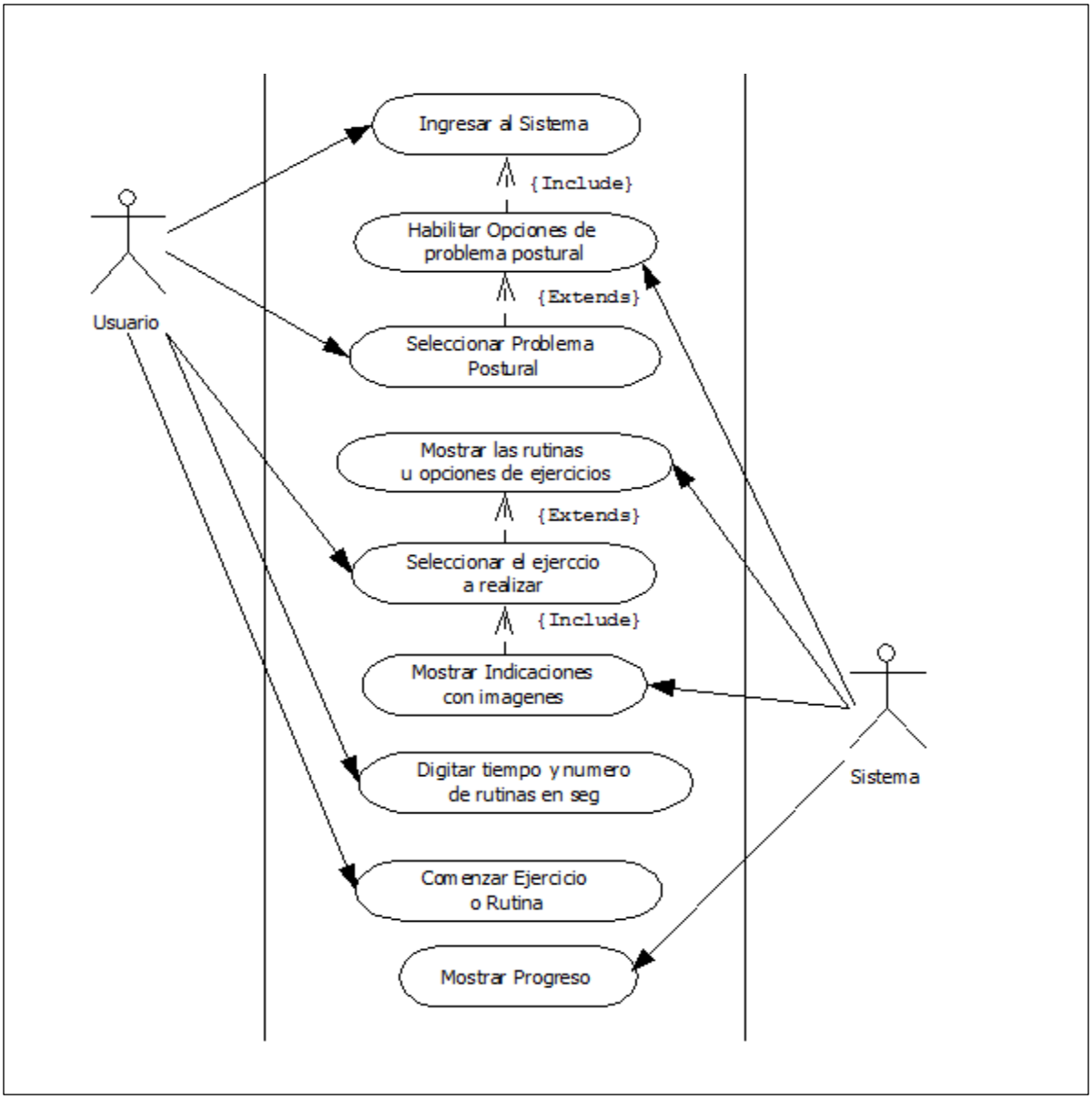


Figura 26. Caso de uso realizar ejercicio

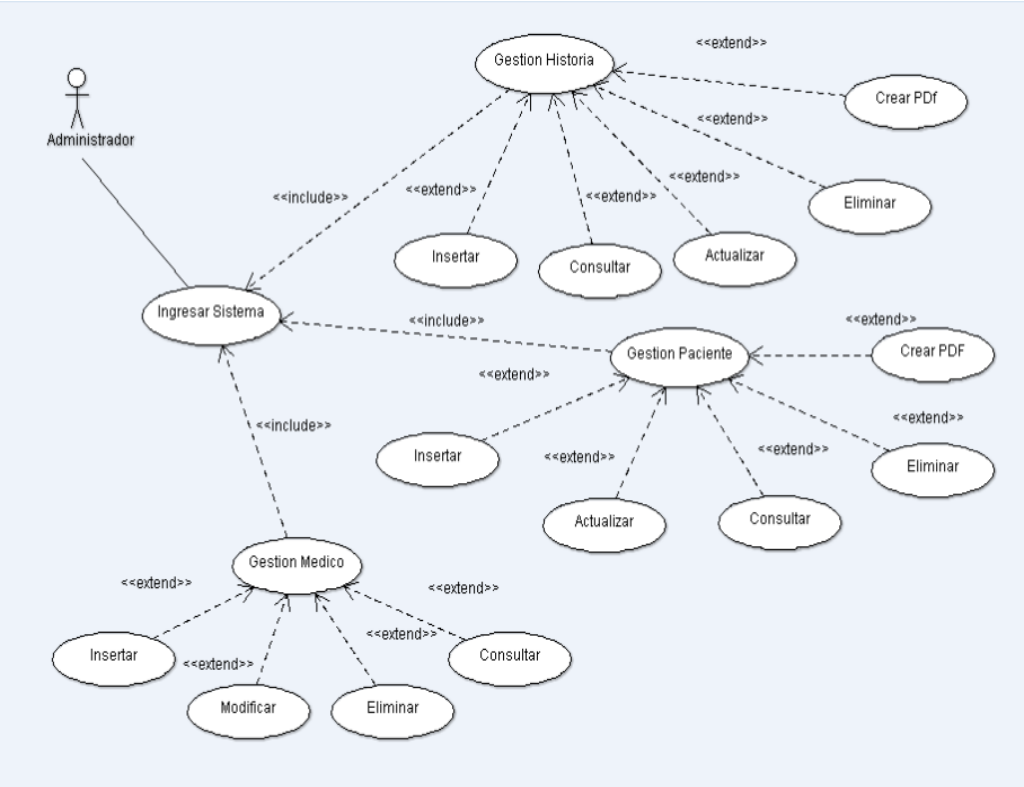


Figura 27. Caso de uso administrador

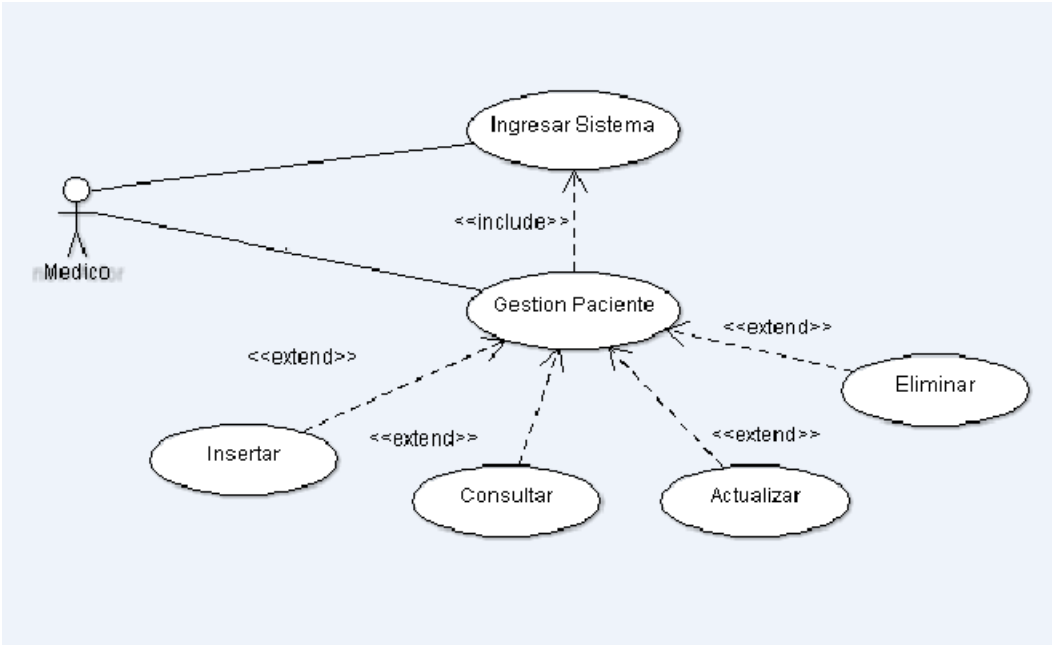


Figura 28. Caso de uso gestión paciente

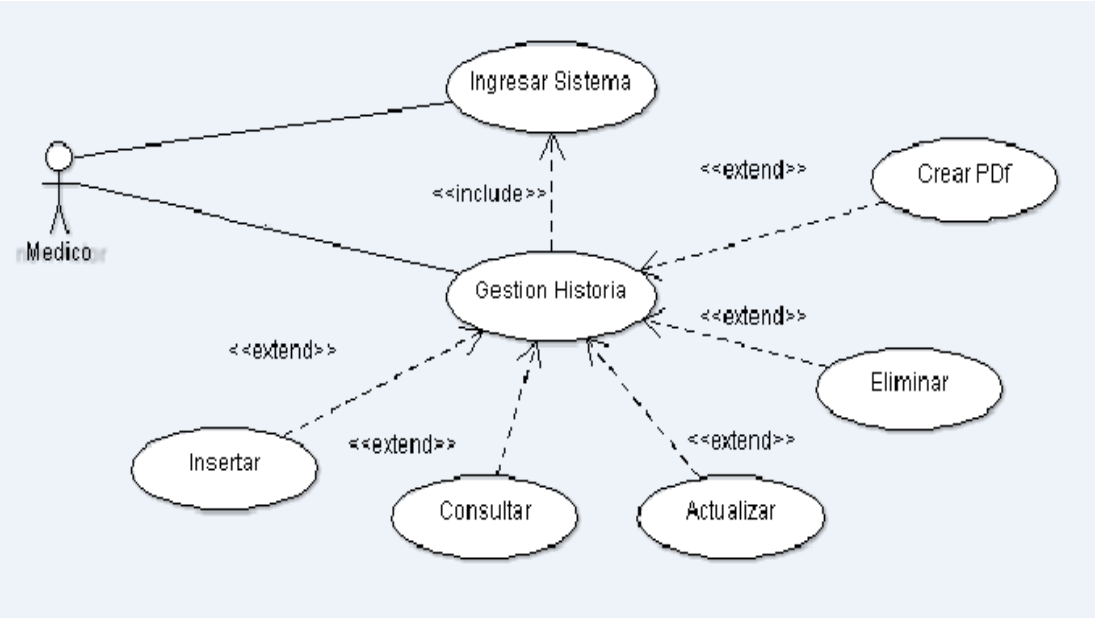


Figura 29. Caso de uso gestión historia

5.2.4.2. Diagramas de secuencia

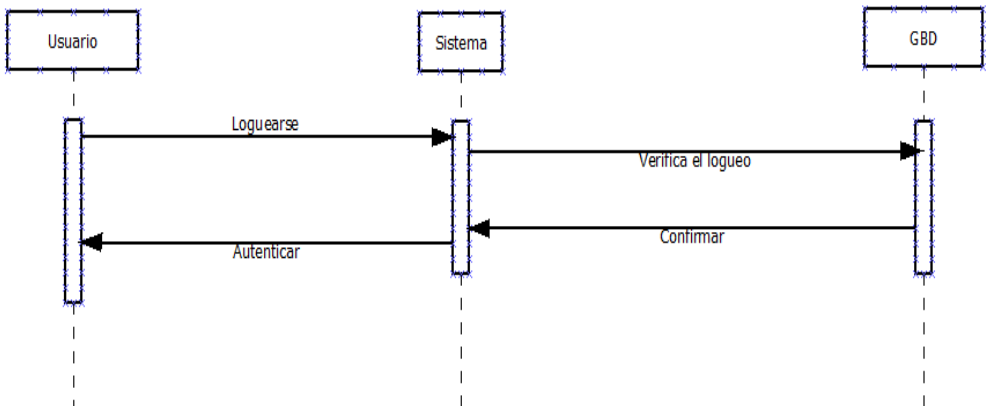


Figura 30. Diagrama de secuencia ingresar al sistema

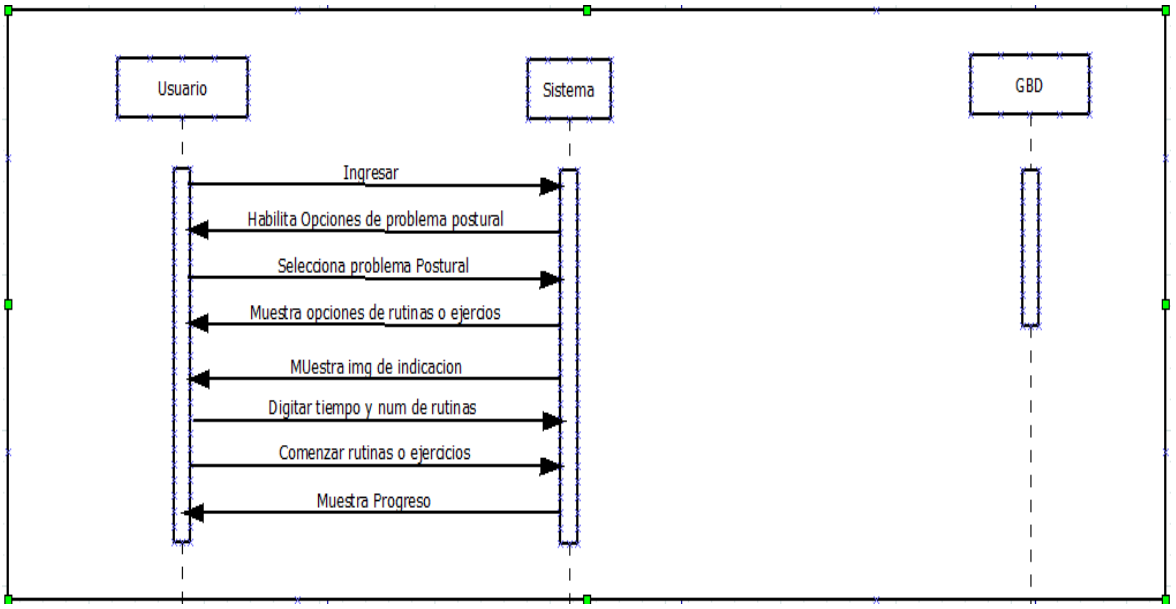


Figura 31. Diagrama de secuencia realizar ejercicio

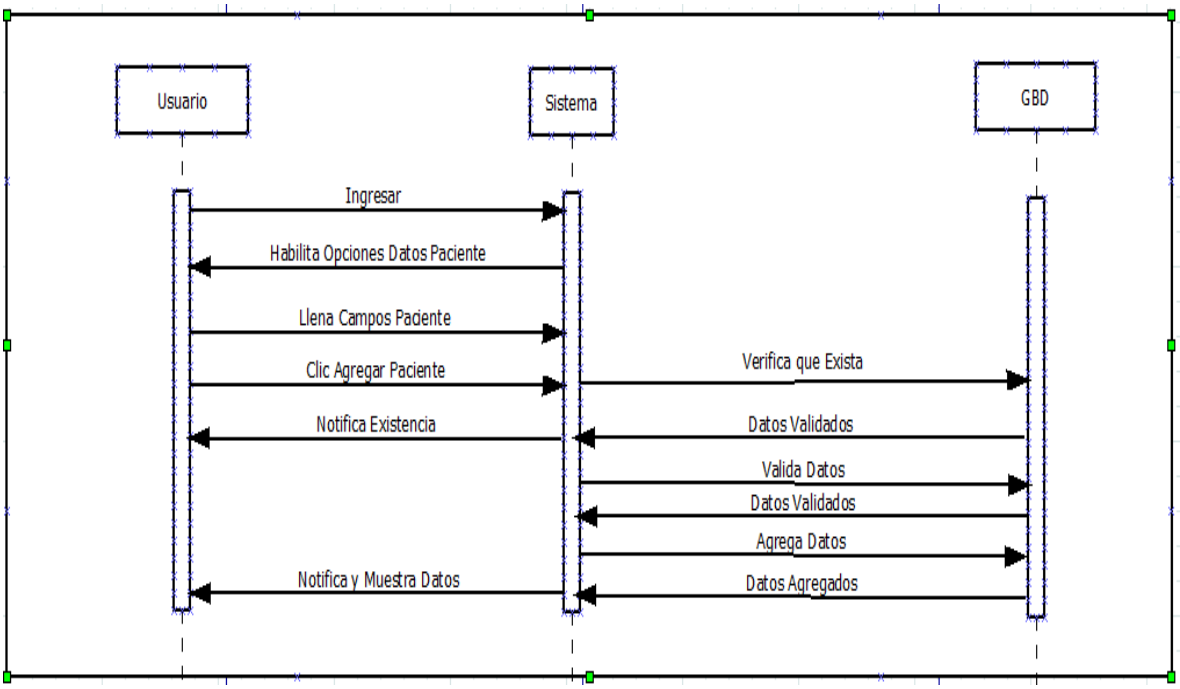


Figura 32. Diagrama de secuencia ingresar paciente

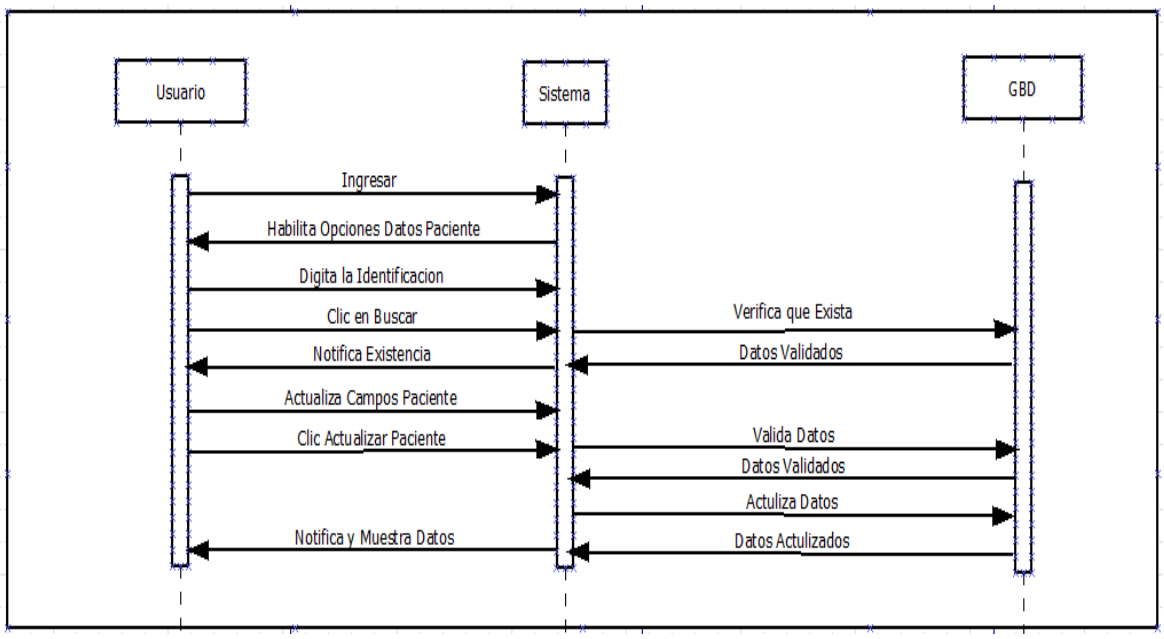


Figura 33. Diagrama de secuencia actualizar paciente

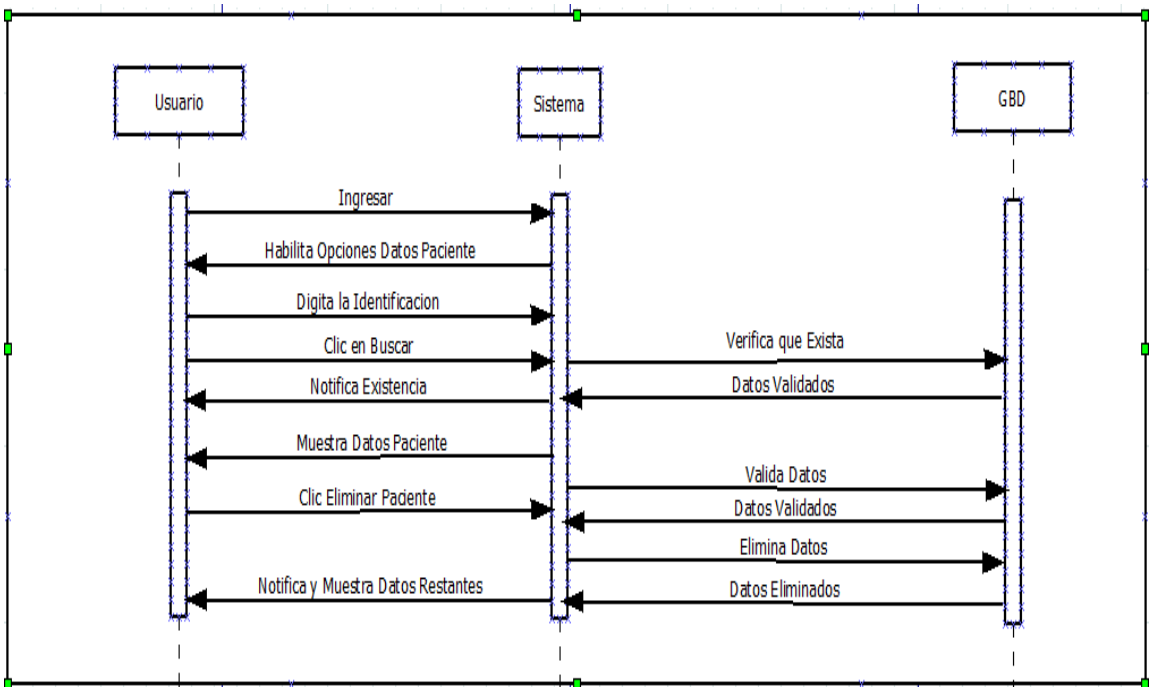


Figura 34. Diagrama de secuencia eliminar paciente.

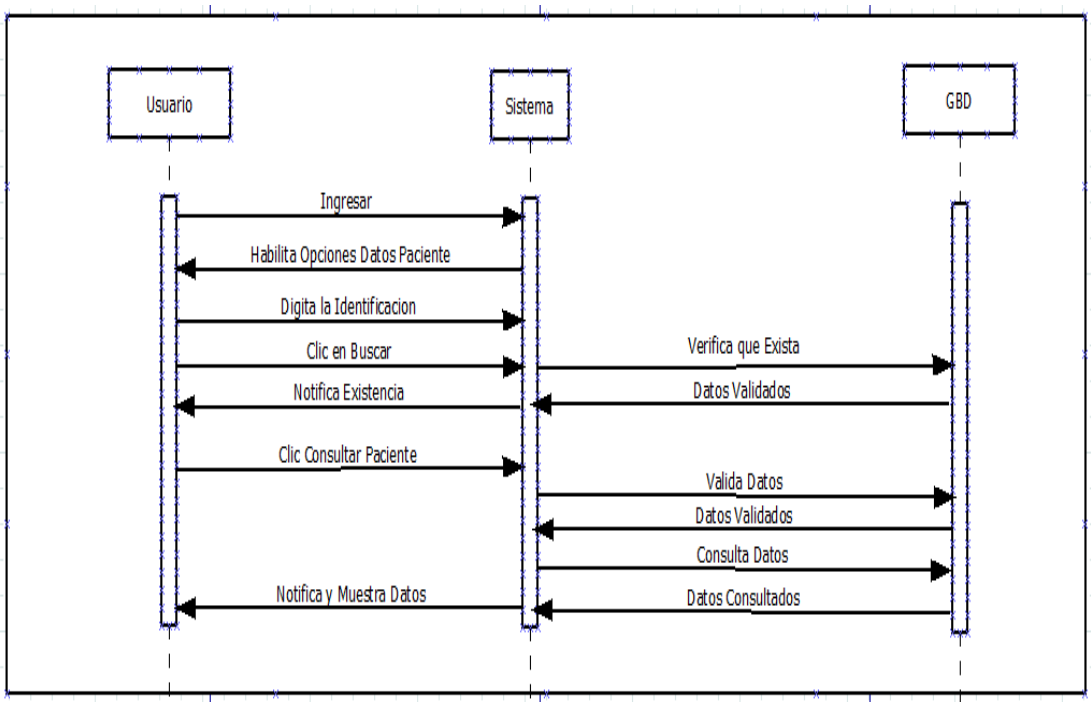


Figura 35. Diagrama de secuencia consultar paciente.

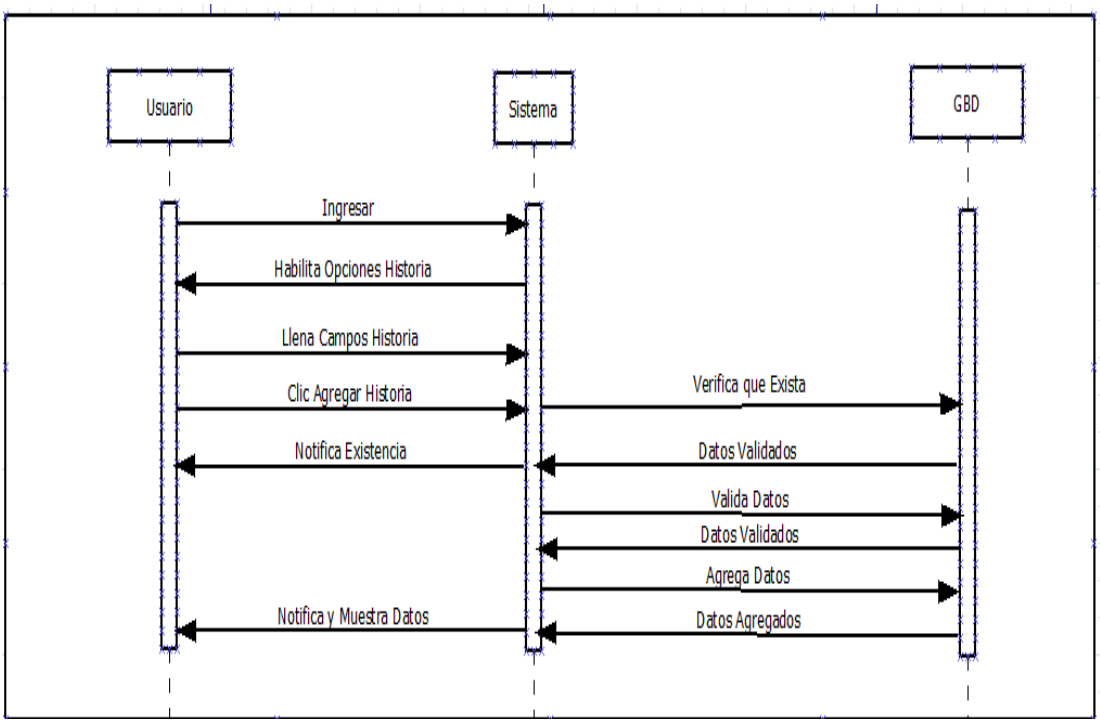


Figura 36. Diagrama de secuencia agregar historia.

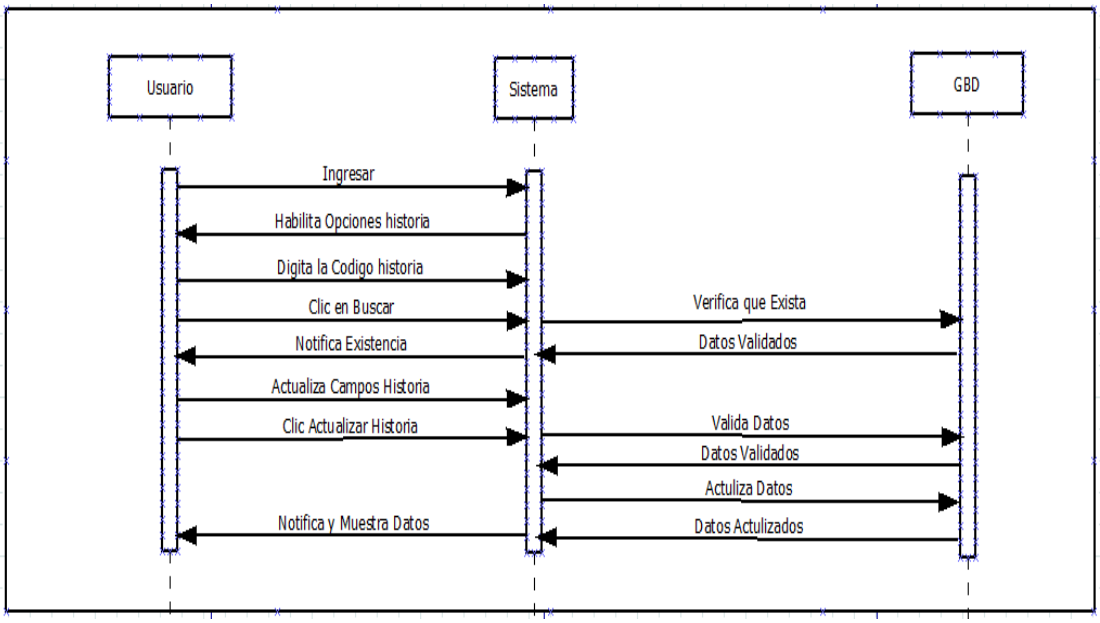


Figura 37. Diagrama de secuencia actualizar historia.

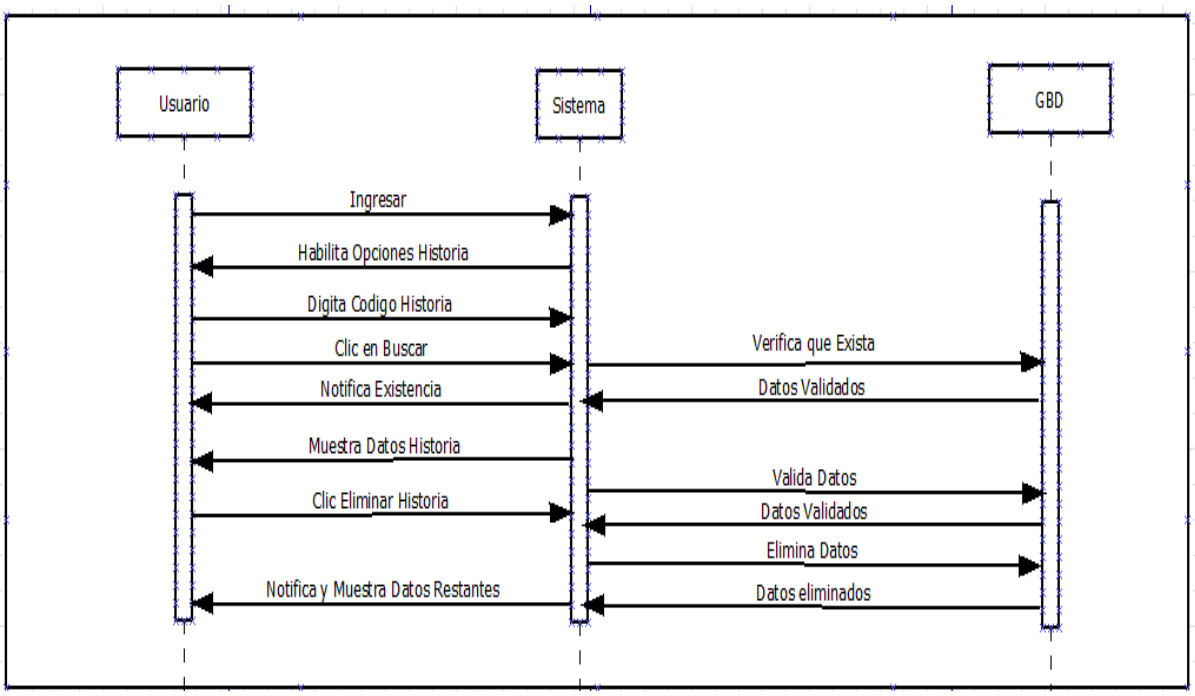


Figura 38. Diagrama de secuencia eliminar historia.

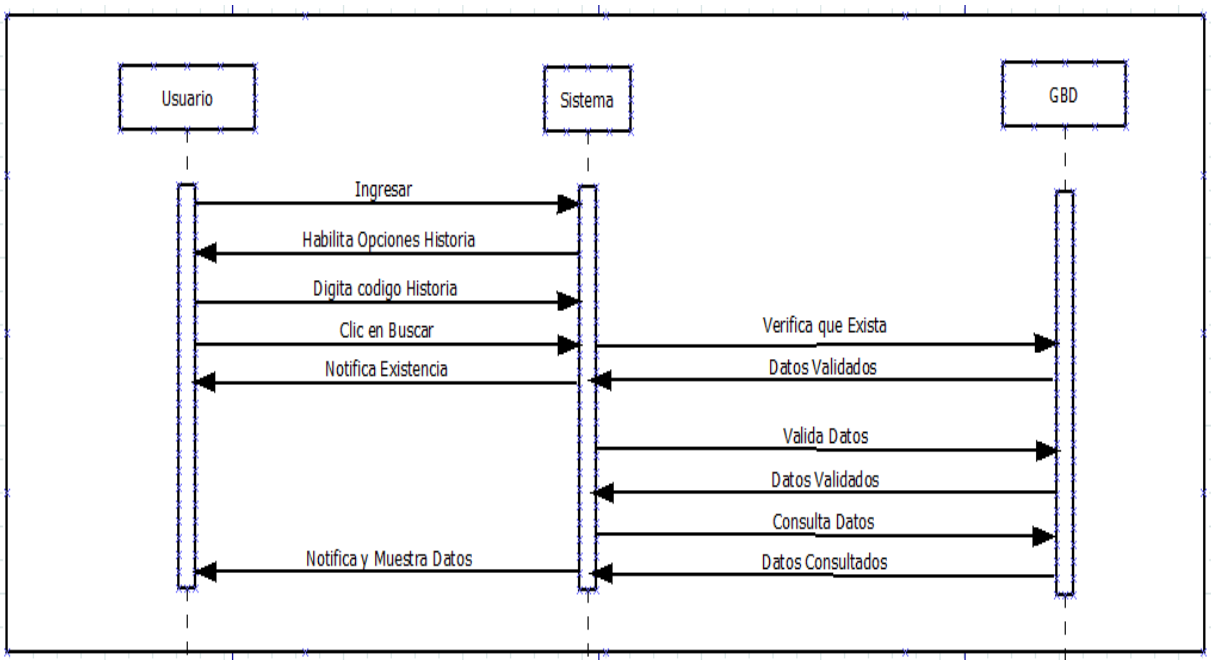


Figura 39. Diagrama de secuencia consultar historia.

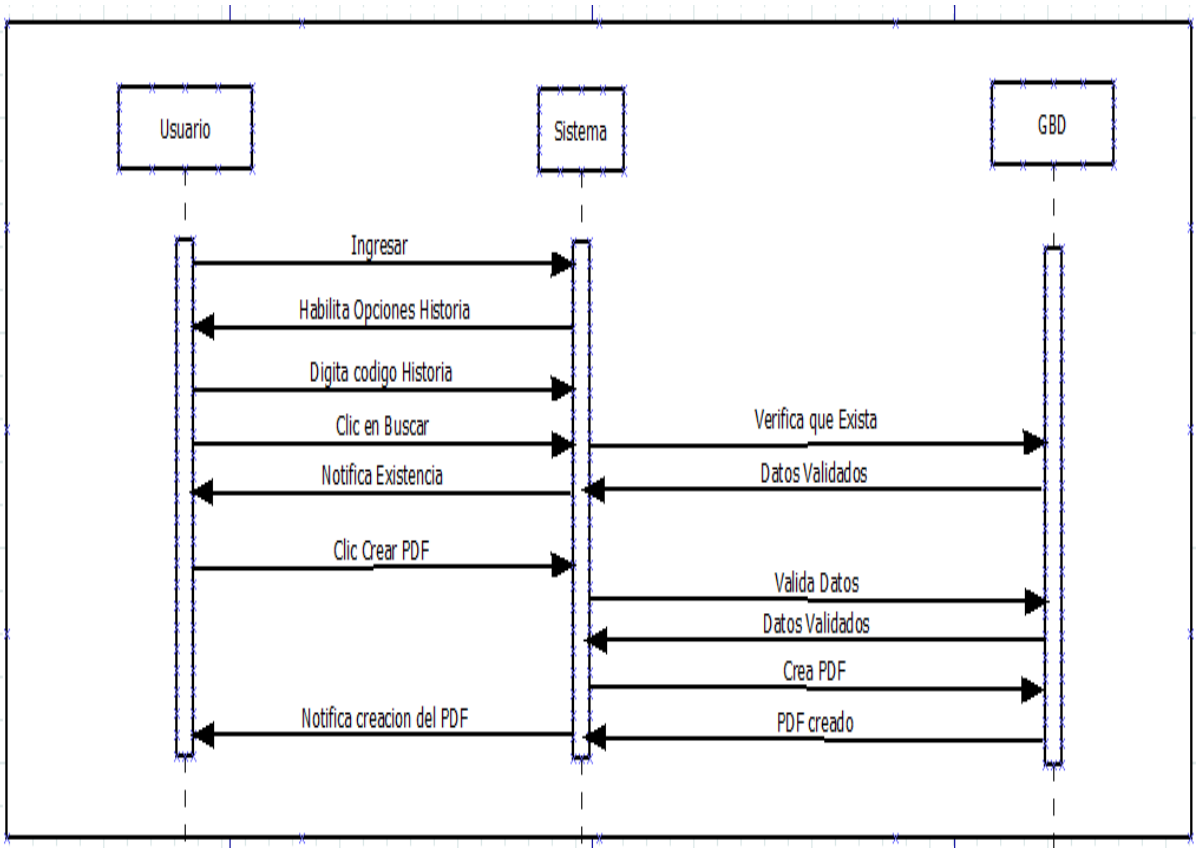


Figura 40. Diagrama de secuencia reporte de historia.

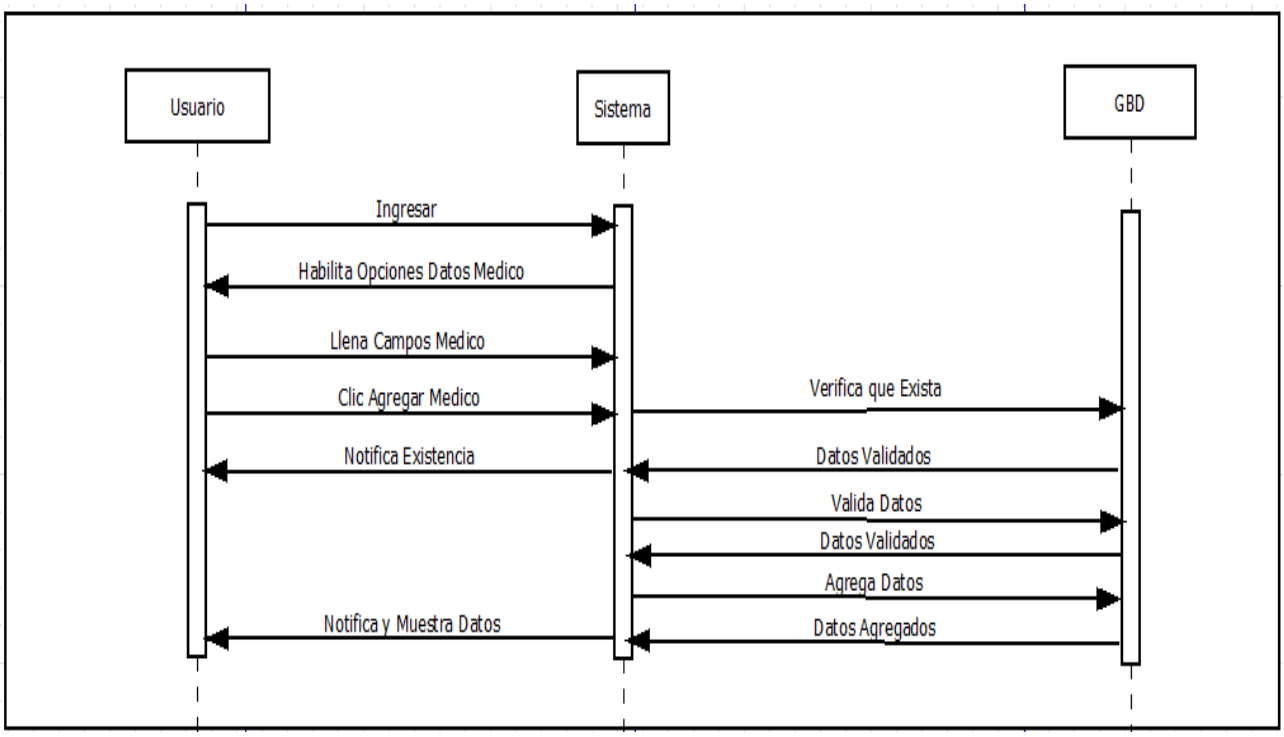


Figura 41. Diagrama de secuencia agregar médico.

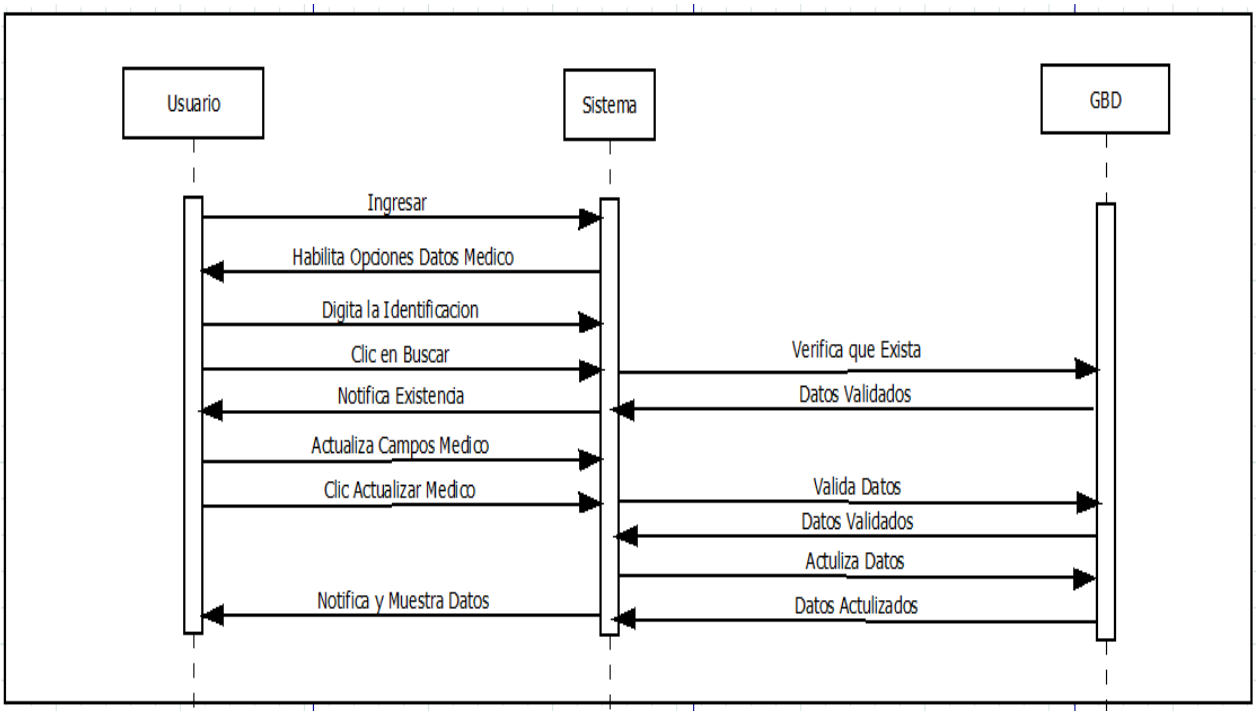


Figura 42. Diagrama de secuencia actualizar médico.

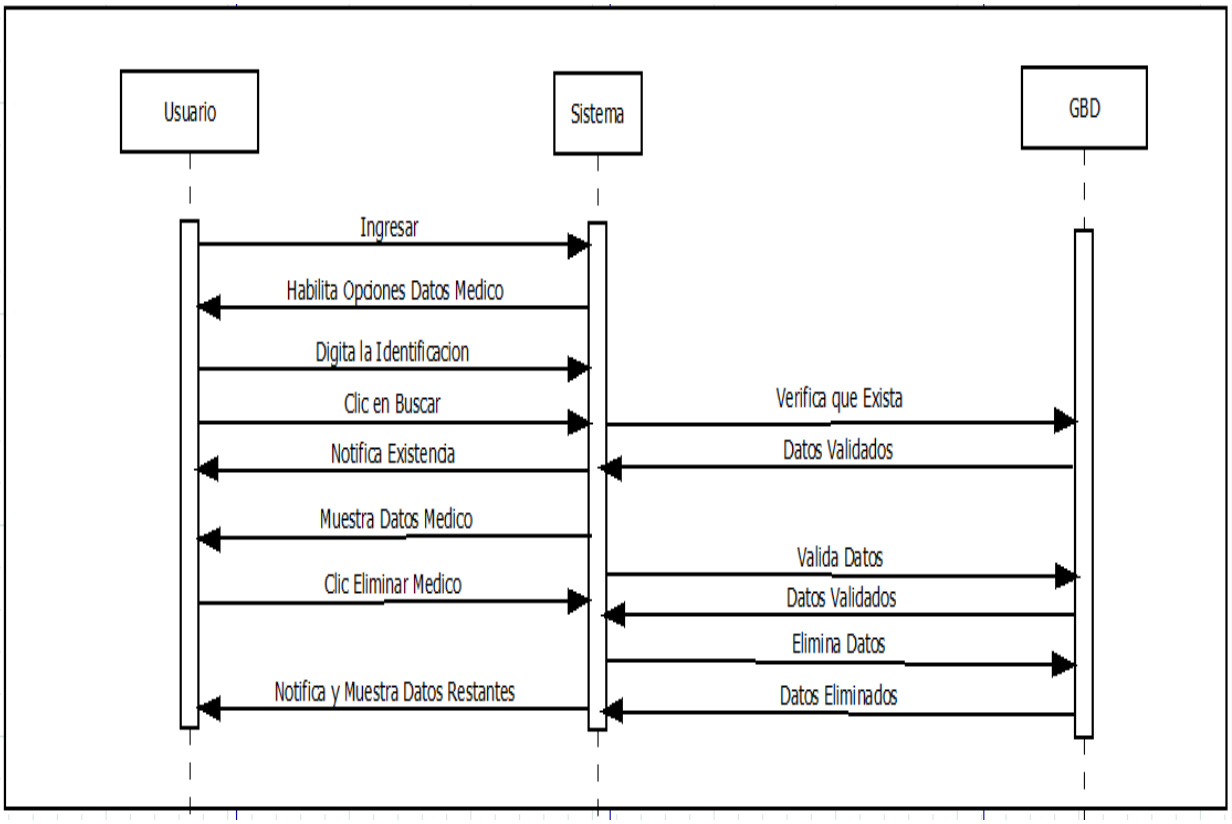


Figura 43. Diagrama de secuencia eliminar médico.

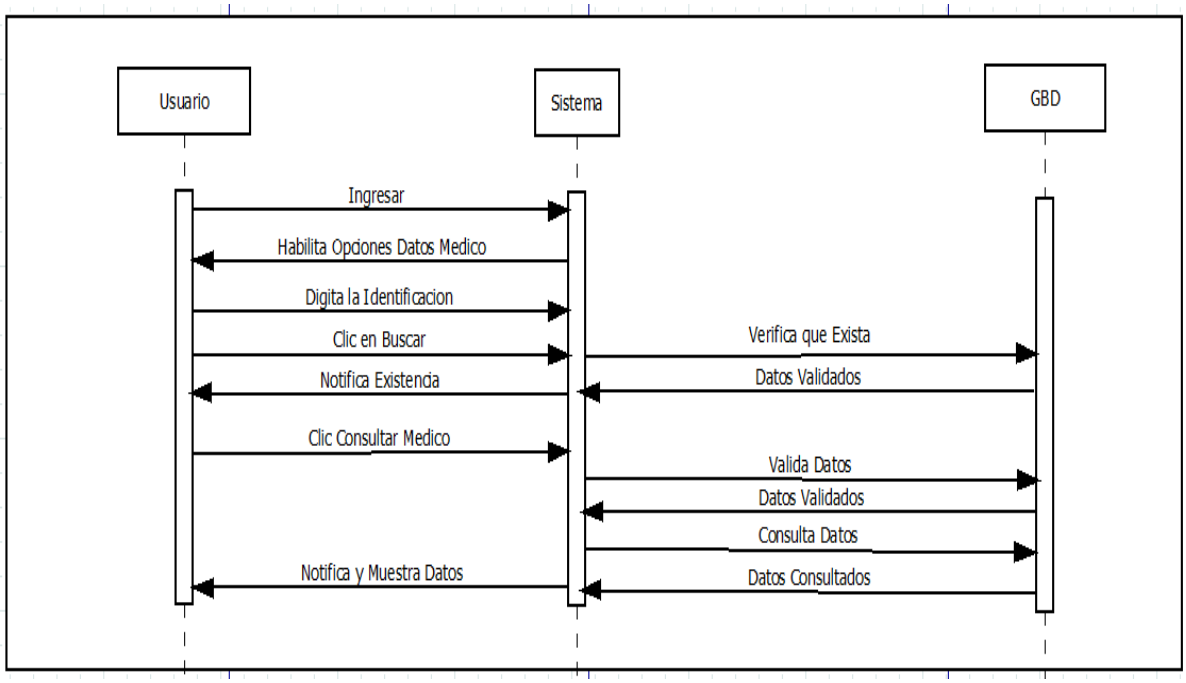


Figura 44. Diagrama de secuencia consultar médico.

5.2.4.3. Diagrama de Actividades

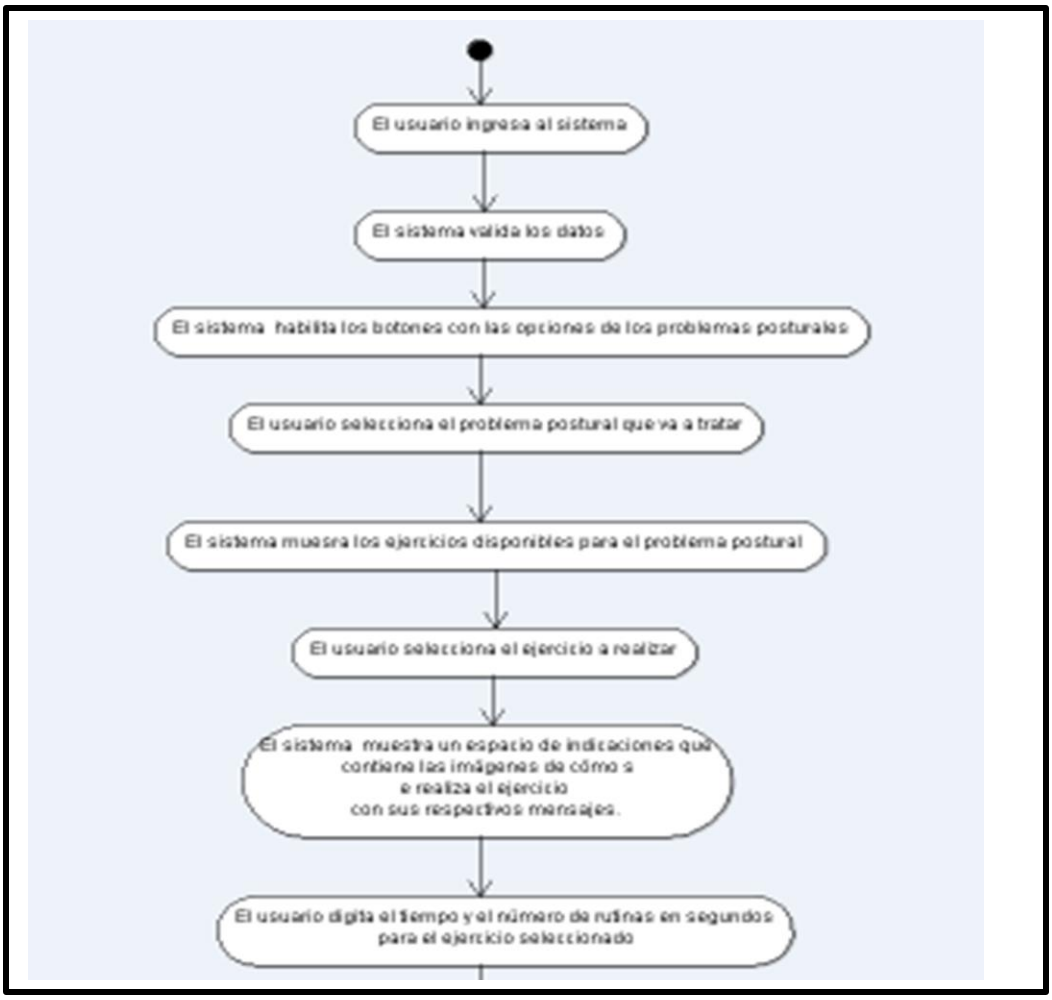


Figura 46. Diagrama de actividades

5.2.5. Elaboración de un archivo de persistencia (.txt)

Para la creación de los respectivos archivos .txt que contiene las coordenadas de los diferentes rutinas de ejercicios, se usó la interfaz del algoritmo DTW (ver Figura 12), y especificación de una ruta donde se guardaron los archivos .txt que almacenan las coordenadas.

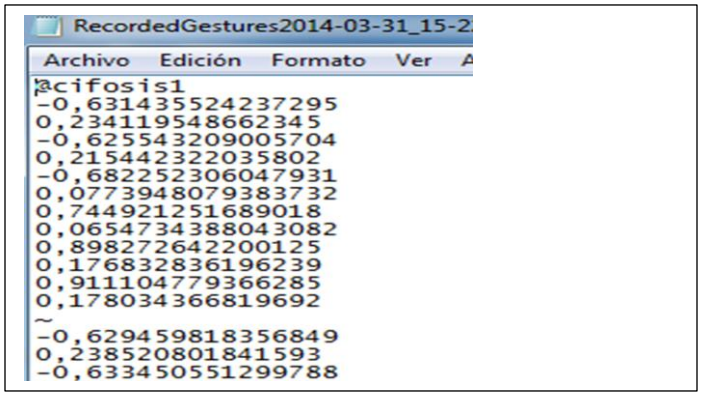


Figura 47. Archivo de persistencia .txt con las coordenadas del ejercicio

Construcción

5.2.6. Diagrama de clases

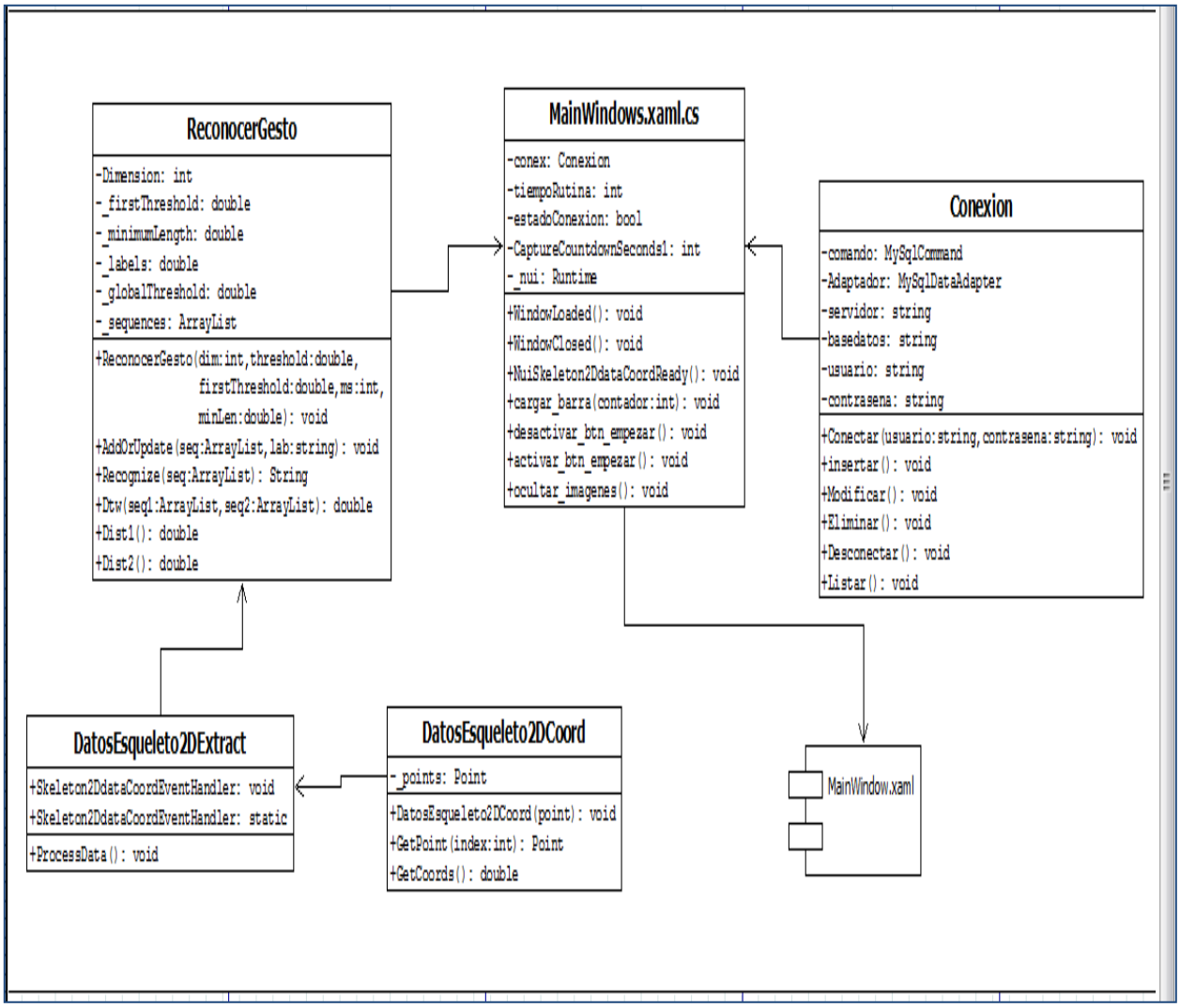


Figura 48. Diagrama de clases

5.2.7. Implementacion

Para la implementación de este proyecto, se utilizó el sdk para Kinect y se procedió a importar las clases necesarias para la aplicación.

1. **Clase MainWindow.xaml.cs**

En esta clase es donde definimos toda la inteligencia asociada a la **interfaz gráfica**.

Los métodos más importantes que encontramos en ella son:

private void WindowLoaded(object sender, RoutedEventArgs e): Una vez que se ha abierto la interfaz gráfica, se llama a este método para inicializar el Kinect.

private static void SkeletonExtractSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e): Cuando estamos intentando reconocer un gesto y Kinect tiene un frame del esqueleto listo, este método es llamado. Él se encarga de pasar todos los datos del esqueleto al algoritmo DTW para compararlo con los patrones.

private void NuiSkeleton2DdataCoordReady(object sender, Skeleton2DdataCoordEventArgs a):

Es el método que contiene toda la inteligencia del reconocimiento de gestos, al cual se le llama cada vez que Kinect tiene un esqueleto preparado

private Point getDisplayPosition(DepthImageFrame depthFrame, Joint joint):

Obtiene la posición donde se muestra una articulación en la representación cualitativa del esqueleto que aparece en la interfaz.

private void NuiSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e):

Cada vez que un esqueleto está listo para ser mostrado, se llama a este método para actualizar el canvas con las líneas que representan los huesos y los puntos que representan las articulaciones.

Prívate void CaptureCountdown1(object sender, EventArgs e) : este método contabiliza el tiempo que tarda en hacerse una rutina.

Prívate void cargar_barra(int valor) : este método se encarga de pintar en la barra de progreso el porcentaje del ejercicio realizado.

Código fuente de la clase (Ver anexo I).

2. Clase DTWReconocerGesto.

Es la clase donde se encuentra implementado el algoritmo de **alineamiento temporal dinámico (DTW)** para la comparación de un gesto realizado y los gestos patrón.

Existe una variable llamada "***globalThreshold***", un umbral que nos determina la máxima distancia DTW que puede haber entre dos gestos para que puedan ser considerados iguales. Si la distancia de un gesto realizado a todos los patrones está por encima de este umbral, el gesto no será reconocido. Podremos determinar el valor de esta variable desde la interfaz gráfica.

Código fuente de la clase (Ver anexo F).

3. Clase DatosEsqueleto2DExtract.

Se encarga de **obtener** las **coordenadas** de la **posición** de **6** partes distintas de nuestro cuerpo, gracias a la funcionalidad de "***skeletal tracking***" del **SDK** de Kinect, y almacenarla en un array ***"p"***de objetos de la clase "***Point***".

Las partes del esqueleto con las que trabajaremos y el orden en el que las tendremos almacenadas se pueden ver en la siguiente tabla.

Posición del array	Parte del cuerpo
p[0]	Mano izquierda
p[1]	Muñeca izquierda
p[2]	Codo izquierdo
p[3]	Codo derecho
p[4]	Muñeca derecha
p[5]	Mano derecha

Además, obtiene las posiciones de ambos hombros y las almacena en las variables "***shoulderRight***" y "***shoulderLeft***". Estas coordenadas serán utilizadas para establecer el origen de nuestro **sistema de referencia** en el **punto medio** del segmento que une ambos hombros.

Por último, para completar la definición del sistema de coordenadas, **normaliza** la distancia de cada punto al nuevo origen de coordenadas dividiendo entre la distancia que separa a los hombros.

Código fuente de la clase (Ver anexo G).

4. Clase DatosEsqueleto2DCoord.

Coge todas las coordenadas que habíamos obtenido y expresado según el nuevo sistema de referencia en la clase ***DatosEsqueleto2DExtract*** y las prepara para poder aplicar el **algoritmo DTW** sobre ellas.

Para ello, crea un nuevo vector de ***12 doubles*** en el que va almacenando las **coordenadas x** en las posiciones **pares** (consideramos que el 0 es par) y las **coordenadas y** en las posiciones **impares**

Este vector será el que pasemos como parámetro a la clase DTWReconocerGesto para calcular distancias. Código fuente de la clase (Ver anexo H).

5. Clase conexión

En esta clase se encuentran los métodos que permiten gestionar la base de datos.

Public bool conectar (string usuario, string contrasena)

Public static void desconectar ();

Public void insertar (SqlString);

Public void modificar (SqlString);

Public void eliminar (SqlString);

Public void buscar (int id);

Public void listar ();

Codigo fuente de la clase (Ver anexo J)

Gestor de base de datos

Para la realización del software se usó mysql como nuestro gestor de base de datos, el siguiente script muestra el código para la realización de la base de datos que almacenara la información de pacientes, médicos e historias clínicas.

Script

```
CREATE TABLE Paciente (
  idPaciente INTEGER UNSIGNED  NOT NULL   AUTO_INCREMENT,
  Nombre VARCHAR(200)  NULL  ,
  Apellido VARCHAR(200)  NULL  ,
  Direccion VARCHAR(500)  NULL  ,
  Celular INTEGER UNSIGNED  NULL    ,
PRIMARY KEY(idPaciente));
CREATE TABLE Medico (
```

```
  idMedico INTEGER UNSIGNED  NOT NULL   AUTO_INCREMENT,
  Nombre VARCHAR(200)  NULL  ,
profesion VARCHAR(100)  NULL    ,
PRIMARY KEY(idMedico));
CREATE TABLE Historia (
  idHistoria INTEGER UNSIGNED  NOT NULL   AUTO_INCREMENT,
  Paciente_idPaciente INTEGER UNSIGNED  NOT NULL  ,
  Medico_idMedico INTEGER UNSIGNED  NOT NULL  ,
  tipoEjercicio VARCHAR(100)  NULL  ,
  progreso INTEGER UNSIGNED  NULL  ,
  diagnostico VARCHAR(300)  NULL  ,
  valoracion VARCHAR(100000)  NULL    ,
PRIMARY KEY(idHistoria)  ,
INDEX Historia_FKIndex1(Medico_idMedico)  ,
INDEX Historia_FKIndex2(Paciente_idPaciente),
  FOREIGN KEY(Medico_idMedico)
    REFERENCES Medico(idMedico)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
  FOREIGN KEY(Paciente_idPaciente)
    REFERENCES Paciente(idPaciente)
      ON DELETE CASCADE
      ON UPDATE CASCADE);
```

Diseño de interfaz

Para el diseño de la interfaz de usuario, se usó wpf (Windows Presentation Foundation) con xaml (eXtensible Application Markup Language) de visual studio 2010, este nos permite agregarles características a los botones, a los paneles, a los labels, agregar imganes, etc. como son colores de borde, fondos, tamaños, color de la letra entre otros, (Ver anexo E).

Transicion:

5.2.7. Pruebas finales

En la realización de las pruebas DE REFPOST con los usuarios finales, se dispuso un consultorio de fisioterapia ubicado en el municipio de santa cruz de lorica, con la asesoría de una fisioterapeuta y se tomaron dos pacientes con edades de 20 y 19 años.

Primero se procedió a instalar el software en el consultorio de la fisioterapeuta, para esto se utilizó adicionalmente una pantalla LCD para observar una visión más grande de la interfaz gráfica. (Ver figura 49).

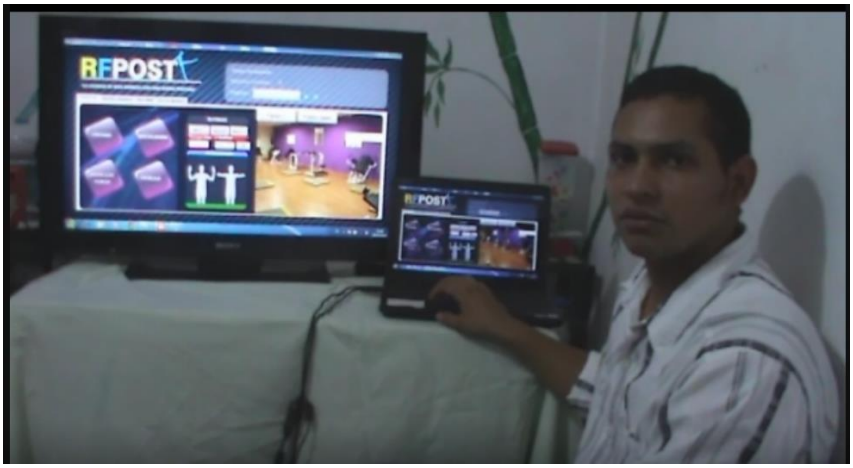


Figura 49. Instalación de RFPOST

Después se le dio el instructivo a la fisioterapeuta de cómo usar el software. (Ver figura 50).

Figura 50. Instructivo de RFPOST

Luego se realizó la consulta con los pacientes para determinar el problema postural a tratar y la fisioterapeuta determino que el primer paciente padecía una cifosis. (Ver figura 51).

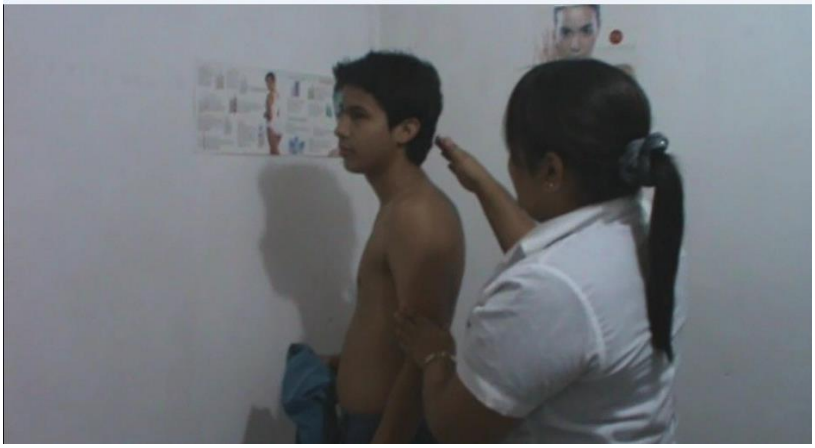


Figura 51. Valoración primer paciente

El segundo paciente presentaba una desviación de cuello. (Ver figura 52).

Figura 52. Valoración segundo paciente

Posteriormente la fisioterapeuta comenzó a agregar los datos del paciente en la base de datos para poder llevar el control del paciente mediante una historia clínica. **(Ver figura 53).**

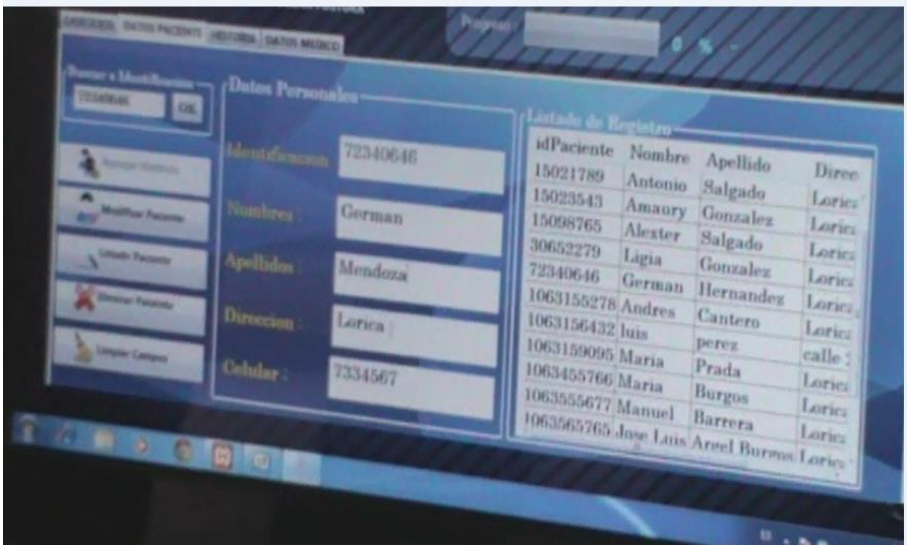


Figura 53. Ingreso de los datos del paciente

Por último se explicó al paciente el funcionamiento del software para que comenzar a realizar la rutina de ejercicios correspondientes a su problema postural. (Ver figura 54 y 55).

Figura 54. Explicación de la rutina de ejercicios al paciente con cifosis

Figura 55. Explicación de la rutina de ejercicios al paciente con desviación de cuello

Inmediatamente comenzaron a realizar la rutina de ejercicios haciendo uso de RFPOST, donde se iba observando si el paciente estaba realizando el ejercicio

correctamente y el progreso que tenía dependiendo del número de rutinas y el tiempo en segundos digitados por la fisioterapeuta. Corroborando de esta manera el correcto funcionamiento del software. (Ver figura 56).

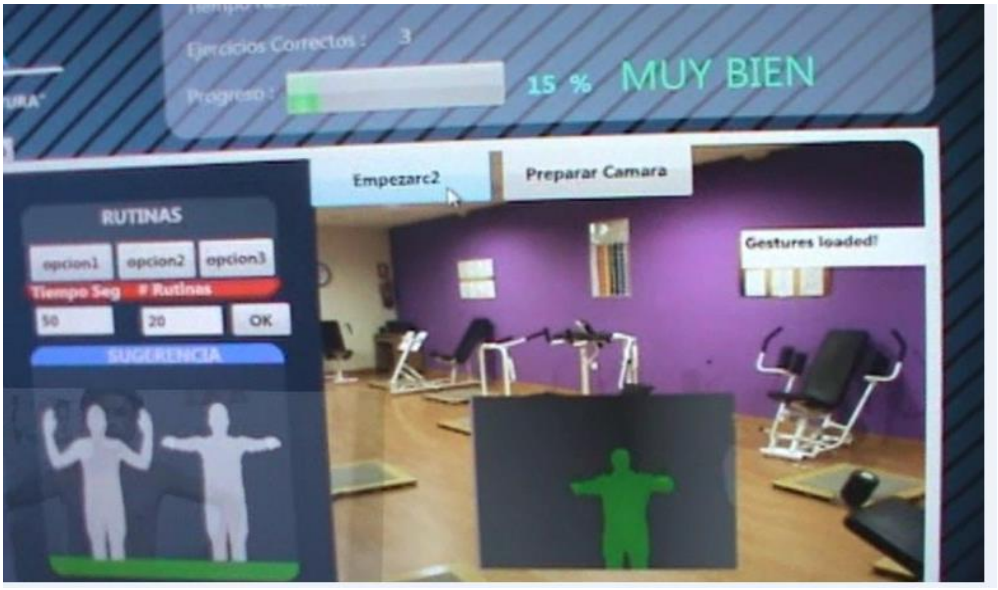


Figura 56. Paciente realizando ejercicios con RFPOST

A continuación, se muestran los resultados arrojados de la pauta final que se les realizo a los usuarios que hicieron uso del software, permitiendo con esto medir el grado de aceptación del sistema por parte de la población de estudio.

En la figura 57 se observa que en promedio, RFPOST manifiesta ser un dispositivo de gran utilidad para apoyar el proceso de rehabilitación de pacientes con problemas posturales.

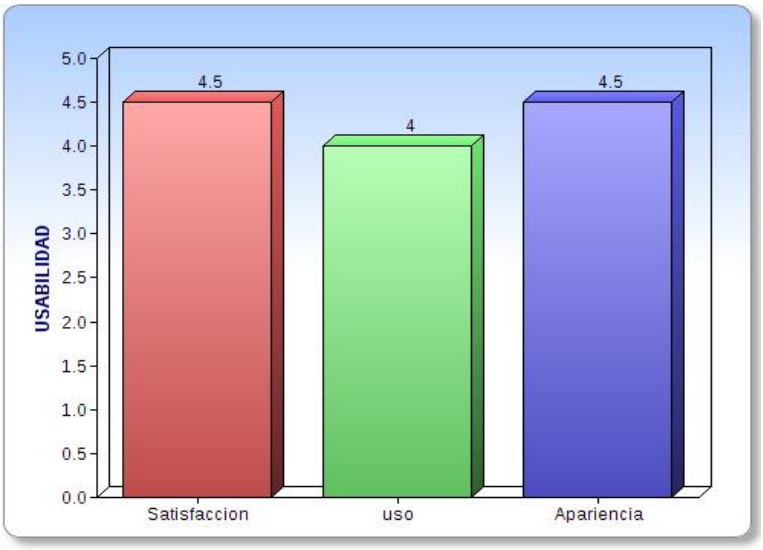


Figura 57. Usabilidad de RFPOST

Para la categoría de “satisfacción, los usuarios manifiestan que el dispositivo fue de su agrado, debido a que este les permitió interactuar con su entorno, logrando realizar el proceso de rehabilitación de una forma diferente y más divertida.

En la categoría de “uso”, los usuarios indican que el dispositivo es fácil de utilizar.

En la categoría apariencia los usuarios manifestaron que el software tiene una apariencia muy agradable y novedosa.

6. CONCLUSIONES

El proyecto RFPOST es una herramienta didáctica e interactiva que sirve de apoyo a una problemática existente como son los problemas posturales**.** Dando la oportunidad de que los fisioterapeutas lo usen como instrumento de apoyo para atender a pacientes que tengan defectos de postura, y a los pacientes para ayudarlos a su recuperación física, de una manera más interesante haciendo que el proceso de rehabilitación sea más divertido, provocando en el usuario un nivel de satisfacción mayor que hace que se adhiera mejor a su terapia.

Con el desarrollo de este software se demostró que es posible implementar tecnologías que cumplan diversos propósitos enfocados a ser útiles a la comunidad en general y en especial en el área de la fisioterapia como por ejemplo el Kinect en este caso.

La finalización del proyecto fue una labor beneficiosa, debido a que se realizó una investigación completa , poniendo en práctica todos los conocimientos obtenidos durante el proceso de formación, además, fue una experiencia motivadora trabajar para un área tan importante de la salud como es la fisioterapia.

Finalmente, este proyecto abre muchas puertas para seguir avanzando en busca de crear y mejorar cada día herramientas tecnológicas que contribuyan con el desarrollo integral de las personas.

Referencias

Villacorta D. y Morales L. (2010). *Analisis de la mecanica corporal en la comunidad universitaria de la universidad autonoma de santa ana* Tesis de postgrado. Universidad autónoma, santa ana, El salvador.

Rosario, J. (2005). La Tecnología de la Información y la Comunicación (TIC). *Observatorio para la cibersociedad,* 12, 100-103

Venegas J. y Alviarez D. (2011). *Kinectsiology.* Tesis de pregrado, Universidad Técnica Federico Santa Mará, Valparaíso, Chile.

Zorilla J. (2012). *Rehabilit-AR.* Tesis de Maestría no publicada, universidad de Burgos, España.

Cts. (1997). *Fisimetrix.* Tesis doctoral no publicada, Universidad de brasilia, Brasilia, Brasil.

Henao O. y Lopez F. (2013). *Sistema de Rehabilitación basado en el Uso de Análisis Biomecánico y Videojuegos mediante el Sensor Kinect.* Tesis de maestría no publicada, universidad tecnológica de pereira, Pereira, Colombia.

Muñoz, J. (2012). Sensor Kinect para el análisis biomecánico y la interacción con los videojuegos. 5ta conferencia de creación de videojuegos serios para la salud, Pereira, Colombia.

Eafit. (2010). *aplicación de video juegos para rehabilitación de pacientes que han sufrido accidentes cerebro-vasculares.* Tesis de postgrado no publicada, universidad de Medellin, Medellin, Colombia.

Tohen, Z. (1970). *Medicina física y rehabilitación* 2ª *ed*. Mexico: DF TheUniversitySociety Mexicana S.A.

Gattoronchieri, V. (2005). *La postura correcta.* Barcelona: De vecchi.

Chávez, A. (2012, julio 10). *Defectos posturales*. Recuperado de: http://www.slideshare.net/gutyparrandas/defectos-posturales-13590452

Kendall, P. (2006). *kendall's músculos pruebas funcionales postura y dolor,* 5ª ed. Madrid: Marban.

Areli E. (2008, abril 28). *La columna vertebral*. Recuperado de: http://seguridadhigiene.wordpress.com/2008/04/

Guerrero, C. (2013, febrero 01). *Las posturas inadecuadas factores de riesgo para patologías de la columna*. Recuperado de http://prezi.com/lpvj_9pedryu/untitled-prezi/

Duran, J. (2008, julio 04). *Columna vertebral II*. Recuperado de: http://www.monografias.com/trabajos63/columna-vertebral/columna-vertebral2.shtml

Valencia, C. (2011,Enero 08). *La postura corporal.* Recuperado de http://www.slideshare.net/cbpilar/cv-6485479

Garcia, T. (2012, diciembre 02). *Ejercicios físicos terapéuticos para la rehabilitación de la deformidad postural*. Recuperado de: http://www.monografias.com/trabajos87/ejercicios-fisicos-terapeuticos-rehabilitacion-deformidad-postural-escoliosis/ejercicios-fisicos-terapeuticos-rehabilitacion-deformidad-postural-escoliosis.shtml

Saluspost, (2013, octubre 20). *La importancia de las tecnologías de la información en la medicina*. Recuperado de: http://blog.saluspot.com/la-importancia-de-las-tecnologias-de-la-informacion-en-medicina/

Murillo, A. (2012, Febrero). *Kinect para developers.* Recuperado de: http://www.kinectfordevelopers.com/author/alejandro-murillo/page/2/

Msdn (2011, Agosto,09). *Detectar posturas con skeletal tracking.* Recuperado de:http://blogs.msdn.com/b/esmsdn/archive/2011/08/09/reto-sdk-de-kinect-detectar-poses-con-skeletal-tracking.aspx

Codeplex. (2011, Julio 30).*kinectsdk dynamic time warping (DTW) gesture recognition*. Recuperado de: http://kinectdtw.codeplex.com

Davara, M. (2005). *Manual de derecho informático,* 7ª ed. Madrid:Aranzadi S.A.

Zuta, J. (2011, Noviembre 10). *Tipos de licencias para software.* Recuperado de: http://www.monografias.com/trabajos88/tipos-licencias-software/tipos-licencias-software.shtml

ANEXOS

ANEXO A

ALCANCES Y LIMITACIONES

Alcances

El presente trabajo investigativo evalúa la propuesta de una infraestructura basada en un dispositivo que permite la rehabilitación física de personas con defectos posturales utilizando como medio la tecnología Kinect con el fin de permitirle a los pacientes una herramienta interactiva con la que puedan corregir estos padecimientos.

Este sistema será capaz de:

- Mostrar opciones de ejercicios según cada problema postural.
- Reconocer un determinado movimiento (ejercicio).
- Sugerir como debe realizarse el ejercicio.
- Mostrar en una barra el progreso del ejercicio realizado.
- Almacenar información correspondiente a pacientes, médicos e historia clínica.
- Actualizar información correspondiente a pacientes, médicos e historia clínica.
- Eliminar información correspondiente a pacientes, médicos e historia clínica.
- Consultar información correspondiente a pacientes, médicos e historia clínica.

- Almacenar información correspondiente a pacientes, médicos e historia clínica.
- Generar un reporte en PDF sobre la historia clínica de un paciente.

Limitaciones

El sistema no será capaz de

- Que el fisioterapeuta pueda acceder al software desde la web y así poder realizar el oficio sin necesidad de estar presente.

- Que más de un paciente realice una rutina de ejercicios simultáneamente.

- Mostrar al paciente a través de un simulador como se realiza el ejercicio.

- Permitir que el fisioterapeuta guarde sus propios patrones de ejercicios

ANEXO B

Marco legal

Aspectos legales del software

A continuación se especifican las distintas herramientas usadas en desarrollo de la aplicación con sus respectivas licencias.

Tabla 2. Herramientas usadas en el desarrollo de RFPOST

HERRAMIENTA	UTILIDAD	APORTE	TIPO LICENCIA	FABRICANTE
Sdk beta 2	Sdk Kinect para Windows	Detección de profundidad, monitoreo de movimientos humanos y reconocimiento de voz y objetos con el uso de la tecnología Kinect en Windows 7.	Free	Microsoft
Sdk dtw	Reconocimiento de gestos	Incluye un grabador de gesto, reconocedor y gestos de la muestra. Usted puede guardar sus gestos en un archivo. Utiliza seguimiento esquelético y apoya vectores 2D.	CDDL	FaisalNahian
Visual studio2010	Herramienta rápida de desarrollo.	Entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows	propietaria	Microsoft

Tabla 3. Aplicaciones usadas en el desarrollo de RFPOST

Aplicación	Tipo	versión	Aporte	Tipo licencia
C#	lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET,	5.0	Lenguaje utilizado para programar la aplicación	
Dia	Software de diagramación	0.97.2	Dibujar diagramas de software	Free
Gantt	software completo de gestión de proyectos		realizar cronograma de actividades	GPL
Xampp	Servidor	5.0	Gestionar bases de datos de mysql y php	Free
WPF	Tecnología de Microsoft, presentada como parte de Windows Vista. Permite el desarrollo de interfaces de interacción en Windows.	4.5	Crear la interfaz grafica	EULA

En la tabla 3, se observa la especificación de las aplicaciones que fueron utilizadas complementarias para llevar a cabo varios de los requerimientos necesarios para el sistema.

El uso de dichos programas se debió a que presentan mejores características que el resto de las aplicaciones que tiene las mismas funciones.
Además de esto, como este sistema va destinado a un fin académico, no se presentan inconvenientes en cuanto a los términos de uso descritos por cada programa. Si este trabajo tuviera un fin comercial, se tendría que hacer uso de las respectivas licencias legales.

“La Licencia de software es una especie de contrato, en donde se especifican todas las normas y cláusulas que rigen el uso de un determinado programa,

principalmente se estipulan los alcances de uso, instalación, reproducción y copia de estos productos" (Davara, 2005).

En el caso del Software, la legislación colombiana lo asimila a la escritura de una obra literaria, permitiendo que el código fuente de un programa esté cubierto por la ley de Derechos de Autor (Ley 23 de 1982)21, la cual indica que todo autor desde el momento de la creación, dispone de unos derechos patrimoniales.

Según Zuta (2011) "Existen diversos tipos de software según su licencia" entre ellos cabe destacar las siguientes:

Software propietario o comercial**:** es un software cerrado, donde el dueño del software controla su desarrollo y no divulga sus especificaciones.

Software libre**:** es un software que, para cualquier propósito, se puede usar, copiar, distribuir y modificar libremente, es decir, es un software que incluye archivos fuentes.

Software de dominio público**:** (public domain software), es un software libre que tiene como particularidad la ausencia de Copyright, es decir, es libre sin derechos de autor. En este caso los autores renuncian a todos los derechos que les puedan corresponder.

Software semi-libre**:** para la FSF (Free Software Foundation) el software semi-libre es software que posee las libertades del software libre pero sólo se puede usar sin fines de lucro, por lo cual lo cataloga como software no libre.

Software freeware**:** es un software que se puede usar, copiar y distribuir libremente pero que no incluye archivos fuentes. Para la FSF, el software freeware no es software libre, aunque tampoco lo califica como semi-libre ni propietario.

La denominación de software libre se debe a la Free Software Foundation (FSF), que significa Fundación de software libre. Esta entidad promueve el uso y desarrollo de software de este tipo. Cuando la FSF habla de software libre se refiere a la nueva filosofía respecto al software, donde prevalecen aspectos como especificaciones abiertas y bienes comunes, sin fines de lucro. Esta organización elabora, mantiene y defiende la Licencia Pública General GNU (GNU/GPL), la licencia de software libre más utilizada, cuya última versión es la GPL versión 3 que fue publicada en forma definitiva en junio de 2007 (Wikipedia, 2013).

Por otra parte la LEY 528 DE 1999
(Septiembre 14) Diario Oficial No. 43.711, de 20 de septiembre de 1999
Poder Público - Rama Legislativa
Por la cual se reglamenta el ejercicio de la profesión de fisioterapia, se dictan normas en materia de ética profesional y otras disposiciones**.**

EL CONGRESO DE COLOMBIA DECRETA:

CAPITULO I

De las relaciones del fisioterapeuta con los usuarios de sus servicios

Artículo 13. Siempre que el fisioterapeuta desarrolle su trabajo profesional, con individuos o grupos, es su obligación partir de una evaluación integral, destinada a establecer un diagnóstico fisioterapéutico, como fundamento de su intervención profesional.

Artículo 28. El fisioterapeuta deberá comprometerse, como parte integral de su ejercicio profesional, con las acciones permanentes de promoción de la salud y prevención primaria, secundaria y terciaria de las alteraciones y complicaciones del movimiento humano.

ANEXO C

Resultados de las encuentas

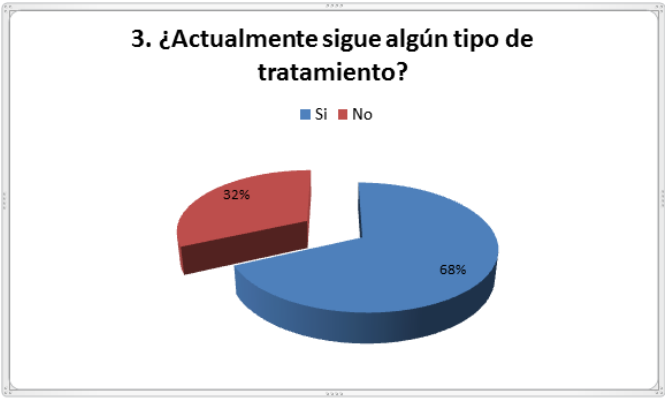


Figura 16. Gráfica de resultado a la pregunta 3 de la encuesta inicial.

El 68% de las personas encuestadas actualmente están sometidos a algún tratamiento para corregir su problema postural y un 32% no están bajo ningún tratamiento.

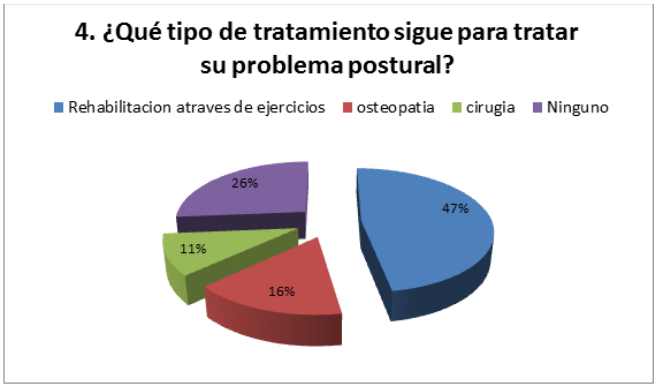


Figura 17. Gráfica de resultado a la pregunta 4 de la encuesta inicial.

Se puede observar que un 47% de las personas encuestadas utilizan la rehabilitación atraves de ejercicios físicos para corregir su problema postural, un 16% se inclina por la osteopatía, un 11% se someterán a intervenciones

quirúrgicas para corregir el problema y 26% no están en ningún tratamiento médico.

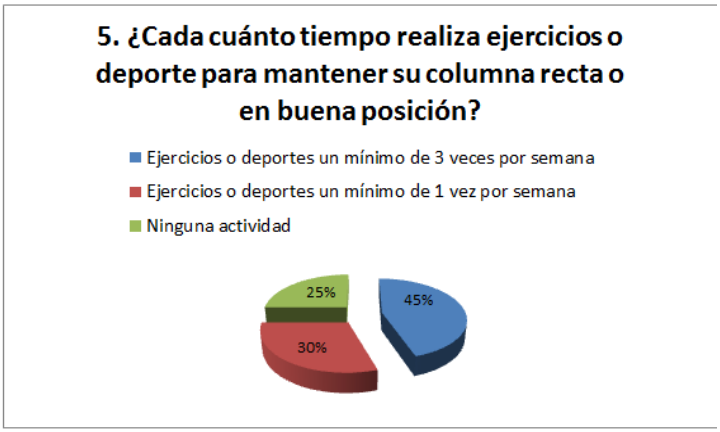


Figura 18. Gráfica de resultado a la pregunta 5 de la encuesta inicial.

La mayoría de las personas encuestadas realizan algún deporte o ejercicio 3 vece por semana.

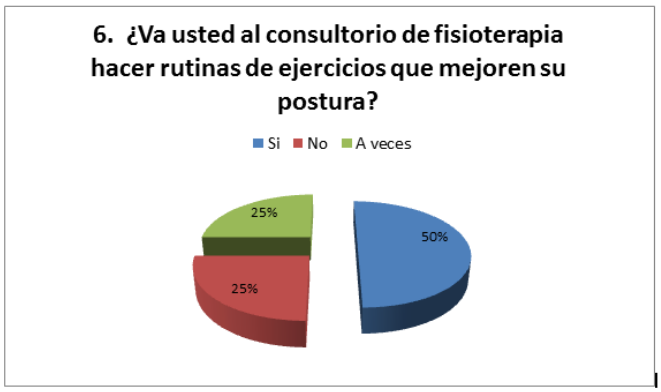


Figura 19. Gráfica de resultado a la pregunta 6 de la encuesta inicial.

La mitad de las personas encuestadas van al fisioterapeuta a realizar ejercicios que mejoren o corrijan su problema postural, el 25% no acude al fisioterapeuta y el otro 25% A veces lo hace.

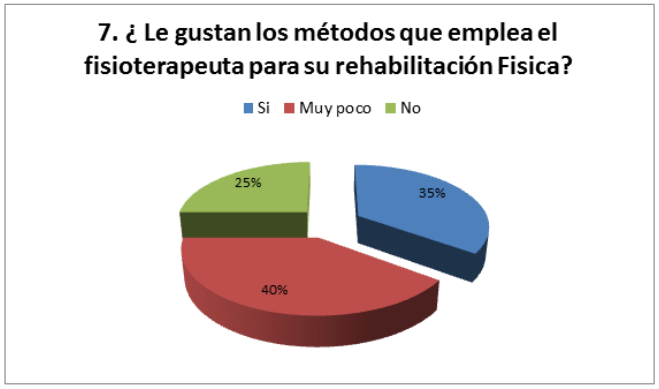


Figura 20. Gráfica de resultado a la pregunta 7 de la encuesta inicial.

El 35% de la población le gustan la forma en que el fisioterapeuta emplea para ayudarlo en la rehabilitación física, el 25% definitivamente no le gustan pues les parece aburrido y poco interactivo, y el 40% muy poco le gustan pues no les parece divertido el proceso de rehabilitación y además poco informativo.

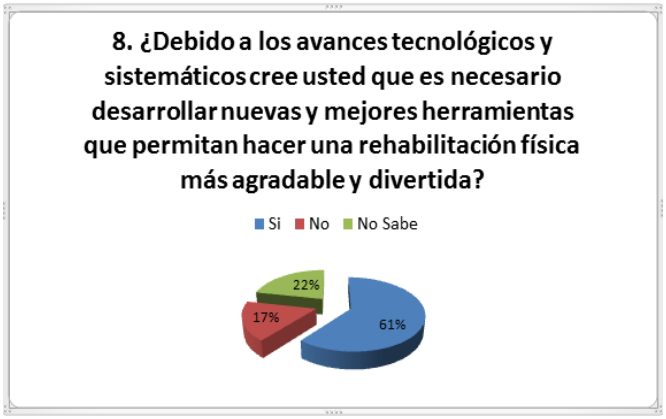


Figura 21. Grafica de resultado a la pregunta 8 de la encuesta inicial.

El 61% de las personas cree que es muy necesario que aprovechando el auge tecnológico de la actualidad desarrollar nuevas herramientas que permitan realizar una rehabilitación física más agradable y divertida.

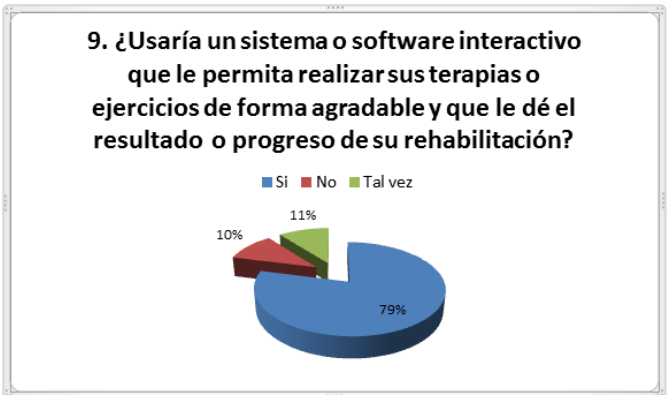


Figura 22. Gráfica de resultado a la pregunta 9 de la encuesta inicial.

El 79% de las personas encuestadas usarían el software interactivo que le permita realizar sus terapias de forma agradable y que le brinde información sobre el progreso de su rehabilitación.

ANEXO D

Descripciones de los casos de uso

Tabla 4. Caso de uso ingresar al sistema.

CÓDIGO	**REC001**		
NOMBRE	Ingresar al sistema		
PRIORIDAD	Alta		
ACTORES	Usuario		
DESCRIPCIÓN	El sistema debe permitir al usuario ingresar al sistema.		
SECUENCIA NORMAL	**Paso**		**Acción**
	1	el usuario solicita ingresar al Sistema.	
	2		El sistema solicita el usuario y contraseña.
	3	El usuario ingresa su nombre de Usuario y su	

		contraseña.	
	4		El sistema Valida los datos y de ser correctos permite que el usuario ingrese.
Paso	**Acción**		
	1	Si el usuario ya existe, se muestra un mensaje.	
PRECONDICIÓN	El Usuario no existe en el sistema.		
POSTCONDICION	El Usuario ha sido ingresado exitosamente.		

Tabla 5. Caso de uso habilitar opciones problema postural

CÓDIGO	**REC002**	
NOMBRE	Habilitar opciones de problema postural.	
PRIORIDAD	Alta	
ACTORES	Sistema	
DESCRIPCIÓN	Cuando el usuario ingresa al sistema la interfaz debe inmediatamente habilitar las opciones de los diferentes problemas posturales.	
SECUENCIA NORMAL	**Paso**	**Acción**
	1	El sistema valida los datos del usuario.
	2	El sistema habilita las opciones de los problemas posturales.
		El sistema muestra al usuario los diferentes botones con los problemas posturales.
CAMINO DE EXCEPCIÓN	**Paso**	**Acción**
	1	El sistema Valida los datos del usuario
	2	El sistema falla al habilitar los botones con las opciones de los problemas posturales.
PRECONDICIÓN	El Usuario debe estar registrado en el sistema	
POSTCONDICION	Las opciones han sido habilitadas satisfactoriamente.	

Tabla 6. Caso de uso seleccionar problema postural

CÓDIGO	**REC003**	
NOMBRE	Seleccionar problema postural.	
PRIORIDAD	Alta	
ACTORES	Usuario.	
DESCRIPCIÓN	El usuario indica al sistema que quiere seleccionar un determinado problema postural.	
SECUENCIA NORMAL	**Paso**	**Acción**
	1	El usuario visualiza los diferentes botones con los problemas posturales.
	2	El usuario selecciona el problema postural que va a tratar.
PRECONDICIÓN	El Usuario debe estar registrado en el sistema	
POSTCONDICION	El problema postural se seleccionó satisfactoriamente.	

Tabla 7. Mostrar las rutinas u opciones de ejercicios.

CÓDIGO	**REC004**	
NOMBRE	Mostrar las rutinas u opciones de ejercicios.	
PRIORIDAD	Alta	
ACTORES	Sistema, usuario	
DESCRIPCIÓN	La interfaz debe mostrar diferentes opciones de ejercicios por cada problema postural seleccionado.	
SECUENCIA NORMAL	**Paso**	**Acción**
	1	El sistema habilita los ejercicios disponibles para el problema postural escogido.
	2	El sistema muestra al usuario los botones con los diferentes ejercicios.
CAMINO DE EXCEPCIÓN	**Paso**	**Acción**
	1	El sistema falla a la hora de mostrar las opciones de ejercicios.
PRECONDICIÓN	El Usuario debe estar registrado en el sistema	
POSTCONDICION	Las opciones de ejercicios se mostraron correctamente.	

.

Tabla 8. Caso de uso Seleccionar el ejercicio a realizar.

CÓDIGO	**REC005**	
NOMBRE	Seleccionar el ejercicio a realizar	
PRIORIDAD	Alta	
ACTORES	Usuario	
DESCRIPCIÓN	El usuario indica a la interfaz que quiere seleccionar un determinado ejercicio.	
SECUENCIA NORMAL	**Paso**	**Acción**
	1	El usuario visualiza los botones con ejercicios disponibles para el problema postural escogido.
	2	El usuario selecciona el ejercicio a realizar.
CAMINO DE EXCEPCIÓN	**Paso**	**Acción**
	1	El sistema falla a la hora de mostrar las opciones de ejercicios.
PRECONDICIÓN		
POSTCONDICION	El usuario selecciono el ejercicio.	

Tabla 9. Caso de uso Mostrar indicaciones con imágenes y mensajes

CÓDIGO	**REC006**	
NOMBRE	Mostrar indicaciones con imágenes y mensajes	
PRIORIDAD	Alta	
ACTORES	Sistema.	
DESCRIPCIÓN	El sistema debe mostrar las indicaciones de cómo se realiza el ejercicio a través de imágenes con sus respectivas instrucciones.	
SECUENCIA NORMAL	**Paso**	**Acción**
	1	El sistema verifica el ejercicio seleccionado.
	2	El sistema muestra un espacio de indicaciones que contiene las imágenes de cómo se realiza el ejercicio con sus respectivos mensajes.
CAMINO DE EXCEPCIÓN	**Paso**	**Acción**
	1	El sistema falla a la hora de mostrar las imágenes.
PRECONDICIÓN		

POSTCONDICION	La interfaz mostro las indicaciones correctamente

Tabla 10. Caso de uso Digitar tiempo y numero de rutinas en segundos

CÓDIGO	**REC007**	
NOMBRE	Digitar tiempo y numero de rutinas en segundos	
PRIORIDAD	Alta	
ACTORES	Usuario.	
DESCRIPCIÓN	El usuario debe digitar el tiempo y el número de rutinas para el ejercicio que va a realizar.	
SECUENCIA NORMAL	**Paso**	**Acción**
	1	El usuario digita el tiempo y el número de rutinas en segundos para el ejercicio seleccionado.
	2	El sistema procesa la información.
CAMINO DE EXCEPCIÓN	**Paso**	**Acción**
	1	El usuario no digite correctamente los tiempos y numero de rutinas.
PRECONDICIÓN		
POSTCONDICION	El usuario digito el tiempo y numero de rutinas del ejercicio.	

Tabla 11. Caso de uso Comenzar ejercicio.

CÓDIGO	**REC008**	
NOMBRE	Comenzar ejercicio.	
PRIORIDAD	Alta	
ACTORES	Usuario.	
DESCRIPCIÓN	Cuando el usuario indica al sistema el ejercicio a realizar, mira las indicaciones y digita el tiempo y el número de rutinas la interfaz muestra un espacio donde se puede iniciar el ejercicio dando clic en el botón empezar.	
SECUENCIA NORMAL	**Paso**	**Acción**
	1	El usuario da clic en el botón empezar
		El usuario inicia el ejercicio.
	2	El sistema muestra como el tiempo digitado comienza a correr.
	3	El sistema muestra si el ejercicio se está realizando bien.
	4	El sistema acaba el ejercicio cuando finaliza

		el tiempo.
PRECONDICIÓN		
POSTCONDICION	El ejercicio se realizó.	

Tabla 12. Caso de uso mostrar progreso.

CÓDIGO	**REC009**	
NOMBRE	Mostar progreso	
PRIORIDAD	Alta	
ACTORES	Sistema	
DESCRIPCIÓN	La interfaz muestra un progreso en porcentaje del ejercicio realizado.	
SECUENCIA NORMAL	**Paso**	**Acción**
	1	El sistema finaliza el ejercicio cuando el tiempo se agota
	2	El sistema muestra el progreso en porcentaje del ejercicio realizado.
PRECONDICIÓN		
POSTCONDICION	El progreso se mostró satisfactoriamente	

Tabla 13. Caso de uso agregar paciente.

CÓDIGO: REC 010		
NOMBRE: gestión paciente(agregar paciente)		
ACTOR(ES): Administrador del sistema, medico		
DESCRIPCIÓN: mediante este caso de uso se registra la información personal de un paciente, el sistema debe permitir ingresar los siguientes datos nombre, apellidos, cedula, fecha de nacimiento, dirección, número de teléfono.		
Flujo principal		
	Actor	**Sistema**
1	Selecciona la opción agregar paciente	
2		solicita los datos personales del paciente
3	Ingresa datos solicitados	

4		Envía un mensaje al usuario indicando que se ha registrado exitosamente el paciente.	
5			
	Camino de excepción	**paso**	**acción**
		1	Si no se han ingresado todos los datos del paciente o este ya existe se envía un mensaje de error al usuario.
Precondiciones el paciente no existe en el sistema			
Postcondiciones el paciente ha sido registrado			

Tabla 14. Caso de uso actualizar paciente.

CÓDIGO: 011		
NOMBRE: Gestión paciente(actualizar paciente)		
ACTOR(ES): administrador del sistema, medico.		
DESCRIPCIÓN: el sistema debe permitir modificar la información de un paciente.		
Flujo principal		
	Actor	**Sistema**
1	Selecciona la opción paciente	
2		Solicita que digite el paciente al que se va a modificar
3	Digita la identificación del paciente	
4	Clic en buscar	
5		Muestra la información del paciente.

6	Modifica la información deseada.	
		Registra los cambios realizados.
Precondiciones el paciente exista en el sistema		
Postcondiciones la información ha sido modificada		

Tabla 15. Caso de uso eliminar paciente.

CÓDIGO: 012		
NOMBRE: Gestión paciente(Eliminar paciente)		
ACTOR(ES): administrador del sistema, medico.		
DESCRIPCIÓN: el sistema debe permitir al usuario eliminar información de un paciente.		
Flujo principal		
	Actor	**Sistema**
1	Selecciona la opción paciente	
2		Pide al usuario digite el paciente
3	Digita la identificación del paciente	
4	.clic en buscar	Muestra los datos del paciente
5	Da clic en eliminar	
6		pide confirmación (si o no)
	cancela o acepta la operación	
		Elimina el paciente.
Precondiciones el paciente debe existir en el sistema .		

Postcondiciones	el paciente ha sido eliminado

Tabla 16. Caso de uso consultar paciente.

CÓDIGO: 013		
NOMBRE: Gestión paciente(consultar paciente)		
ACTOR(ES): administrador del sistema, medico.		
DESCRIPCIÓN: El sistema debe permitir consultar información de un paciente.		
Flujo principal		
	Actor	**Sistema**
1	Selecciona la opción paciente.	
2		Pide al usuario que digite la identificación del paciente.
3	Digita la identificación del paciente	
4	Da clic en consultar paciente.	
5		Muestra la información del paciente.
Precondiciones	el paciente debe existir en el sistema.	
Postcondiciones	el paciente ha sido consultado	

Tabla 17. Caso de uso agregar historia.

<table>
<tr><td colspan="5">CODIGO 014</td></tr>
<tr><td colspan="5">NOMBRE: Gestión historia(agregar historia)</td></tr>
<tr><td colspan="5">ACTOR(ES): Administrador del sistema, medico</td></tr>
<tr><td colspan="5">Descripción: mediante este caso de uso se registra la información de la historia médica del paciente</td></tr>
<tr><td colspan="5">Flujo principal</td></tr>
<tr><td></td><td>Actor</td><td colspan="3">Sistema</td></tr>
<tr><td>1</td><td>Selecciona la opción historia</td><td colspan="3"></td></tr>
<tr><td>2</td><td></td><td colspan="3">solicita los datos de la historia del paciente</td></tr>
<tr><td>3</td><td>Ingresa datos solicitados</td><td colspan="3"></td></tr>
<tr><td>4</td><td>Da clic en agregar historia</td><td colspan="3"></td></tr>
<tr><td>5</td><td></td><td colspan="3">Envía un mensaje al usuario indicando que se han registrado exitosamente los datos.</td></tr>
<tr><td rowspan="2"></td><td rowspan="2">Camino de excepción</td><td>paso</td><td colspan="2">acción</td></tr>
<tr><td rowspan="2">1</td><td colspan="2" rowspan="2">Si no se han ingresado todos los datos o se envía un mensaje de error al usuario.</td></tr>
<tr><td></td></tr>
<tr><td colspan="5">Precondiciones el paciente no existe en el sistema</td></tr>
<tr><td colspan="5">Postcondiciones la historia ha sido registrada</td></tr>
</table>

Tabla 18. Caso de uso actualizar historia.

CODIGO REC 015
NOMBRE: gestión historia(actualizar historia)
ACTOR(ES): Administrador del sistema, medico.
DESCRIPCIÓN: el sistema debe permitir modificar la información de una historia

clínica.		
Flujo principal		
	Actor	**Sistema**
1	Selecciona la opción historia	
2		Solicita que digite el código de la historia a actualizar.
3	Digita el código de la historia.	
4	Clic en buscar	
5		Muestra la información de la historia
6	Modifica la información deseada.	
		Registra los cambios realizados.
Precondiciones el paciente exista en el sistema		
Postcondiciones la información ha sido modificada		

Tabla 19. Caso de uso eliminar historia

CODIGO REC 016		
NOMBRE: Gestión historia(eliminar historia)		
ACTOR(ES): Administrador del sistema, medico.		
DESCRIPCIÓN: el sistema debe permitir al usuario eliminar una historia clínica.		
Flujo principal		
	Actor	**Sistema**
1	Selecciona la opción historia	
2		Pide al usuario digite el código de la historia

3	Digita el código de la historia	
4	.clic en buscar	Muestra los datos de la historia
5	Da clic en eliminar	
6		pide confirmación (si o no)
	cancela o acepta la operación	
		Elimina la historia.
Precondiciones	el paciente debe existir en el sistema .	
Postcondiciones	la historia ha sido eliminada	

Tabla 20. Caso de uso consultar historia.

CODICO REC 017		
NOMBRE: Gestión historia(consultar historia)		
ACTOR(ES): Administrador del sistema, medico.		
DESCRIPCIÓN: El sistema debe permitir consultar la historia clínica de un paciente.		
Flujo principal		
	Actor	**Sistema**
1	Selecciona la opción historia.	
2		pide al usuario que digite el código de la historia
3	Digita el código de la historia	
4	Da clic en consultar.	
5	.	Muestra la información de la historia.

Precondiciones	el paciente debe existir en el sistema.
Postcondiciones	la historia ha sido consultada

Tabla 21. Caso de uso reporte historia.

CÓDIGO REC 018			
NOMBRE: Reporte historia clínica			
ACTOR(ES): administrador, medico			
DESCRIPCIÓN: el sistema debe permitir generar un reporte de la historia clínica de un determinado paciente.			
Flujo principal			
	Actor	**Sistema**	
1	Selecciona la opción historia		
2		Pide que digite el código de la historia.	
3	Digita el código de la historia		
4	da clic en buscar		
5		Muestra la historia	
6	Selecciona la opción generar reporte		
7		Genera el reporte estadístico.	
	Camino de excepción	**Paso**	**Acción**
			Datos de ingreso inválidos. Si se digita el nombre de usuario y contraseña incorrectos se le informa al

		1	usuario lo anterior a través de un mensaje de error y se continua a partir del segundo paso del flujo normal
Precondiciones	el usuario y contraseñas deben ser validos		
Postcondiciones	el usuario ha ingresado al sistema		

Tabla 22. Caso de uso agregar médico.

CÓDIGO REC 019			
NOMBRE: gestión medico(agregar medico)			
ACTOR(ES): Administrador del sistema, medico.			
DESCRIPCIÓN: mediante este caso de uso se registra la información personal de un medico el sistema debe permitir ingresar los siguientes datos nombre, apellidos, cedula, fecha de nacimiento, dirección, número de teléfono.			
Flujo principal			
	Actor	**Sistema**	
1	Selecciona la opción medico		
2		solicita los datos personales del medico	
3	Ingresa datos solicitados		
4	Da clic en agregar medico		
5		Envía un mensaje al usuario indicando que se ha registrado exitosamente el medico	
	Camino de excepción	**paso**	**acción**
		1	Si no se han ingresado todos los datos del medico o este ya existe se envía un

			mensaje de error al usuario.
Precondiciones el médico no existe en el sistema			
Postcondiciones el médico ha sido registrado			

Tabla 23. Caso de uso actualizar médico.

CODIGO REC 020		
NOMBRE: Gestión medico(actualizar medico)		
ACTOR(ES): Administrador del sistema, medico.		
DESCRIPCIÓN: el sistema debe permitir modificar la información de un paciente.		
Flujo principal		
	Actor	**Sistema**
1	Selecciona la opción medico	
2		Solicita que digite el médico al que se va a modificar
3	Digita la identificación del medico	
4	Clic en buscar	
5		muestra la información del medico
6	Modifica la información deseada.	
		Registra los cambios realizados.
Precondiciones el médico exista en el sistema		
Postcondiciones la información ha sido modificada		

Tabla 24. Caso de uso eliminar médico.

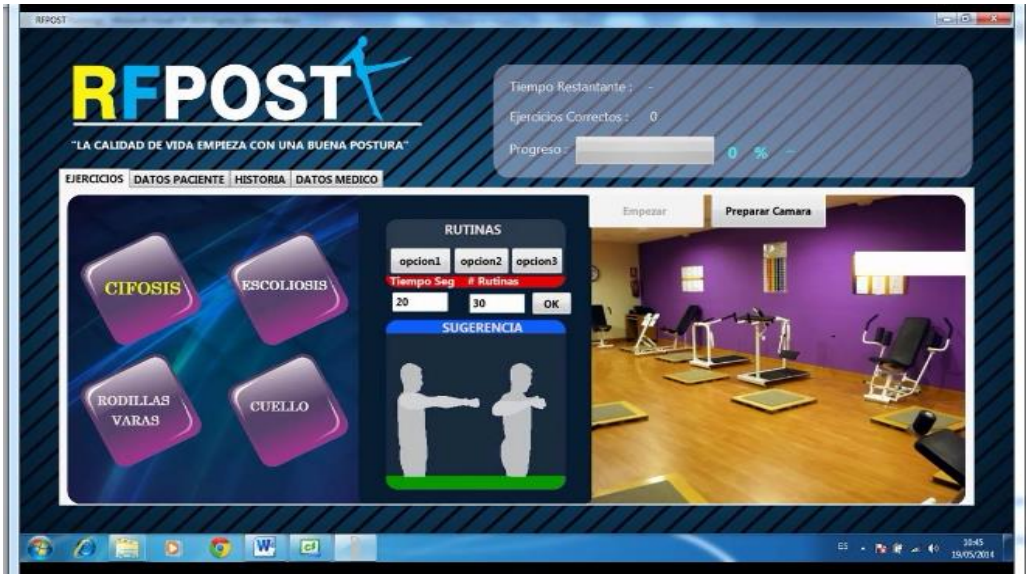
CODIGO REC 021		
NOMBRE: Gestión medico(eliminar medico)		
ACTOR(ES): Administrador del sistema, medico.		
DESCRIPCIÓN: el sistema debe permitir al usuario eliminar información de un médico.		
Flujo principal		
	Actor	**Sistema**
1	Selecciona la opción médico.	
2		Pide al usuario digite el médico.
3	Digita la identificación del médico.	
4	.clic en buscar	Muestra los datos del médico.
5	Da clic en eliminar	
6		pide confirmación (si o no)
7	cancela o acepta la operación	
8		Elimina el médico.
Precondiciones el médico debe existir en el sistema .		
Postcondiciones el medico ha sido eliminado		

Tabla 25. Caso de uso consultar médico.

CODIGO REC 022		
NOMBRE: Gestión medico(consultar medico)		
ACTOR(ES): Administrador del sistema, medico.		
DESCRIPCIÓN: El sistema debe permitir consultar información de un médico.		
Flujo principal		
	Actor	**Sistema**
1	Selecciona la opción médico.	
2		Pide al usuario que digite la identificación del médico.
3	Digita la identificación del médico.	
4	.da clic en consultar	
5		Muestra la información del médico.
Precondiciones el médico debe existir en el sistema.		
Postcondiciones el médico ha sido consultado		

ANEXO E

Diseño de interfaz gráfica con xaml



Código xaml

```
<Window x:Class="RFpost.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="RFPOST" Height="712" Width="1372" Loaded="WindowLoaded" Closed="WindowClosed"
WindowState="Maximized" Icon="/RFpost;component/Images/13.png">
    <Window.Background>
        <ImageBrush ImageSource="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\img4.jpg"></ImageBrush>
    </Window.Background>
    <Window.Resources>
        <ImageBrush x:Key="fondo" ImageSource="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\JESURAG6.jpg">
        </ImageBrush>
        <ImageBrush x:Key="fondo2" ImageSource="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\fondo1.png">
        </ImageBrush>
        <RadialGradientBrush x:Key="GlowFX" GradientOrigin=".5,1" Center=".5,1">
            <!--  <GradientStop Offset="0" Color="#12ECE8"></GradientStop>-->
            <GradientStop Offset=".1" Color="Purple"></GradientStop>
            <GradientStop Offset=".1" Color="White"></GradientStop>
            <GradientStop Offset="1" Color="Aquamarine"></GradientStop>
         </RadialGradientBrush>

    </Window.Resources>



    <Grid Height="628" Width="1285">
        <Grid.ColumnDefinitions>
            <ColumnDefinition Width="648*" />
            <ColumnDefinition Width="637*" />
        </Grid.ColumnDefinitions>
        <!--  <Image Source="/DTWGestureRecognition;component/HeaderBG.png" />-->
        <TabControl  Height="482" HorizontalAlignment="Left" Name="tabControl1" VerticalAlignment="Top"
Width="1260" Margin="12,157,0,0" Grid.ColumnSpan="2">
            <TabItem Header="EJERCICIOS" Name="tabItem1" FontWeight="Bold" FontSize="15">
                <Grid Height="434" Width="1236" Background="{StaticResource fondo}">
```

```
                    <Button Content="opcion2" Height="36" HorizontalAlignment="Right" Margin="0,74,629,0"
Name="opc2" VerticalAlignment="Top" Width="75" Click="button6_Click" IsEnabled="False" />
                    <Button Content="Preparar Camara" Height="49" HorizontalAlignment="Right"
Margin="0,0,192,388" Name="button4" VerticalAlignment="Bottom" Width="159" Click="button4_Click" />
                    <Button Visibility="Visible"  Content="Empezar " Margin="0,0,360,388"
Name="bEmpezarCifosis1" Click="button1_Click" HorizontalAlignment="Right" Width="159"  Height="49"
VerticalAlignment="Bottom" />
                    <Button Content="opcion1" HorizontalAlignment="Left" Margin="444,74,0,324" Name="opc1"
Width="83" Click="button2_Click" IsEnabled="False" />
                    <TextBox FontSize="14" Name="status" Margin="1083,79,0,0" Focusable="False"
BorderThickness="0" Text="" TextAlignment="Left" HorizontalAlignment="Left" VerticalAlignment="Top"
Width="153" Height="34" />
                    <Image Name="depthImage" Stretch="Uniform" Margin="829,227,141,38" />
                    <Canvas Visibility="Hidden" Name="skeletonCanvas" Background="Black"
ClipToBounds="True" Margin="1062,115,-5,296" />
                    <Label  Content="CIFOSIS" MouseMove="MouseMoveCifosis"
MouseLeave="labelCifosis_MouseLeave" MouseLeftButtonDown="labelCifosis_MouseLeftButtonDown" Cursor="Hand"
Height="41" HorizontalAlignment="Left" Margin="46,110,0,0" Name="labelCifosis" VerticalAlignment="Top"
Width="116" Foreground="#FFECE3E3" FontSize="18" FontWeight="Bold" FontFamily="Century" />
                    <Label Content="ESCOLIOSIS" MouseMove="MouseMoveEscoliosis"
MouseLeave="Escoliosis_MouseLeave" MouseLeftButtonDown="labelEscoliosis_MouseLeftButtonDown" Cursor="Hand"
FontFamily="Century" FontSize="18" FontWeight="Bold" Foreground="#FFECE3E3" Height="41"
Margin="0,110,836,0" Name="labelEscoliosis" VerticalAlignment="Top" HorizontalAlignment="Right" Width="166"
/>
                    <Label Content="RODILLAS" MouseMove="MouseMoveRodillas"
MouseLeave="Rodillas_MouseLeave" MouseLeftButtonDown="labelRodillas_MouseLeftButtonDown" Cursor="Hand"
FontFamily="Century" FontSize="18" FontWeight="Bold" Foreground="#FFECE3E3" Height="41"
HorizontalAlignment="Left" Margin="36,271,0,0" Name="labelRodillas" VerticalAlignment="Top" Width="136" />
                    <Label Content="VARAS"  FontFamily="Century" FontSize="18" FontWeight="Bold"
Foreground="#FFECE3E3" Height="41" HorizontalAlignment="Left" Margin="56,299,0,0" Name="labelVaras"
VerticalAlignment="Top" Width="116" />
                    <Label Content="CUELLO" MouseMove="MouseMoveCuello" MouseLeave="Cuello_MouseLeave"
MouseLeftButtonDown="labelCuello_MouseLeftButtonDown" Cursor="Hand" FontFamily="Century" FontSize="18"
FontWeight="Bold" Foreground="#FFECE3E3" HorizontalAlignment="Left" Margin="246,279,0,114"
Name="labelCuello" Width="128" />
                    <Button Content="opcion3" HorizontalAlignment="Right" Margin="0,74,555,324" Name="opc3"
Width="70" IsEnabled="False" Click="opc3_Click" />
                    <Label Content="RUTINAS" Height="31" HorizontalAlignment="Left" Margin="514,31,0,0"
Name="label6" VerticalAlignment="Top" Foreground="#FFECDDDD" FontSize="18" FontStyle="Normal"
FontWeight="Bold" />
                    <Label Content="SUGERENCIA" FontSize="18" FontStyle="Normal" FontWeight="Bold"
Foreground="#FFECDDDD" Height="31" HorizontalAlignment="Left" Margin="511,168,0,0" Name="label7"
VerticalAlignment="Top" />
                    <Label Content="-" Height="28" HorizontalAlignment="Left" Margin="1046,181,0,0"
Name="label11" VerticalAlignment="Top" Width="158" />
                    <Button Content="Empezarc2 " Height="49" HorizontalAlignment="Right"
Margin="0,0,360,388" Name="bEmpezarCifosis2" VerticalAlignment="Bottom" Visibility="Hidden" Width="159"
Click="bEmpezarCifosis2_Click" />
                    <Button Content="Empezar " Height="49" HorizontalAlignment="Right" Margin="0,0,360,388"
Name="bEmpezarCifosis3" VerticalAlignment="Bottom" Visibility="Hidden" Width="159" Click="button1_Click_1"
/>
                    <Button Visibility="Hidden" Content="opcion2e" Height="36" IsEnabled="False"
Margin="532,74,629,0" Name="ope2" VerticalAlignment="Top" Click="ope2_Click" />
                    <Button Visibility="Hidden" Content="opcion1e" IsEnabled="False" Margin="444,74,0,324"
Name="ope1" HorizontalAlignment="Left" Width="83" Click="ope1_Click" />
                    <Button Visibility="Hidden" Content="opcion3e" HorizontalAlignment="Right"
IsEnabled="False" Margin="0,0,555,324" Name="ope3" Width="70" Height="36" VerticalAlignment="Bottom"
Click="ope3_Click" />
                    <Button Content="opcion2" Visibility="Hidden" Height="36" HorizontalAlignment="Left"
IsEnabled="False" Margin="532,74,0,0" Name="opcu2" VerticalAlignment="Top" Width="75" Click="opcu2_Click"
/>
                    <Button Content="opcion1" Visibility="Hidden" HorizontalAlignment="Left"
IsEnabled="False" Margin="444,74,0,324" Name="opcu1" Width="83" Click="opcu1_Click" />
                    <Button Content="opcion3" Visibility="Hidden" HorizontalAlignment="Right"
IsEnabled="False" Margin="0,74,555,324" Name="opcu3" Width="70" Click="opcu3_Click" />
                    <Button Content="opcion2" Visibility="Hidden" Height="36" HorizontalAlignment="Left"
IsEnabled="False" Margin="532,74,0,0" Name="oprv2" VerticalAlignment="Top" Width="75" Click="oprv2_Click"
/>
                    <Button Content="opcion1" Visibility="Hidden" HorizontalAlignment="Left"
IsEnabled="False" Margin="444,74,0,324" Name="oprv1" Width="83" Click="oprv1_Click" />
                    <Button Content="opcion3" Visibility="Hidden" HorizontalAlignment="Right"
IsEnabled="False" Margin="0,74,555,324" Name="oprv3" Width="70" Click="oprv3_Click" />
                    <
```

```
                    <!-- <MediaElement Height="96" HorizontalAlignment="Left" LoadedBehavior="Play"
Margin="968,141,16,44" Source="/RFpost;component/Images/g2.gif" Stretch="Fill" VerticalAlignment="Top"
Width="146"></MediaElement>-->
                </Grid>
            </TabItem>
            <TabItem Header="DATOS PACIENTE" Name="tabItem2" FontWeight="Bold" FontSize="15">
                <Grid Background="{StaticResource fondo2}">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="122*" />
                        <ColumnDefinition Width="137*" />
                        <ColumnDefinition Width="991*" />
                    </Grid.ColumnDefinitions>
                    <Button  Height="48" Name="bAgregardatos" HorizontalAlignment="Left" Margin="2,135,0,0"
VerticalAlignment="Top" Width="242" Grid.ColumnSpan="2" Click="bAgregardatos_Click" >
                        <DockPanel>
                            <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\addusuario.png" Height="46" Width="40"></Image>
                            <TextBlock Height="33" Width="141">Agregar Paciente</TextBlock>
                        </DockPanel>

                    </Button>
                    <Button Height="50" HorizontalAlignment="Left" Margin="2,189,0,0"
Name="bmodificardatos" VerticalAlignment="Top" Width="242" Grid.ColumnSpan="2"
Click="bmodificardatos_Click">
                        <DockPanel>
                            <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\modusuario.png" Height="46" Width="40"></Image>
                            <TextBlock Height="24" Width="140">Modificar Paciente</TextBlock>
                        </DockPanel>
                    </Button>
                    <Button  Height="50" HorizontalAlignment="Left" Margin="4,248,0,0" Name="button3"
VerticalAlignment="Top" Width="240" Grid.ColumnSpan="2" Click="button3_Click_1" ></Button>
                    <Button  Height="48" HorizontalAlignment="Left" Margin="4,307,0,0" Name="button5"
VerticalAlignment="Top" Width="240" Grid.ColumnSpan="2" Click="button5_Click_2"></Button>
                    <GroupBox Header="Listado de Registro" Height="401" Foreground="White"
HorizontalAlignment="Left" Margin="447,29,0,0" Name="groupBox1" VerticalAlignment="Top" Width="491"
FontFamily="Century" FontSize="22" Grid.Column="2">
                        <Grid>
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="190*" />
                                <ColumnDefinition Width="370*" />
                            </Grid.ColumnDefinitions>
                            <DataGrid AutoGenerateColumns="true" Height="370" HorizontalAlignment="Left"
Margin="6,0,0,0" Name="dataGridRegistros" VerticalAlignment="Top" Width="474" Grid.ColumnSpan="2"
Foreground="#FF080841" />
                        </Grid>
                    </GroupBox>
                    <GroupBox Header="Datos Personales" Height="405" FontFamily="Century"
FontWeight="Black" Foreground="White" HorizontalAlignment="Left" Margin="136,25,0,0" Name="groupBox2"
VerticalAlignment="Top" Width="442" FontSize="24" Grid.Column="1" Grid.ColumnSpan="2">
                        <Grid>
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="190*" />
                                <ColumnDefinition Width="370*" />
                            </Grid.ColumnDefinitions>
                            <TextBox Height="51" HorizontalAlignment="Left" Margin="32,35,0,0"
Name="txtidentificacion" VerticalAlignment="Top" Width="246" Grid.Column="1" />
                            <Label Content="Identificacion " Height="50" HorizontalAlignment="Left"
Margin="-2,47,0,0" Name="label14" VerticalAlignment="Top" Width="174" Grid.ColumnSpan="2"
Foreground="#FFF3E50F" />
                            <TextBox Height="51" HorizontalAlignment="Left" Margin="32,108,0,0"
Name="txtnombres" VerticalAlignment="Top" Width="246" Grid.Column="1" />
                            <Label Content="Nombres :" Grid.ColumnSpan="2" Height="50"
                            <Label Content="Direccion :" Grid.ColumnSpan="2" Height="50"
HorizontalAlignment="Left" Margin="4,249,0,0" Name="label17" VerticalAlignment="Top" Width="174"
Foreground="#FFF3E50F" />
                            <TextBox Height="51" HorizontalAlignment="Left" Margin="30,303,0,0"
Name="txtcelular" VerticalAlignment="Top" Width="246" Grid.Column="1" />
                            <Label Content="Celular :" Grid.ColumnSpan="2" Height="50"
HorizontalAlignment="Left" Margin="6,313,0,0" Name="label18" VerticalAlignment="Top" Width="174"
Foreground="#FFF3E50F" />
                        </Grid>
                    </GroupBox>
```

```
                    <GroupBox Grid.ColumnSpan="2" Header="Buscar x Identificacion" Height="75"
HorizontalAlignment="Left" Margin="4,33,0,0" Name="groupBox4" VerticalAlignment="Top" Width="240"
Foreground="White" FontSize="18" FontFamily="Century">
                        <Grid>
                            <TextBox Height="30" HorizontalAlignment="Left" Margin="14,6,0,0"
Name="txtbuscar" VerticalAlignment="Top" Width="144" />
                            <Button Content="OK" Height="35" HorizontalAlignment="Right" Margin="0,3,10,0"
Name="button2" VerticalAlignment="Top" Width="48" Click="button2_Click_1" />
                        </Grid>
                    </GroupBox>
                    <DockPanel Grid.ColumnSpan="2" Margin="32,247,56,155">
                        <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\mostrar.png" Height="46" Width="40"></Image>
                        <TextBlock Height="26" Width="131">Listado Paciente</TextBlock>
                    </DockPanel>
                    <DockPanel Grid.ColumnSpan="2" Margin="32,305,64,99">
                        <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\cancelar.png" Height="46" Width="40"></Image>
                        <TextBlock Height="30" Width="123">Eliminar Paciente</TextBlock>
                    </DockPanel>
                    <Button Grid.ColumnSpan="2" Height="48" HorizontalAlignment="Left" Margin="4,365,0,0"
Name="blimpiardatos" VerticalAlignment="Top" Width="240" Click="blimpiardatos_Click">
                        <DockPanel>
                            <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\limpiar.png" Height="46" Width="40"></Image>
                            <TextBlock Height="24" Width="140">Limpiar Campos</TextBlock>

            </TabItem>
            <TabItem Header="HISTORIA" FontWeight="Bold" FontSize="15">
                <Grid Background="{StaticResource fondo2}">
                    <GroupBox Header="Opciones" Height="398" HorizontalAlignment="Left" Margin="33,20,0,0"
Name="groupBox5" VerticalAlignment="Top" Width="290" Foreground="#FFFFFCFC" FontSize="24"
FontFamily="Century">
                        <Grid>
                            <TextBox Height="33" HorizontalAlignment="Left" Margin="10,47,0,0"
Name="textBoxcodhistoria" VerticalAlignment="Top" Width="142" />
                            <Label Content="Cod Historia" Foreground="#FFF3E50F" Height="50"
HorizontalAlignment="Left" Margin="2,8,0,0" Name="label21" VerticalAlignment="Top" Width="174"
FontFamily="Century" />
                            <Button Content="Buscar" Height="41" HorizontalAlignment="Right"
Margin="0,38,9,0" Name="button6" VerticalAlignment="Top" Width="101" Click="button6_Click_2"
FontFamily="Arial" FontSize="18" />
                            <Button  Height="44" HorizontalAlignment="Left" Margin="1,148,0,0"
Name="button7" VerticalAlignment="Top" Width="270" Click="button7_Click_1" FontSize="24"
FontFamily="Arial">
                                <DockPanel>
                                    <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\addhistoria.png" Height="46" Width="40"></Image>
                                    <TextBlock Height="24" Width="134" FontSize="15">Agregar
Historia</TextBlock>
                                </DockPanel>

                            </Button>
                            <Button  Height="48" HorizontalAlignment="Left" Margin="2,195,0,0"
Name="button8" VerticalAlignment="Top" Width="267" Click="button8_Click">
                                <DockPanel>
                                    <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\modhistoria.png" Height="46" Width="40"></Image>
                                    <TextBlock FontFamily="Arial" FontSize="15" Height="30"
Width="134">Actualizar Historia</TextBlock>
                                </DockPanel>
                            </Button>
                            <Button  Height="51" HorizontalAlignment="Left" Margin="2,248,0,0"
Name="button9" VerticalAlignment="Top" Width="267" Click="button9_Click">
                                <DockPanel>
                                    <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\elimhistoria.png" Height="46" Width="40"></Image>
                                    <TextBlock FontFamily="Arial" Height="20" Width="131"
FontSize="15">Eliminar Historia</TextBlock>
                                </DockPanel>
                            </Button>
                            <TextBox Height="33" HorizontalAlignment="Left" Margin="94,97,0,0"
Name="txtmedicohistoria" VerticalAlignment="Top" Width="178" />
```

```
                        <Label Content="Medico" Foreground="#FFF3E50F" Height="50"
HorizontalAlignment="Left" Margin="-2,92,0,0" Name="label30" VerticalAlignment="Top" Width="126" />
                        <Button Height="51" HorizontalAlignment="Left" Margin="2,302,0,0"
Name="button11" VerticalAlignment="Top" Width="267" Click="button11_Click">
                            <DockPanel>
                                <Image Height="46" Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\imgpdf.jpg" Width="40" />
                                <TextBlock FontFamily="Arial" FontSize="15" Height="20"
Width="131">Crear PDF</TextBlock>
                            </DockPanel>
                        </Button>
                    </Grid>
                </GroupBox>
                <GroupBox Header="Historia" Height="398" HorizontalAlignment="Left" Margin="333,22,0,0"
Name="groupBox6" VerticalAlignment="Top" Width="888" Foreground="#FFFFFCFC" FontSize="24"
FontFamily="Century">
                    <Grid>
                        <TextBox Height="177" HorizontalAlignment="Left" Margin="14,168,0,0"
Name="txtValoracion" VerticalAlignment="Top" Width="837" AcceptsReturn="True" TextWrapping="Wrap"
VerticalScrollBarVisibility="Auto" FontFamily="Times New Roman" FontSize="18" />
                        <TextBox Height="33" HorizontalAlignment="Left" Margin="8,32,0,0"
                        <Label Content="Tipo Ejercicio" Foreground="#FFF3E50F" Height="50"
HorizontalAlignment="Left" Margin="560,-4,0,0" Name="label29" VerticalAlignment="Top" Width="174" />
                    </Grid>
                </GroupBox>
            </Grid>
        </TabItem>
        <TabItem Header="DATOS MEDICO" Name="tabItem3" FontWeight="Bold" FontSize="15">
            <Grid Background="{StaticResource fondo2}">
                <Grid.ColumnDefinitions>
                    <ColumnDefinition Width="122*" />
                    <ColumnDefinition Width="137*" />
                    <ColumnDefinition Width="991*" />
                </Grid.ColumnDefinitions>
                <Button  Height="48" Name="bAgregardatosMedico" HorizontalAlignment="Left"
Margin="2,135,0,0"  VerticalAlignment="Top" Width="242" Grid.ColumnSpan="2"
Click="bAgregardatosMedico_Click" >
                    <DockPanel>
                        <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\addmedico1.jpg" Height="46" Width="40"></Image>
                        <TextBlock Height="33" Width="141">Agregar Medico</TextBlock>
                    </DockPanel>

                </Button>
                <Button Height="50" HorizontalAlignment="Left" Margin="2,189,0,0"
Name="bmodificardatosMedico" VerticalAlignment="Top" Width="242" Grid.ColumnSpan="2"
Click="bmodificardatosMedico_Click">
                    <DockPanel>
                        <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\modusuario.png" Height="46" Width="40"></Image>
                        <TextBlock Height="24" Width="140">Modificar Medico</TextBlock>
                    </DockPanel>
                </Button>
                <Button  Height="50" HorizontalAlignment="Left" Margin="4,248,0,0" Name="blistarMedico"
VerticalAlignment="Top" Width="240" Grid.ColumnSpan="2" Click="blistarMedicos_Click_1" ></Button>
                <Button  Height="48" HorizontalAlignment="Left" Margin="4,307,0,0"
Name="beliminarMedico" VerticalAlignment="Top" Width="240" Grid.ColumnSpan="2"
Click="bEliminarMedico_Click_2"></Button>
                <GroupBox Header="Listado de Registro" Height="401" Foreground="White"
HorizontalAlignment="Left" Margin="447,29,0,0" Name="groupBox11" VerticalAlignment="Top" Width="491"
FontFamily="Century" FontSize="22" Grid.Column="2">
                    <Grid>
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="190*" />
                            <ColumnDefinition Width="370*" />
                        </Grid.ColumnDefinitions>
                        <DataGrid AutoGenerateColumns="true" Height="370" HorizontalAlignment="Left"
Margin="6,0,0,0" Name="dataGridRegistrosMedicos" VerticalAlignment="Top" Width="474" Grid.ColumnSpan="2"
Foreground="#FF080841" />
                    </Grid>
                </GroupBox>
                <GroupBox Header="Datos Personales" Height="405" FontFamily="Century"
FontWeight="Black" Foreground="White" HorizontalAlignment="Left" Margin="136,25,0,0" Name="groupBox22"
VerticalAlignment="Top" Width="442" FontSize="24" Grid.Column="1" Grid.ColumnSpan="2">
```

```
                        <Grid>
                            <Grid.ColumnDefinitions>
                                <ColumnDefinition Width="190*" />
                                <ColumnDefinition Width="370*" />
                            </Grid.ColumnDefinitions>
                            <TextBox Height="51" HorizontalAlignment="Left" Margin="32,35,0,0"
                            <Label Content="Celular :" Grid.ColumnSpan="2" Height="50"
HorizontalAlignment="Left" Margin="6,313,0,0" Name="label188" VerticalAlignment="Top" Width="174"
Foreground="#FFF3E50F" />
                        </Grid>
                    </GroupBox>
                    <GroupBox Grid.ColumnSpan="2" Header="Buscar x Identificacion" Height="75"
HorizontalAlignment="Left" Margin="4,33,0,0" Name="groupBox44" VerticalAlignment="Top" Width="240"
Foreground="White" FontSize="18" FontFamily="Century">
                        <Grid>
                            <TextBox Height="30" HorizontalAlignment="Left" Margin="14,6,0,0"
Name="txtbuscarMedico" VerticalAlignment="Top" Width="144" />
                            <Button Content="OK" Height="35" HorizontalAlignment="Right" Margin="0,3,10,0"
Name="bbuscarMedico" VerticalAlignment="Top" Width="48" Click="buscarMedico_Click_1" />
                        </Grid>
                    </GroupBox>
                    <DockPanel Grid.ColumnSpan="2" Margin="32,247,56,155">
                        <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\mostrar.png" Height="46" Width="40"></Image>
                        <TextBlock Height="26" Width="131">Listado Medico</TextBlock>
                    </DockPanel>
                    <DockPanel Grid.ColumnSpan="2" Margin="32,305,64,99">
                        <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\elimmedico1.jpg" Height="46" Width="40"></Image>
                        <TextBlock Height="30" Width="123">Eliminar Medico</TextBlock>
                    </DockPanel>
                    <Button Grid.ColumnSpan="2" Height="48" HorizontalAlignment="Left" Margin="4,365,0,0"
Name="blimpiardatosMedico" VerticalAlignment="Top" Width="240" Click="blimpiardatosMedico_Click">
                        <DockPanel>
                            <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\limpiar.png" Height="46" Width="40"></Image>
                            <TextBlock Height="24" Width="140">Limpiar Campos</TextBlock>
                        </DockPanel>
                    </Button>
                </Grid>

            </TabItem>
        </TabControl>
        <Image Height="125" HorizontalAlignment="Left" Margin="26,-28,0,0" Name="image1" Stretch="Fill"
VerticalAlignment="Top" Width="486" Source="/RFpost;component/Images/img3.png" />
        <Label Content="&amp;quot;LA CALIDAD DE VIDA EMPIEZA CON UNA BUENA POSTURA&amp;quot;" Height="40"
HorizontalAlignment="Left" Margin="24,108,0,0" Name="label5" VerticalAlignment="Top" Width="480"
Foreground="#FFF0E9E9" FontSize="18" Grid.Column="1" />
        <Label Content="Progreso :" FontSize="18" Foreground="#FFF0E9E9" Height="36"
HorizontalAlignment="Left" Margin="627,112,0,0" Name="label12" VerticalAlignment="Top" Grid.ColumnSpan="2"
/>
        <ProgressBar Height="40" HorizontalAlignment="Left" Margin="75,112,0,0" Name="progressBar1"
VerticalAlignment="Top" Width="191" Grid.Column="1" Maximum="100" />
        <Label Content="0" Height="39" HorizontalAlignment="Left" Margin="280,114,0,0" Name="label3"
            <Grid Height="116" Width="454">
                <TextBox Height="39" HorizontalAlignment="Right" Margin="0,12,191,0" Name="txtusuario"
VerticalAlignment="Top" Width="166" />
                <Label Content="Usuario :" HorizontalAlignment="Left" Margin="-6,17,0,61" Name="label19"
Width="92" Foreground="#FFEFEF18" FontSize="20" />
                <Label Content="Contraseña :" Height="43" HorizontalAlignment="Left" Margin="-21,50,0,0"
Name="label20" VerticalAlignment="Top" Foreground="#FFEFEF18" FontSize="20" Width="127" />
                <PasswordBox Height="37" HorizontalAlignment="Left" Margin="97,56,0,0" Name="pwcontrasena"
VerticalAlignment="Top" Width="166" />
                <Button  Height="59" HorizontalAlignment="Left" Margin="281,20,0,0" Name="button1"
VerticalAlignment="Top" Width="147" Click="button1_Click_2" Foreground="#FF4631DF" ></Button>
                <DockPanel Margin="292,26,40,37">
                    <Image Source="C:\Users\JESU\Documents\Visual Studio
2010\Projects\RFpost\imagenes\llave.png" Height="46" Width="40"></Image>
                    <TextBlock Height="26" Width="54" FontSize="15">Entrar</TextBlock>
                </DockPanel>
            </Grid>
        </GroupBox>
    </Grid>
</Window>
```

ANEXO F

Codigo fuente de la clase ReconocerGesto.cs

Clase ReconocerGesto.cs

```
using System;
using System.Collections.Generic;
using System.Collections;
using System.Linq;
using System.Text;

namespace RFpost
{
    class ReconocerGesto
    {
        /// <summary>
        /// Tamano del vector de observaciones.
        /// </summary>
        private readonly int _dimension;

        /// <summary>
        /// Distancia máxima entre las últimas observaciones de cada secuencia.
        /// </summary>
        private readonly double _firstThreshold;

        /// <summary>
        /// La longitud mínima de un gesto antes de que pueda ser reconocido
        /// </summary>
        private readonly double _minimumLength;

        /// <summary>
        ///   Determina la máxima distancia DTW que
        /// puede haber entre dos gestos para que puedan ser considerados iguales.
        /// </summary>
        private readonly double _globalThreshold;

        /// <summary>
        /// Los nombres gesto. Índice coincide con el de la matriz de secuencias en _sequences
        /// </summary>
        private readonly ArrayList _labels;

        /// <summary>
        ///Pasos verticales u horizontales máximos en una fila.
        /// </summary>
        private readonly int _maxSlope;

        /// <summary>
        /// secuencias de gestos grabadas.
        /// </summary>
        private readonly ArrayList _sequences;

        /// <summary>
        /// inicializa una nueva instancia de la clase Dtwreconocergesto.
        /// Primera DTW constructor
        /// </summary>
        /// <param name="dim">tamaño del vector</param>
        /// <param name="threshold">Distancia máxima entre las últimas observaciones de cada
secuencia</param>
        /// <param name="firstThreshold">Minimo umbral</param>
        public ReconocerGesto(int dim, double threshold, double firstThreshold, double minLen)
        {
            _dimension = dim;
            _sequences = new ArrayList();
            _labels = new ArrayList();
            _globalThreshold = threshold;
            _firstThreshold = firstThreshold;
            _maxSlope = int.MaxValue;
            _minimumLength = minLen;
        }
```

```
        /// <summary>
        ///  inicializa una nueva instancia de la clase Dtwreconocergesto.
        /// segunda DTW constructor
        /// </summary>
        /// <param name="dim">Tamano del vector</param>
        /// <param name="threshold">Distancia máxima entre las últimas observaciones de cada
secuencia</param>
        /// <param name="firstThreshold">Minimo umbral</param>
        /// <param name="ms">Maximos pasos horizontales o verticales en una fila</param>
        public ReconocerGesto(int dim, double threshold, double firstThreshold, int ms, double minLen)
        {
            _dimension = dim;
            _sequences = new ArrayList();
            _labels = new ArrayList();
            _globalThreshold = threshold;
            _firstThreshold = firstThreshold;
            _maxSlope = ms;
            _minimumLength = minLen;
        }

        /// <summary>
        /// Añadir una seqence con una etiqueta a la biblioteca de secuencias conocidas.
        /// El gesto debe comenzar en la primera observación de la secuencia y finalizará el último.
        /// Las secuencias pueden tener diferentes longitudes.
        /// </summary>
        /// <param name="seq">la secuencia</param>
        /// <param name="lab">nombre de la secuencia</param>
        public void AddOrUpdate(ArrayList seq, string lab)
        {
            // Primero comprobamos si ya hay una grabación de esta etiqueta. Si es así sobrescribir, añadir
lo contrario una nueva entrada
            int existingIndex = -1;

            for (int i = 0; i < _labels.Count; i++)
            {
                if ((string)_labels[i] == lab)
                {
                    existingIndex = i;
                }
            }

            // Si tenemos un partido de eliminar las entradas en el índice existente
            //para evitar duplicados.Vamos a añadir las nuevas entradas más tarde de todos modos

            if (existingIndex >= 0)
            {
                _sequences.RemoveAt(existingIndex);
                _labels.RemoveAt(existingIndex);
            }

            // adicione nuevas entradas
            _sequences.Add(seq);
            _labels.Add(lab);
        }

        /// <summary>
        /// Reconocer gesto en el orden indicado
        /// Siempre va a suponer que el gesto se termina en la última observación de esa secuencia.
        /// Si la distancia entre las últimas observaciones de cada secuencia es demasiado grande, o si la
distancia total entre los dos DTW secuencia es demasiado grande, será reconocido ningún gesto.
        /// </summary>
        /// <param name="seq">la secuencia a reconocer</param>
        /// <returns>el nombre del gesto reconocido</returns>
        public string Recognize(ArrayList seq)
        {
            double minDist = double.PositiveInfinity;
            string classification = "__UNKNOWN";
            for (int i = 0; i < _sequences.Count; i++)
            {
                var example = (ArrayList)_sequences[i];
                ////Debug.WriteLine(Dist2((double[]) seq[seq.Count - 1], (double[]) example[example.Count -
1]));
```

```
                if (Dist2((double[])seq[seq.Count - 1], (double[])example[example.Count - 1]) <
_firstThreshold)
                {
                    double d = Dtw(seq, example) / example.Count;
                    if (d < minDist)
                    {
                        minDist = d;
                        classification = (string)_labels[i];
                    }
                }
            }

            return (minDist < _globalThreshold ? classification : "__UNKNOWN") + " "
/*+minDist.ToString()*/;
        }

        //Incio-prueba

        public int Recognize1(ArrayList seq)
        {
            int cont = 0;
            double minDist = double.PositiveInfinity;
            // int classification = 0;
            for (int i = 0; i < _sequences.Count; i++)
            {
                var example = (ArrayList)_sequences[i];
                ////Debug.WriteLine(Dist2((double[]) seq[seq.Count - 1], (double[]) example[example.Count -
1]));
                if (Dist2((double[])seq[seq.Count - 1], (double[])example[example.Count - 1]) <
_firstThreshold)
                {
                    double d = Dtw(seq, example) / example.Count;
                    if (d < minDist)
                    {
                        minDist = d;
                        //    classification =(int)_labels[i];
                        cont++;
                    }
                }
            }

            //    return (minDist < _globalThreshold ? classification : 0) /*+minDist.ToString()*/;
            return cont;
        }

        //fin-prueba

        /// <summary>
        /// Recupera una represeantation texto del _label y su _sequence asociado
        /// Para el uso en dispaying información de depuración y para guardar en archivo
        /// </summary>
        /// <returns>Una cadena que contiene todos los gestos grabados y sus nombres</returns>
        public string RetrieveText()
        {
            string retStr = String.Empty;

            if (_sequences != null)
            {
                // Iterar a traves de cada gesto.
                for (int gestureNum = 0; gestureNum < _sequences.Count; gestureNum++)
                {
                    // Echo the label
                    retStr += _labels[gestureNum] + "\r\n";

                    int frameNum = 0;

                    //Iterar a través de cada cuadro de este gesto
                    foreach (double[] frame in ((ArrayList)_sequences[gestureNum]))
                    {
                        // Extract each double
                        foreach (double dub in (double[])frame)
                        {
                            retStr += dub + "\r\n";
                        }
```

```
                    // Signifies end of this double
                    retStr += "~\r\n";

                    frameNum++;
                }

                // Signifies end of this gesture
                retStr += "----";
                if (gestureNum < _sequences.Count - 1)
                {
                    retStr += "\r\n";
                }
            }
        }

        return retStr;
    }

    /// <summary>
    /// Calcule la distancia min DTW entre seq2 y todos los finales posibles de seq1.
    /// </summary>

    public double Dtw(ArrayList seq1, ArrayList seq2)
    {
        // Init
        var seq1R = new ArrayList(seq1);
        seq1R.Reverse();
        var seq2R = new ArrayList(seq2);
        seq2R.Reverse();
        var tab = new double[seq1R.Count + 1, seq2R.Count + 1];
        var slopeI = new int[seq1R.Count + 1, seq2R.Count + 1];
        var slopeJ = new int[seq1R.Count + 1, seq2R.Count + 1];

        for (int i = 0; i < seq1R.Count + 1; i++)
        {
            for (int j = 0; j < seq2R.Count + 1; j++)
            {
                tab[i, j] = double.PositiveInfinity;
                slopeI[i, j] = 0;
                slopeJ[i, j] = 0;
            }
        }

        tab[0, 0] = 0;

        // Cálculo dinámico de la matriz DTW.
        // En primer lugar, construye una matriz de distancias acumuladas entre
        // En primer lugar, construye una matriz de distancias acumuladas entre cada par de
        //elementos de las secuencias y a continuación encuentra la mejor manera de alinearlas
secuencias buscando el camino óptimo.
        //El camino de alineamiento optimo, es un camino de alineamiento que entre todos los posibles
caminos es el costo totalminimo. y la distancia DTW asociada a el
        for (int i = 1; i < seq1R.Count + 1; i++)
        {
            for (int j = 1; j < seq2R.Count + 1; j++)
            {
                if (tab[i, j - 1] < tab[i - 1, j - 1] && tab[i, j - 1] < tab[i - 1, j] &&
                    slopeI[i, j - 1] < _maxSlope)
                {
                    tab[i, j] = Dist2((double[])seq1R[i - 1], (double[])seq2R[j - 1]) + tab[i, j - 1];
                    slopeI[i, j] = slopeJ[i, j - 1] + 1;
                    slopeJ[i, j] = 0;
                }
                else if (tab[i - 1, j] < tab[i - 1, j - 1] && tab[i - 1, j] < tab[i, j - 1] &&
                         slopeJ[i - 1, j] < _maxSlope)
                {
                    tab[i, j] = Dist2((double[])seq1R[i - 1], (double[])seq2R[j - 1]) + tab[i - 1, j];
                    slopeI[i, j] = 0;
                    slopeJ[i, j] = slopeJ[i - 1, j] + 1;
                }
                else
                {
```

```
                        tab[i, j] = Dist2((double[])seq1R[i - 1], (double[])seq2R[j - 1]) + tab[i - 1, j -
1];
                        slopeI[i, j] = 0;
                        slopeJ[i, j] = 0;
                    }
                }
            }

            // Encuentra el mejor entre seq2 y un final (sufijo) del seq1.
            double bestMatch = double.PositiveInfinity;
            for (int i = 1; i < (seq1R.Count + 1) - _minimumLength; i++)
            {
                if (tab[i, seq2R.Count] < bestMatch)
                {
                    bestMatch = tab[i, seq2R.Count];
                }
            }

            return bestMatch;
        }

        /// <summary>
        /// Calcula una distancia 2 entre dos observaciones. (también conocido como la distancia
manhattan).        .
        /// </summary>
        /// <param name="a">Point a (double)</param>
        /// <param name="b">Point b (double)</param>
        /// <returns>Manhattan distancia entre dos puntos</returns>
        private double Dist1(double[] a, double[] b)
        {
            double d = 0;
            for (int i = 0; i < _dimension; i++)
            {
                d += Math.Abs(a[i] - b[i]);
            }

            return d;
        }

        /// <summary>
        /// Calcula una distancia 2 entre dos observaciones. (también conocido como la distancia
euclidiana).
        /// </summary>
        /// <param name="a">Point a (double)</param>
        /// <param name="b">Point b (double)</param>
        /// <returns>distancia euclidiana entre dos puntos</returns>
        private double Dist2(double[] a, double[] b)
        {
            double d = 0;
            for (int i = 0; i < _dimension; i++)
            {
                d += Math.Pow(a[i] - b[i], 2);
            }

            return Math.Sqrt(d);
        }

    }
}
```

ANEXO G

Codigo fuente de la clase DatosEsqueleto2DExtract.cs

Clase DatosEsqueleto2DExtract.cs

```
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace RFpost
{

    using System;
    using System.Windows;
    using Microsoft.Research.Kinect.Nui;

    class DatosEsqueleto2DExtract
    {
        /// <summary>
        /// Skeleton2DdataCoordEventHandler delegado
        /// </summary>
        /// <param name="sender">The sender object</param>
        /// <param name="a">Skeleton 2Ddata Coord Event Args</param>
        public delegate void Skeleton2DdataCoordEventHandler(object sender, DatosEsqueleto2DCoord a);

        /// <summary>
        /// The Skeleton 2Ddata Coord Ready evento
        /// </summary>
        public static event Skeleton2DdataCoordEventHandler Skeleton2DdataCoordReady;

        /// <summary>
        /// Abdominales datos Esqueleto de Kinect SDK en un formato más útil para DTW
        /// </summary>
        /// <param name="data">Kinect SDK's Skeleton Data</param>
        public static void ProcessData(SkeletonData data)
        {
            // Extracto de las coordenadas de los puntos.
            var p = new Point[6];
            Point shoulderRight = new Point(), shoulderLeft = new Point();
            foreach (Joint j in data.Joints)
            {
                switch (j.ID)
                {
                    case JointID.HandLeft:
                        p[0] = new Point(j.Position.X, j.Position.Y);
                        break;
                    case JointID.WristLeft:
                        p[1] = new Point(j.Position.X, j.Position.Y);
                        break;
                    case JointID.ElbowLeft:
                        p[2] = new Point(j.Position.X, j.Position.Y);
                        break;
                    case JointID.ElbowRight:
                        p[3] = new Point(j.Position.X, j.Position.Y);
                        break;
                    case JointID.WristRight:
                        p[4] = new Point(j.Position.X, j.Position.Y);
                        break;
                    case JointID.HandRight:
                        p[5] = new Point(j.Position.X, j.Position.Y);
                        break;
                    case JointID.ShoulderLeft:
                        shoulderLeft = new Point(j.Position.X, j.Position.Y);
                        break;
                    case JointID.ShoulderRight:
                        shoulderRight = new Point(j.Position.X, j.Position.Y);
                        break;
                }
            }
```

```
            //Centro de los datos
            //Además, obtiene las posiciones de ambos hombros y las almacena en las variables
            //“shoulderRight” y “shoulderLeft”. Estas coordenadas serán utilizadas para
            //establecer el origen de nuestro sistema de referencia en el punto medio del
            //segmento que une ambos hombros.

            var center = new Point((shoulderLeft.X + shoulderRight.X) / 2, (shoulderLeft.Y +
shoulderRight.Y) / 2);
            for (int i = 0; i < 6; i++)
            {
                p[i].X -= center.X;
                p[i].Y -= center.Y;
            }

            // Normalization de las coordenadas. Por último, para completar la definición del sistema
            //de coordenadas, normaliza la distancia de cada punto al nuevo origen de coordenadas
dividiendo
            //entre la distancia que separa a los hombros.

            double shoulderDist =
                Math.Sqrt(Math.Pow((shoulderLeft.X - shoulderRight.X), 2) +
                          Math.Pow((shoulderLeft.Y - shoulderRight.Y), 2));
            for (int i = 0; i < 6; i++)
            {
                p[i].X /= shoulderDist;
                p[i].Y /= shoulderDist;
            }

            // Launch the event!
            Skeleton2DdataCoordReady(null, new DatosEsqueleto2DCoord(p));
        }

    }
}
```

ANEXO H

Codigo fuente de la clase DatosEsqueleto2DCoord.cs

Clase DatosEsqueleto2DCoord.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;

namespace RFpost
{
    class DatosEsqueleto2DCoord
    {
        /// <summary>
        /// Las posiciones de los codos, las muñecas y las manos (colocados de izquierda a derecha)
        /// </summary>
        private readonly Point[] _points;

        /// <summary>
        /// inicializa una nueva instancia de la clase Skeleton2DdataCoordEventArgs
        /// </summary>
        /// <param name="points">Los puntos que necesitamos para manejar en esta clase</param>
        public DatosEsqueleto2DCoord(Point[] points)
        {
            _points = (Point[])points.Clone();
        }

        /// <summary>
```

```
        /// Obtiene el punto en un determinado índice
        /// </summary>
        /// <param name="index">El índice que queremos recuperar</param>
        /// <returns>El punto en el índice enviado</returns>
        public Point GetPoint(int index)
        {
            return _points[index];
        }

        /// <summary>
        /// Obtiene las coordenadas de nuestros _POINTS
        /// </summary>
        /// <returns>Las coordenadas de nuestros _points</returns>
        internal double[] GetCoords()
        {
            var tmp = new double[_points.Length * 2];
            for (int i = 0; i < _points.Length; i++)
            {
                tmp[2 * i] = _points[i].X;
                tmp[(2 * i) + 1] = _points[i].Y;
            }

            return tmp;
        }
```

ANEXO I

Codigo fuente de la clase MainWindows.xaml.cs

Clase MainWindows.xaml.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Windows;
using controls = System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Collections;
using System.Diagnostics;
using Microsoft.Research.Kinect.Nui;
using System.Windows.Forms;
using mensajes = System.Windows;
using System.IO;
using System.Data;
using iTextSharp.text;
using iTextSharp.text.pdf;
using par = iTextSharp.text;
using imgpdf=System.Drawing;
using pat = System.IO;



namespace RFpost
{

    /// <summary>
    /// interaccion logica para  MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
```

```
        Conexion1 conex = new Conexion1();
        DataSet ds = new DataSet();
        DataSet ds1 = new DataSet();


        private const int RedIdx = 2;
        private int tiempoRutina = 0;
        private int tiempoGeneral = 0;
        private int contador = 1;
        private bool estadoConexion=false;


        private const int GreenIdx = 1;

        private const int BlueIdx = 0;

        /// <summary>
        /// La cantidad de marcos de ignorar esqueleto (_flipFlop)
        /// 1 = capturar cada fotograma , 2 =capturar cada segundo marco etc.
        /// </summary>
        private const int Ignore = 2;

        /// <summary>
        /// La cantidad de marcos esqueleto de almacenar en el _video buffer
        /// </summary>
        private const int BufferSize = 32;

        /// <summary>
        /// El número minumum de tramas en el búfer _Video antes de intentar iniciar gestos
        /// </summary>
        private const int MinimumFrames = 6;


        private const int CaptureCountdownSeconds = 3;

        private const int CaptureCountdownSeconds1 = 20; //tiempo en seg que dura en finalizar el tiempo

        /// <summary>
        /// ruta para guardar posturas o coordenadas
        /// </summary>
        private const string GestureSaveFileLocation = @"C:\Users\JESU\Desktop\poses1\";

        /// <summary>
        /// inicio del nombre del archivo
        /// </summary>
        private const string GestureSaveFileNamePrefix = @"RecordedGestures";

        /// <summary>
        /// Diccionario de todas las articulaciones q Kinect SDK es capaz de seguimiento. Puede que no
quieras usar siempre a todos, pero que se incluyen aquí para thouroughness.
        /// </summary>
        private readonly Dictionary<JointID, Brush> _jointColors = new Dictionary<JointID, Brush>
        {
            {JointID.HipCenter, new SolidColorBrush(Color.FromRgb(169, 176, 155))},
            {JointID.Spine, new SolidColorBrush(Color.FromRgb(169, 176, 155))},
            {JointID.ShoulderCenter, new SolidColorBrush(Color.FromRgb(168, 230, 29))},
            {JointID.Head, new SolidColorBrush(Color.FromRgb(200, 0, 0))},
            {JointID.ShoulderLeft, new SolidColorBrush(Color.FromRgb(79, 84, 33))},
            {JointID.ElbowLeft, new SolidColorBrush(Color.FromRgb(84, 33, 42))},
            {JointID.WristLeft, new SolidColorBrush(Color.FromRgb(255, 126, 0))},
            {JointID.HandLeft, new SolidColorBrush(Color.FromRgb(215, 86, 0))},
            {JointID.ShoulderRight, new SolidColorBrush(Color.FromRgb(33, 79,  84))},
            {JointID.ElbowRight, new SolidColorBrush(Color.FromRgb(33, 33, 84))},
            {JointID.WristRight, new SolidColorBrush(Color.FromRgb(77, 109, 243))},
            {JointID.HandRight, new SolidColorBrush(Color.FromRgb(37,  69, 243))},
            {JointID.HipLeft, new SolidColorBrush(Color.FromRgb(77, 109, 243))},
            {JointID.KneeLeft, new SolidColorBrush(Color.FromRgb(69, 33, 84))},
            {JointID.AnkleLeft, new SolidColorBrush(Color.FromRgb(229, 170, 122))},
            {JointID.FootLeft, new SolidColorBrush(Color.FromRgb(255, 126, 0))},
            {JointID.HipRight, new SolidColorBrush(Color.FromRgb(181, 165, 213))},
            {JointID.KneeRight, new SolidColorBrush(Color.FromRgb(71, 222, 76))},
            {JointID.AnkleRight, new SolidColorBrush(Color.FromRgb(245, 228, 156))},
            {JointID.FootRight, new SolidColorBrush(Color.FromRgb(77, 109, 243))}
```

```
        };

        /// <summary>
        /// La matriz de bytes marco profundidad. Sólo es compatible con 320 * 240 en este momento
        /// </summary>
        private readonly byte[] _depthFrame32 = new byte[320 * 240 * 4];

        /// <summary>
        /// Flag para mostrar si el reconocedor gesto está capturando una nueva pose
        /// </summary>
        private bool _capturing;

        /// <summary>
        /// Objeto Deformación Dinámica de Tiempo
        /// </summary>
        private ReconocerGesto _dtw;

        /// <summary>
        /// ¿Cuántos frames se produjo "último tiempo". Se utiliza para el cálculo de fotogramas por
segundo
        /// </summary>
        private int _lastFrames;

        /// <summary>
        /// La fecha y hora "última vez". Se utiliza para el cálculo de fotogramas por segundo
        /// </summary>
        private DateTime _lastTime = DateTime.MaxValue;

        /// <summary>
        /// El tiempo de ejecución Natural User Interface
        /// </summary>
        private Runtime _nui;

        /// <summary>
        /// Número total de enmarcado que se han producido. Se utiliza para el cálculo de fotogramas por
segundo
        /// </summary>
        private int _totalFrames;

        /// <summary>
        /// Interruptor utiliza para ignorar ciertos marcos esqueleto
        /// </summary>
        private int _flipFlop;

        /// <summary>
        /// ArrayList de coordenadas que se registran en secuencia para definir un gesto
        /// </summary>
        private ArrayList _video;


        private DateTime _captureCountdown = DateTime.Now;


        private Timer _captureCountdownTimer;

        /// <summary>
        /// Initializes a new instance of the MainWindow class
        /// </summary>
        public MainWindow()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Abre el archivo de texto enviado y crea una secuencia de gestos _dtw registrado
        /// </summary>
        /// <param name="fileLocation"></param>
        public void LoadGesturesFromFile(string fileLocation)
        {
            int itemCount = 0;
            string line;
            string gestureName = String.Empty;
```

```
            ArrayList frames = new ArrayList();
            double[] items = new double[12];

            // Lea el archivo y mostrarlo línea por línea.
            System.IO.StreamReader file = new System.IO.StreamReader(fileLocation);
            while ((line = file.ReadLine()) != null)
            {
                if (line.StartsWith("@"))
                {
                    gestureName = line;
                    continue;
                }

                if (line.StartsWith("~"))
                {
                    frames.Add(items);
                    itemCount = 0;
                    items = new double[12];
                    continue;
                }

                if (!line.StartsWith("----"))
                {
                    items[itemCount] = Double.Parse(line);
                }

                itemCount++;

                if (line.StartsWith("----"))
                {
                    _dtw.AddOrUpdate(frames, gestureName);
                    frames = new ArrayList();
                    gestureName = String.Empty;
                    itemCount = 0;
                }
            }

            file.Close();
        }

        /// <summary>
        ///Llamado cada vez que un marco de esqueleto está listo. Pasa los datos del esqueleto al
procesador DTW
        /// </summary>
        /// <param name="sender">The sender object</param>
        /// <param name="e">Skeleton Frame Ready Event Args</param>
        private static void SkeletonExtractSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
        {
            SkeletonFrame skeletonFrame = e.SkeletonFrame;
            foreach (SkeletonData data in skeletonFrame.Skeletons)
            {
                DatosEsqueleto2DExtract.ProcessData(data);
            }
        }

        /// <summary>
        /// Convierte un marco de profundidad de escala de grises de 16 bits que incluye índices de jugador
en una trama de 32 bits que muestra los diferentes actores en diferentes colores
        /// </summary>
        /// <param name="depthFrame16">La matriz de bytes marco profundidad</param>
        /// <returns>Matriz de bytes que contiene el marco de profundidad una imagen de jugador</returns>
        private byte[] ConvertDepthFrame(byte[] depthFrame16)
        {

            for (int i16 = 0, i32 = 0; i16 < depthFrame16.Length && i32 < _depthFrame32.Length; i16 += 2,
i32 += 4)
            {
                int player = depthFrame16[i16] & 0x07;
                int realDepth = (depthFrame16[i16 + 1] << 5) | (depthFrame16[i16] >> 3);

                // transformar la información de profundidad de 13 bits en una intensidad de 8 bits
apropiada
                // para la exhibición (no tenemos en cuenta la información en el bit más significativo)
                var intensity = (byte)(255 - (255 * realDepth / 0x0fff));
```

```
                _depthFrame32[i32 + RedIdx] = 0;
                _depthFrame32[i32 + GreenIdx] = 0;
                _depthFrame32[i32 + BlueIdx] = 0;

                // choose different display colors based on player
                switch (player)
                {
                    case 0:
                        _depthFrame32[i32 + RedIdx] = (byte)(intensity / 2);
                        _depthFrame32[i32 + GreenIdx] = (byte)(intensity / 2);
                        _depthFrame32[i32 + BlueIdx] = (byte)(intensity / 2);
                        break;
                    case 1:
                        _depthFrame32[i32 + RedIdx] = intensity;
                        break;
                    case 2:
                        _depthFrame32[i32 + GreenIdx] = intensity;
                        break;
                    case 3:
                        _depthFrame32[i32 + RedIdx] = (byte)(intensity / 4);
                        _depthFrame32[i32 + GreenIdx] = intensity;
                        _depthFrame32[i32 + BlueIdx] = intensity;
                        break;
                    case 4:
                        _depthFrame32[i32 + RedIdx] = intensity;
                        _depthFrame32[i32 + GreenIdx] = intensity;
                        _depthFrame32[i32 + BlueIdx] = (byte)(intensity / 4);
                        break;
                    case 5:
                        _depthFrame32[i32 + RedIdx] = intensity;
                        _depthFrame32[i32 + GreenIdx] = (byte)(intensity / 4);
                        _depthFrame32[i32 + BlueIdx] = intensity;
                        break;
                    case 6:
                        _depthFrame32[i32 + RedIdx] = (byte)(intensity / 2);
                        _depthFrame32[i32 + GreenIdx] = (byte)(intensity / 2);
                        _depthFrame32[i32 + BlueIdx] = intensity;
                        break;
                    case 7:
                        _depthFrame32[i32 + RedIdx] = (byte)(255 - intensity);
                        _depthFrame32[i32 + GreenIdx] = (byte)(255 - intensity);
                        _depthFrame32[i32 + BlueIdx] = (byte)(255 - intensity);
                        break;
                }
            }

            return _depthFrame32;
        }

        /// <summary>
        /// Se llama cuando cada fotograma profundidad está listo
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void NuiDepthFrameReady(object sender, ImageFrameReadyEventArgs e)
        {
            PlanarImage image = e.ImageFrame.Image;
            byte[] convertedDepthFrame = ConvertDepthFrame(image.Bits);

            depthImage.Source = BitmapSource.Create(
                image.Width, image.Height, 96, 96, PixelFormats.Bgr32, null, convertedDepthFrame,
image.Width * 4);

            ++_totalFrames;

            DateTime cur = DateTime.Now;
            if (cur.Subtract(_lastTime) > TimeSpan.FromSeconds(1))
            {
                int frameDiff = _totalFrames - _lastFrames;
                _lastFrames = _totalFrames;
                _lastTime = cur;
               // frameRate.Text = frameDiff + " fps";
            }
```

```
        }

        /// <summary>
        /// Obtiene la posición de la pantalla de una Articulación
        /// </summary>
        /// <param name="joint">Kinect NUI Joint</param>
        /// <returns>Point mapped location of sent joint</returns>
        private Point GetDisplayPosition(Joint joint)
        {
            float depthX, depthY;
            _nui.SkeletonEngine.SkeletonToDepthImage(joint.Position, out depthX, out depthY);
           depthX = Math.Max(0, Math.Min(depthX * 320, 320)); // convertir a 320, 240 space
           depthY = Math.Max(0, Math.Min(depthY * 240, 240)); // convertir a 320, 240 space


            int colorX, colorY;
            var iv = new ImageViewArea();

            //Sólo ImageResolution.Resolution640x480 se apoya en este punto
            _nui.NuiCamera.GetColorPixelCoordinatesFromDepthPixel(ImageResolution.Resolution640x480, iv,
(int)depthX, (int)depthY, 0, out colorX, out colorY);

            //  volver a skeleton.Width & skeleton.Height
            return new Point((int)(skeletonCanvas.Width * colorX / 640.0), (int)(skeletonCanvas.Height *
colorY / 480));
        }

        /// <summary>
        ///Funciona cómo dibujar una línea ("hueso") para articulaciones enviados
        /// </summary>

        private Polyline GetBodySegment(JointsCollection joints, Brush brush, params JointID[] ids)
        {
            var points = new PointCollection(ids.Length);
            foreach (JointID t in ids)
            {
                points.Add(GetDisplayPosition(joints[t]));
            }

            var polyline = new Polyline();
            polyline.Points = points;
            polyline.Stroke = brush;
            polyline.StrokeThickness = 5;
            return polyline;
        }

        /// <summary>
        /// cada vez que un marco de esqueleto está listo. Actualiza la lona esqueleto con nuevas
ubicaciones conjuntas y polilínea.
        /// </summary>
        /// <param name="sender">The sender object</param>
        /// <param name="e">Skeleton Frame Event Args</param>
        private void NuiSkeletonFrameReady(object sender, SkeletonFrameReadyEventArgs e)
        {
            SkeletonFrame skeletonFrame = e.SkeletonFrame;
            int iSkeleton = 0;
            var brushes = new Brush[6];
            brushes[0] = new SolidColorBrush(Color.FromRgb(255, 0, 0));
            brushes[1] = new SolidColorBrush(Color.FromRgb(0, 255, 0));
            brushes[2] = new SolidColorBrush(Color.FromRgb(64, 255, 255));
            brushes[3] = new SolidColorBrush(Color.FromRgb(255, 255, 64));
            brushes[4] = new SolidColorBrush(Color.FromRgb(255, 64, 255));
            brushes[5] = new SolidColorBrush(Color.FromRgb(128, 128, 255));

            skeletonCanvas.Children.Clear();
            foreach (SkeletonData data in skeletonFrame.Skeletons)
            {
                if (SkeletonTrackingState.Tracked == data.TrackingState)
                {
                    // Dibuja los huesos
                    Brush brush = brushes[iSkeleton % brushes.Length];
                    skeletonCanvas.Children.Add(GetBodySegment(data.Joints, brush, JointID.HipCenter,
JointID.Spine, JointID.ShoulderCenter, JointID.Head));
```

```
                    skeletonCanvas.Children.Add(GetBodySegment(data.Joints, brush, JointID.ShoulderCenter,
JointID.ShoulderLeft, JointID.ElbowLeft, JointID.WristLeft, JointID.HandLeft));
                    skeletonCanvas.Children.Add(GetBodySegment(data.Joints, brush, JointID.ShoulderCenter,
JointID.ShoulderRight, JointID.ElbowRight, JointID.WristRight, JointID.HandRight));
                    skeletonCanvas.Children.Add(GetBodySegment(data.Joints, brush, JointID.HipCenter,
JointID.HipLeft, JointID.KneeLeft, JointID.AnkleLeft, JointID.FootLeft));
                    skeletonCanvas.Children.Add(GetBodySegment(data.Joints, brush, JointID.HipCenter,
JointID.HipRight, JointID.KneeRight, JointID.AnkleRight, JointID.FootRight));

                    //Dibuja las Articulaciones
                    foreach (Joint joint in data.Joints)
                    {
                        Point jointPos = GetDisplayPosition(joint);
                        var jointLine = new Line();
                        jointLine.X1 = jointPos.X - 3;
                        jointLine.X2 = jointLine.X1 + 6;
                        jointLine.Y1 = jointLine.Y2 = jointPos.Y;
                        jointLine.Stroke = _jointColors[joint.ID];
                        jointLine.StrokeThickness = 6;
                        skeletonCanvas.Children.Add(jointLine);
                    }
                }

                iSkeleton++;
            } // for each skeleton
        }

        /// <summary>
        /// Llamado cada vez que un vídeo (RGB) marco está listo
        /// </summary>
        /// <param name="sender">The sender object</param>
        /// <param name="e">Image Frame Ready Event Args</param>
        private void NuiColorFrameReady(object sender, ImageFrameReadyEventArgs e)
        {
            // 32-bit per pixel, RGBA image
            PlanarImage image = e.ImageFrame.Image;
        //    videoImage.Source = BitmapSource.Create(
          //      image.Width, image.Height, 96, 96, PixelFormats.Bgr32, null, image.Bits, image.Width *
image.BytesPerPixel);
        }

        /// <summary>
        /// Se ejecuta después de la ventana, se haya cargado
        /// </summary>
        /// <param name="sender">The sender object</param>
        /// <param name="e">Routed Event Args</param>



private void WindowLoaded(object sender, RoutedEventArgs e)
        {
            ocultar_imagenes();
            desactivar_btn_empezar();
            ocultar_progreso();
            tabControl1.IsEnabled = false;

           _nui = new Runtime();

            try
            {
                _nui.Initialize(RuntimeOptions.UseDepthAndPlayerIndex | RuntimeOptions.UseSkeletalTracking
|
                                RuntimeOptions.UseColor);
            }
            catch (InvalidOperationException)
            {
                System.Windows.MessageBox.Show("Runtime initialization failed. Please make sure Kinect
device is plugged in.");
                return;
            }

        }
private void WindowClosed(object sender, EventArgs e)
        {
```

```
            Debug.WriteLine("Stopping NUI");
            _nui.Uninitialize();
            Debug.WriteLine("NUI stopped");
            Environment.Exit(0);
        }
private void NuiSkeleton2DdataCoordReadyCifosis1(object sender, DatosEsqueleto2DCoord a)
        {

           // currentBufferFrame.Text = _video.Count.ToString();

            // We need a sensible number of frames before we start attempting to match gestures against
remembered sequences

            if (_video.Count > MinimumFrames && _capturing == false)
            {

                ////Debug.WriteLine("Reading and video.Count=" + video.Count);
                string s = _dtw.Recognize(_video);
             //   results.Text = "Reconoida as: " + s;
                int s1 = _dtw.Recognize1(_video);


                if (s.Contains("@cifosis1") && tiempoRutina > 0)
                {

                    pruebat.Content = "MUY BIEN!!!";
                    //   cargar_barra(cont);
                    label4.Content = contador + 1;
                    contador = contador + 1;
                    cargar_barra(contador);

                }
                else
                {
                    pruebat.Content = "INTENTALO";



                }


                if (!s.Contains("__UNKNOWN"))
                {
                    // There was no match so reset the buffer
                    _video = new ArrayList();
                    //   pruebat.Text = "entro aqui";

                }

            }

            // Asegura que recordemos sólo los x últimos fotogramas
            if (_video.Count > BufferSize)
            {
                // si actualmente estamos capturando
                if (_capturing)
                {
                    DtwStoreClick(null, null);
                }
                else
                {
                    // Retire el primer fotograma en la memoria intermedia
                    _video.RemoveAt(0);
                }
            }

          // Decida qué marcos esqueleto para capturar.
          //Sólo hacerlo si las tramas en realidad devuelven un número.
            if (!double.IsNaN(a.GetPoint(0).X))
            {
                // Opcionalmente registrarse sólo 1 marco de cada n
                _flipFlop = (_flipFlop + 1) % Ignore;
                if (_flipFlop == 0)
                {
```

```
                    _video.Add(a.GetCoords());
                }
            }

            //Actualizar la ventana de depuración con información Secuencias
            //dtwTextOutput.Text = _dtw.RetrieveText();


        }

private void CaptureCountdown(object sender, EventArgs e)
        {
            if (sender == _captureCountdownTimer)
            {
                if (DateTime.Now < _captureCountdown)
                {
                    status.Text = "Wait " + ((_captureCountdown - DateTime.Now).Seconds + 1) + " seconds";
                }
                else
                {
                    _captureCountdownTimer.Stop();
                    status.Text = "Recording gesture";
                    StartCapture();
                }
            }
        }
private delegate void DelegadoActualizaProgressBar(System.Windows.DependencyProperty dp, Object valor);

        private void cargar_barra(int valor)
        {
            //Configuración del ProgressBar
            progressBar1.Minimum = 0;//valor mínimo (inicio de la barra de carga)
            progressBar1.Maximum = 100;//valor máximo(hasta donde se carga, como ejemplo 100)
            progressBar1.Value = 0;//valor de inicio


            DelegadoActualizaProgressBar DelegadoActualizaPB = new
DelegadoActualizaProgressBar(progressBar1.SetValue);


            double porcentaje = 0;
            int numeroRutinas = int.Parse(txnumrutinas.Text); // aqui hace una pequeña regla
            porcentaje = (valor * 100) / numeroRutinas ;      // de tres simple
            label3.Content = "" + porcentaje; //muestrar el porcentaje en label

            if (progressBar1.Value != progressBar1.Maximum)
            {
                    Dispatcher.Invoke(DelegadoActualizaPB,
                    System.Windows.Threading.DispatcherPriority.Background,
                    new object[] { controls.ProgressBar.ValueProperty, porcentaje });

                  /*
                   * esta parte es la que pinta la barra y se va mostrando el color verde en este caso
                   * de acuerdo al porcentaje
                   */
            }

        }

        private void button5_Click(object sender, RoutedEventArgs e)
        {
            int cont = 0;
            while (tiempoRutina > 0)
            {
                if (!pruebat.Content.Equals(null))
                {
                    cont++;
                    label4.Content = cont;

                }
            }
        }
```

ANEXO J

Codigo fuente de la clase Conexion.cs

Clase Conexion.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using MySql.Data.MySqlClient;
using System.Windows.Forms;
using System.Windows.Data;
using MySql.Data;
using System.Data;

namespace RFpost
{
    class Conexion1
    {
        static MySqlConnection Conex = new MySqlConnection();

        static MySqlCommand Comando = new MySqlCommand();
        static MySqlDataAdapter Adaptador = new MySqlDataAdapter();

        public bool Conectar(string usuario, string contrasena)
        {

            bool es = false;

          string  servidor = "Server=localhost;";
          string  basedatos = "Database=rfpost;";
                  usuario = "UID="+usuario+";";
                  contrasena = "Password ="+contrasena;

            string cadenaconex = servidor + basedatos + usuario + contrasena;

            try
            {
                Conex.ConnectionString = cadenaconex;
                Conex.Open();
             //   MessageBox.Show("La BD esta ahora conectada");
                es = true;
            }
            catch
            {
             //   MessageBox.Show("Ocurrio un error al conectar a la BD "+e.Message);
                es = false;


            }
            return es;
        }



        public static void Desconectar()
        {
            Conex.Close();
        }
```

ANEXO K

Trabajo futuro

Con la culminación del proyecto surgen algunas ideas en pos de realizar algunas mejoras a futuro, como por ejemplo:

- Que el software se pueda manejar sin necesidad de usar el mouse, si no que el mismo usuario con su mano lo pueda manipular.

- Que el fisioterapeuta pueda acceder al software desde la Web y así poder realizar el oficio sin necesidad de estar presente.

- Que el software este en capacidad de que más de un paciente realice una rutina de ejercicios simultáneamente.

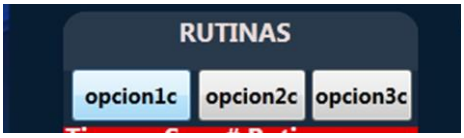
ANEXO L

Manual de Usuario
RFPOST

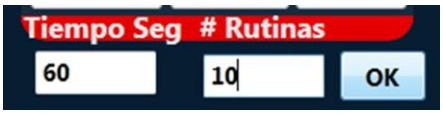
0. **Ingreso al Sistema:** Digitar el usuario con su respectiva contraseña y clic en el botón Entrar

1. **Escoger Problema Postural:** El medico selecciona la opción de acuerdo al problema a tratar.

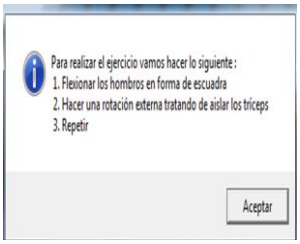
2. **Escoger una opción de Rutina:** El medico escoge una de las opciones para realizar la rutina.



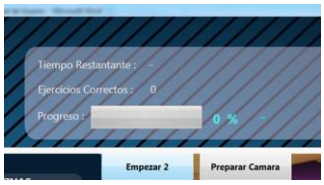
3. **Digitar Tiempo y numero de rutinas:** el medico digita el tiempo que debe durar haciendo la rutina y el número de veces que debe realizar dicho movimiento o ejercicio y clic en “ok”

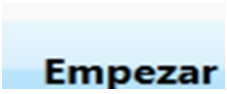


4. **Sugerencia:** En este espacio muestra cómo debe realizarse el ejercicio.



5. **Empezar:** En esta opción se da clic para iniciar la rutina





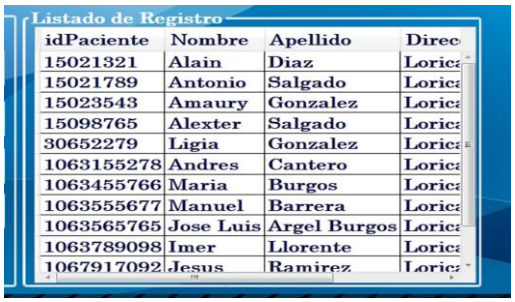
6. **Cámara Activada:** En esta parte el sensor se activa y refleja el movimiento en tiempo real.

7. **Progreso:** en este espacio muestra el tiempo restante, el número de ejercicios correctos y el porcentaje de progreso, en tiempo real a medida que el usuario ejecute la rutina.

8. **Agregar Paciente:** Selección la pestaña datos paciente, llena los campos y clic en "agregar paciente".

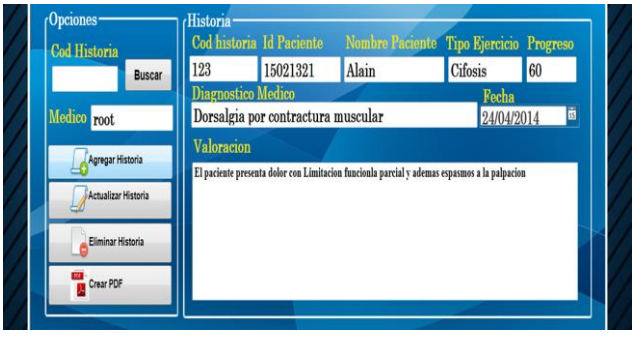
9. **Modificar y Eliminar paciente :** Se digita la identificación en el campo buscar x identificación, se da clic en "ok" y aparece la información respectiva del paciente, si se va a modificar entonces se actualiza la información y luego clic en "Modificar paciente" y si se va a eliminar dar clic en la opción "eliminar paciente"

10. **Listar Pacientes:** En este espacio se muestra una lista de todos los pacientes agregados con sus datos personales al dar clic en el botón "Listar Pacientes".



12. **Limpiar Campos:** Esta opción limpia los campos de los datos personales y además ubica el cursor en el campo de búsqueda.

13. **Crear una Historia:** se llenan los campos correspondientes y se da clic en "agregar historia"



14. **Buscar Historia:** Escribir el código de la historia en Opciones "Cod.Historia" y clic en "buscar"

15. **Actualizar y Eliminar Historia:** Se digita el código en el campo buscar "CodHistoria", se da clic en "buscar" y aparece la información respectiva de la historia, si se va a modificar entonces se actualiza la información y luego clic en "Actualizar historia" y si se va a eliminar dar clic en la opción "Eliminar Historia"

16. **Crear PDF:** En esta opción el medico tiene la oportunidad de crear un documento PDF de cada historia, se busca primero la historia y luego clic en el botón "crear PDF" o en su defecto se crea primero la historia y luego ejecuta esta opción.

17. **Agregar Medico:** Selección la pestaña datos Medico, llena los campos y clic en "agregar Medico".

18. **Modificar y Eliminar Medico:** Se digita la identificación en el campo buscar x identificación, se da clic en "ok" y aparece la información respectiva del Médico, si se va a modificar entonces se actualiza la información y luego clic en "Modificar Medico" y si se va a eliminar dar clic en la opción "Eliminar Medico"

19. **Listar Medico:** En este espacio se muestra una lista de todos los médicos agregados con sus datos personales al dar clic en el botón "Listar Pacientes".

20. **Limpiar Campos:** Esta opción limpia los campos de los datos personales y además ubica el cursor en el campo de búsqueda.

21. **Visión General**