UNIVERSIDAD POLITÉCNICA DE PUEBLA

INGENIERÍA EN INFORMÁTICA

PROYECTO DE ESTADÍA PROFESIONAL

DESARROLLO DE UNA APP EN ANDROID PARA GENERAR Y CONTROLAR SESIONES DE EJERCICIOS DE REHABILITACIÓN A PERSONAS CON PARÁLISIS FACIAL

XICALE MONTES ERIKA

ASESOR TÉCNICO

DR. FRANCISCO JAVIER RENERO CARRILLO

ASESOR ACADÉMICO

MTRA. REBECA RODRÍGUEZ HUESCA

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El presente documento muestra el proceso de desarrollo de una aplicación, así como las herramientas empleadas y la metodología implementada. Esta aplicación tiene como fin detectar el rostro humano en estado de reposo, movimientos faciales como lo son el levantamiento de cejas, labios contraídos, sonrisa, ojos cerrados con fuerza, cada uno de estos movimientos son empleados por pacientes con parálisis facial durante sus sesiones de rehabilitación.

Se pretende brindar apoyo a los médicos que atienden al problema de la parálisis facial, debido a que el proceso de rehabilitación que se realiza en la actualidad es doctor y paciente cara a cara, donde el paciente realiza sus ejercicios de rehabilitación y el médico toma medidas del rostro, haciendo uso de un vernier este método se considera un método incómodo para el paciente, es por esto que se tomaron métricas del rostro por medio del procesamiento digital de imágenes, facilitando la métrica y el seguimiento de ejercicios de rehabilitación.

CAPITULO 1. ANTECEDENTES DE LA EMPRESA O DEL PROYECTO

1.1 Antecedentes Históricos

El Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE) fue creado por decreto presidencial el 11 de noviembre de 1971 como un organismo descentralizado, de interés público, con personalidad jurídica y patrimonio propio, ubicado en Tonantzintla, Puebla, con los siguientes objetivos:

- Preparar investigadores, profesores especializados, expertos y técnicos en astrofísica, óptica y electrónica.
- Procurar la solución de problemas científicos y tecnológicos relacionados con las citadas disciplinas.
- Orientar sus actividades de investigación y docencia hacia la superación de las condiciones y resolución de los problemas del país.
- Con este decreto el INAOE tiene la facultad de impartir cursos y otorgar grados de maestría y doctorado en las diversas disciplinas que en él se desarrollan.

EL INAOE es heredero de una gran tradición científica que data de 1942, cuando Luis Enrique Erro fundó el Observatorio Astrofísico Nacional de Tonantzintla.

Con la Cámara Schmidt de Tonantzintla se inauguró el Observatorio, abriendo las puertas a la astronomía moderna en México y Latinoamérica. La importancia del Observatorio Astrofísico de Tonantzintla traspasó las fronteras de México, siendo reconocida la labor realizada por astrónomos reconocidos internacionalmente, entre los que figuraron el mismo fundador Luis Enrique Erro; el Dr. Guillermo Haro, el Prof. Luis Rivera Terrazas, el Dr. Luis Munch y el astrónomo Enrique Chavira, entre otros.

En 1972 se fundó el Departamento de Óptica, y dos años después inició sus actividades el Departamento de Electrónica. Desde su creación uno de los principales objetivos del INAOE ha sido la preparación de investigadores jóvenes, capaces de identificar y resolver problemas científicos y tecnológicos en astrofísica, óptica, electrónica y áreas afines. En 1972 se iniciaron los estudios de maestría en Óptica y en 1974 los de Electrónica. En 1984 se inició el programa de doctorado en Óptica, y en 1993 los programas de doctorado en Electrónica; así como la maestría y doctorado en Astrofísica. Finalmente, en agosto de 1998 se inició el programa de maestría y doctorado en Ciencias Computacionales [2].

1.2 Misión

Contribuir como centro público de investigación a la generación, avance y difusión del conocimiento para el desarrollo del país y de la humanidad, por medio de la identificación y solución de problemas científicos y tecnológicos y de la formación de especialistas en las áreas de Astrofísica, Óptica, Electrónica, Ciencias Computacionales y áreas afines.

1.3 Visión

El INAOE será un centro público de investigación con un alto liderazgo a nivel internacional en el ámbito de la investigación científica, el desarrollo tecnológico y la formación de recursos humanos dentro de las áreas de Astrofísica, Óptica, Electrónica,

Ciencias Computacionales y áreas afines, comprometido con el desarrollo nacional a través de la promoción de valores sociales de solidaridad, creatividad y alta competitividad.

1.4 Facultades

- Desarrollar investigaciones e impartir enseñanzas para la consecución de los objetivos previstos.
- Organizar sus planes de investigación y enseñanza.
- Adoptar métodos adecuados para evaluar sus actividades de investigación y enseñanza.
- Conceder grados y otorgar diplomas.

1.5 Coordinación de Óptica

La Coordinación de Óptica fue creada en 1972, justo después de la fundación del INAOE. Sus dos postgrados, Maestría y Doctorado en Ciencias con especialidad en Óptica, son los más antiguos del INAOE. Actualmente el programa de maestría está catalogado como de Competencia Internacional, mientras que el de doctorado como Consolidado, en el Padrón Nacional de Posgrados de Calidad del CONACYT [3].

1.5.1Misión de la Coordinación de Óptica

- Realizar investigación básica de vanguardia,
- Realizar investigación aplicada orientada a satisfacer las necesidades de la sociedad y
- Formar recursos humanos capaces de resolver problemas científicos y tecnológicos de alta relevancia.

Todo lo anterior dentro de las siguientes líneas generales de investigación:

- Biofotónica
- Fotónica
- Instrumentación Óptica y Metrología
- Óptica Cuántica
- Óptica Estadística
- Óptica Física
- Optoelectrónica
- Procesado de Imágenes

1.8 Organigrama

A continuación, se muestra en la figura 1 el organigrama del instituto Nacional de Astrofísica Óptica y Electrónica de manera general debido a que existen más de 300 plazas distribuidas entre investigadores, técnicos académicos, ingenieros, área administrativa, mandos medios y superiores.

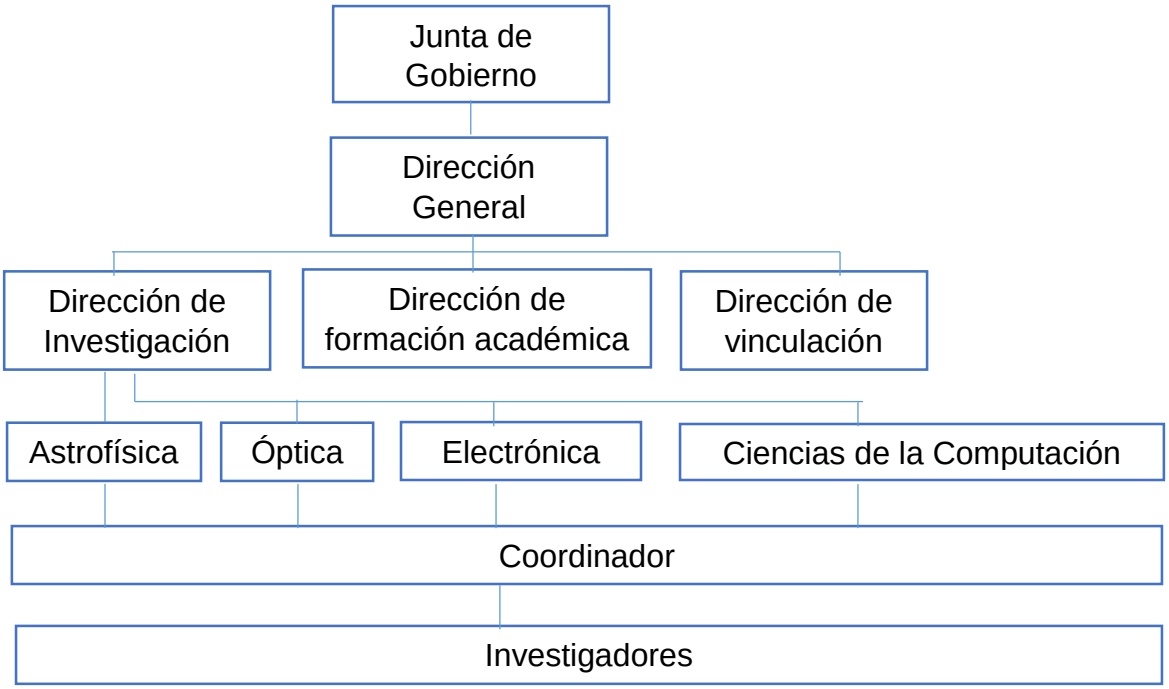

Figura 1. Organigrama del INAOE

CAPITULO 2. METODOLOGÍA Y HERRAMIENTAS DE DESARROLLO

En este capítulo se describe la metodología, que se implementara en el desarrollo de esta aplicación para dispositivos móviles Android, así como las herramientas requeridas para el mismo.

2.1 Metodología

Con el propósito de completar el desarrollo del proyecto se ha optado por implementar una metodología ágil del desarrollo de software, como lo es la metodología de programación extrema, porque esta metodología nos permite estar fuertemente implicados con el cliente estableciendo las prioridades de los requerimientos del sistema, con la posibilidad de que el cliente sea parte del equipo de desarrollo, proporcionando la ventaja de recoger las necesidades del cliente en historias de usuario, una vez que estas tarjetas se han realizado se permitirá estimar el esfuerzo requerido para su implementación, además se le concede al cliente dar la prioridad a las historias con lo cual se obtienen resultados que pueden ser utilizados inmediatamente por el cliente, entre otras de sus ventajas por lo que fue seleccionada esta metodología es el permitir que en caso de que algún requerimiento cambie las historias sin implementar sean cambiadas o descartadas permitiendo una adaptación a nuevos requerimientos establecidos por el usuario.

Esta metodología adopta un desarrollo iterativo, permitiendo la construcción de varias versiones del software y la implementación de pruebas de las mismas. De manera general la programación XP, consiste en seis fases: Exploración, Planificación de la Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto [7].

Fase de exploración: En esta fase inicial el cliente desarrolla historias de usuarios describiendo las funcionalidades con que contará el sistema para la primera entrega, cada una incluye características que serán adicionadas al programa. Por otra parte, el equipo de desarrollo se relaciona con las herramientas, tecnologías y practicas a emplear.

Fase de Planificación de la entrega (Release): Durante esta fase el cliente determina la prioridad de cada historia de usuario mientras que los programadores, realizan una estimación del esfuerzo requerido en cada una de ellas, en base a esta información se establece un cronograma de actividades, así como un contrato de contenido para la primera entrega. El tiempo de la programación de la primera entrega normalmente no excede los 3 meses y el tiempo de la fase como tal toma un par de días.

Fase de Iteraciones: Dentro de esta fase se incluyen las diferentes iteraciones a cerca del sistema antes de ser entregado. La programación se determina en la etapa anterior y es dividida en un numero de iteraciones donde cada una tomará de una a cuatro semanas para ser implementada.

Fase de Producción: En esta fase, se pueden encontrar nuevos cambios y se toma la decisión si serán incluidos en la entrega actual, debido a cambios durante esta fase. Después que la primera entrega es producida para el uso del cliente, el proyecto XP debe

mantener el sistema en producción corriendo mientras que también se estén produciendo nuevas iteraciones.

Fase de Mantenimiento: Mientras la primera versión se encuentra en producción, el proyecto XP, debe mantener el sistema en funcionamiento al mismo tiempo que se desarrollan nuevas iteraciones, para esto se requiere un esfuerzo para soportar las tareas de los clientes. Así, la velocidad del desarrollo puede desacelerarse después de que el sistema está en producción. Esta fase puede requerir incorporar nuevas personas al equipo y cambiar la estructura del mismo.

Fase muerte del Proyecto: Esta fase sucede cuando el cliente no tiene más historias para ser implementadas. Esto requiere que el sistema satisfaga también las necesidades del cliente en otros aspectos, como por ejemplo lo concerniente a la ejecución y la confiabilidad. Este es el momento en el proceso XP cuando la documentación necesaria del sistema es finalmente escrita porque no habrá más cambios en la arquitectura, diseño o código. La muerte puede ocurrir si el sistema no está entregando los beneficios deseados o si se está convirtiendo muy costoso implementarlo.

A continuación, en la Figura 2 se muestran el ciclo de entrega en la programación extrema [4]:

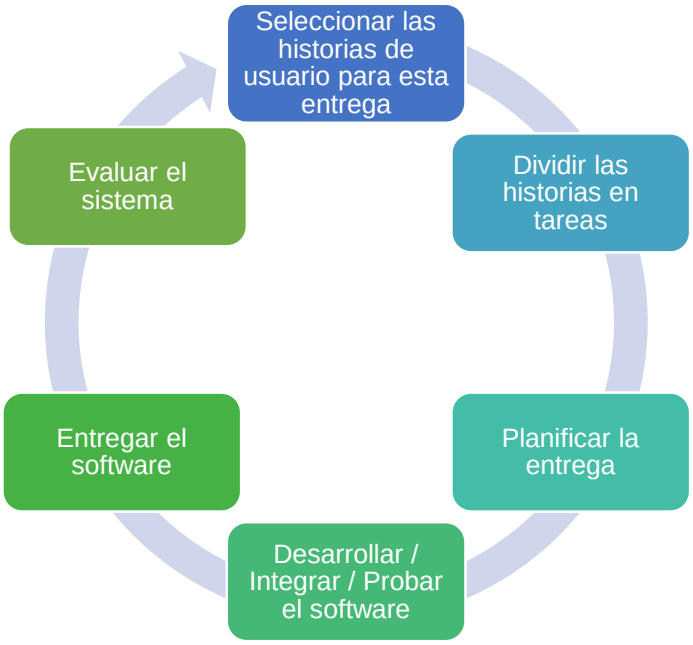

Figura 2. Ciclo de entrega en la programación extrema

A continuación, en la Tabla 1 se describen las prácticas que lleva a cabo la programación extrema [4].

Tabla 1: Prácticas de la programación extrema

Principio o practica	Descripción
Planificación incremental	Los requerimientos se registran en tarjetas de historias y las historias a incluir a una entrega se determinan según el tiempo disponible y su prioridad relativa
Entregas pequeñas	El mínimo conjunto útil de funcionalidad que proporcione valor de negocio se desarrolló primero. Las entregas del sistema son frecuentes e incrementalmente añaden funcionalidad a la primera entrega
Diseño sencillo	Solo se lleva a cabo el diseño necesario para cumplir los requerimientos actuales
Desarrollo previamente probado	Se utiliza un sistema de pruebas de unidad automatizado para escribir pruebas para nuevas funcionalidades antes de que estas se implementen
Refactorización	Se espera que todos los desarrolladores refactoricen el código continuamente tan pronto como encuentren posibles mejoras en el código. Esto conserva el código sencillo y mantenible
Programación en parejas	Los desarrolladores trabajan en parejas, verificando cada uno el trabajo del otro y proporcionando la ayuda necesaria para hacer siempre un buen trabajo
Propiedad colectiva	Las parejas de desarrolladores trabajan en todas las áreas del sistema, de modo que no desarrollen islas de conocimientos y todos los desarrolladores posean todo el código. Cualquiera puede cambiar cualquier cosa
Integración continua	En cuanto acaba el trabajo en una tarea, se integra el sistema entero. Después de la integración, se deben pasar al sistema todas las pruebas de unidad
Ritmo sostenible	No se consideran aceptables grandes cantidades de horas extras, ya que a menudo el efecto que tienen es que se reduce la calidad del código y la productividad a medio plazo
Cliente presente	Debe estar disponible al equipo de la XP un representante de los usuarios finales del sistema (el cliente) a tiempo completo. En un proceso de la programación

	extrema el cliente es miembro del equipo de desarrollo y es responsable de formular al equipo los requerimientos del sistema para su implementación

2.2 Herramientas de Desarrollo

Las herramientas requeridas para el correcto desarrollo de esta aplicación fueron:

Android Studio: Es el IDE oficial para el desarrollo de aplicaciones en dispositivos móviles Android, se encuentra basado en IntelliJ IDEA de la compañía JetBrains, ofreciendo diferentes ventajas sobre el plugin ADT de Eclipse el cual fue anteriormente el IDE recomendado por Google para el desarrollo de aplicaciones para Android [5].

Su objetivo es ofrecer a los desarrolladores un entorno dedicado exclusivamente a la programación de aplicaciones de dispositivos móviles Android, además de funciones que aumentan la productividad de compilación dicho sistema es flexible debido a que está basado en Gradle el cual automatiza la construcción de proyectos, herramientas Lint permitiendo la detección de problemas de rendimiento, uso, compatibilidad conversiones en el Sistema Operativo Android, es compatible con el lenguaje de programación C/C++ por medio de NDK (Native Development Kit) [6].

Ventajas

- Todas sus funciones son gratuitas
- Integra un emulador el cual permite la ejecución de la aplicación en tiempo real
- Permite ejecutar la aplicación en desarrollo por medio de dispositivos móviles físicos
- Es un IDE multiplataforma

Desventajas

- Actualizaciones constantes las cuales demoran en ser instaladas
- Los requisitos de hardware son elevados por lo cual no puede ser ejecutado en computadoras con bajos recursos
- El emulador consume altos recursos provocando el calentamiento del PC

OpenCV**:** Es una librería de visión por computadora la cual se puede utilizar para uso académico y comercial. Puede ser utilizado por diferentes lenguajes por ejemplo C, C++, Java, Python etc. Además, es compatible con diversos sistemas operativos como Windows, Linux, Mac OS, iOS y Android.

OpenCV fue escrito en C / C++ y construido para proporcionar una infraestructura común para aplicaciones de visión por computadora y para acelerar el uso de la percepción de una computadora en los productos comerciales [1].

Ventajas

- Gratuito
- Librería multiplataforma
- Contiene más de 500 funciones de visión por computadora
- Permite aplicar técnicas de procesamiento de imágenes digitales
- Permite Aplicar diversos algoritmos de reconocimiento de patrones

Desventajas

- Versiones en evolución por lo cual el manejo del mismo cambia conforme al lenguaje de programación
- Alto consumo de recursos de una PC al compilar

CAPITULO 3. DESARROLLO DEL TRABAJO EN LA EMPRESA O DE INVESTIGACIÓN

En esta sección se describen las fases de las actividades realizadas en la metodología de desarrollo de la aplicación, así como las herramientas utilizadas en la misma. Como se mencionó en el capítulo anterior la programación XP consiste en 5 fases dentro de estas existe un ciclo en cada una de las iteraciones.

3.1 Fases de la Programación extrema

3.1.1 Fase de exploración: En esta fase se definió el alcance general del proyecto, así como los requerimientos necesarios en la aplicación desarrollada, como herramienta se utilizaron "historias de usuarios" en las cuales se siguió un formato mostrado en la tabla 2, así como la descripción de sus componentes, es importante recalcar que la interacción oral entre los desarrolladores y usuarios, es prioridad y no la comunicación escrita por este motivo se evitaron detalles considerados innecesarios e inconvenientes buscando principalmente obtener de los usuarios los requerimientos pertinentes al funcionamiento.

Objetivo general

Desarrollo de una aplicación para dispositivos móviles Android, permitiendo el control de sesiones de rehabilitación en parálisis facial, por medio de la librería OPENCV.

Numero: Identifica el número correspondiente de la historia de usuario desarrollada

Usuario: Se coloca la persona que utilizará la funcionalidad del sistema descrito en la historia correspondiente

Nombre de historia: Describe de manera general la historia de usuario

Prioridad de estimados: Valor de complejidad que una historia de usuario representa al equipo de desarrollo

Iteración Asignada: Número de iteración en que el cliente deseaba que se implementara la historia de usuario

Programador Responsable: Persona encargada de codificar y diseñar cada historia de usuario

Descripción: Información detallada de la historia de usuario

Observaciones: Este campo es opcional con el fin de aclarar, si es necesario, el requerimiento descrito de una historia de usuario.

Tabla 2. Formato de las historias de usuario

Historia de usuario	
Número:	**Usuario:**
Nombre historia:	
Prioridad en negocio:	
Iteración asignada:	
Programador responsable:	
Descripción:	
Observaciones:	

Tabla 3. Historia de usuario 01 Inicio de sesión del paciente

Historia de usuario	
Número: 01	Usuario: Pacientes
Nombre historia: Inicio de sesión del paciente	
Prioridad en negocio: Alta	
Iteración asignada:1	
Programador responsable: Erika Xicale Montes	
Descripción: Al iniciar la aplicación el usuario ingresa su nombre de usuario y contraseña	
Observaciones:	

Tabla 4. Historia de usuario 02 Selección del médico a cargo

Historia de usuario	
Número: 02	Usuario: Pacientes
Nombre historia: Selección del médico a cargo	
Prioridad en negocio: Alta	
Iteración asignada:1	
Programador responsable: Erika Xicale Montes	
Descripción: Se muestra una lista de los médicos registrados en la BD, el usuario(paciente) selecciona en la lista de médicos al médico que registró sus sesiones de rehabilitación a realizar.	
Observaciones:	

Tabla 5. Historia de usuario 03 Detección del rostro

Historia de usuario	
Número: 03	Usuario: Pacientes
Nombre historia: Detección del rostro	
Prioridad en negocio: Alta	
Iteración asignada:2	
Programador responsable: Erika Xicale Montes	
Descripción: Se detecta el rostro en estado de reposo por medio de la cámara frontal	
Observaciones:	

Tabla 6. Historia de usuario 04 Detección de levantamiento de cejas

Historia de usuario	
Número: 04	Usuario: Pacientes
Nombre historia: Detección de levantamiento de cejas	
Prioridad en negocio: Alta	
Iteración asignada:3	
Programador responsable: Erika Xicale Montes	
Descripción: Se debe detectar el levantamiento de cejas por medio de la cámara frontal	
Observaciones:	

Tabla 7. Historia de usuario 05 Detección de contracción de labios

Historia de usuario	
Numero: 05	Usuario: Pacientes
Nombre historia: Detección de contracción de labios	
Prioridad en negocio: Alta	
Iteración asignada:4	
Programador responsable: Erika Xicale Montes	
Descripción: Se debe detectar el movimiento de contraer labios por medio de la cámara frontal	
Observaciones:	

Tabla 8. Historia de usuario 06 Detección de sonrisa

Historia de usuario	
Numero: 06	Usuario: Pacientes
Nombre historia: Detección de sonrisa	
Prioridad en negocio: Alta	
Iteración asignada:5	
Programador responsable: Erika Xicale Montes	
Descripción: Se debe detectar el movimiento de una sonrisa por medio de la cámara frontal	
Observaciones:	

Tabla 9. Historia de usuario 07 Detección de ojos cerrados

Historia de usuario	
Numero: 07	Usuario: Pacientes
Nombre historia: Detección de ojos cerrados	
Prioridad en negocio: Alta	
Iteración asignada:6	
Programador responsable: Erika Xicale Montes	
Descripción: Se debe detectar el movimiento de cerrar ojos por medio de la cámara frontal	
Observaciones:	

Tabla 10. Historia de usuario 08. Mostrar número de sesiones a realizar

Historia de usuario	
Numero: 08	Usuario: Pacientes
Nombre historia: Muestra el número de sesiones a realizar	
Prioridad en negocio: Baja	
Iteración asignada:7	
Programador responsable: Erika Xicale Montes	
Descripción: Mostrar el número de ejercicios que debe realizar el paciente por sesión, dicha información estará registrada en la Base de datos por parte del doctor	
Observaciones:	

Tabla 11. Historia de usuario 09 Envió de métricas obtenidos a la Base de datos

Historia de usuario	
Numero: 09	Usuario: Pacientes
Nombre historia: Envió de métricas obtenidos a la Base de datos	
Prioridad en negocio: Baja	
Iteración asignada:7	
Programador responsable: Erika Xicale Montes	
Descripción: Envió de métricas calculadas del rostro en las sesiones realizadas.	
Observaciones:	

3.1.2 Fase de Planificación

En esta fase nos reunimos los desarrolladores de dos aplicaciones con el cliente del proyecto, al cual se le hará mención como el asesor del proyecto y el co-asesor de este el Mtro. Javier Caldera Miguel, fue necesario debido a que ambas aplicaciones se complementarán a futuro, una estará enfocada al control de registros que los médicos realizan acerca de sus pacientes, la otra aplicación sobre la cual se trabajó en este proyecto está orientada a ser utilizada por los pacientes de esta manera se acordaron los posibles tiempos que se realizarían por iteración, esto se realizó en la primera reunión ya que el plan de entregas podría variar dependiendo de cada entrega.

Tabla 12. Tabla de iteraciones

Iteración	Numero de historia	Duración de iteraciones
1 Iteración	01 y 02	1 semana
2 Iteración	03	1 a 2 semanas
3 Iteración	04	1 semana
4 Iteración	05	1 semana
5 Iteración	06	1 semana
6 Iteración	07	1 semana
7 Iteración	08 y 09	1 a 2 semanas

3.1.3 Fase de Iteraciones

Se realizaron diversas iteraciones con el cliente las cuales consistieron en la entrega de cada historia de usuario desarrollada de acuerdo al orden establecido siguiendo un ciclo. En cada reunión se planifico la interacción estableciendo las tareas específicas de programación, así como cada prueba de aceptación fueron empleadas al término de cada ciclo de iteración y al final de cada uno de los ciclos siguientes de esta manera se verificaba que las subsiguientes iteraciones no afectaran a las anteriores.

Iteración 1: Esta fue la primera iteración que se realizó en la cual se realizaron las historias de usuario 01 Inicio de sesión del paciente (Tabla 3) y 02 selección del médico a cargo (Tabla 4). Esta iteración se solicitó que fuera realizada superficialmente el acceso con permiso de usuario y la extracción de datos sería llevada a cabo a futuro ya que la prioridad es la detección y metrología de movimientos faciales.

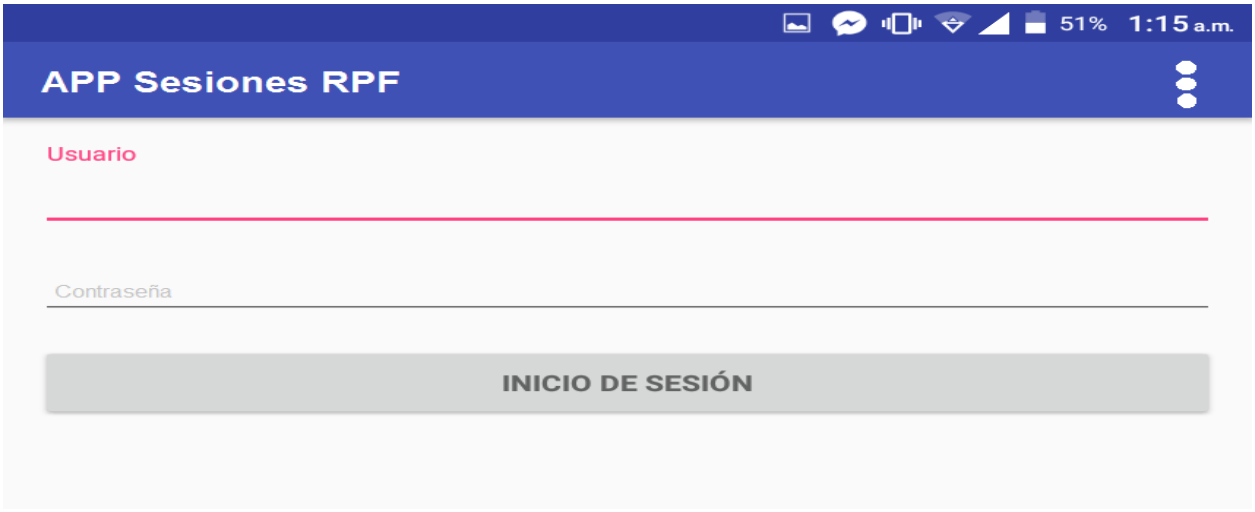

Figura 3. Interfaz de inicio de sesión del paciente

Figura 4. Interfaz de Búsqueda y selección del médico

Iteración 2. En esta iteración se realizó solo la historia de usuario 03 Detección del rostro (Tabla 05). En esta iteración se hizo entrega de la detección del rostro de una persona en estado de reposo y se realizó una metrología de ojos, nariz, boca y el rostro esto serviría de apoyo los siguientes cálculos de movimientos faciales.

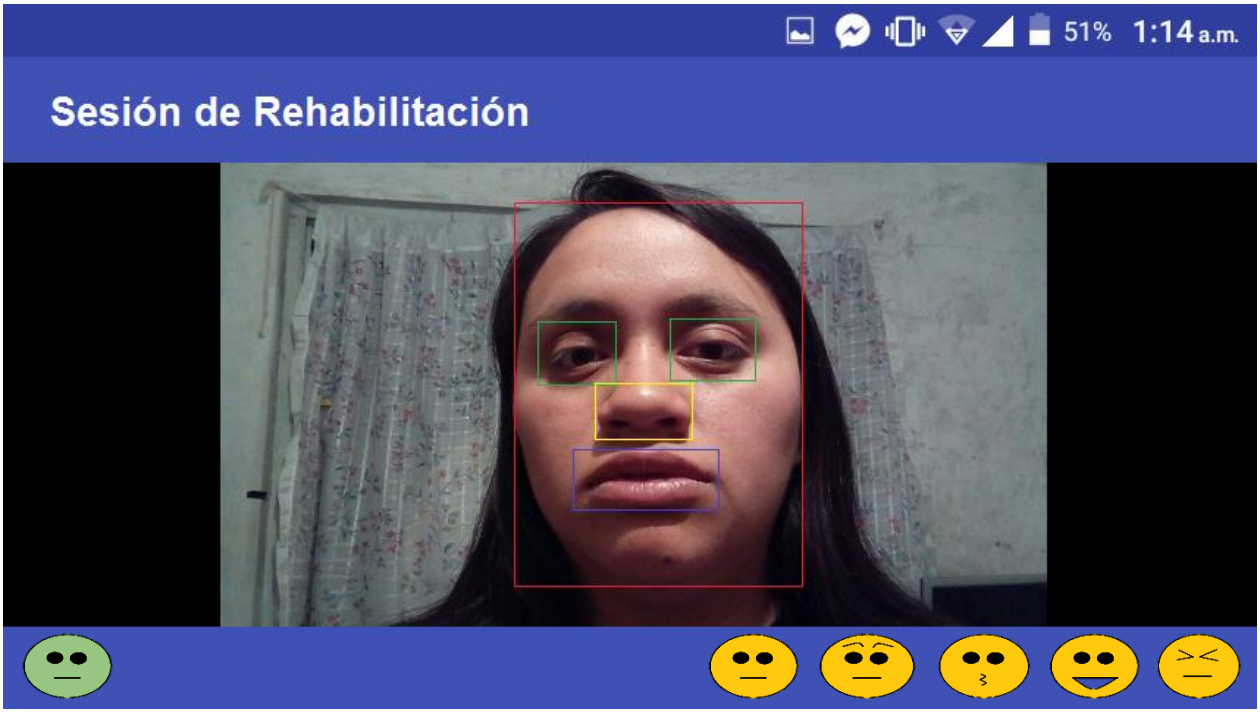

Figura 5. Interfaz de Detección del rostro en estado de reposo

Iteración 3: En esta iteración se realizó la historia de usuario 04 Detección de levantamiento de cejas (Tabla 6). Por medio de la iteración anterior (Iteración 2) se pudo marcar el punto donde aparecen las cejas en estado de reposo y posteriormente se implantaron métodos sugeridos por el co-asesor del proyecto con los cuales se pudo determinar el limite promedio en el que una persona alza una ceja con ambos valores se puede valorar si el paciente cumple con el ejercicio.

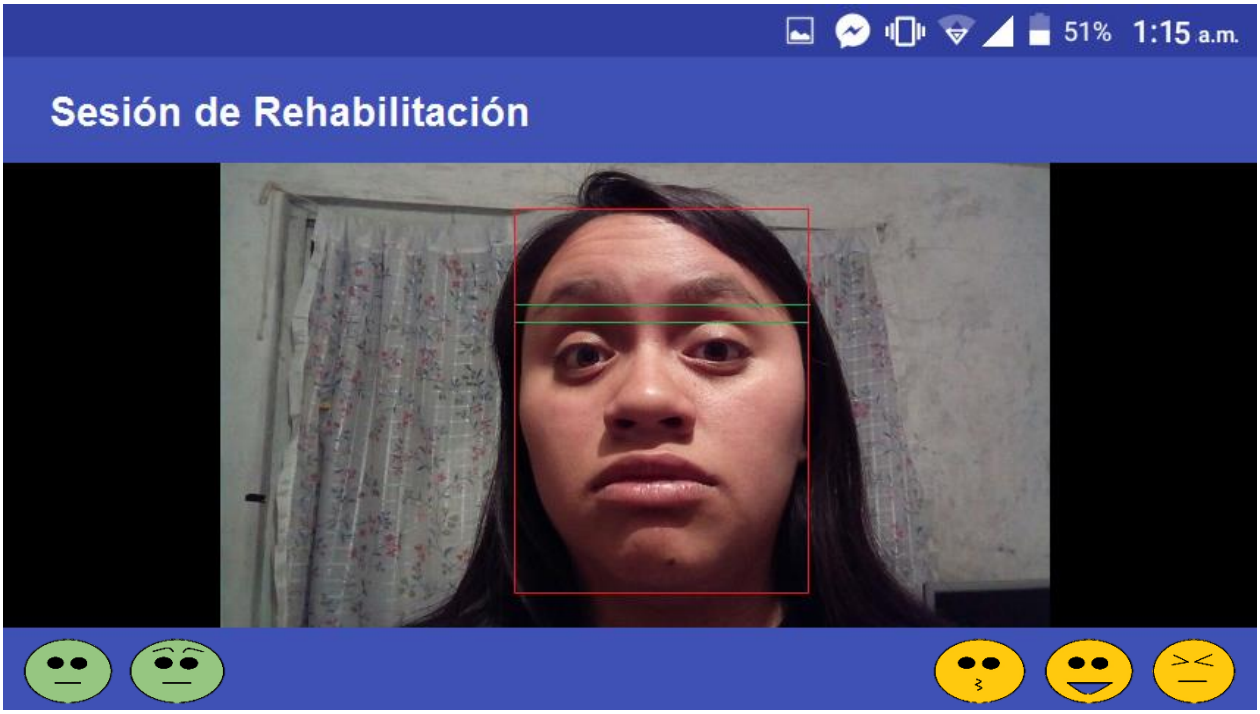

Figura 6. Interfaz de Detección del rostro en estado de reposo

Iteración 4: En esta iteración se realizó la historia de usuario 05 Detección de contracción de labios (Tabla 7). La detección de labios contraídos es comparada con las métricas obtenidas en el rostro en estado de reposo brindando después de un proceso se pude permite definir si el paciente ha cumplido con el ejercicio.

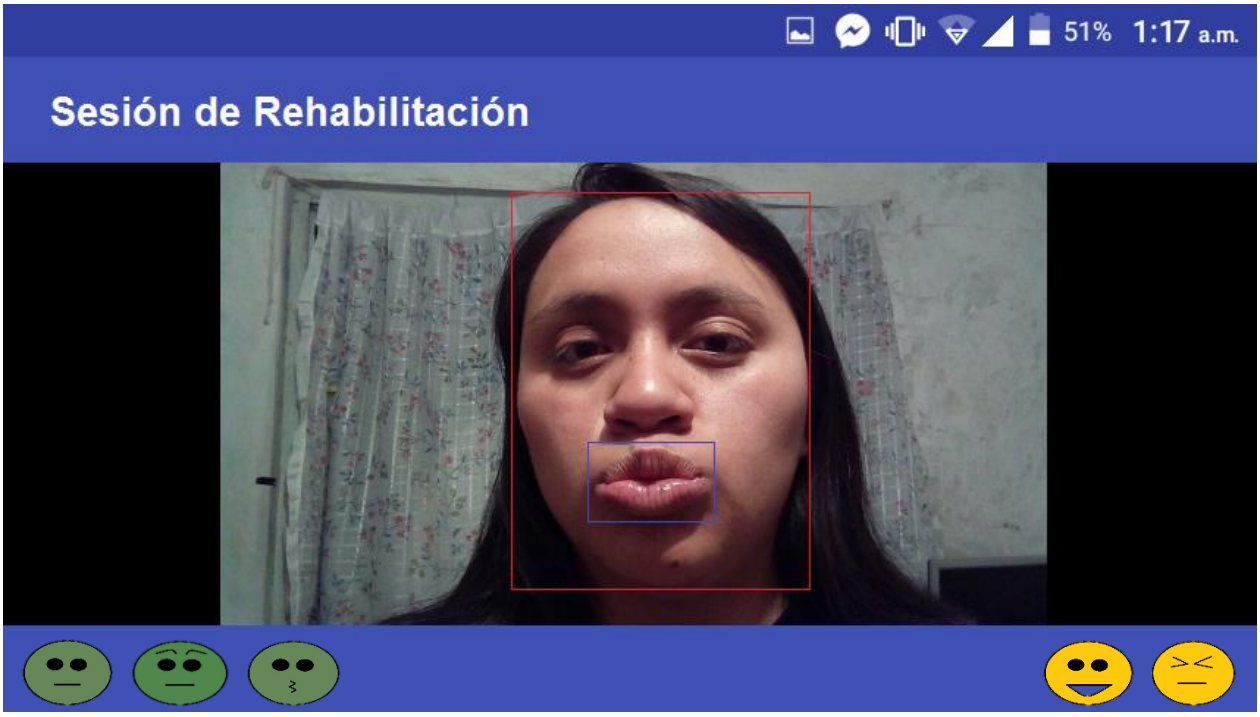

Figura 7. Interfaz de Detección de labios contraídos

Iteración 5: En esta iteración se realizó la historia de usuario 06 Detección de sonrisa (Tabla 8). Al momento de detectar este movimiento facial se encontraron errores de cálculo en la forma en que se está realizando la métrica de la boca, ya que el algoritmo detectó como sonrisa parte de las mejillas, esto es provocado por que al momento de sonreír generamos sombra en dicha zona, por este motivo y para no retrasar la entrega de la aplicación se acordó con el asesor del proyecto que este movimiento facial fuera descartado, sin embargo, a futuro este será aplicado.

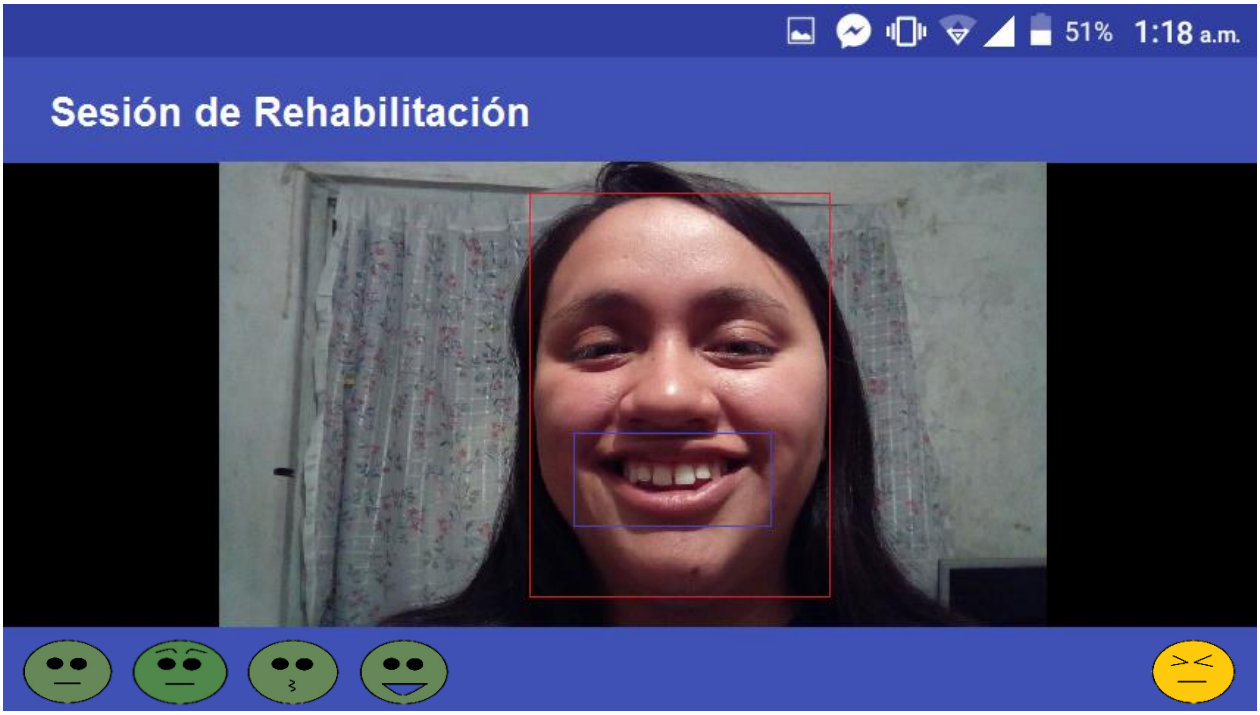

Figura 8. Interfaz de Detección de sonrisa

Iteración 6: En esta iteración se realizó la historia de usuario 07 Detección de ojos cerrados fuertemente (Tabla 9). La métrica de este movimiento facial fue descartada debido a que al momento de cerrar con fuerza los ojos, se generan sombras al lado de ambos ojos las cuales afectan el cálculo de los mismos. No obstante, el asesor junto con el co-asesor acordó que a futuro este movimiento podría ser cambiado para que al momento de que se cerrarán los ojos sea de una manera suave permitiendo realizar una métrica adecuada.

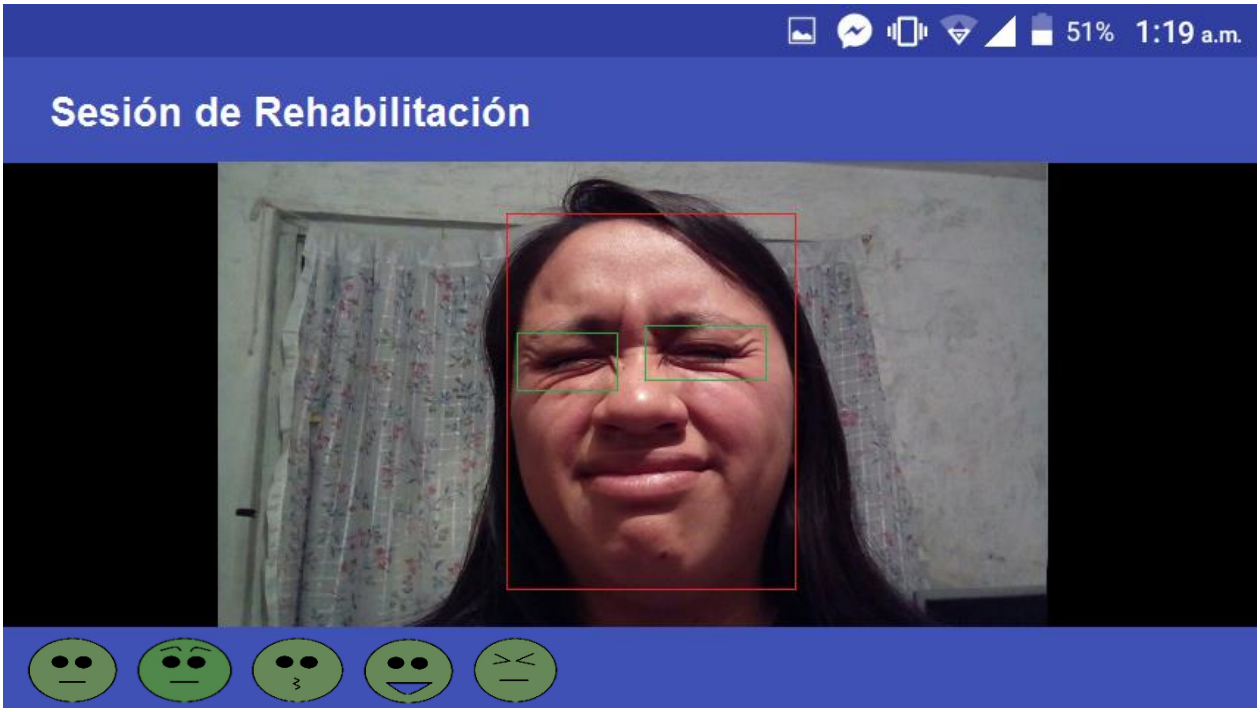

Figura 9. Interfaz de Detección de cerrar ojos

Iteración 7: En esta iteración corresponden las historias de usuario 08 y 09 las cuales están enfocadas a la adquisición y almacenamiento de datos, ambas historias de usuario, se llevarán a cabo posteriormente a la entrega final del proyecto, esto fue lo que se acordó con el asesor dado que por problemas administrativos el proyecto estuvo detenido un tiempo y se realizó un compromiso de continuar con el proyecto posteriormente la entrega final. Es importante recalcar que se realizaron las funciones marcadas como prioridad para el funcionamiento de la Aplicación.

3.1.4 Fase de Producción

Conforme cada iteración fue aprobada se fue avanzando en el desarrollo de esta aplicación es importante mencionar que se realizaron pruebas las cuales fueron llevadas a cabo de manera informal por este motivo no se tiene un registro formal correspondientes a una prueba, sin embargo, se muestran los resultados.
Se acordó que las pruebas formales se realizarán a futuro con pacientes con parálisis facial reales, sin embargo, su lanzamiento aun no es probable debido a que existen funciones por realizar.

Durante las pruebas realizadas se obtuvieron los siguientes resultados:

La detección del rostro no era posible si la posición del celular se encontraba de manera vertical por este motivo se colocó por default la posición de “ladscape”, para que la aplicación trabajara bajo el modo horizontal.

Cuando se detectaba el rostro se trazaba una elipse alrededor de este y en los ojos círculos, los cuales fueron cambiados de manera que se trazarán rectángulos con el fin de facilitar los procedimientos en el cálculo de las métricas.

Las fotografías que eran tomadas al aire libre en algunas ocasiones provocaban detección de rostros inexistentes por este motivo se tomó la decisión que las fotografías fueran tomadas en entornos cerrados o con un fondo blanco.

Las ejecuciones de la aplicación en lugares con poca luz a veces causaban que el rostro no fuera encontrado, por este motivo el uso de la aplicación se debe realizar en lugares con iluminación adecuada.

Se realizaron pruebas de la ejecución en un dispositivo móvil Bmobile AX600 con las siguientes características:

- Sistema Operativo: Android 4.4.2 (KitKat)
- Cámara frontal: 2 Megapíxeles
- Procesador: Dual-core 1,2 GHz
- Memoria RAM: 512 Mb
- Memoria teléfono: 4Gb

Como resultado en este dispositivo la detección del rostro era lenta. Por este motivo se ejecutaron las iteraciones a partir de la 04 a la 07 en 2 dispositivos móviles diferentes, en estos la detección del rostro mejoro en tiempo. A continuación, se muestra el modelo de ambos dispositivos y sus características de hardware:

Alcatel OneTouch PIXI 4:

- Cámara frontal: 8 Megapíxeles
- Sistema Operativo: Android 6.0 Marshmallow
- Procesador: Quadcore 1.3Ghz
- Memoria RAM: 1 GB

- Memoria de teléfono: 8Gb

LG x180g zone:

- Cámara frontal: 8 Megapíxeles
- Sistema Operativo: Android 5.1
- Procesador: Octa Core 1.4 GHz
- Memoria RAM: 1 Gb
- Memoria de teléfono: 16 Gb

Cuando se realizó la ejecución de detección de rostro en personas con fleco o barba provocaba errores de detección. Se recomienda que la ejecución se realice con el rostro descubierto sin fleco o barba.

La detección del rostro ojos, boca, nariz durante cada ejercicio de rehabilitación provocaba confusión del ejercicio que se tenía que realizar por este motivo se tomó la decisión de solo mostrar la detección del segmento del rostro al que el ejercicio hiciera referencia junto con un emoticón en la barra inferior para mostrarle al usuario el ejercicio al que correspondía.

Durante el cálculo de la sonrisa al momento de aplicar el filtro de blanco y negro se obtuvieron valores que no correspondían a la boca ya que se tomaban valores del filtro en las mejillas. Con la intención de no retrasar la entrega de la aplicación el cliente sugirió que este movimiento facial fuera descartado, sin embargo, a futuro este será aplicado. Así mismo se realizará una investigación acerca de los filtros o cambios que se necesiten realizar para poder realizar un cálculo correcto.

Al mismo tiempo en la aplicación del filtro de blanco y negro en la detección de ojos cerrados se obtuvieron valores equivocados ya que se generaban sombras al lado de los ojos las cuales afectan el cálculo de los mismos. Para resolver este problema se acordó que a futuro este movimiento podría ser cambiado para que al momento de que se cerrarán los ojos sea de una manera suave permitiendo realizar una métrica adecuada.

Los requerimientos mínimos para ejecutar esta aplicación en un dispositivo móvil son:

- Sistema Operativo: Android 4.4.2 KitKat
- Cámara frontal: 8 Megapíxeles
- Procesador: Quadcore 1.3Ghz
- Memoria RAM: 1 GB
- Memoria de teléfono libre: 150 Mb

3.1.5 Fase de mantenimiento

Aún no se llegado a esta fase ya que esta etapa es considerada como “el estado normal de un proyecto de XP”, para hacer esto posible la aplicación debería estar liberada para posteriormente mantenerla sin problemas ya sea agregando nuevas funciones, además se permitiría la incorporación o modificación de los integrantes del equipo.

3.1.6 Fase de Muerte del proyecto

Debido a que aún existen historias de usuario por realizar aún no se ha llegado a esta, cuando se realice esta fase se realizara la documentación pertinente a la aplicación.

CONCLUSIONES Y RECOMENDACIONES

El desarrollo de aplicaciones móviles en los últimos años se está convirtiendo en una tendencia importante dentro de la sociedad estos dispositivos permiten realizar grandes funciones por medio de aplicaciones en las cuales muchas veces sólo nos enfocamos en áreas como redes sociales y entretenimiento cuando también podemos desarrollar para el área de la salud, este es un sector que comúnmente he notado que no tomamos en cuenta porque pensamos que se necesitan equipos especializados pero la realidad es diferente.

Durante la estadía puede implementar algunas cosas de lo que se enseñó dentro de la Universidad Politécnica de Puebla, por ejemplo, el cómo llevar el proceso de desarrollo como lo es el levantamiento de requerimientos en este caso mencionadas como historias de usuarios, el proceso de iteraciones con el asesor, así como en la implementación de la lógica de programación.

Como recomendación para los futuros proyectos es que cuando se pretenda realizar una aplicación para las plataformas de dispositivos móviles de Android y Apple se desarrolle de primera instancia para la plataforma Android debido a que la empresa de Apple solicita licencia de desarrollador para poder controlar las funciones de las cámaras las cuales son elementos esenciales en el reconocimiento de rostro utilizando la librería Opencv.

GLOSARIO

ADT : (Android Developer Tools). Es un plugin para Eclipse que permite construir aplicaciones para Android.

Compilar: Traducir con un compilador un programa en lenguaje de alto nivel a lenguaje de la máquina.

IDE: (Integrated Development Environment). Es una herramienta que permite a los desarrollares diseñar y programar de una manera amigable sus aplicaciones, brindando ayuda visual en la sintaxis, plantillas, wizards, plugins y sencillas opciones para probar y hacer un debug.

NDK: (Native Development Kit). Permite a los desarrolladores reutilizar código escrito en C/C++ introduciéndolo en las aplicaciones.

XP: (Extreme Programming). Programación Extrema Metodología de desarrollo ágil de software.

ANEXO

En este apartado se muestra parte del código utilizado para el funcionamiento de esta aplicación no se permitió mostrar código sobre la detección del rostro con las metrologías debido a que dentro del instituto Nacional Astrofísica Óptica y Electrónica (INAOE), se continúa trabajando con estos algoritmos los cuales no pueden ser mostrados hasta que la aplicación este liberado en su totalidad esto implica los módulos que se planean integrar a futuro.

Anexo 1. Código para importar la librería de Opencv a Android estudio desde el CMakeList.txt

```
# Sets the minimum version of CMake required to build the native library.
# library. You should either keep the default value or only pass a
# value of 3.4.0 or lower

set(pathToProject C:/Users/0.0.0.0/AndroidStudioProjects/MyApplication)
set(pathToOpenCv C:/OpenCV-2.4.9-android-sdk)

cmake_minimum_required(VERSION 3.4.1)

set(CMAKE_VERBOSE_MAKEFILE on)
set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -std=gnu++11")
# OpenCV
include_directories(${pathToOpenCv}/sdk/etc/haarcascades/)
include_directories(${pathToOpenCv}/sdk/native/jni/include/)
add_library( lib_opencv SHARED IMPORTED )
set_target_properties(lib_opencv PROPERTIES IMPORTED_LOCATION
${CMAKE_CURRENT_SOURCE_DIR}/src/main/jniLibs/${ANDROID_ABI}/libopencv_java.so)
# Creates and names a library, sets it as either STATIC
# or SHARED, and provides the relative paths to its source code.
# You can define multiple libraries, and CMake builds them for you.
# Gradle automatically packages shared libraries with your APK.

add_library( # Sets the name of the library.
```

```
            com_example_xicale_myapplication_OpencvNativeClass

            # Sets the library as a shared library.
            SHARED

            # Provides a relative path to your source file(s).
src/main/jni/com_example_xicale_myapplication_OpencvNativeClass.cpp )

# Searches for a specified prebuilt library and stores the path as a
# variable. Because CMake includes system libraries in the search path by
# default, you only need to specify the name of the public NDK library
# you want to add. CMake verifies that the library exists before
# completing its build.

find_library( # Sets the name of the path variable.
              log-lib

              # Specifies the name of the NDK library that
              # you want CMake to locate.
              log )
# Specifies libraries CMake should link to your target library. You
# can link multiple libraries, such as libraries you define in this
# build script, prebuilt third-party libraries, or system libraries.

target_link_libraries( # Specifies the target library.
                    com_example_xicale_myapplication_OpencvNativeClass

                       # Links the target library to the log library
                       # included in the NDK.
                       $\{log-lib} lib_opencv)
```

Anexo 2. Código para la activación de la cámara frontal

```
package com.example.xicale.myapplication;
```

```
import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import org.opencv.android.BaseLoaderCallback;

import org.opencv.android.CameraBridgeViewBase;

import org.opencv.android.JavaCameraView;

import org.opencv.android.LoaderCallbackInterface;

import org.opencv.android.OpenCVLoader;

import org.opencv.core.CvType;

import org.opencv.core.Mat;


public class MainActivity extends AppCompatActivity implements
CameraBridgeViewBase.CvCameraViewListener2 {

    private static final String TAG = "MainActivity";


    // Verifica que la conexion a OpenCv sea correcta

    static {

        if (OpenCVLoader.initDebug()) {

            Log.d(TAG, "Opencv conecto exitosamente");

        } else {

            Log.d(TAG, "Opencv no ha conectado");

        }

    }

    // Used to load the '"com_example_xicale_myapplication_OpencvNativeClass'
library on application startup.


    static {


System.loadLibrary("com_example_xicale_myapplication_OpencvNativeClass");

    }


    JavaCameraView javaCameraView;
```

```
    Mat mRgba;
    BaseLoaderCallback mLoaderCallback = new BaseLoaderCallback(this) {
        @Override
        public void onManagerConnected(int status) {
            switch (status) {
                case BaseLoaderCallback.SUCCESS:
                javaCameraView.enableView();
            break;
                    default:
                super.onManagerConnected(status);
            break;
        }
    }
};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        // Example of a call to a native method
     //   TextView tv = (TextView) findViewById(R.id.sample_text);
      //  tv.setText(stringFromJNI());
        javaCameraView= (JavaCameraView)findViewById(R.id.java_camara_view);
        javaCameraView.setVisibility(View.VISIBLE);
        javaCameraView.setCameraIndex(1);
        javaCameraView.setCvCameraViewListener(this);

    }

    /**
     * A native method that is implemented by the 'native-lib' native library,
```

```
     * which is packaged with this application.
     */
    // public native String stringFromJNI();

    @Override
    public void onCameraViewStarted(int width, int height) {
        mRgba = new Mat(height, width, CvType.CV_8UC4);
    }

    @Override
    public void onCameraViewStopped() {
        mRgba.release();
    }

    @Override
    public Mat onCameraFrame(CameraBridgeViewBase.CvCameraViewFrame
inputFrame) {
        mRgba = inputFrame.rgba();
        OpencvNativeClass.faceDetection(mRgba.getNativeObjAddr());
        return mRgba;

    }
    @Override
    protected void onPause(){
        super.onPause();
        if(javaCameraView!=null)
            javaCameraView.disableView();
    }

    protected void onDestroy(){
        super.onDestroy();
        if(javaCameraView!= null)
```

```
            javaCameraView.disableView();
    }


    protected  void onResume(){
        super.onResume();
        if (OpenCVLoader.initDebug()) {
            Log.i(TAG, "OpenCV cargo exitosamente 2");

mLoaderCallback.onManagerConnected(LoaderCallbackInterface.SUCCESS);
        }else{
            Log.i(TAG, "Opencv no ha cargado 2");
            OpenCVLoader.initAsync(OpenCVLoader.OPENCV_VERSION_2_4_9, this,
mLoaderCallback);
        }
    }
}
```

BIBLIOGRAFÍA

[1] http://docs.opencv.org/2.4/modules/core/doc/intro.html#. Extensión de la página oficial de Opencv en esta se puede consultar información acerca del funcionamiento de esta la librería.

[2] http://www.inaoe.edu.mx/historia.php?movil=0Extensión de la Página oficial del Instituto Nacional de Astrofísica Óptica y Electrónica, donde se describe brevemente la historia de la institución

[3] http://www-optica.inaoep.mx/?movil=0 Extensión de la Página oficial del Instituto Nacional de Astrofísica Óptica y Electrónica, en la cual se muestra información acerca del departamento del departamento de Óptica

[4] Ian Sommerville “Ingeniería del software” Editorial Pearson Educación, Madrid, 2005

[5] Chryssa Aliferi “Android Programming Cookbook “Editorial Java Code Geeks, n/a, 2016

[6] Sylvain Hébuterne, Sébastien Pérochon “Android Guía de desarrollo de aplicaciones para Smartphones y Tabletas” Editorial ENI, n/a, 2014

[7] Beck, K. “Extreme Programming Explained Embrace Change” Pearson Education, n/a, 1999.