**UNIVERSIDAD POLITÉCNICA DE PUEBLA**

INGENIERÍA EN INFORMÁTICA

PROYECTO DE ESTADÍA PROFESIONAL

# ANALISIS, DISEÑO E IMPLEMENTACION DE UNA BASE DE DATOS PARA LA ADMINISTRACION Y CONTROL DE PACIENTES CON PARALISIS FACIAL A TRAVES DE UNA APP EN ANDROID

CAMPOS DOLORES EDUARDO

ASESOR TÉCNICO

DR. FRANCISCO JAVIER RENERO CARRILLO

ASESOR ACADÉMICO

MTRA. REBECA RODRÍGUEZ HUESCA

# Índice

## Índice de figuras

# Introducción

La gestión de los pacientes por parte de un hospital es de gran importancia para una correcta administración tanto del hospital como para obtener los mejores resultados en lo que concierne a la atención del paciente. Una de las problemáticas que afecta a muchos hospitales es el no contar con un control automatizado. El presente trabajo pretende dar solución a dicho problema y de igual manera facilitar y hacer más rápida cada una de las operaciones que se realizan en un hospital. Mediante una aplicación móvil se busca agilizar el registro de pacientes y la consulta de información del mismo, de esta manera se ahorrará tiempo entre consultas médicas y así atender a más personas por día. El desarrollo se realizará en dispositivos con sistema Android debido a que este es más utilizado en comparación con otros sistemas móviles y además de que un dispositivo con dicha característica es más accesible.

# CAPITULO I
# INAOE

## 1.1 Antecedentes

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 [1].

## 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 Organigrama

A continuación se muestra en la siguiente figura el organigrama de la institución (INAOE).

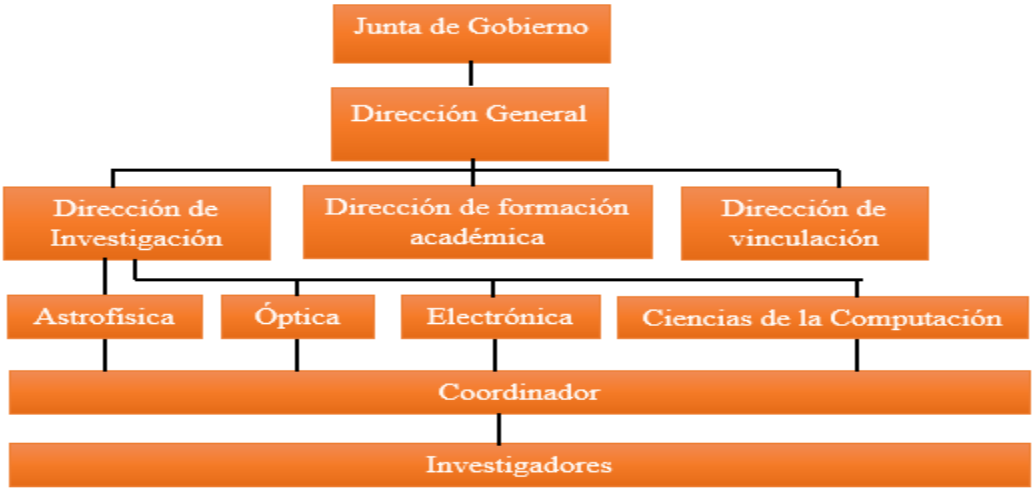



*Figura 1 Organigrama INAOE*

## 1.6 Coordinación 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.

## 1.7 Misión Coordinación 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

# CAPITULO II

# METODOLOGIA Y HERRAMIENTAS PARA EL DESARROLLO

## 2.1 Metodología

### 2.1 Programación Extrema (XP)

La programación Extrema (XP) es posiblemente el método ágil más conocido y ampliamente utilizado. El nombre fue acuñado por Beck (Beck, 2000) debido a que el enfoque fue desarrollado utilizando buenas practicas reconocidas, como el desarrollo iterativo, y con la participación del cliente en niveles extremos. En la programación extrema, todos los requerimientos se expresan como escenarios (llamados historias de usuario), los cuales se implementan directamente como una serie de tares. Los programadores trabajan en parejas y desarrollan pruebas para cada tarea antes de escribir el código. Todas las pruebas deben ejecutar satisfactoriamente cuando el código nuevo se integre al sistema. Existe un pequeño espacio de tiempo entre las entregas del sistema [2].

El ciclo de vida ideal de XP consiste de seis fases [3]: Exploración, Planificación de la Entrega (*Release*), Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

**Fase I Exploración**: En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto.

**Fase II Planificación de la Entrega:** En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente. Una entrega debería obtenerse en no más de tres meses.

**Fase III Iteraciones:** Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Al final de la última iteración el sistema estará listo para entrar en producción.

**Fase IV Producción:** La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

**Fase V Mantenimiento:** Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla

nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción.

**Fase VI Muerte del Proyecto:** Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

## 2.2 Herramientas

### 2.2.1 Android Studio

Android Studio es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJ IDEA . Además del potente editor de códigos y las herramientas para desarrolladores de IntelliJ, Android Studio ofrece aún más funciones que aumentan la productividad durante la compilación de apps para Android. Esta herramienta es muy intuitiva y fácil de usar, es mejor para el diseño de las interfaces en comparación con otras el único inconveniente es que los requisitos que piden son muy elevados, para que funcione bien el emulador se debe de tener una buena máquina [4].

### 2.2.2 PHP

PHP es un lenguaje de código abierto muy popular, adecuado para desarrollo web y que puede ser incrustado en HTML. Es popular porque un gran número de páginas y portales web están creadas con PHP. Código abierto significa que es de uso libre y gratuito para todos los programadores que quieran usarlo. Incrustado en HTML significa que en un mismo archivo vamos a poder combinar código PHP con código HTML, siguiendo unas reglas. PHP se utiliza para generar páginas web dinámicas. Recordar que llamamos página estática a aquella cuyos contenidos permanecen siempre igual, mientras que llamamos páginas dinámicas a aquellas cuyo contenido no es el mismo siempre. Por ejemplo, los contenidos pueden cambiar en base a los cambios que haya en una base de datos, de búsquedas o aportaciones de los usuarios, etc.

### 2.2.3 MySQL

MySQL es un sistema de administración de bases de datos (*Database Management System, DBMS*) para bases de datos relacionales. Así, MySQL no es más que una aplicación que permite gestionar archivos llamados de bases de datos. Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información. MySQL tiene velocidad al realizar las operaciones y eso lo hace uno de los mejores gestores con mejor rendimiento, es fácil la configuración y su instalación, soporta gran variedad de sistemas operativos y además posee un buen control de acceso de usuarios y seguridad en los datos.

# CAPITULO III

# ANALISIS, DISEÑO E IMPLEMENTACION DE UNA BASE DE DATOS PARA LA ADMINISTRACION Y CONTROL DE PACIENTES CON PARALISIS FACIAL A TRAVES DE UNA APP EN ANDROID

## 3.1 Instalación de herramientas

### 3.1.1 Instalación de Android Studio

Android Studio es la herramienta desarrollada por Google especialmente para los desarrolladores de aplicaciones de Android. Gracias a ella es posible programar dichas aplicaciones dentro de un entorno especializado a la vez que nos brinda una serie de posibilidades, como por ejemplo la posibilidad de crear máquinas virtuales y emuladores de Android para poder disponer de este sistema operativo desde nuestro ordenador. Para poder utilizar las ventajas del Android Studio en nuestro ordenador lo primero que debemos hacer es instalar el entorno de desarrollo así como el SDK que nos permite hacer uso de las funciones de Android en nuestro PC y poder programar para este sistema operativo.

Para ello lo primero que debemos hacer es descargar la versión más reciente de Android Studio desde la página web principal de desarrolladores de Android.


*Figura 2 Página web principal de desarrolladores de Android*

Una vez descargado el instalador correspondiente a nuestro sistema operativo (por defecto la propia web detectará nuestro sistema operativo y nos ofrecerá la mejor versión acorde a él) lo ejecutamos en nuestro equipo para comenzar con la instalación. Lo primero que veremos será el asistente de instalación de Android Studio.

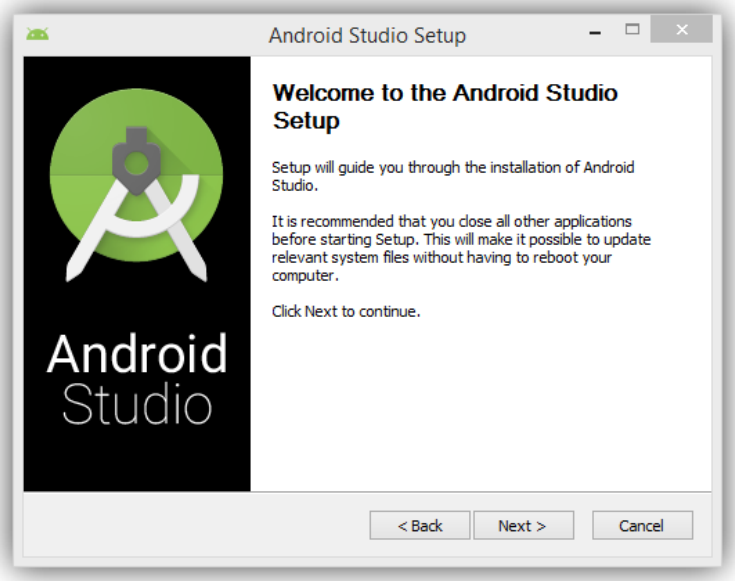



*Figura 3 Asistente de instalación Android Studio*

El proceso de instalación es muy sencillo, como prácticamente cualquier otra aplicación para nuestro sistema operativo, aunque de todas formas vamos a analizar el proceso paso a paso. Seguimos con el asistente y nos preguntará por los componentes que queremos instalar de esta suite de programación.

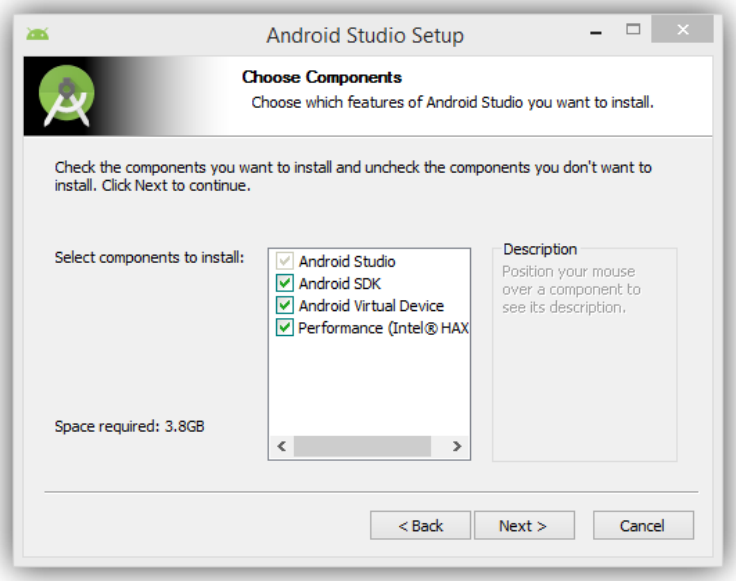



*Figura 4 Componentes a instalar*

Lo más recomendable es instalar todos para tener todas las funciones disponibles en caso de que queramos utilizarlas. Seguimos con el asistente y llegaremos a la licencia y a los términos de uso, que debemos aceptar para poder seguir con la instalación.

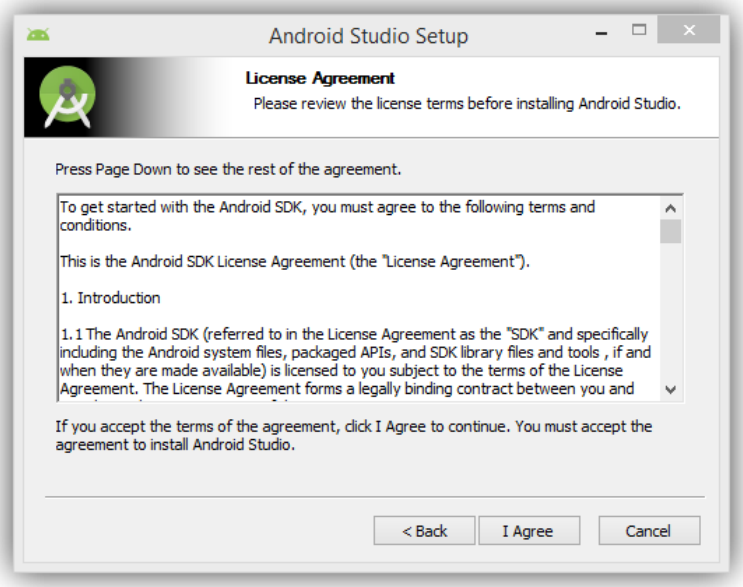



*Figura 5 Licencia y términos de uso*

En el siguiente paso debemos elegir la ruta donde instalaremos nuestro Android Studio. Debemos elegir una ruta para el programa en sí y otra diferente para instalar el SDK, con bastante espacio disponible ya que las descargas y actualizaciones de los componentes de este suelen ocupar bastante espacio.

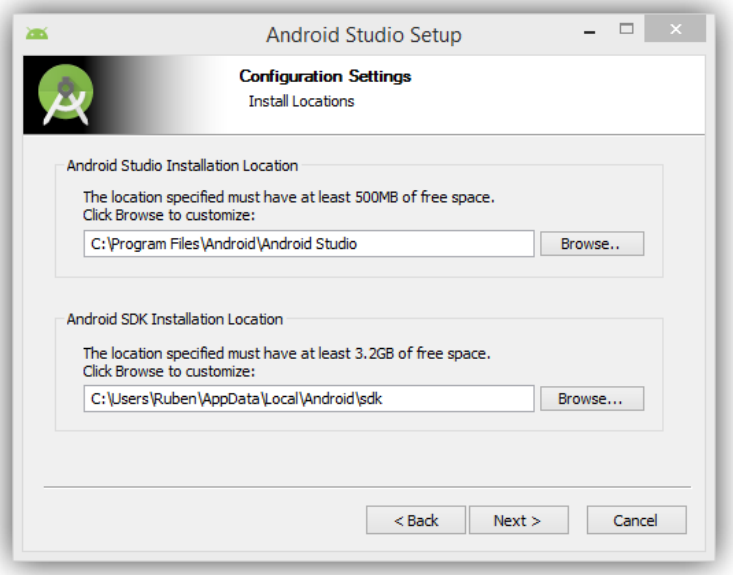



*Figura 6 Elegir ruta donde instalar Android Studio*

En el siguiente paso el asistente nos preguntará por la cantidad de memoria RAM que queremos asignar para el uso de máquinas virtuales y emuladores de Android.

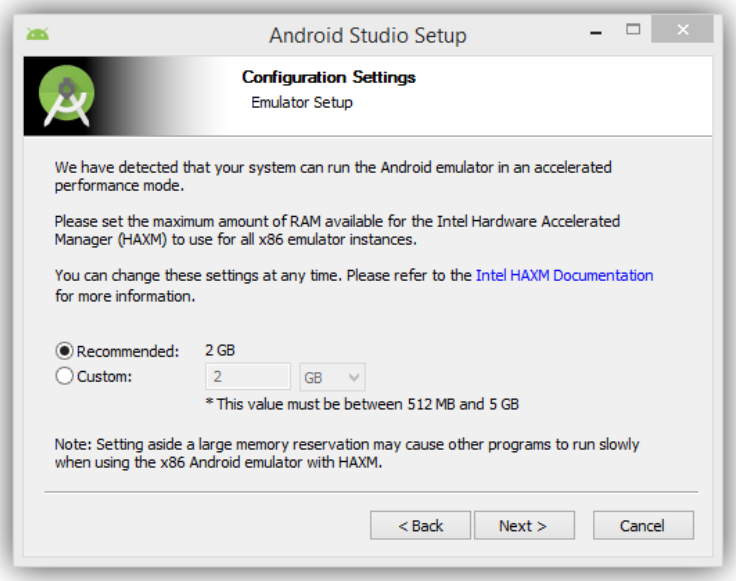



*Figura 7 Asignar memoria RAM para uso de emuladores de Android*

Con esto comenzará ya la copia de los archivos al disco duro. Este proceso puede tardar más o menos tiempo según la velocidad de nuestro sistema.

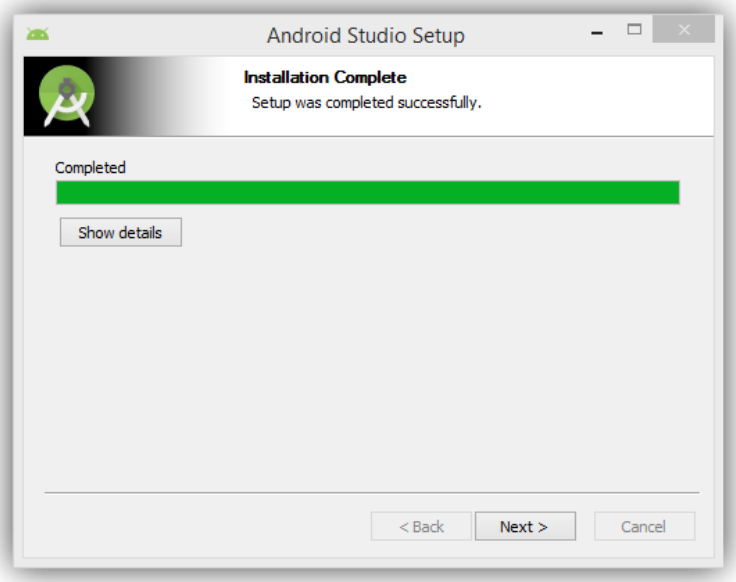



*Figura 8 Copia de archivos al disco duro*

Una vez finalice la instalación Android Studio se conectará a Internet y descargará los elementos del SDK necesarios para funcionar correctamente.

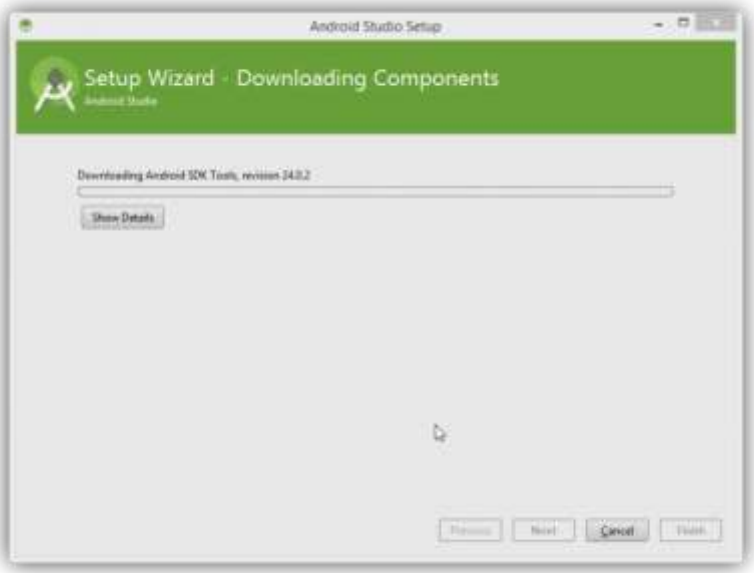



*Figura 9 Descarga elementos del SDK*

Una vez finalice esta copia de datos ya estaremos listos para utilizar nuestro entorno de programación de Android.

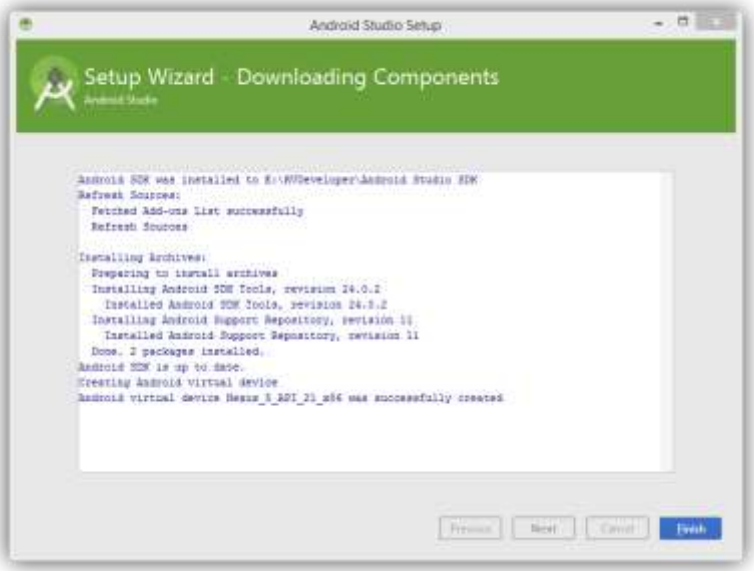



*Figura 10 Instalación completa*

### 3.1.2 Servidor

La configuración del servidor físico no fue posible por problemas administrativos debido que no se dio acceso al mismo en el tiempo establecido, es por esto que se optó por el uso de un servidor en la nube para así llevar acabo el desarrollo del proyecto.

## 3.2 Fases de la metodología

### 3.2.1 Fase de Exploración

En esta fase el cliente definió mediante unas historias de usuario lo que tiene que realizar la aplicación solicitada. Se identificaron 8 historias de usuarios, las cuales se muestran a continuación:

| **Historia de Usuario** | |
|---|---|
| **Número: 1** | **Usuario:** Administrador |
| **Nombre historia:** Gestión de médicos | |
| **Prioridad:** Alta | |
| **Iteración asignada:** 1 | |
| **Programador responsable:** Eduardo Campos | |
| **Descripción:**<br>Llevaremos el alta, baja y modificación de los datos relacionados con los médicos. | |
| **Observaciones:**<br>Se solicitó que el ID se generara con los datos ingresados dando como resultado un RFC. | |

*Figura 11 Historia de usuario Gestión de médicos*

| **Historia de Usuario** | |
|---|---|
| **Número: 2** | **Usuario:** Médicos |
| **Nombre historia:** Gestión de pacientes | |
| **Prioridad:** Alta | |
| **Iteración asignada:** 2 | |
| **Programador responsable:** Eduardo Campos | |
| **Descripción:**<br>Llevaremos el alta y modificación de los datos relacionados con los Pacientes. | |
| **Observaciones:**<br>Se solicitó que el ID se generara con los datos ingresados dando como resultado un RFC. | |

*Figura 12 Historia de usuario Gestión de Pacientes*

| Historia de Usuario | |
|---|---|
| **Número: 3** | **Usuario:** Médicos |
| **Nombre historia:** Registro Antecedentes Personales Patológicos | |
| **Prioridad:** Alta | |
| **Iteración asignada:** 2 | |
| **Programador responsable:** Eduardo Campos | |
| **Descripción:**<br>Llevaremos el registro del historial clínico del paciente en este caso los antecedentes personales patológicos. | |
| **Observaciones:** | |

*Figura 13 Historia de usuario Registro Antecedentes Personales Patológicos*

| Historia de Usuario | |
|---|---|
| **Número: 4** | **Usuario:** Médicos |
| **Nombre historia:** Registro Antecedentes Personales No Patológicos | |
| **Prioridad:** Alta | |
| **Iteración asignada:** 3 | |
| **Programador responsable:** Eduardo Campos | |
| **Descripción:**<br>Llevaremos el registro del historial clínico del paciente en este caso los antecedentes personales no patológicos. | |
| **Observaciones:** | |

*Figura 14 Historia de usuario Registro Antecedentes Personales No Patológicos*

| Historia de Usuario | |
|---|---|
| **Número: 5** | **Usuario:** Médicos |
| **Nombre historia:** Registro Antecedentes Gineco-obstetricos | |
| **Prioridad:** Alta | |
| **Iteración asignada:** 4 | |
| **Programador responsable:** Eduardo Campos | |
| **Descripción:**<br>Llevaremos el registro del historial clínico del paciente en este caso los antecedentes gineco-obstetricos | |
| **Observaciones:** | |

*Figura 15 Historia de usuario Registro Antecedentes Gineco-obstetricos*

| Historia de Usuario | |
|---|---|
| **Número: 6** | **Usuario:** Médicos |
| **Nombre historia:** Registro Exploración Física | |
| **Prioridad:** Alta | |
| **Iteración asignada:** 4 | |
| **Programador responsable:** Eduardo Campos | |
| **Descripción:**<br>Llevaremos el registro del historial clínico del paciente en este caso la exploración física. | |
| **Observaciones:** | |

*Figura 16 Historia de usuario Registro Exploración Física*

| Historia de Usuario | |
|---|---|
| **Número: 7** | **Usuario:** Médicos |
| **Nombre historia:** Registro Antecedentes Familiares | |
| **Prioridad:** Alta | |
| **Iteración asignada:** 4 | |
| **Programador responsable:** Eduardo Campos | |
| **Descripción:**<br>Llevaremos el registro del historial clínico del paciente en este caso los antecedentes familiares. | |
| **Observaciones:** | |

*Figura 17 Historia de usuario Registro Antecedentes Familiares*

| Historia de Usuario | |
|---|---|
| **Número: 8** | **Usuario:** Todos |
| **Nombre historia:** Control de acceso de usuarios | |
| **Prioridad en negocio:** Alta | |
| **Iteración asignada:** 5 | |
| **Programador responsable:** Eduardo Campos | |
| **Descripción:**<br>Antes de iniciar la aplicación se solicita el nombre de usuario y su clave para que tenga acceso a los datos que corresponden a su categoría de usuario.<br>Hay tres tipos de usuario: administrador, médico y paciente, con distintos permisos de acceso a los menús de acceso a las funcionalidades que les corresponden. | |

*Figura 18 Historia de usuario Control de acceso de usuarios*

### 3.2.2 Fase de Planificación

En esta fase se describe la planificación que ha seguido el proyecto a lo largo de su desarrollo, en la fase inicial y a lo largo de las 5 iteraciones. Este plan se encarga de mostrar las historias de usuario que serán implementadas en cada una de las iteraciones, así como la duración estimada de cada una y el orden en que se implementarán.

| Iteraciones | Orden de las historias de Usuario | Duración de las iteraciones |
|---|---|---|
| 1ra Iteración | Gestión de Médicos | 1 a 2 semanas |
| 2da Iteración | Gestión de Pacientes<br>Registro Antecedentes Personales Patológicos | 1 a 2 semanas |
| 3ra Iteración | Registro Antecedentes Personales No Patológicos<br>Registro Antecedentes Gineco-obstetricos | 1 a 2 semanas |
| 4ta Iteración | Registro Exploración Física<br>Registro Antecedentes Familiares | 1 a 2 semanas |
| 5ta Iteración | Control de acceso de usuarios | 1 semana |

*Figura 19 Duración de las iteraciones*

### 3.2.3 Fase Iteraciones

Esta es la fase principal en el ciclo de desarrollo. Las funcionalidades fueron desarrolladas generando al final de cada una de ellas un entregable funcional que se implementaron en las historias de usuario asignadas en la iteración. Cabe mencionar que el cliente participo activamente para recabar todos los datos necesarios para el desarrollo de cada funcionalidad.

#### 1ra Iteración

En esta iteración se le dará cumplimiento a la historia de usuario Gestión de Médicos que es de prioridad alta. Las siguientes figuras muestran los resultados:


*Figura 20 Menú para usuario Administrador*


*Figura 21 Formulario para registrar Medico*


*Figura 22 Buscar Medico por ID*

**2da Iteración**

Esta iteración tendrá como objetivo darle cumplimientos a las historias de usuario Gestión de Pacientes y Registro Antecedentes Personales Patológicos. Las siguientes figuras muestran los resultados.


*Figura 23 Menú para usuario Medico*


*Figura 24 Formulario para registrar un Paciente*

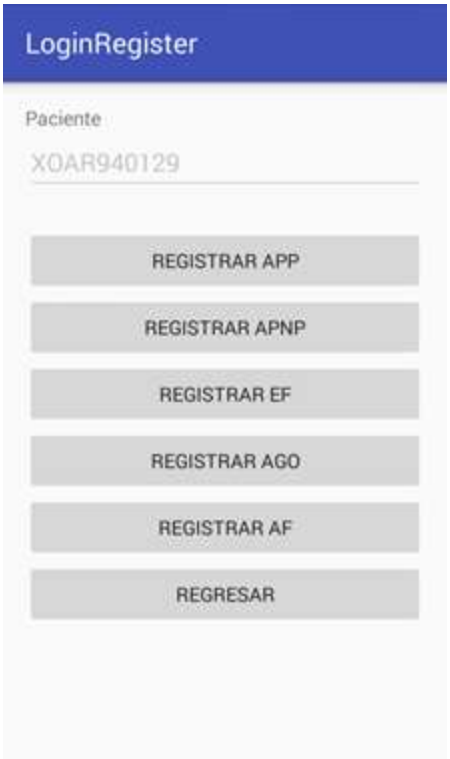



*Figura 25 Menú para registrar Historial Clínico*

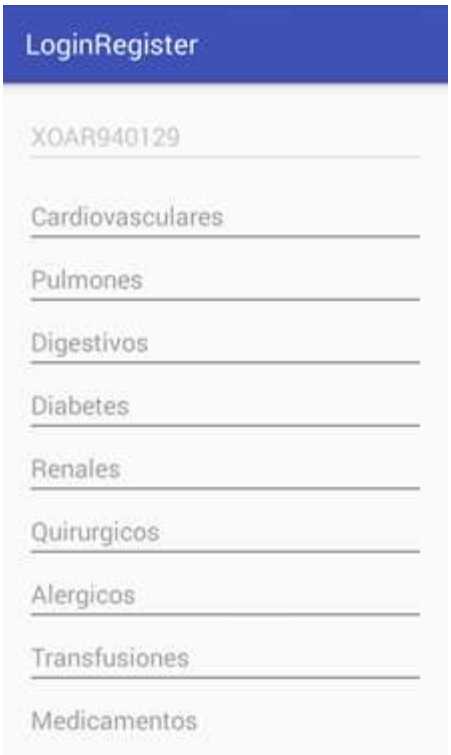



*Figura 26 Formulario para Registrar APP*

**3ra Iteración**

En esta iteración se implementaran las historias de usuarios Registro Antecedentes Personales No Patológicos y Registro Antecedentes Gineco-obstetricos

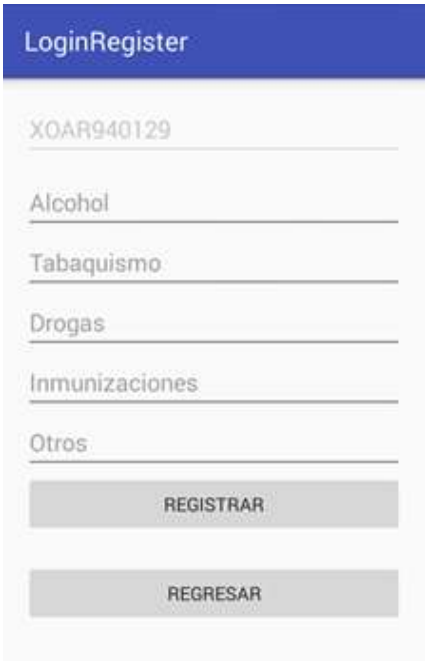



*Figura 27 Formulario para Registrar APNP*


*Figura 28 Formulario para Registrar AGO*

**4ta Iteración**

La implementación de las historias Registro Exploración Física y Registro Antecedentes Familiares en esta iteración proporcionará una idea completa de la aplicación.

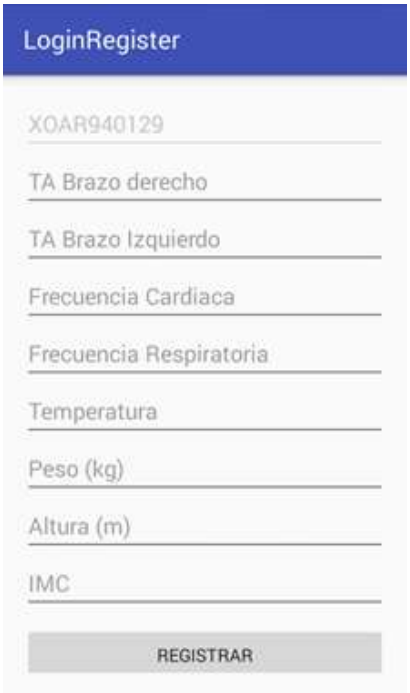



*Figura 29 Formulario para Registrar EF*

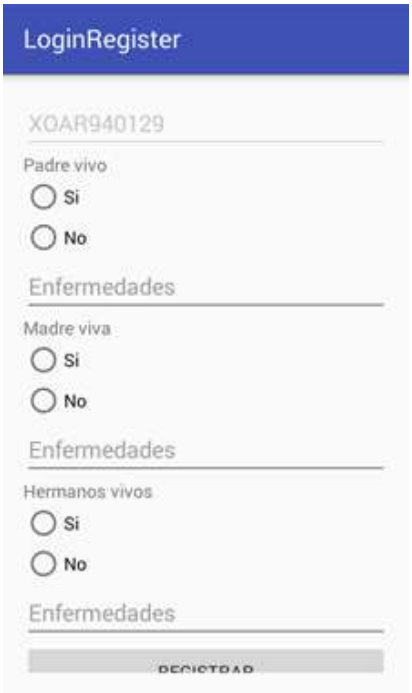



*Figura 30 Formulario para Registrar AF*

**5ta Iteración**

En esta iteración se le dará cumplimiento a la historia de usuario Control de acceso de usuarios, al finalizar su implementación quedará terminado el sistema.


*Figura 31 Pantalla de Login*

### 3.2.4 Fase Producción

Este sistema fue diseñado para facilitar la gestión de la información de pacientes y médicos y para lograrlo el diseño visual es de gran importancia, por ese motivo la aplicación presenta un diseño simple y sencillo para que sea más fácil su uso y los usuarios se sientan satisfechos al momento de utilizarlo. El diseño de la base de datos fue realizado con la herramienta SQL Power Architect, la misma está compuesta por 9 tablas que cumplen con las normas establecidas para el diseño de base de datos.

**Pruebas**

Una historia de usuario no se considera completa hasta que no ha pasado por sus pruebas funcionales es por eso que se realizaron diferentes pruebas funcionales a la aplicación, las mismas se muestran a continuación:

| Prueba Funcional | |
|---|---|
| **Numero de Prueba:** 1 | **Numero de historia de usuario:** 1 |
| **Nombre de la prueba:** Registro Medico | |
| **Descripción:** El usuario vera un formulario en la que se le solicitara los datos personales para registrar un médico. El usuario debe introducir estos campos y si deja campos importantes vacíos la aplicación no registra la información. | |
| **Condiciones de ejecución:** Ninguna | |
| **Entrada:**<br>-El usuario presiona botón "Registrar medico"<br>-Aparece un formulario donde solicita los datos personales para registrar a un medico<br>-El usuario introduce los datos y presiona el botón "Registrar Medico"<br>-La aplicación verifica que el campo Nombre, Apellido Materno, Apellido Paterno, Fecha de nacimiento no estén vacíos<br>-La aplicación muestra un mensaje de error "Verifique campos" y se muestra el mismo formulario | |
| **Resultado esperado:** Si el campo Nombre, Apellido Materno, Apellido Paterno y Fecha de Nacimiento estas vacíos la aplicación no debe registrar los datos | |
| **Evaluación de la prueba:** Prueba satisfactoria | |

*Figura 32 Prueba Funcional Primera Iteración*

| Prueba Funcional | |
|---|---|
| **Numero de Prueba:** 2 | **Numero de historia de usuario:** 2 |
| **Nombre de la prueba:** Registro Paciente | |
| **Descripción:** El usuario vera un formulario en la que se le solicitara los datos para registrar un paciente. El usuario debe introducir estos campos y si deja campos importantes vacíos la aplicación no registra la información. | |
| **Condiciones de ejecución:** Ninguna | |
| **Entrada:**<br>-El usuario presiona botón "Registrar paciente"<br>-Aparece un formulario donde solicita los datos para registrar a un paciente<br>-El usuario introduce los datos y presiona el botón "Registrar Paciente"<br>-La aplicación verifica que el campo Nombre, Apellido Materno, Apellido Paterno, Fecha de nacimiento no estén vacíos<br>-La aplicación muestra un mensaje de error "Verifique campos" y se muestra el mismo formulario | |
| **Resultado esperado:** Si el campo Nombre, Apellido Materno, Apellido Paterno y Fecha de Nacimiento estas vacíos la aplicación no debe registrar los datos | |
| **Evaluación de la prueba:** Prueba satisfactoria | |

*Figura 33 Prueba Funcional Segunda Iteración*

| Prueba Funcional | |
|---|---|
| **Numero de Prueba:** 3 | **Numero de historia de usuario:** 3 |
| **Nombre de la prueba:** Registro APP | |
| **Descripción:** El usuario vera un formulario en la que se le solicitara los datos para registrar el historial clínico de paciente, en este caso Antecedentes Personales Patológicos. | |
| **Condiciones de ejecución:** Registrar a Paciente | |
| **Entrada:**<br>-El usuario presiona botón "Registrar APP"<br>-Aparece un formulario donde solicita los datos para registrar los Antecedentes Personales Patológicos de un paciente<br>-El usuario introduce los datos y presiona el botón "Registrar"<br>-La aplicación verifica que los campos no tengan números<br>-La aplicación muestra un mensaje "No se permiten números" y se muestra el mismo formulario | |
| **Resultado esperado:** Si en un campo fue escrito un numero la aplicación no debe registrar los datos | |
| **Evaluación de la prueba:** Prueba satisfactoria | |

*Figura 34 Prueba Funcional Segunda Iteración*

| Prueba Funcional | |
|---|---|
| **Numero de Prueba:** 4 | **Numero de historia de usuario:** 4 |
| **Nombre de la prueba:** Registro APNP | |
| **Descripción:** El usuario vera un formulario en la que se le solicitara los datos para registrar el historial clínico de paciente, en este caso Antecedentes Personales No Patológicos. | |
| **Condiciones de ejecución:** Registrar a Paciente | |
| **Entrada:**<br>-El usuario presiona botón "Registrar APNP"<br>-Aparece un formulario donde solicita los datos para registrar los Antecedentes Personales No Patológicos de un paciente<br>-El usuario introduce los datos y presiona el botón "Registrar"<br>-La aplicación verifica que los campos no tengan números<br>-La aplicación muestra un mensaje "No se permiten números" y se muestra el mismo formulario | |
| **Resultado esperado:** Si en un campo fue escrito un numero la aplicación no debe registrar los datos | |
| **Evaluación de la prueba:** Prueba satisfactoria | |

*Figura 35 Prueba Funcional Tercera Iteración*

| Prueba Funcional | |
|---|---|
| **Numero de Prueba:** 5 | **Numero de historia de usuario:** 5 |
| **Nombre de la prueba:** Registro AGO | |
| **Descripción:** El usuario vera un formulario en la que se le solicitara los datos para registrar el historial clínico de paciente, en este caso Antecedentes Gineco-Obstétricos. | |
| **Condiciones de ejecución:** Registrar a Paciente | |
| **Entrada:**<br>-El usuario presiona botón "Registrar AGO"<br>-Aparece un formulario donde solicita los datos para registrar los Antecedentes Gineco-Obstétricos de un paciente<br>-El usuario introduce los datos y presiona el botón "Registrar"<br>-La aplicación verifica que el campo FUM no tenga letras<br>-La aplicación muestra un mensaje "No se permiten letras, formato fecha" y se muestra el mismo formulario | |
| **Resultado esperado:** Si en el campo FUM se escriben letras la aplicación no debe registrar los datos | |
| **Evaluación de la prueba:** Prueba satisfactoria | |

*Figura 36 Prueba Funcional Tercera Iteración*

| Prueba Funcional | |
|---|---|
| **Numero de Prueba:** 6 | **Numero de historia de usuario:** 6 |
| **Nombre de la prueba:** Registro EF | |
| **Descripción:** El usuario vera un formulario en la que se le solicitara los datos para registrar el historial clínico de paciente, en este caso Exploración Física | |
| **Condiciones de ejecución:** Registrar a Paciente | |
| **Entrada:**<br>-El usuario presiona botón "Registrar EF"<br>-Aparece un formulario donde solicita los datos para registrar Exploración física de un paciente<br>-El usuario introduce los datos y presiona el botón "Registrar"<br>-La aplicación verifica que el campo peso y/o altura no tenga letras<br>-La aplicación muestra un mensaje "No se permiten letras" y se muestra el mismo formulario | |
| **Resultado esperado:** Si en el campo peso o altura se escriben letras no debe registrar los datos, además de que no se podrá calcular el IMC. | |
| **Evaluación de la prueba:** Prueba satisfactoria | |

*Figura 37 Prueba Funcional Curta Iteración*

| Prueba Funcional | |
|---|---|
| **Numero de Prueba:** 7 | **Numero de historia de usuario:** 7 |
| **Nombre de la prueba:** Registro AF | |
| **Descripción:** El usuario vera un formulario en la que se le solicitara los datos para registrar el historial clínico de paciente, en este caso Antecedentes Familiares | |
| **Condiciones de ejecución:** Registrar a Paciente | |
| **Entrada:**<br>-El usuario presiona botón "Registrar AF"<br>-Aparece un formulario donde solicita los datos para registrar los Antecedentes Familiares de un paciente<br>-El usuario introduce los datos y presiona el botón "Registrar"<br>-La aplicación verifica que se seleccionen los RadioButton<br>-La aplicación muestra un mensaje "Debe seleccionar una opción" y se muestra el mismo formulario | |
| **Resultado esperado:** Si no se selecciona un RadioButton la aplicación no debe registrar los datos | |
| **Evaluación de la prueba:** Prueba satisfactoria | |

*Figura 38 Prueba Funcional Cuarta Iteración*

| Prueba Funcional | |
|---|---|
| **Numero de Prueba:** 8 | **Numero de historia de usuario:** 8 |
| **Nombre de la prueba:** Verificación de usuario | |
| **Descripción:** El usuario al iniciar la aplicación vera una ventana de acceso en la que se le solicitara el nombre de usuario y la contraseña. El usuario debe introducir estos campos y si el usuario no existe, no tendrá acceso a la aplicación. | |
| **Condiciones de ejecución:** Ninguna | |
| **Entrada:**<br>-El usuario ejecuta la aplicación<br>-Aparece un cuadro de texto en el que se solicita el usuario y contraseña<br>-El usuario introduce los datos y presiona el botón "Aceptar"<br>-La aplicación verifica los datos en la base de datos y comprueba que no existe el usuario<br>-La aplicación muestra un mensaje "Usuario y/o Contraseñas incorrectos" y se muestra el mismo formulario | |
| **Resultado esperado:** Soló los usuarios dados de alta tienen acceso a la aplicación | |
| **Evaluación de la prueba:** Prueba satisfactoria | |

*Figura 39 Prueba Funcional Quinta Iteración*

### 3.2.5 Fase Muerte del proyecto

Debido a que ya no hay más historias de usuario que incluir al sistema y además el usuario quedo satisfecho en todos los aspectos. A partir de aqui ya no se generó nada y tampoco se realizó algún cambio de ningún tipo.

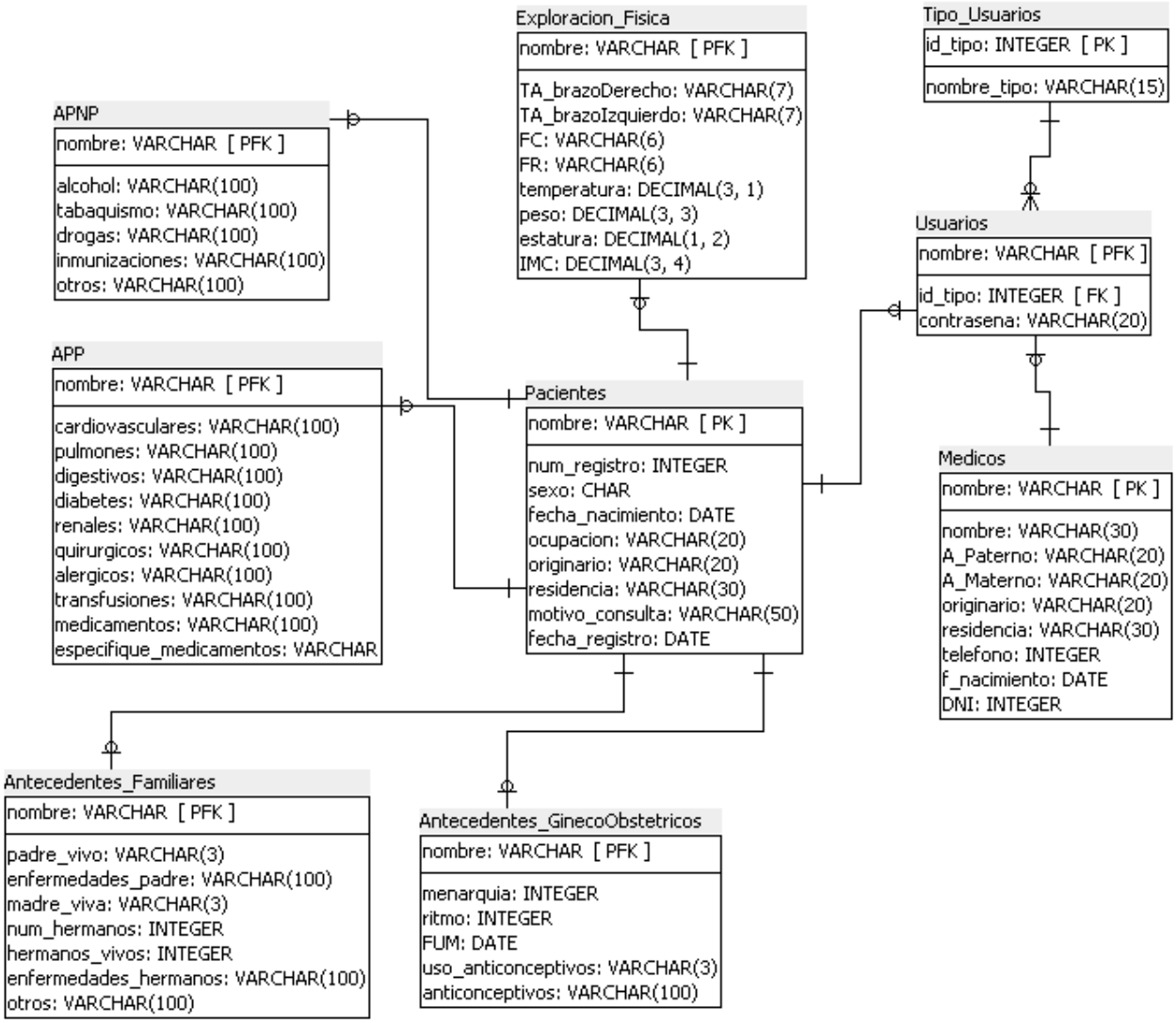
# Conclusión

El uso del sistema operativo Android para la aplicación es de gran ayuda para el cliente ya que puede encontrar mucha más variedad de dispositivos en el mercado, con los cuales nuestra aplicación es compatible.

Android al ser un sistema operativo libre (open source) nos brinda la libertad a los usuarios de adquirirlo y usarlo. Puedo concluir que es mucho más fácil la implementación en este sistema operativo ya que nos permite desarrollar aplicaciones con herramientas gratuita, también de alguna manera se me facilito ya que Android emplea lenguaje Java, con el cual trabaje durante el transcurso de mi carrera.

# Anexos

## Anexo 1 Diagrama de Base de datos de la aplicación



## Anexo 2 Código para el login

```
package com.familiar_cd.loginregister;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import com.android.volley.Request;
import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.toolbox.Volley;

import org.json.JSONException;
import org.json.JSONObject;

public class LoginActivity extends AppCompatActivity {
    private EditText editUsu;

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

        editUsu = (EditText) findViewById(R.id.editUsu);
        final EditText editContra = (EditText) findViewById(R.id.editContra);
        final Button bLog = (Button) findViewById(R.id.bAceptar);

        bLog.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final String username = editUsu.getText().toString();
                final String password = editContra.getText().toString();

                Response.Listener<String> responseListener = new
Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        try {
                            JSONObject jsonResponse = new
JSONObject(response);
                            boolean success =
jsonResponse.getBoolean("success");

                            if (success){
                                AlertDialog.Builder builder = new
AlertDialog.Builder(LoginActivity.this);
                                builder.setMessage("Bienvenido")
                                        .setPositiveButton("Aceptar", null)
                                        .create()
                                        .show();
```

```
                                String user =
jsonResponse.getString("username");

                                Intent UserAreIntent = new
Intent(LoginActivity.this, UseAreaAct.class);
                                //UserAreIntent.putExtra("username", user);

LoginActivity.this.startActivity(UserAreIntent);
                                GuardarPreferences();
                            }else{
                                AlertDialog.Builder builder = new
AlertDialog.Builder(LoginActivity.this);
                                builder.setMessage("Usuario y/o Contraseña
invalido")
                                        .setNegativeButton("Aceptar", null)
                                        .create()
                                        .show();
                            }
                        }catch (JSONException e){
                            e.printStackTrace();
                        }
                    }
                };
                LoginRequest loginRequest = new LoginRequest(username,
password, responseListener);
                RequestQueue queue =
Volley.newRequestQueue(LoginActivity.this);
                queue.add(loginRequest);
            }
        });
    }

    /*public void CargarPreferences(){
        SharedPreferences mispreferencias =
getSharedPreferences("PreferenciasUsuario", Context.MODE APPEND);
        editUsu.setText(mispreferencias.getString("usuario", ""));
    }*/

    public void GuardarPreferences(){
        SharedPreferences mispreferencias =
getSharedPreferences("PreferenciasUsuario", Context.MODE_APPEND);
        SharedPreferences.Editor editor = mispreferencias.edit();
        String user1 = editUsu.getText().toString();
        editor.putString("usuario", user1);
        editor.commit();
    }
}
```

## Anexo 3 Código para el login

```
package com.familiar_cd.loginregister;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.toolbox.StringRequest;

import java.util.HashMap;
import java.util.Map;

/**
 * Created by Familiar_CD.
 */

public class LoginRequest extends StringRequest{
    private static final String LOGIN_REQUEST_URL =
"http://rehabili.esy.es/Login.php";
    private Map<String, String> params;

    public LoginRequest(String username, String password,
Response.Listener<String> listener) {

        super(Request.Method.POST, LOGIN_REQUEST_URL, listener, null);
        params = new HashMap<>();
        params.put("username", username);
        params.put("password", password);
    }
        public Map<String, String> getParams(){
            return params;
        }

}
```

## Anexo 4 Código para Registrar Pacientes

```
package com.familiar_cd.loginregister;

import android.content.Context;
import android.content.Intent;
import android.content.SharedPreferences;
import android.support.v7.app.AlertDialog;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import com.android.volley.RequestQueue;
import com.android.volley.Response;
import com.android.volley.toolbox.Volley;

import org.json.JSONException;
import org.json.JSONObject;

public class RegisterPacActivity extends AppCompatActivity {
    private String nomPac;

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

        final EditText editNom = (EditText) findViewById(R.id.editNom);
        final EditText ediApePa = (EditText) findViewById(R.id.editApePa);
        final EditText editApeMa = (EditText) findViewById(R.id.editApeMa);
        final EditText editFecNac = (EditText) findViewById(R.id.editFecNac);
        final EditText editOcu = (EditText) findViewById(R.id.editOcu);
        final EditText editOri = (EditText) findViewById(R.id.editOri);
        final EditText editReside = (EditText) findViewById(R.id.editReside);
        final EditText editMotCon = (EditText) findViewById(R.id.editMotCon);

        final Button bReg = (Button) findViewById(R.id.bReg);
        final Button bAtr = (Button) findViewById(R.id.bAtr);

        bAtr.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent atrIntent = new Intent(RegisterPacActivity.this,
UseAreaAct.class);
                RegisterPacActivity.this.startActivity(atrIntent);
            }
        });



        bReg.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                final String nom = editNom.getText().toString().toUpperCase();
```

```
                final String apePa =
ediApePa.getText().toString().toUpperCase();
                final String apeMa =
editApeMa.getText().toString().toUpperCase();
                final String fecNac = editFecNac.getText().toString();
                final String ocu = editOcu.getText().toString();
                final String ori = editOri.getText().toString();
                final String reside = editReside.getText().toString();
                final String motCon = editMotCon.getText().toString();

                Response.Listener<String> responseListener = new
Response.Listener<String>() {
                    @Override
                    public void onResponse(String response) {
                        try{
                            JSONObject jsonResponse = new
JSONObject(response);
                            boolean success =
jsonResponse.getBoolean("success");
                            if(success){
                                AlertDialog.Builder builder = new
AlertDialog.Builder(RegisterPacActivity.this);
                                builder.setMessage("Datos guardados")
                                        .create()
                                        .show();


                                nomPac = jsonResponse.getString("nomPac");
                                Intent pacACT = new
Intent(RegisterPacActivity.this, PacActivity.class);
                                //pacACT.putExtra("nomPac", nomPac);

RegisterPacActivity.this.startActivity(pacACT);

                                editNom.setText("");
                                ediApePa.setText("");
                                editApeMa.setText("");
                                editFecNac.setText("");
                                editOcu.setText("");
                                editOri.setText("");
                                editReside.setText("");
                                editMotCon.setText("");
                                GuardarPreferences();



                            }else{
                                AlertDialog.Builder builder = new
AlertDialog.Builder(RegisterPacActivity.this);
                                builder.setMessage("No se guardaron los
datos")
                                        .setNegativeButton("Aceptar", null)
                                        .create()
                                        .show();
                            }
                        }catch (JSONException e){
                            e.printStackTrace();
```

```
                        }
                    }
                };
                RegisterPac registerPac = new RegisterPac(nom, apePa, apeMa,
fecNac, ocu, ori, reside, motCon, responseListener);
                RequestQueue queue =
Volley.newRequestQueue(RegisterPacActivity.this);
                queue.add(registerPac);
            }
        });
    }

    public void GuardarPreferences(){
        SharedPreferences mispreferencias =
getSharedPreferences("PreferenciasUsuario", Context.MODE_APPEND);
        SharedPreferences.Editor editor = mispreferencias.edit();
        editor.putString("nomPac", nomPac);
        editor.commit();
    }
}
```

## Anexo 5 Código para Registrar Pacientes

```
package com.familiar_cd.loginregister;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.toolbox.StringRequest;

import java.util.Date;
import java.util.HashMap;
import java.util.Map;

/**
 * Created by Familiar_CD on 06/02/2017.
 */

public class RegisterPac extends StringRequest{
    private static final String REGISTER_PAC_URL =
"http://rehabili.esy.es/Registrar_Pacien.php";
    private Map<String, String> params;

    public RegisterPac(String nom, String apePa, String apeMa, String fecNac,
String ocu, String ori,
                       String reside, String motCon, Response.Listener<String>
listener){

        super(Request.Method.POST, REGISTER_PAC_URL, listener, null);
        params = new HashMap<>();
        params.put("nom", nom);
        params.put("apePa", apePa);
        params.put("apeMa", apeMa);
        params.put("fecNac", fecNac);
        params.put("ocu", ocu);
        params.put("ori", ori);
        params.put("reside", reside);
        params.put("motCon", motCon);
    }

    public Map<String, String> getParams() {

        return params;
    }
}
```

# Bibliografía


[1] URL: http://inaoep.mx/: Página principal del Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), en ella se puede consultar información acerca de las carreras y servicios que ofrece.

[2] Ian Sommerville. “Ingeniería del software”.Pearson Education, 2005.

[3] Beck, K. (1999). *Extreme Programming Explained. Embrace Change*: Pearson Education

[4] URL: https://developer.android.com/studio/intro/index.html?hl=es-419 Página donde se puede consultar manual de usuario de Android Studio

[5] Tomas, Jesus. El gran libro de Android Segunda Edicion. Alfaomega Grupo Editor