UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de una aplicación móvil para consultas sobre neoplasias usando la Clasificación Internacional de Enfermedades para Oncología (CIE-O) tercera edición

TESIS

Que para obtener el título de

Ingeniero en Computación

P R E S E N T A

Eloy Guillermo Vidal García

DIRECTOR DE TESIS

M.C. Alejandro Velázquez Mena

Ciudad Universitaria, Cd. Mx., 2018

Índice

Introducción 5

Capítulo I. Marco teórico 7
1.1 Tipos de aplicaciones 7
1.1.1 Stand alone 7
1.1.2 Cliente – servidor 7
1.2 Persistencia de datos 7
1.3 Diseño móvil 8
1.4 Layout 8
1.4.1 Layout lineal 8
1.4.2 Layout en rejilla 9
1.4.3 Layouts dinámicos.....9
1.5 Experiencia de Usuario: Concepto UX (User eXperience) 10
1.5.1 Aspectos a considerar en una experiencia de usuario .12
1.6 Metodologías de desarrollo de software 13
1.6.1 Metodologías ágiles 13
1.6.1.1 Scrum 14
1.6.1.2 Extreme Programming (XP) 15
1.6.1.3 Lean Programming 16
1.7 Sistemas operativos móviles 16
1.7.1 Android 18
1.7.2 iOS 18
1.7.3 Windows Phone 19

Capítulo II. Análisis 20
2.1 Definición del problema 20
2.2 Cuestiones a tomar en cuenta 20
2.3 Posibles soluciones 21
2.4 Propuesta de solución 22
2.5 Análisis de los requerimientos de software 22
2.6 ¿Qué es la Clasificación Internacional de Enfermedades para Oncología Tercera edición (CIE-O-3)? 23
2.6 Marco de desarrollo de software a utilizar 24
2.7 Preparación de datos 25
2.7.1 Captura de datos 26
2.8 Plataforma de desarrollo a emplear 29
2.9 Entorno de desarrollo a utilizar 30

Capítulo III. Diseño 31
3.1 Traducción de los requerimientos de software 31
3.1.1 Características (features) a incluir en la aplicación móvil 31
3.2 Primer esquema de la interfaz gráfica 33
3.3 Funcionamiento de la aplicación 35
3.4 Interacción del usuario con la aplicación 35
3.5 Funcionamiento interno de la aplicación 36
3.6 Product backlog para el análisis y desarrollo de los módulos empleando Scrum 38
3.6.1 Módulos de la aplicación 39
3.6.1.1 Base de datos portátil 39

3.6.1.2 Campo de búsqueda 40
3.6.1.3 Botón y algoritmo de búsqueda 41
3.6.1.4 Campo de resultados 42
3.6.1.5 Despliegue de resultados 44
3.6.1.6 Botón y función de limpieza de campos 44
3.6.1.7 Mensajes al usuario 44
3.6.1.8 Función de copiado de texto al portapapeles .45
3.7 Layout 45

Capítulo IV. Desarrollo 47
4.1 Ícono de la aplicación 47
4.2 Archivos XML 48
4.2.1 activity_main.xml 48
4.2.2 strings.xml 48
4.3 Componentes de la interfaz de usuario 49
4.3.1 Eventos asociados a los componentes 50
4.4 Responsividad en la interfaz gráfica 51
4.5 Main activity 52
4.5.1 Función "buscar()" 53
4.5.2 Función "limpiar()" 61
4.5.3 Función "copiaTextView()" 62
4.5.4 Función "muestra()" 63

Capítulo V. Prueba e instalación 65
5.1 Primera versión completa 65
5.2 Casos de prueba 66
5.2.1 Dispositivos y versiones de Android utilizadas 66
5.2.2 Pruebas de escritorio 67
5.2.2.1 Interfaz gráfica 67
5.2.2.2 Botones 68
5.2.2.3 Entrada del editText 69
5.2.2.4 Portapapeles 70
5.3 Correcciones 72
5.3.1 Texto perdido en el textView 72
5.3.2 Componentes fuera de pantalla 72
5.3.3 Entradas inválidas en editText 73
5.4 Manual de usuario 74
5.4.1 Requerimientos 74
5.4.2 Instalación 74
5.4.3 Modo de uso 75
5.5 Disponibilidad en Play Store 77

Capítulo VI. Resultados 78
6.1 Desempeño en los distintos dispositivos 78
6.1.1 Interfaz de usuario 78
6.1.2 Distinto hardware 79
6.2 Comodidad 80
6.3 Usabilidad 81
6.4 Aceptación de los usuarios finales 82

Conclusiones 84

Glosario **86**

Referencias **88**

Anexos **90**

Introducción

La tecnología en la computación ha avanzado de manera sorprendente desde sus orígenes hace ya varias décadas. Puede resultar difícil creer cómo aquellas enormes primeras computadoras del tamaño de habitaciones han sido reemplazadas a estos días por algo tan pequeño y liviano que cabe en el bolsillo. La portabilidad se ha convertido en una de las principales ventajas de la tecnología del cómputo. Tener una computadora a la mano en cualquier momento es algo que quizá nunca se imaginaron los científicos e ingenieros que vieron los primeros pasos de la computación.

En la actualidad, gracias a la disminución de los precios y a la necesidad en la que se han convertido, no es nada raro poseer al menos un smartphone (llamado también *teléfono inteligente* en español) u otro dispositivo móvil. Desde los jóvenes que los usan para visitar sus redes sociales o ver sus series favoritas en prácticamente cualquier parte, hasta los profesionistas que lo usan como una herramienta de comunicación más efectiva y, a la larga, más económica que las llamadas telefónicas convencionales.

En este trabajo primeramente se dará un marco teórico el cual permitirá identificar y definir algunos de los conceptos fundamentales o necesarios para poder analizar y dar solución al problema que se planteará en este escrito. Se definirán dichos conceptos y se justificará a lo largo de este mismo texto la mención de los mismos. Estos conceptos ayudarán además a poder justificar y elegir las opciones más viables para resolver el problema.

Concluidas dichas definiciones, se procederá a realizar un análisis del problema en cuestión. Es dicho apartado se mencionará cuál es la problemática que se desea resolver, además de que se incluirán las consideraciones e información que se crean necesarias o relevantes para tratar de resolver el problema a tratar. Teniendo esto, se analizarán soluciones potenciales, mencionando tanto ventajas que presenten como razones para ser descartadas. Todo este análisis se realizará empleando un enfoque de ingeniería de software, es decir, se incluirá un análisis de requerimientos de software para abordar el problema. Más aún, toda esta tarea se coordinará empleando alguna metodología o marco de desarrollo de software, esto con el fin de estructurar y coordinar el desarrollo de la misma.

Durante el diseño se interpretarán los requerimientos de software recolectados durante el análisis realizado, además de que se esbozará la solución que se planea desarrollar. Se analizarán ventajas y desventajas que proporciona cada una de las opciones de las que se dispone para el desarrollo, así como se justificará la elección de cada una de ellas. Se añadirán diagramas y bocetos que fungirán como apoyo visual para poder explicar la funcionalidad y propósito que

presenta cada uno de los componentes que conforman a la aplicación móvil.

En el capítulo titulado "Desarrollo" se procederá a transformar y adaptar los modelos creados durante el análisis y el diseño de software con el fin de obtener el código fuente que satisfaga las necesidades obtenidas del problema que se planteó. Se justificará la codificación realizada y se mencionará el porqué de cada elección cuando se presenten distintas opciones durante el desarrollo.

En el apartado de nombre "Prueba e Instalación" se describirán las verificaciones de software que fueron realizadas una vez concluido el desarrollo de la aplicación en cuestión. Se explicará su justificación, se mencionará cada uno de los problemas encontrados en tal aplicación y se indicarán las correcciones o modificaciones que se aplicaron para resolverlos. Adicionalmente, se mostrará el modo de instalar la aplicación móvil anexando un sencillo pero completo manual de usuario destinado a los usuarios finales de la app.

Finalmente, se analizarán los resultados obtenidos de la elaboración de este trabajo. Se analizarán cada uno de los aspectos relevantes de la aplicación móvil desarrollada en cuanto al desempeño de la misma. Por último, se mencionará y examinará la aceptación que los usuarios finales tuvieron después de probar la versión final de dicho software.

Capítulo I. Marco Teórico

En este capítulo se proporcionará parte de la teoría necesaria para abordar el problema a tratar en este escrito. Se incluirán algunas definiciones y temas pertinentes, además, se hablará sobre varios aspectos que serán considerados durante los capítulos posteriores con la finalidad de comprender mejor la problemática a tratar.

1.1 Tipos de aplicaciones

De acuerdo al tipo de autonomía que presente una aplicación para su funcionamiento, podemos dividirlas en dos principales arquitecturas: stand alone (lo que puede traducirse al español como "autónomo") y la llamada cliente-servidor.

1.1.1 Stand alone

La arquitectura stand alone hace referencia a toda aquella tecnología que no requiere conectarse a una red para funcionar, es decir, es completamente autónomo. Es el mismo sistema el que gestiona su funcionamiento sin requerir recursos de terceros.

La principal ventaja de esta arquitectura, para el caso de dispositivos móviles, recae en no necesitar una conexión a otro dispositivo para operar. Se puede disponer de sus servicios de manera más fácil sin la dependencia que presenta requerir de una conexión a red, por ejemplo.

1.1.2 Cliente - servidor

La arquitectura cliente - servidor se refiere a un modelo de comunicación entre dispositivos que comparten tareas o recursos en una red. El funcionamiento consiste en un usuario (el cliente) el cual obtiene servicios de una máquina remota que provee el servicio (el servidor). El servidor en este caso proporciona un puerto de comunicación a través del cual se conectan todos los clientes que obtienen servicios de él. En este modelo se establece un socket en el cliente y otro en el servidor y se comunican entre sí por el puerto que se proporciona.

1.2 Persistencia de datos

Existen ocasiones en la que nos interesa conservar cierta información o datos una vez terminada la ejecución de algún

programa, esto con el fin de poder visualizarlos después, hacer uso de ellos por parte del programa mismo en ejecuciones posteriores, o bien como algún tipo de respaldo. La persistencia de datos consiste en la conservación de los mismos y que no sean perdidos si es requerido preservarlos. Como ejemplos de esto se tienen el almacenamiento en archivos y las bases de datos.

1.3 Diseño móvil

La interfaz de una aplicación es una de las partes más importantes de la misma. No sólo representa un atractivo visual o el lado llamativo del programa, sino que también consiste en la principal forma de interacción entre el usuario y la aplicación. Los botones, los íconos, los gráficos, los mensajes y los fondos representan la parte visual de la aplicación.

La tarea del diseño móvil no solamente consiste en darle a la interfaz un estilo y diseño vistoso, sino también requiere hacerlas fáciles de usar, intuitivas, coherentes, originales y amigables para los usuarios. Ya no basta nada más hacer aplicaciones eficientes y rápidos para realizar las tareas del usuario. Estas razones son el porqué de la importancia que tiene el diseño móvil hoy en día.

1.4 Layout

El término layout, lo cual puede traducirse en este contexto al español como “disposición” u “organización” (traducido por la página oficial de Android y muchos otros sitios como “diseño”), hace referencia a la disposición de los elementos visuales en una interfaz de usuario. El layout puede verse como un lienzo en donde se dispondrán los elementos a mostrar en la pantalla. Este es un aspecto de suma importancia para la comodidad y el atractivo visual en una página web, en un programa o una aplicación móvil.

A continuación, se mencionan algunos de los principales layouts empleados en el diseño de aplicaciones móviles.

1.4.1 Layout lineal

Conocido comúnmente como “linear layout”, se trata de un diseño o disposición sencillo de elementos en la interfaz de usuario en el cual todos los elementos se organizan uno tras otro en una sola dirección, ya sea de forma vertical u horizontal.

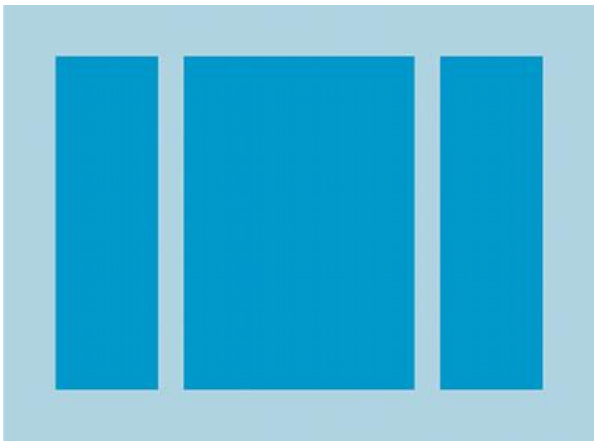

Figura 1.4.1. Ejemplo de un layout lineal horizontal.

1.4.2 Layout en rejilla

Más popularmente conocido por su término en inglés como "grid layout" o simplemente "grid", es otro tipo de acomodo de elementos en una interfaz de usuarios en el cual los elementos se ordenan en un arreglo bidimensional, es decir, se disponen como si estuviesen en una cuadricula.

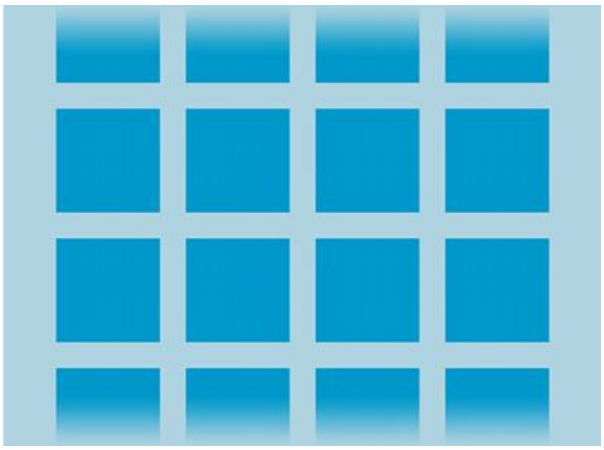

Figura 1.4.2. Ejemplo de un grid layout.

1.4.3 Layouts dinámicos

Los layouts de este tipo permiten organizar objetos en una vista de una aplicación móvil de manera más flexible y de manera dinámica. Permiten dimensionar y colocar widgets estableciendo relaciones entre los mismos y la vista que contiene al mismo layout.

Como ventaja más notable, se destaca el hecho de permitir realizar interfaces responsivas de manera más cómoda y sencilla para el programador, lo cual beneficia tanto al desarrollador como al usuario. Abajo se muestran algunos ejemplos de diseños adaptables "en cadena", los cuales son posibles con los layouts responsivos.

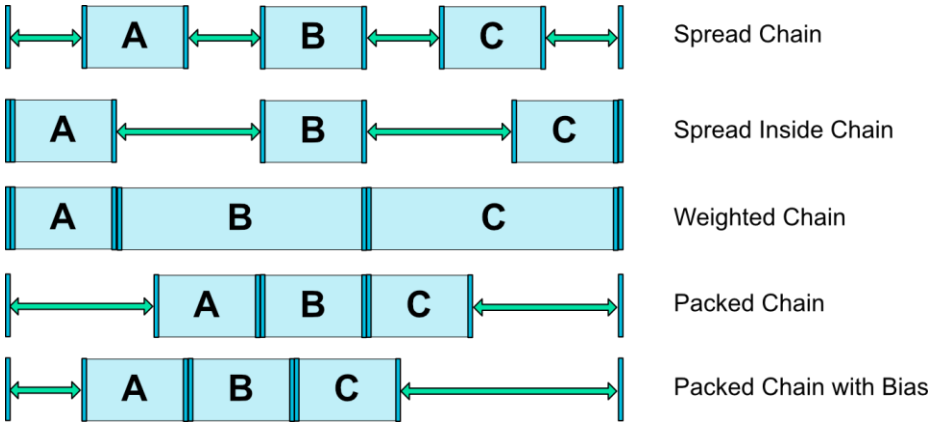

Figura 1.4.3. Ejemplos de diseño Chain Style.

1.5 Experiencia de Usuario: Concepto UX (User eXperience)

El concepto de experiencia de usuario (abreviado y popularizado muchas veces como "UX") consiste en todos los elementos (percepciones y respuestas) derivados de la interacción de un usuario con un producto, sistema o servicio. La experiencia de usuario incluye sus emociones, preferencias, percepciones, respuestas psicológicas, comportamientos e incluso sentimientos; presentados durante y después de usar dicho producto.

Este concepto toma cierta importancia en el desarrollo de software para el diseño de interfaces, tanto para desarrollo web como desarrollo móvil. Resulta importante tomar en cuenta la experiencia de usuario para poder desarrollar interfaces agradables, amigables y eficientes para quien vaya a hacer uso nuestro software.

A diferencia de lo que se pueda dar a entender en algunos sitios web que tratan de hablar sobre el tema, la experiencia de usuario no se ve limitada al diseño de la interfaz gráfica o sólo al estudio del usuario como cliente, sus requerimientos y sus necesidades; sino que estas últimas son solamente algunas de las actividades que forman parte del proceso que conlleva a desarrollar y mejorar la experiencia de usuario.

Podemos mencionar algunas de las tareas más importantes (más no las únicas) que se llevan a cabo para proporcionar una experiencia de usuario adecuada:

- Entender qué es lo que debe realizar el producto que vamos a desarrollar.
- Estudiar el entorno de los usuarios finales del producto.
- Satisfacer los requerimientos de software por completo.
- Definir a nuestro usuario final (entender a quién va dirigido lo que se está desarrollando).
- Diseñar interfaces intuitivas y cómodas para el usuario.
- Estudiar, comprender y evaluar cuál es la experiencia del usuario con el software o producto que se le presenta.

A continuación, se presenta un diagrama que resume un poco los elementos presentes en la Experiencia de Usuario. Cabe resaltar que se hace diferencia entre la interfaz de software y un sistema de hipertexto.

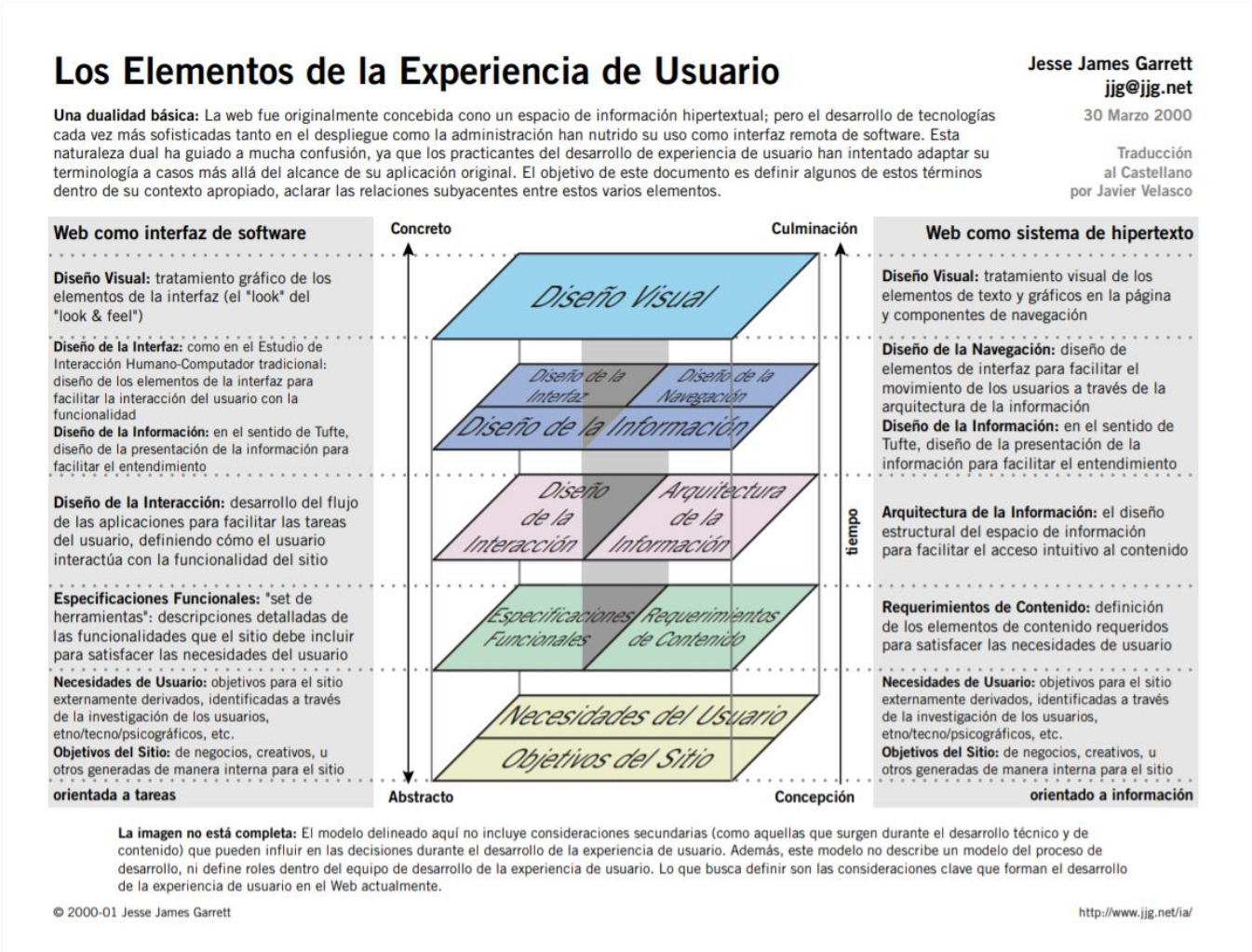

Figura 1.5. Elementos de la User Experience (UX) orientados a la web.

1.5.1 Aspectos a considerar en una experiencia de usuario

Como puede apreciarse en la columna izquierda de la imagen anterior, son varios los aspectos a considerar en una interfaz de software para obtener una buena experiencia de usuario. Cada aspecto mencionado deberá analizarse y adecuarse a las necesidades del usuario y optimizar su experiencia lo mejor posible.

Muchas veces esta adecuación a los usuarios no resulta tan fácil como puede sonar. Una de las principales causas de esta dificultad recae en la medición de algunos aspectos del diseño de interfaces o páginas web. ¿Cómo medir qué tan intuitiva es una interfaz? ¿Cómo medir la facilidad con que se puede entender nuestro software? Parecen preguntas difíciles de responder, sin embargo, existen libros dedicados a medir (o al menos tratar) estos aspectos para

poder comprender mejor cómo crear la mejor experiencia de usuario posible.

A estos aspectos se le pueden sumar otras dificultades como el desconocimiento o malinterpretación de los requisitos de software o el desconocimiento del usuario final en sí, pues resultará difícil desarrollar software o sitios de internet adecuados si no se sabe qué es lo que se va a desarrollar o hacia quién o quiénes van dirigidos los mismos. Por esta razón, resulta muy importante considerar esto durante el análisis de requerimientos de software.

1.6 Metodologías de desarrollo de software

Se puede definir a las metodologías de desarrollo de software (también conocidas como marcos de desarrollo de software) como paradigmas de planeación, organización y control para la elaboración de proyectos de software de manera que se facilite, se estructure y se optimice el proceso de desarrollo de software durante un proyecto. Estas metodologías traen consigo varias ventajas para tanto desarrolladores y diseñadores, como para líderes de proyecto, entre las cuales destacan:

- ✓ Facilitar la planeación del proyecto
- ✓ Hacer eficiente el tiempo de trabajo del que se dispone
- ✓ Obtener un mejor control y organización en el proyecto
- ✓ Facilitar el encontrar y luego solucionar errores durante el proceso de desarrollo

1.6.1 Metodologías ágiles

Las metodologías ágiles se definen como aquellas metodologías de desarrollo enfocadas a disminuir la documentación, concentrarse en la funcionalidad del software y ofrecer una mayor flexibilidad a los cambios en los requerimientos del proyecto junto con una mejor adaptación a la necesidad del mercado de desarrollar y validar versiones de software en periodos limitados. Todas estas metodologías dinámicas representan una opción viable para satisfacer los requerimientos de software en la actualidad, los programadores actuales, y también las tecnologías, las cuales resultan tan cambiantes de hoy en día. Entre las principales ventajas de las metodologías ágiles se tienen:

- ✓ Permitir la entrega de avances de proyecto al trabajar en módulos independientes sin esperar a que se terminen otros de

los que no depende

- ✓ Facilitar la aplicación de cambios en los requerimientos de software durante la marcha del proyecto
- ✓ Ahorrar tiempo de proyecto al no retrasar un módulo mientras se termina otro
- ✓ Mejorar la eficiencia del equipo de trabajo

A continuación, se mencionan las principales metodologías de desarrollo ágil usadas en la actualidad.

1.6.1.1 Scrum

La metodología scrum está enfocada en separar tareas y simplificar proyectos complejos, además de promover un trabajo más colaborativo entre los miembros del proyecto. Otro de sus puntos a favor es el constante seguimiento de los avances que lleva a cabo, logrando un mejor control y supervisión del proyecto. Los pasos claves del scrum se resumen en:

- ➢ Crear una pila de lo que se desea desarrollar, llamada product backlog, en la cual los elementos con mayor prioridad van hasta arriba con el fin de ser realizadas primero. Esta lista es elaborada por el product owner, el cual equivale al líder de proyecto, quien se encarga de dirigir y visualizar que el producto satisfaga los requerimientos y expectativas del cliente.
- ➢ Se eligen algunos elementos del tope de la lista para decidir plazos y procedimientos a utilizar. Este paso se conoce como sprint plannig.
- ➢ Se proporciona un periodo de entre una y cuatro semanas para que el equipo pueda completar las tareas seleccionadas.
- ➢ Se hacen reuniones diarias para mostrar progresos. A esto se le llama daily scrum. Aquí se requiere la participación de un líder llamado scrum master, el cual además de liderar las reuniones y mantener enfocado al grupo en los objetivos, facilitará y ayudará con los problemas que hayan surgido durante el sprint.

- Al finalizar cada sprint, se genera un entregable para el cliente, el cual ya puede ser mostrado como un avance.

- Se genera un sprint review en el cual se resume lo que se ha obtenido durante el sprint.

- Se realiza una retrospectiva para analizar objetivos cumplidos, analizar errores, y encontrar mejoras que pueda haber. A este paso se le llama sprint retrospective.

- Se escogen nuevas actividades del tope de la lista para trabajar y se comienza con el siguiente sprint.

Fig 1.6.1.1. Diagrama de los procedimientos realizados en scrum.

1.6.1.2 Extreme Programming (XP)

La extreme programming (en español "programación extrema"), conocida simplemente como metodología XP, consiste en una metodología orientada a proyectos con pocos programadores, a proyectos pequeños y orientados a un solo cliente. Es ideal para proyectos en los que se pueden llevar a cabo muy pocos procedimientos en paralelo.

Una de sus principales ventajas recae en la gran facilidad para adaptarse que presenta ante cambios de requerimientos u otros

imprevistos. Por otro lado, presenta como desventajas la dificultad para realizar documentación y la poca planeación para largo plazo que presenta.

Este paradigma consiste principalmente en:

- ✓ Realizar el diseño, desarrollo e implementación lo más rápido posible
- ✓ Saltar la documentación, si es necesario
- ✓ Modificar los requisitos a lo largo del proyecto conforme estos vayan cambiando sin mayores dificultades
- ✓ Adaptarse a los requerimientos que vayan surgiendo durante el proyecto conforme se desarrolla el mismo
- ✓ Una buena comunicación entre los miembros del equipo, así como del equipo con el cliente

1.6.1.3 Lean Programming

Conocido en ocasiones en español como "desarrollo ligero" o simplemente "lean", es una metodología de desarrollo centrada en conseguir exactamente lo que el cliente requiere. Para lograr esto, se opta por una optimización de los procedimientos y en evitar acciones que se consideran innecesarias o poco útiles durante el desarrollo, optando por utilizar prácticas más eficientes.

Más que procesos, se tienen algunos principios para el lean programming, entre los que destacan:

- ✓ Eliminar desperdicios (lo cual incluye procesos extra, características extras, defectos, entre otros)
- ✓ Decidir tarde, es decir, tomar algunas decisiones hasta que se tiene claros algunos requerimientos y evitar cambios repentinos en los mismos.
- ✓ Entregar tan rápido como sea posible
- ✓ Capacitar al equipo
- ✓ Ver todo como un conjunto

1.7 Sistemas operativos móviles

El sistema operativo móvil es el software encargado de gestionar y administrar los recursos del dispositivo móvil, además de proporcionar una interfaz más accesible para facilitar la interacción entre el usuario y el dispositivo. Podemos decir que el sistema operativo se encarga de facilitar para el usuario el manejo de cierto dispositivo y sus componentes (tanto software como hardware) de manera eficiente, eficaz y segura.

El hecho de enfocarse específicamente en los sistemas operativos móviles se basa en la cantidad de usuarios de dispositivos móviles que hay en la actualidad, haciendo hincapié en cómo estos han superado en cantidad a los usuarios de dispositivos de escritorio (también llamado "desktop"). A continuación, se muestra un gráfico comparativo de los usuarios de equipos de escritorio (desktop) contra los usuarios de dispositivos móviles (mobile) a nivel mundial.

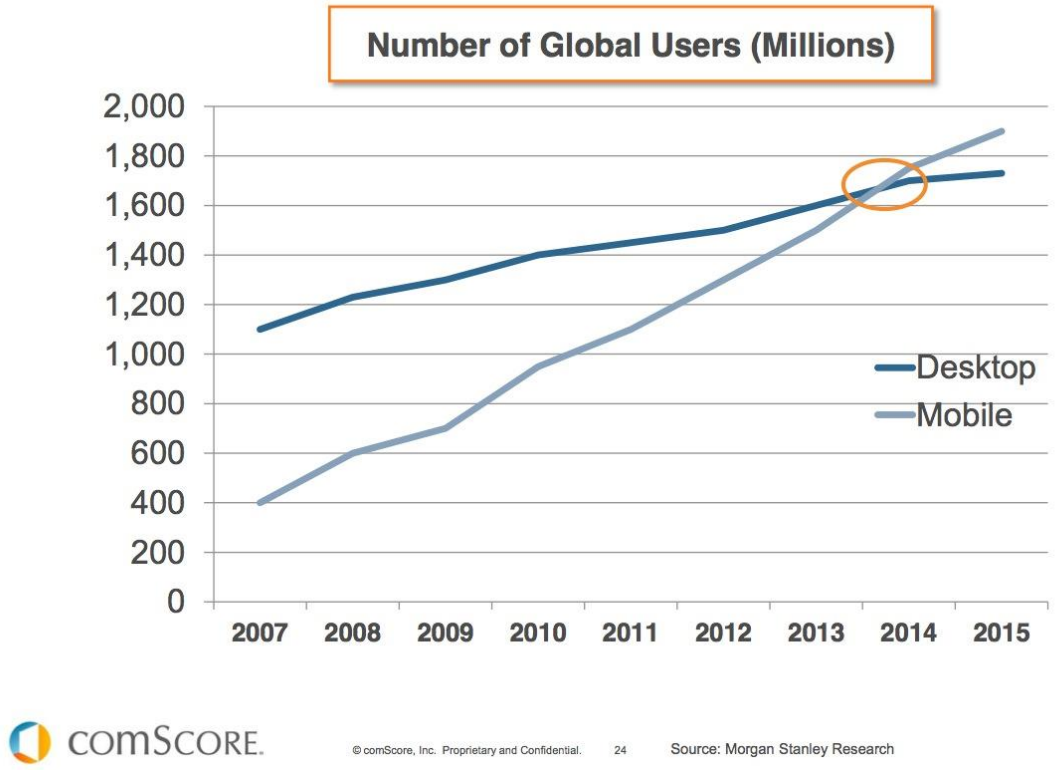

Figura 1.7. Comparativo de usuarios de escritorio contra usuarios móviles.

Dicho esto, se hará mención de los principales sistemas operativos para dispositivos móviles que existen en el mercado en la actualidad.

1.7.1 Android

Uno de los sistemas operativos móviles más populares en la actualidad. Fue desarrollado originalmente por Android Inc. (empresa que fue después adquirida por Google) y presentado en 2007. Está basado en el núcleo de Linux y es de código abierto.

Para el desarrollo de aplicaciones bajo esta plataforma se emplea el lenguaje de programación Java, empleando comúnmente el IDE (Entorno de Desarrollo Integrado) Android Studio, además del Android SDK (Software Development Kit), estos dos últimos provistos de manera gratuita por Google desde el sitio web de Android Studio. Todas estas facilidades hacen de Android una de las mejores opciones para el desarrollo de aplicaciones móviles respecto a otros sistemas operativos móviles.

Abajo se puede observar una comparación de los sistemas operativos más populares a nivel mundial. Se aprecia el alcance del sistema operativo móvil Android, muy similar al que tienen los sistemas operativos de Microsoft, Windows.

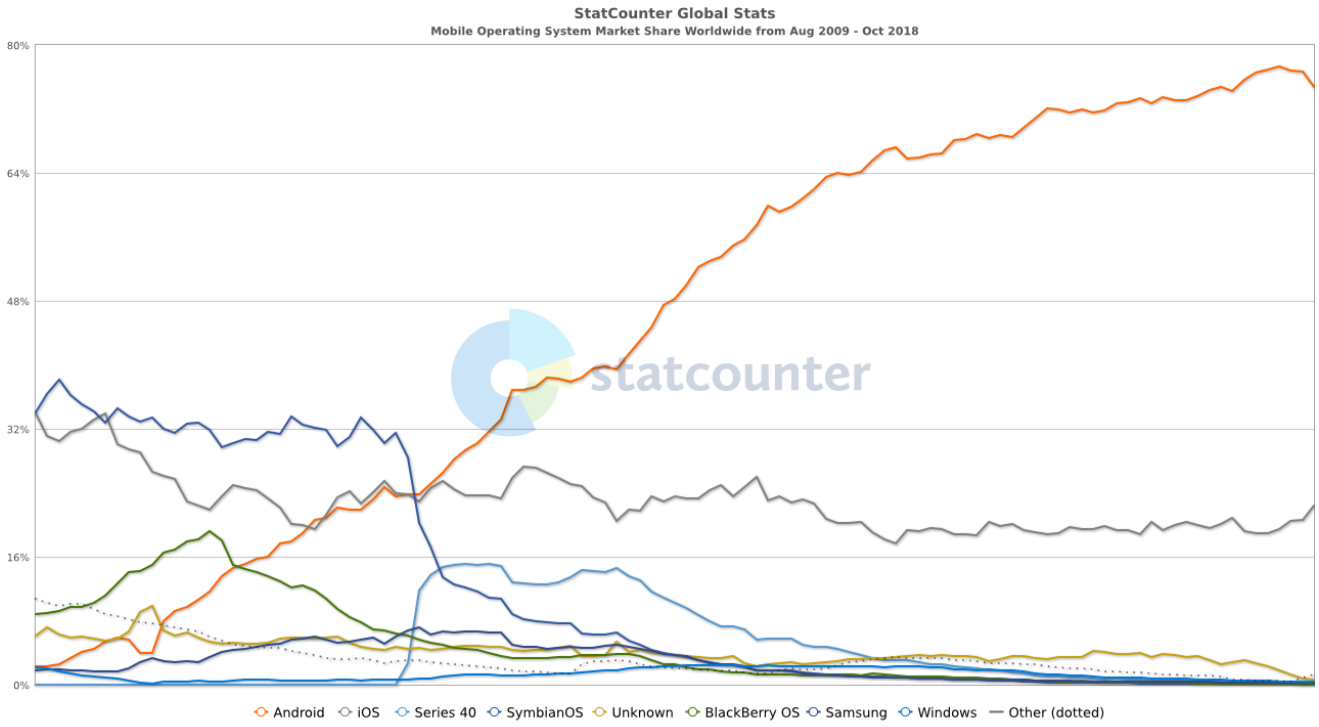

Figura 1.7.1. Comparativo de los sistemas operativos más usados.

1.7.2 iOS

El sistema operativo desarrollado por Apple Inc. en 2007 para su línea de productos móviles (iPad, iPhone, iPod Touch), por tal razón, este sistema operativo no está disponible para dispositivos ajenos a la marca.

La principal desventaja del desarrollo en iOS consiste en el costo que implica esto. Primeramente, es preciso contar con una Mac para poder usar las herramientas de desarrollo proporcionadas por Apple. Además, se debe estar registrado en el Apple Developer Program, cuyo costo varía según la versión de desarrollo que se requiera.

1.7.3 Windows Phone

Éste es el sistema operativo móvil desarrollado por Microsoft para dispositivos móviles lanzado en 2010. A pesar de la buena crítica que recibió por su originalidad y diseño, se ha visto desplazada a lo largo de su historia por los otros dos sistemas operativos ya mencionados.

Actualmente ya no recibe soporte por parte de Microsoft para enfocarse y ser reemplazado por Windows 10 Mobile y apoyar la compatibilidad con el sistema operativo Windows 10.

Capítulo II. Análisis

A continuación, se procederá a hacer un estudio del problema que se aborda en este trabajo. Se definirá el problema planteado y se procederá a proporcionar una solución al mismo.

Posteriormente, se analizará dicha solución propuesta. Se examinará su viabilidad, su ejecución y los requerimientos para llevar a cabo la misma.

2.1 Definición del problema

La consulta y búsqueda de información en libros físicos presenta a veces una tarea tediosa, tardada e incluso molesta, sobre todo cuando se trata buscar en libros bastante voluminosos o que poseen varios tomos. Este tipo de búsqueda tradicional resulta en una gran inversión de tiempo y esfuerzo para los usuarios. Más aún, el manejo, cuidado, almacenamiento y traslado de dichos ejemplares físicos resultan tareas extra para esta situación.

Para el caso del libro Clasificación Internacional de Enfermedades para Oncología, en su tercera edición, se tiene un archivo digital de 246 páginas, de las cuales los códigos representan la gran mayoría de las páginas en el libro. Dicha situación trae como consecuencia que la consulta de los códigos para enfermedades o partes del cuerpo humano sea una de las actividades menos deseables que se puedan realizar.

Se requiere una solución que resuelva o bien, facilite todas estas consideraciones de una manera eficiente, presentando ninguna o la menor cantidad de desventajas posibles respecto al problema a resolver, además de hacerlo sin alterar los resultados de dichas actividades.

2.2 Cuestiones a tomar en cuenta

Estos son algunos aspectos a tomar en cuenta para poder dar solución al problema en cuestión:

- ✓ Los códigos mencionados deben estar relacionados de manera fiel y sin alteración alguna a su descripción, tal y como aparecen en el libro.

- ✓ Obtener un ahorro de tiempo de la propuesta de solución debe ser considerablemente mayor al que se tendría en una búsqueda

tradicional en un libro físico.

- ✓ Se debe proporcionar una solución portable.
- ✓ Se debe proporcionar una solución intuitiva y fácil de usar por quienes consultan dichos libros.

2.3 Posibles soluciones

A continuación, se presentarán algunas propuestas tentativas las cuales fueron tomadas en cuenta para dar poder dar solución al problema planteado, indicando a su vez tanto las ventajas como las desventajas que presenta cada una de ellas.

1. Diseñar y desarrollar un programa de computadora que permita buscar y consultar los códigos de la Clasificación Internacional de Enfermedades para Oncología.

   El realizar la búsqueda de información en dicho programa presenta un gran ahorro de tiempo para los usuarios. Lamentablemente, esta alternativa no presenta una solución al problema de la portabilidad, siendo obligatorio disponer de un dispositivo no más pequeño que una laptop, la cual no es tan portátil como otros dispositivos que hay en la actualidad.

2. Diseñar y desarrollar una aplicación para dispositivos móviles que permita visualizar las páginas que contienen los códigos de la Clasificación Internacional de Enfermedades para Oncología.

   Aunque esta posible solución resuelve el problema que se planteó sobre la portabilidad, se mantiene el problema del tiempo que se invierte en las consultas que se realizan en los libros físicos, es decir, resulta casi la misma situación que realizar la consulta de información buscando página por página y luego código por código, el cual es uno de los problemas que se deben resolver.

Por los motivos ya mencionados, lamentablemente ambas propuestas se deben descartar como soluciones, al no resultar del todo viables y/o representar solamente una solución parcial al problema.

2.4 Propuesta de solución

Elaborar una aplicación para dispositivos móviles que pueda realizar búsquedas mediante palabras clave de los códigos y descripciones contenidos en el libro Clasificación Internacional de Enfermedades para Oncología, en su tercera edición. La aplicación mostrará los resultados de búsqueda en pantalla para poder ser empleados por el usuario.

Esta propuesta es la que presenta una solución óptima, viable y que satisface por completo todas las consideraciones presentadas anteriormente. Por estas razones, esta propuesta será la elegida para dar solución al problema que se ha planteado.

2.5 Análisis de los requerimientos de software

Escogida una solución al problema mencionado, se procederá a realizar el análisis de los requerimientos de software que permita analizar detalladamente qué es lo que se requiere diseñar, desarrollar e implementar. Estos son las características que se precisan:

- Se requiere diseñar y desarrollar una aplicación para dispositivos móviles la cual pueda acceder a una base de datos que contenga los códigos del estándar de Clasificación CIE (Clasificación Internacional de Enfermedades) para enfermedades oncológicas.

- Se requiere permitir buscar entre tales códigos mediante palabras claves de la descripción del padecimiento o bien, mediante el código asociado a dicho para facilitar la consulta de la información necesaria para el usuario.

- La aplicación deberá mostrar los resultados de la búsqueda de acuerdo a las coincidencias encontradas, resaltando además las coincidencias encontradas en la búsqueda.

- La aplicación debe ser autónoma y funcionar sin conexión o recursos externos (emplear tecnología stand-alone).

- Se requiere elaborar la base de datos que almacene tanto los códigos como sus respectivas descripciones, las cuales están contenidas en el libro Clasificación Internacional de Enfermedades para Oncología Tercera edición (CIE-O-3). Se debe respetar el contenido original como tal cual aparece en el libro, sin ninguna modificación tanto en los códigos como en las descripciones que les correspondan.

2.6 ¿Qué es la Clasificación Internacional de Enfermedades para Oncología Tercera edición (CIE-O-3)?

La Clasificación Internacional de Enfermedades para Oncología (CIE-O) es un estándar creado en 1976 el cual es empleado para registrar, clasificar y codificar distintas neoplasias (formaciones anormales del tipo tumoral) que se conocen, tanto su ubicación en el cuerpo, así como sus características y descripción.

Este estándar recibe actualizaciones menores cada año y actualizaciones mayores cada tres años. En la actualidad este estándar se encuentra en su tercera versión publicada en el año 2000 y su versión oficial en español en el año 2003.

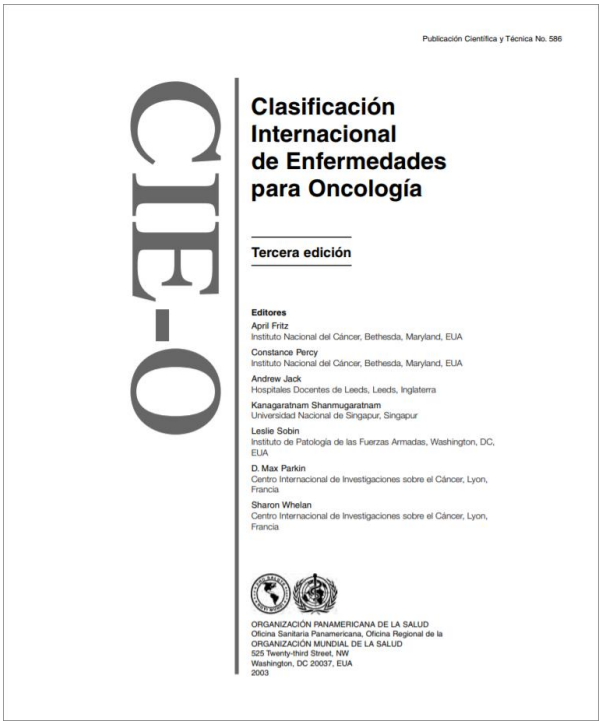

Figura 2.6 Libro de la Clasificación Internacional de Enfermedades para Oncología, Tercera edición (CIE-O-3).

Sabiendo que las actualizaciones para este caso son mínimas y los cambios grandes o considerables tardan varios años, se ha optado por escoger una manipulación de datos manual (captura y organización de los mismos). Esto presenta como ventajas:

- ✓ Verificar que al momento de realizar la captura y ordenamiento de datos no se presenten errores, ya que, como se mencionó anteriormente, es necesario que los datos sean

fieles y correctos a como aparecen en el libro.

- ✓ No se requerirá desarrollar software adicional para poder realizar esta tarea de forma automatizada, lo cual implicaría una inversión de tiempo o de dinero extra (en caso de que se tuviese que pagar a un tercero por desarrollar software especializado para dicha tarea).

- ✓ El desarrollo de la aplicación móvil en sí se muestra como una opción más viable, al no requerir hacer cambios considerables en la misma para su funcionamiento y su actualización de datos, es decir, los cambios necesarios serán llevados a cabo sobre los códigos tomados del libro y no sobre el código fuente de la aplicación a elaborar.

2.6 Marco de desarrollo de software a utilizar

Con el fin de dar estructura, dar control y poder planear el desarrollo de la aplicación a realizar, se ha optado por emplear algún marco de desarrollo de software. Esto resultará una herramienta útil que ayudará a lo largo del proyecto para poder organizar y controlar las actividades necesarias, conforme al tiempo y recursos de los que se dispone.

Para ser capaz de elegir una de las metodologías existentes en la actualidad para el desarrollo de software, que pueda resultar adecuada y viable para las necesidades de este proyecto específico, se han tomado en cuenta varios aspectos importantes relacionados a la aplicación en sí y a los recursos disponibles, los cuales serán analizados uno por uno a continuación:

- Tamaño del proyecto: para este caso, se considera al proyecto relativamente pequeño, dados los pocos requerimientos de software que fueron obtenidos durante el análisis de requerimientos.

- Complejidad del proyecto: de nuevo, con base a los requerimientos de software obtenidos y tomando en cuenta los conocimientos de los que dispone el desarrollador, se considera que el proyecto de moderada complejidad.

- Duración del proyecto: dado el tiempo que se proporcionó para terminar la primera entrega funcional (que se entienda por "funcional" que la aplicación satisfaga los requerimientos mínimos para ser operacional), se considerará a éste como un proyecto de corta duración.

- Personal para realizar el proyecto: siendo esta aplicación desarrollada por una sola persona a nivel diseño, desarrollo y pruebas, el equipo de desarrollo será tomado en cuenta como un equipo de desarrollo pequeño o individual, siendo más estrictos.

- Posibles cambios en los requerimientos de software: previendo algunas posibles ocurrencias, correcciones o ajustes en las necesidades de la aplicación móvil que puedan surgir, se tomarán en cuentan también cambios en los requisitos de software.

Dichas todas estas cuestiones a tomar en cuenta, se ha optado por escoger el marco de desarrollo ágil similar Scrum. Una metodología adaptada tipo Scrum resulta uno de las metodologías más adecuadas para proyectos pequeños y que requieren traslapar tareas en varios puntos del desarrollo. Esta metodología traerá como beneficios a este proyecto:

- ✓ Se presentará una mejor organización de las tareas pendientes en el desarrollo.

- ✓ Se adapta bien para controlar y administrar un proyecto con pocos módulos como éste.

- ✓ Presenta flexibilidad ante los posibles cambios de requerimientos que puedan surgir durante el desarrollo.

- ✓ Al ser una metodología ágil, es recomendado para un proyecto con poca duración.

- ✓ Es ideal para proyectos de software con pocos miembros (una sola persona en este caso).

2.7 Preparación de datos

Para poder generar una base de datos con los registros del libro (para este caso, los que se necesitarán para poder realizar las tareas de la aplicación), es necesario realizar una selección de aquellos datos útiles, almacenarlos y organizarlos para poder disponer de ellos de una manera más eficiente y cómoda para poder ser consultados desde la aplicación durante su ejecución. El hecho de limitarse a almacenar solamente los códigos del libro ayudará a ahorrar espacio en la aplicación, evitando añadir texto que no se requerirá (para la finalidad de la aplicación móvil que se va a desarrollar), índices y tablas que no serán empleadas en la

aplicación, además de ahorrar tiempo de búsqueda durante la ejecución de la misma (aunado al desarrollo de un algoritmo de búsqueda eficiente durante la fase de programación).

En la siguiente imagen se aprecia un ejemplo de cómo aparecen los códigos y sus descripciones en el archivo pdf original que fue proporcionado. En esta captura de pantalla se puede apreciar la correspondencia requerida en los datos (código/descripción del código) de izquierda a derecha, respectivamente. Con esto se puede visualizar cómodamente dicha información.

Morfología - Lista tabular (continuación)

8280/0	**Adenoma acidófilo (C75.1)** Adenoma eosinofílico (C75.1)
8280/3	**Carcinoma acidófilo (C75.1)** Adenocarcinoma acidófilo (C75.1) Adenocarcinoma eosinofílico (C75.1) Carcinoma eosinofílico (C75.1)
8281/0	**Adenoma acidófilo-basófilo mixto (C75.1)**
8281/3	**Carcinoma basófilo-acidófilo mixto (C75.1)**
8290/0	**Adenoma oxifílico** Adenoma oncocítico Oncocitoma Adenoma de células de Hurthle (C73.9) Adenoma folicular, células oxifílicas (C73.9) Tumor de células de Hurthle (C73.9)
8290/3	**Adenocarcinoma oxifílico** Adenocarcinoma oncocítico Carcinoma oncocítico Adenocarcinoma de células de Hurthle (C73.9) Carcinoma de células de Hurthle (C73.9) Carcinoma folicular, células oxifílicas (C73.9)
8300/0	**Adenoma basófilo (C75.1)** Adenoma de células mucoides (C75.1)
8300/3	**Carcinoma basófilo (C75.1)** Adenocarcinoma basófilo (C75.1) Adenocarcinoma de células mucoides (C75.1)
8310/0	**Adenoma de células claras**
8310/3	**Adenocarcinoma de células claras, SAI** Carcinoma de células claras Adenocarcinoma de células claras, mesonefroide
8313/3	**Adenocarcinofibroma de células claras (C56.9)** Cistadenocarcinofibroma de células claras (C56.9)
8314/3	**Carcinoma rico en lípidos (C50._)**
8315/3	**Carcinoma rico en glucógeno**
8316/3	**Carcinoma de células renales asociado a un quiste (C64.9)**
8317/3	**Carcinoma de células renales, tipo cromófobo (C64.9)** Carcinoma cromófobo de células renales (C64.9)
8318/3	**Carcinoma de células renales, sarcomatoide (C64.9)** Carcinoma de células renales, células fusiformes (C64.9)
8319/3	**Carcinoma del conducto colector (C64.9)** Carcinoma del conducto de Bellini (C64.9) Carcinoma renal, tipo conducto colector (C64.9)
8320/3	**Carcinoma de células granulares** Adenocarcinoma de células granulares
8321/0	**Adenoma de células principales (C75.0)**
8322/0	**Adenoma de células claroacuosas (C75.0)**
8322/3	**Adenocarcinoma de células claroacuosas (C75.0)** Carcinoma de células claroacuosas (C75.0)
8323/0	**Adenoma de células mixtas**

Figura 2.7 Captura de pantalla del archivo pdf.

2.7.1 Captura de datos

Como se mencionó previamente, se ha optado por realizar una captura manual de los códigos de descripción y ubicación para crear la base de datos portátil. Para este caso es preciso disponer de un formato de archivo que permita copiar el texto para después ser trasladado

a otro archivo, esto con el fin de satisfacer el requerimiento de mantener íntegros los códigos con sus respectivas descripciones (sin errores ortográficos, palabras faltantes o sobrantes, o alguna incongruencia entre cierto código y la descripción correspondiente, por mencionar algunos ejemplos). El archivo generado podrá ser de utilidad después para quien decida hacer uso de los datos almacenados con un fin similar o distinto.

Se ha decidido hacer una conversión de formato del archivo original que se disponía en formato PDF (Portable Document Format), a un archivo de texto en formato DOC (el cual es además compatible con los sistemas operativos más populares). Este formato de archivo, a diferencia del formato PDF, permite una mejor manipulación de texto que el formato PDF; más aún, permitirá visualizar correctamente la relación código-descripción que se necesita mantener en los datos a almacenar en la base de datos.

Los códigos serán copiados columna por columna (o línea por línea si es necesario o el archivo no permite copiar más líneas) a una hoja de cálculo de un libro de Excel para ser almacenados, corregidos (de ser requerido), ordenados y organizados adecuadamente y poder generar posteriormente la base de datos de la aplicación. En la imagen de abajo se aprecia un poco del trabajo realizado para almacenar códigos y descripciones en una hoja de cálculo de Excel de la paquetería Office de Windows.

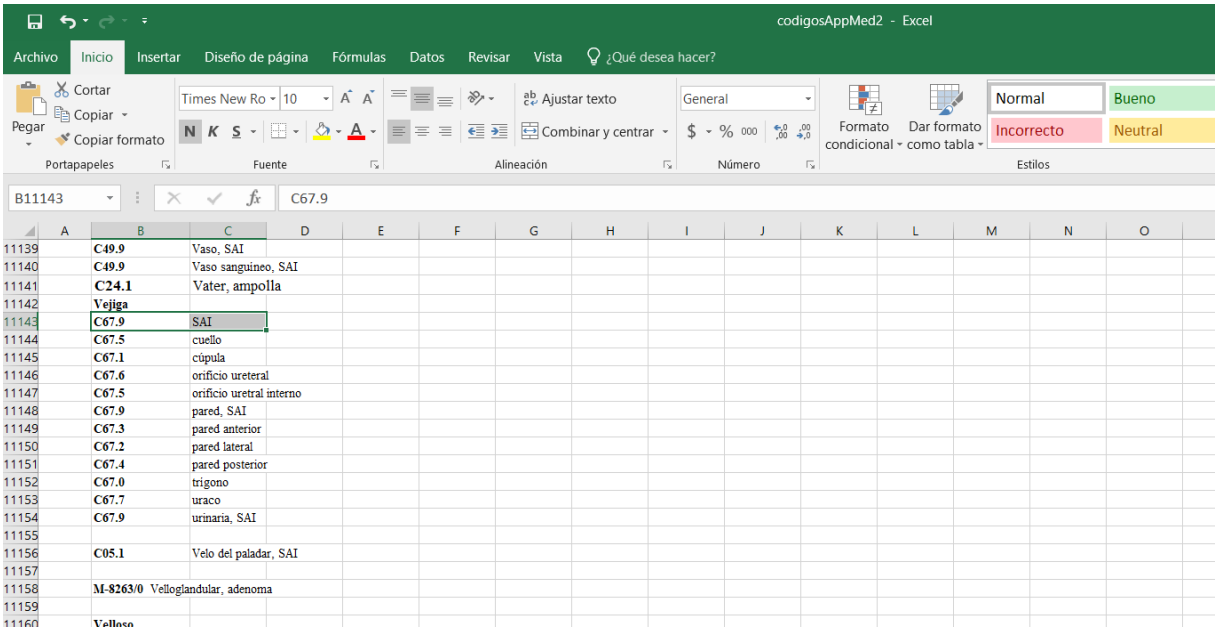

Figura 2.7.1. Códigos y descripciones siendo capturadas.

Una vez concluida la captura de datos, se ha decidido guardar el archivo en un formato CSV (Comma-Separated Values), el cual es un formato de archivo sencillo para almacenar datos en forma de tablas (similar a como lo hace un manejador de bases de datos). Este formato presenta varias ventajas respecto a otros tipos de formato:

- ✓ Si se requiere, puede ser fácilmente convertido a un formato txt sin alterar su contenido.
- ✓ Es más fácil de trabajar con él con ciertos lenguajes de programación en comparación a otros formatos de archivo.
- ✓ Es un formato soportado por muchos tipos de software para su manejo (incluidos muchos sistemas operativos).

Para evitar errores o ambigüedades al momento de realizar búsquedas dentro del archivo desde la aplicación, se reemplazó el carácter de separación de columnas (por defecto el carácter coma (,)) por un carácter que no aparezca dentro de los códigos o las descripciones en los datos almacenados, para este caso el signo de porcentaje (%). Elegir un carácter que no aparece dentro de los datos en el archivo evitará que, al momento de interpretar las distintas columnas desde el código de la aplicación, no se confunda con las comas existentes dentro de las descripciones; más aún, emplear dicho carácter evitará que también se confundan los puntos y coma (;), las barras verticales (|) y otros caracteres popularmente usados en los archivos CSV, los cuales también aparecen en el texto de las descripciones de los códigos.

En la figura 2.6.1a, se aprecia como el carácter '%' fue empleado como carácter separador para evitar confusiones durante la búsqueda.

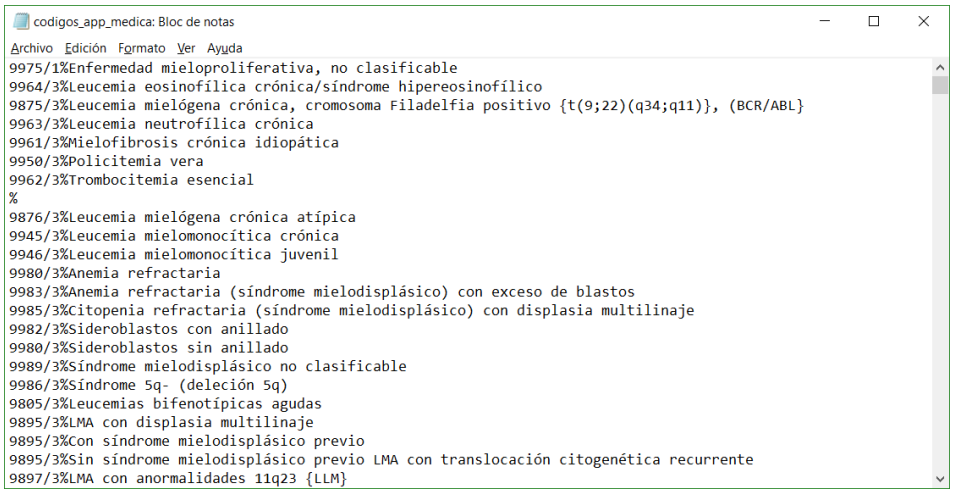

Figura 2.6.1a. Archivo CSV visto como texto plano.

2.8 Plataforma de desarrollo a emplear

Con el objetivo de realizar y validar rápidamente una primera entrega funcional de la aplicación, se ha optado por desarrollar esta primera versión de software bajo un solo sistema operativo. Esto permitirá validar de forma rápida la aplicación, ahorrándose el tiempo de desarrollar la aplicación para otras plataformas móviles, lo cual todavía puede realizarse después según convenga, se solicite o sea requerido.

Después de todo lo mencionado con respecto a los sistemas operativos en el capítulo del marco teórico, se ha optado por desarrollar la aplicación móvil bajo la plataforma de los sistemas operativos Android. Esto se justifica por ser éste el sistema operativo móvil más usado a nivel mundial (superando incluso en 2017 a los sistemas operativos de la familia Windows, incluyendo a sus versiones para escritorio), lo que implica poder llevar la aplicación a una mayor cantidad de usuarios de dispositivos móviles sin tener que adquirir un dispositivo adicional al que posean. Más aún, es bajo los ambientes de desarrollo de esta plataforma sobre los que se posee más conocimiento por parte del desarrollador del software a crear.

En la figura siguiente se puede observar mejor esta relación de usuarios respecto a los principales sistemas operativos. Se muestran los sistemas operativos móviles más usados respecto al total de usuarios a nivel mundial. Se aprecia como Android abarca ampliamente la mayoría de los usuarios desde hace años.

Period	Android	iOS	Windows Phone	Others
2015Q4	79.6%	18.7%	1.2%	0.5%
2016Q1	83.5%	15.4%	0.8%	0.4%
2016Q2	87.6%	11.7%	0.4%	0.3%
2016Q3	86.8%	12.5%	0.3%	0.4%

Source: IDC, Nov 2016

Figura 2.8. Comparativo de los sistemas operativos móviles más usados.

2.9 Entorno de desarrollo a utilizar

Una vez seleccionada la plataforma de desarrollo para la cual se elaborará la aplicación móvil, se procederá a seleccionar un entorno de desarrollo en el cual trabajar. Para el desarrollo de esta aplicación, considerando que se realizará un desarrollo nativo para Android, se hará uso del Entorno de Desarrollo Integrado (IDE, por sus siglas en inglés) oficial de la plataforma Android, Android Studio. Este entorno creado por Google para el desarrollo de aplicaciones bajo cualquiera de las versiones de la plataforma Android, proporciona una herramienta bastante completa, confiable y segura para diseñar, desarrollar y depurar (realizar "debugging") aplicaciones para todos los dispositivos que soporten dicho sistema operativo móvil.

Capítulo III. Diseño

En este capítulo se hará una interpretación de todos los requerimientos de software obtenidos durante el capítulo de análisis, para así ser capaz de crear un modelo o maqueta de lo que se desea desarrollar para posteriormente ser traducido a una aplicación móvil que satisfaga la solución que fue propuesta. Esto ayudará a proporcionar una mejor idea de lo que se desea programar, cómo se planea hacer y lo que se espera que realice el software resultante, lo cual hará que resulte más comprensible y legible para terceros lo que ha sido diseñado en este apartado. Como adicional, se describirá la forma en la que se trabajará con la metodología tipo Scrum para organizar y estructurar las etapas del desarrollo.

3.1 Traducción de los requerimientos de software

Con base a los requerimientos obtenidos durante el análisis de requerimientos de software que fue realizado, se procederá a realizar una explicación general o abstracción y una interpretación de lo que se desea que realice la aplicación móvil a programar. Con esto se podrá proporcionar una idea más concreta de los módulos que la integrarán y poder así crear y añadir a cada uno la funcionalidad que presentarán en el software.

Adicional a esto y acorde a la metodología de desarrollo que se emplea en este proyecto, permitirá crear más adelante el product backlog (pila de producto) el cual es requerido en el marco de desarrollo similar a Scrum con las actividades o funcionalidades de la aplicación a realizar. Como actividad momentánea primordial, de momento se realizará un análisis y diseño de la funcionalidad de la aplicación.

3.1.1 Características (features) a incluir en la aplicación móvil

Primeramente, se desplegará la interfaz que permitirá al usuario realizar las consultas en la base de datos que se elaboró, apegado a lo que indican los requerimientos mencionados. En esta interfaz de usuario se requiere y planea mostrar los elementos con los que interactuará el usuario, tanto para visualizar los contenidos de su interés, así como para poder elegir y manejar las funcionalidades que presenta la misma. Las funcionalidades que se requieren, de acuerdo a los requerimientos de software solicitados:

- Almacenar dentro de la aplicación móvil la base de datos con los códigos y descripciones del libro, la cual fue elaborada

y mencionada durante el capítulo de análisis.

- Poder ingresar un criterio de búsqueda por parte del usuario, el cual constará de una o varias palabras claves para poder realizar dicha búsqueda.

- Poder solicitar una búsqueda de acuerdo al criterio de búsqueda ingresado por el usuario.

- Desplegar los resultados o coincidencias encontrados en la base de datos de la aplicación que contiene los códigos y descripciones tomadas del libro.

- Se deberán resaltar o hacer distinguibles dentro de los resultados de búsqueda, las coincidencias encontradas entre todos los resultados desplegados y los criterios de búsqueda del usuario.

- La aplicación sebe ser completamente autónoma para su funcionamiento.

Además de todas las funcionalidades mencionadas, se añadirán otras adicionales que no interfieran con los requerimientos solicitados. Se agregarán estas características extras con el fin de mejorar la experiencia de usuario y de esta manera presentar una aplicación móvil más cómoda, intuitiva y agradable para los usuarios:

- Interfaz sencilla y minimalista.
  Tomando en cuenta lo mencionado por varios autores respecto a la experiencia de usuario, la interfaz como uno de los aspectos a considerar para poder proporcionar una experiencia agradable a los usuarios, se planea desarrollar una aplicación móvil con una interfaz fácil de comprender e intuitiva para el usuario, se optará por crear una interfaz sencilla y que además presente mucha usabilidad.

- Agregar mensajes al usuario.
  Con la finalidad de hacer a la aplicación más interactiva y más intuitiva, se agregarán mensajes al usuario respecto a las acciones relevantes para que pueda comprender mejor la forma de usar la aplicación y se le notifique los sucesos relevantes a la misma.

- Botón de limpieza de búsqueda y resultados.
  De nuevo, considerando proporcionar comodidad a los usuarios al hacer uso de este software, se añadirá esta característica. Esto agilizará y hará más cómodas las

búsquedas siguientes al no necesitar vaciar los campos de búsqueda manualmente (seleccionar todo el texto y elegir borrar).

- Facilitar el copiar el texto de los resultados de búsqueda al portapapeles del sistema operativo.
  Tomando en cuenta una vez más la experiencia de usuario orientada en el sentido de la comodidad, se considera y se analiza que en muchas de las ocasiones el usuario podría requerir disponer del texto con los resultados de la búsqueda para poder ser empleado por otras aplicaciones (procesadores de texto, hojas de datos, correo electrónico, mensajería instantánea, bloc de notas, por mencionar algunos). Por tal razón, se facilitará una opción para poder guardar dichos resultados en el portapapeles de Android, sin tener que verse obligado a seleccionarlos todos y seleccionar la opción copiar. Se planea reemplazar este procedimiento por una opción más sencilla y cómoda al usuario.

3.2 Primer esquema de la interfaz gráfica

Dicho todo esto, se creará un primer boceto o idea de la interfaz gráfica de usuario que se planea desarrollar para la aplicación móvil, la cual se apegará a las características o features y apegado a la experiencia de usuario mencionadas en el punto anterior.

Los siguientes son los primeros bocetos diseñados de la interfaz gráfica. Se incluyen tanto vista horizontal como vista vertical.

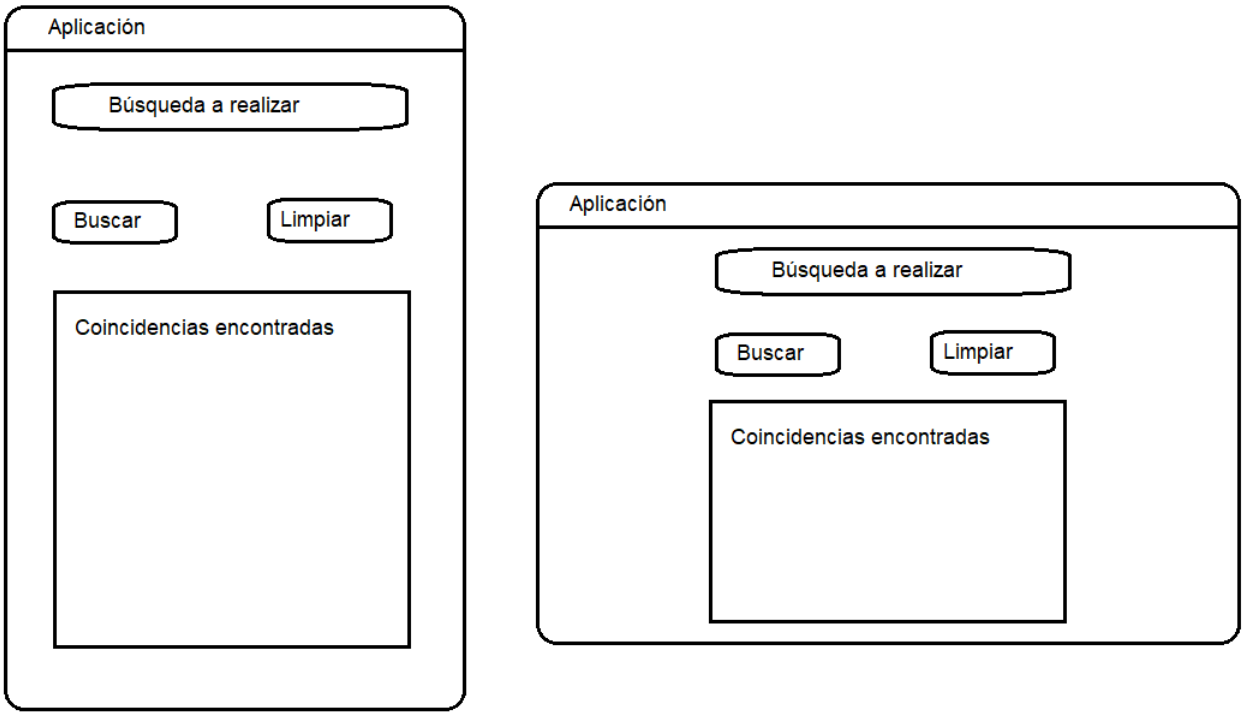

Figura 3.2. Bocetos de la interfaz gráfica de la aplicación a desarrollar.

Como se puede apreciar, se incluyen:

- Campo de búsqueda (indicado en la imagen como "Búsqueda a realizar").
- Botón de búsqueda (indicado en la imagen como "Buscar").
- Botón de limpieza de campos de búsqueda y de resultados (indicado en la imagen como "Limpiar").
- Campo de resultados (indicado en la imagen como "Coincidencias encontradas").

Como se mencionó, acorde a los conceptos de la experiencia de usuario, se ha decidido presentar una interfaz gráfica lo más sencilla e intuitiva posible para quienes hagan uso de la aplicación. Los elementos presentan una sencillez para poder ser entendidos por el usuario intuitivamente sin requerir mucha explicación de los mismos. Se ha considerado también el mostrar pocos elementos y que

todos están a la vista para la comodidad del usuario. Al poseer pocos elementos, el usuario se encuentra con las herramientas que necesita para manejar la aplicación, al alcance de la mano, sin requerir navegación adicional entre muchos menús o pantallas, haciendo la interfaz bastante intuitiva y cómoda.

3.3 Funcionamiento de la aplicación

Ahora se explicará el funcionamiento de la app para poder visualizar y explicar cómo trabajará la misma como aplicación de usuario.

1. Al iniciar la aplicación, el usuario deberá ingresar uno o varios criterios de búsqueda dentro del campo de búsqueda.

2. Deberá presionar el botón "buscar" para poder iniciar la búsqueda de resultados.

3. La aplicación tomará los criterios proporcionados en el campo de búsqueda e iniciará un algoritmo de búsqueda.

4. La función de búsqueda procederá a localizar las coincidencias encontradas dentro de la base de datos de los códigos y descripciones del libro, con las palabras claves proporcionadas por el usuario en el campo de búsqueda.

5. La aplicación desplegará y permitirá visualizar las coincidencias encontradas en el campo de resultados. Se resaltarán las coincidencias entre las palabras claves y los resultados de búsqueda.

6. El usuario podrá ingresar una nueva búsqueda (cuantas veces lo desee) o bien copiar al portapapeles de Android los resultados encontrados del campo de resultados.

7. El usuario podrá presionar el botón "Limpiar" en cualquier momento para despejar tanto el campo de búsqueda, como el campo de resultados.

3.4 Interacción del usuario con la aplicación

A continuación, se presenta un diagrama que permite visualizar las interacciones que se le permitirá al usuario tener con la aplicación móvil. Esto permitirá crear los módulos a programar necesarios para permitir estas interacciones aplicación - usuario.

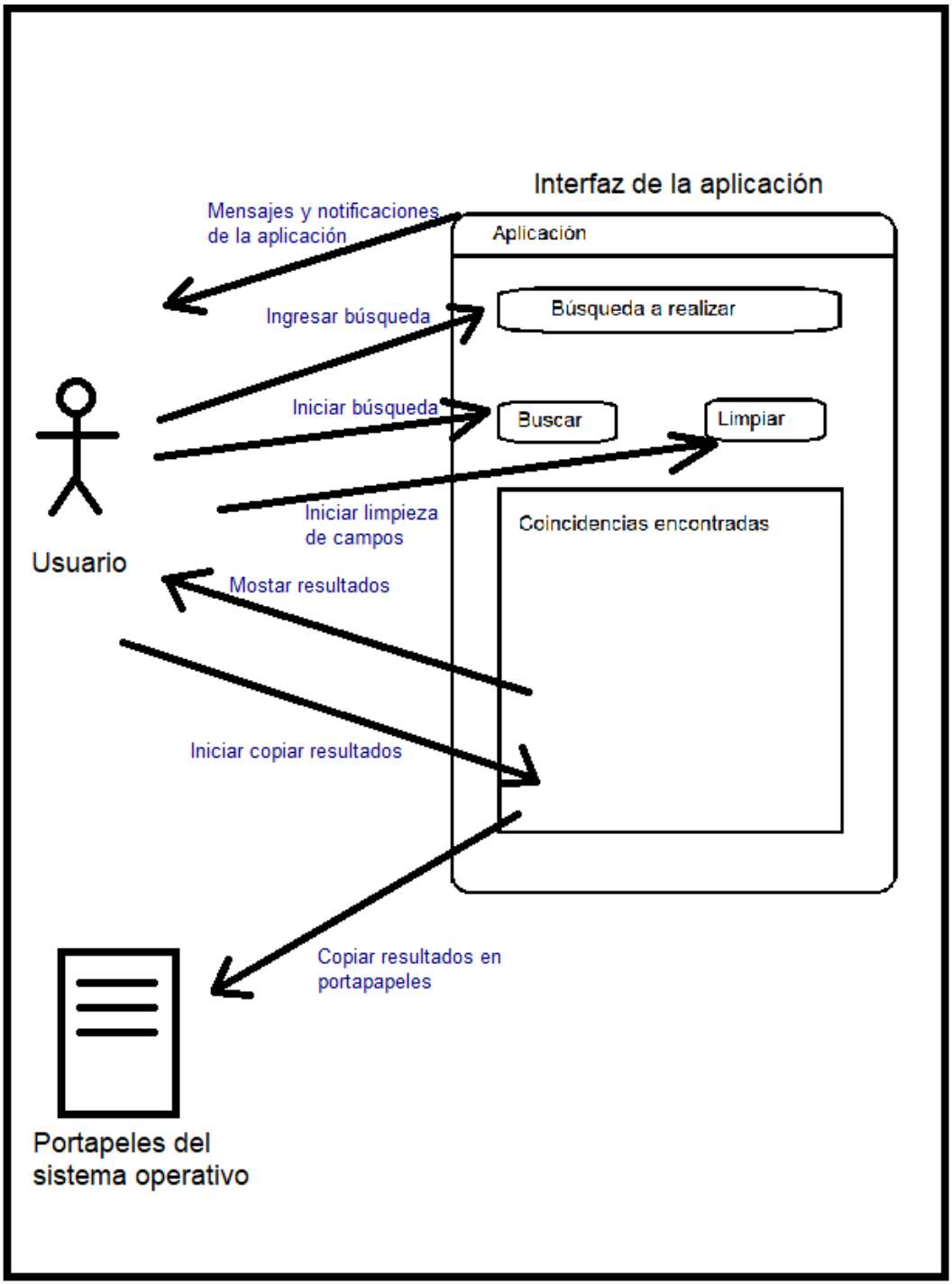

Figura 3. Diagrama que muestra las interacciones del usuario con la interfaz de usuario.

3.5 Funcionamiento interno de la aplicación

Ahora se procederá a explicar las funcionalidades internas de la aplicación, es decir, aquellas que resultan invisibles al usuario o de las cuales no es necesario enterarse para poder hacer uso de la misma. De nuevo se empleará un diagrama para poder bosquejar lo que se espera se realice internamente por la aplicación móvil. De aquí se extraerán los nuevos módulos a desarrollar que sean pertinentes. Este diagrama muestra las interacciones interfaz - aplicación. Como se aprecia, el funcionamiento de las mismas resulta invisible para el usuario.

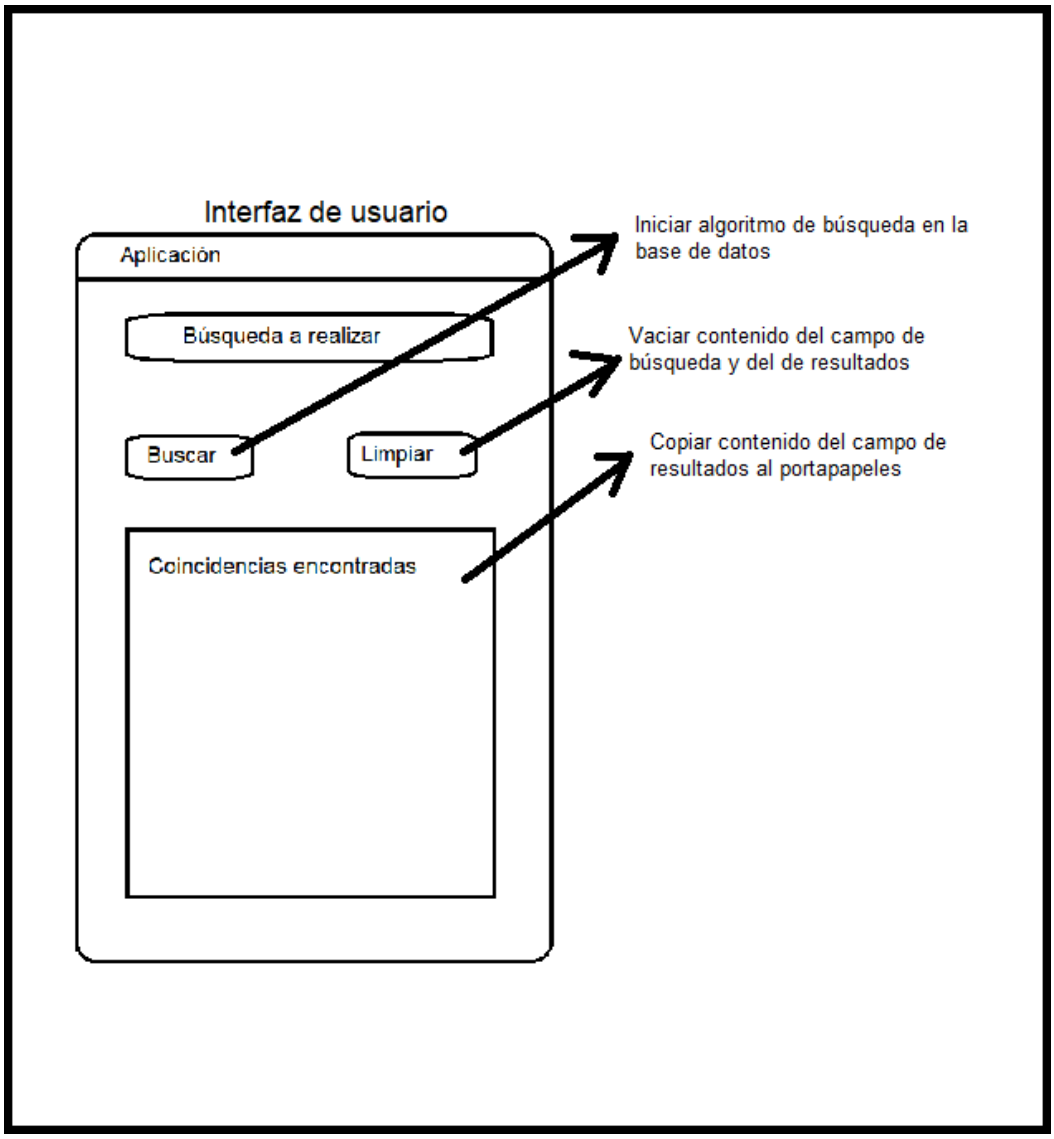

Figura 3.. Diagrama que muestra las funciones solicitadas por el usuario de la aplicación.

Como se puede apreciar en el diagrama, son cuatro actividades principales que se pueden realizar en el programa desde la interfaz de usuario, obviamente, por el usuario y cuando éste las solicite. Se considerará a estas peticiones como otros módulos a desarrollar.

Una vez realizado el diseño de las interacciones mostradas, se ha decidido que se emplearán y desarrollarán (de ser necesario), de momento, los siguientes módulos o componentes para la aplicación móvil:

- Campo de búsqueda
- Botón de búsqueda
- Botón de limpieza de campos
- Campo de resultados
- Base de datos portátil (la cual ya ha sido elaborada)
- Algoritmo de búsqueda de resultados en la base de datos
- Despliegue de resultados de la búsqueda
- Función de limpieza (vaciado) de los contenidos en el campo de búsqueda y el de resultados
- Función de mostrar mensajes y notificaciones al usuario
- Función de copiado de texto al portapapeles

3.6 Product backlog para el análisis y desarrollo de los módulos empleando Scrum

Apegado a la metodología tipo Scrum que se está empleando, se añadirá el análisis y posterior desarrollo de cada uno de estos módulos, como actividades a realizar en el product backlog. Antes de añadirlos, se considerará asignar una prioridad correspondiente a todas de ellas (como indica la metodología), esto para poder presentar un avance funcional de lo realizado al momento.

Para poder presentar este avance, se requiere del desarrollo de los componentes básicos de la interfaz gráfica, obligatoriamente. También es requerido que realice las consultas, es otras palabras, se requiere del algoritmo de búsqueda, la base de datos, y el despliegue de resultados. Luego, se les proporcionará a estos módulos alta prioridad:

- ✓ Campo de búsqueda
- ✓ Botón de búsqueda
- ✓ Campo de resultados
- ✓ Base de datos portátil
- ✓ Algoritmo de búsqueda
- ✓ Despliegue de resultados

Por no ser necesarios para la primera entrega o versión del software o por depender de otros módulos de mayor prioridad, los siguientes módulos quedarán como actividades de segunda prioridad. Estos serán elaborados después de los de alta prioridad ya mencionados:

- ✓ Botón de limpieza
- ✓ Función de limpieza
- ✓ Función de mostrar mensajes y notificaciones
- ✓ Función de copiado de texto al portapapeles

3.6.1 Módulos de la aplicación

En esta sección se explicará lo más detalladamente posible la funcionalidad que poseerá cada uno de los módulos que se planean emplear y/o desarrollar para la aplicación móvil, los cuales después serán traducidos a su respectivo código durante el desarrollo (o adaptados si ya se tiene disposición de ellos). Esto incluye sus funciones dentro de la aplicación y su justificación.

3.6.1.1 Base de datos portátil

Esta base de datos es la que almacena la información que será requerida del libro Clasificación Internacional de Enfermedades para Oncología, tercera edición. Como se mencionó durante el capítulo de análisis, la base de datos se limitará a almacenar tanto códigos de enfermedades y de partes del cuerpo, así como su respectiva descripción, esto para agilizar la búsqueda y no hacerla más pesada de lo que realmente se requiere, además se ahorrará espacio de almacenamiento empleando un archivo más pequeño. Esto, obviamente, causará un menor espacio requerido en almacenamiento del dispositivo móvil.

Para este caso, se ha requerido el hacer uso un repositorio de datos portátil. Esto se realiza con el fin de hacer a la aplicación totalmente autónoma una vez instalada en el dispositivo (tecnología stand-alone precisa por los requerimientos de software), es decir, que no requiera conectarse a alguna red ni a otro dispositivo para poder operar sus funciones después de la instalación (realizar consultas, principalmente).

Dado que no se requiere almacenar ningún tipo de formato con los datos, en este caso se ha decido emplear un archivo de texto plano (también llamado simplemente "archivo txt"). Esto trae consigo varias ventajas que también fueron tomadas en cuenta:

- ✓ Se ahorra almacenamiento en disco, dado el poco peso que presentan este tipo de archivos al no poseer formato.

- ✓ Se facilita el manejo del archivo bajo otras plataformas al ser este formato compatible con las versiones de Windows y los sistemas operativos de la familia Unix y Linux (por consiguiente, para Android también).

- ✓ Es un formato fácil de trabajar durante el desarrollo gracias a las múltiples bibliotecas del lenguaje de programación Java.

3.6.1.2 Campo de búsqueda

En este campo el usuario deberá ingresar su criterio de búsqueda conforme a lo que desee consultar. Se verificará que los datos ingresados por el usuario sean válidos para una consulta (poseer suficientes caracteres o ingresar solamente caracteres válidos, por ejemplo) y así poder buscar entre los campos de la base de datos.

Al poseer bastantes registros la base de datos, es preciso considerar una longitud mínima en los criterios de búsqueda. Esto con el fine de evitar que, al momento de comenzar a buscar coincidencias con secuencias de caracteres muy cortas, se cree un retardo en la búsqueda al encontrarlas en muchas ocasiones. Más aún, el desplegado de caracteres arrojaría todas estas coincidencias haciendo de poca utilidad la búsqueda al tener que volver a buscar (ahora por parte del usuario) los datos de interés.

Por otro lado, la validación de caracteres se debe principalmente a que posiblemente se requiere emplear una expresión regular durante la ejecución del algoritmo de búsqueda. Estas expresiones regulares, según el lenguaje (Java, en este caso), pueden presentar errores o excepciones al tratar de procesar cadenas de texto que contengan algunos de los caracteres reservados para expresiones regulares. Durante la validación de caracteres se deberán reemplazar o eliminar

(si es posible) estos caracteres con el fin de evitar excepciones durante alguna búsqueda que pudiera contenerlos.

3.6.1.3 Botón y algoritmo de búsqueda

Para poder realizar una localización de las coincidencias existentes entre el o los criterios de búsqueda ingresados en el campo de búsqueda por parte del usuario (asumiendo que dicha entrada ya fue validada) y los registros existentes en la base de datos de la aplicación, es necesario diseñar y desarrollar un algoritmo de búsqueda y ubicación de coincidencias existentes entre ambos. Se necesita que dicho algoritmo realice un recorrido campo por campo de la base de datos elaborada y realizar la comparación de caracteres.

Se deberá realizar una comparación tanto con los códigos, como con las descripciones de los mismos. Esto se realizará tomando en cuenta que el usuario puede ingresar un código como los que aparecen en el libro, o bien, una o varias palabras claves de la descripción de interés de los mismos.

El siguiente diagrama representa el recorrido y la comparación de cadenas durante la búsqueda entre los campos de la base de datos.

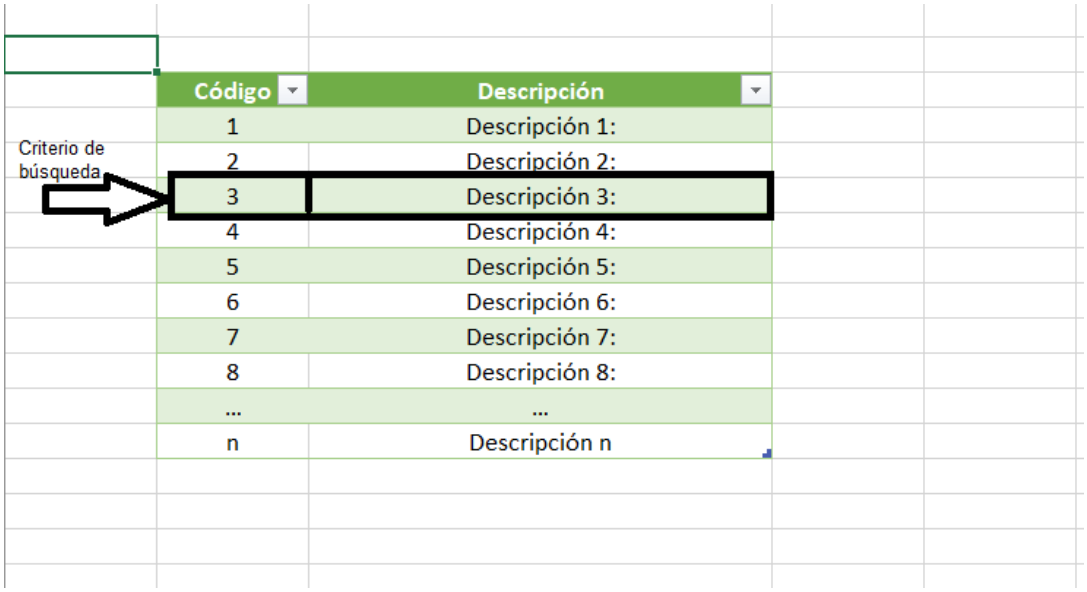

Figura 3.3.a. Diagrama que representa la búsqueda realizada.

Tal algoritmo debe verificar si alguno de los criterios de búsqueda de la instancia actual está contenido o es idéntico al campo de la

base de datos que se está comparando en dicho momento. Esto presentará tres posibles situaciones:

1) La secuencia de caracteres de la búsqueda es idéntica al campo de la base de datos que se compara.

2) La secuencia de caracteres está contenida en el campo de la base de datos.

3) La secuencia de caracteres no aparece en el campo de la base de datos.

Para los dos primeros casos, se considerará que existe una coincidencia en la búsqueda y se tomará el registro localizado (tanto a dicho campo, como su campo adjunto, es decir, se tomará todo el renglón o registro) como uno de los resultados de búsqueda. Para el tercer caso, se considera que no hay coincidencia alguna en el campo de la base de datos, se ignorará tal campo y se procederá comparar los siguientes campos.

Durante el recorrido de los campos de la base de datos de la aplicación, se irán concatenando dentro de una variable temporal los resultados de búsqueda (registros completos) para que dichas cadenas no sean perdidas durante las próximas coincidencias encontradas y poder ser manejados para poder mostrarse al final como requiere la aplicación, tarea que es correspondiente a otro módulo y será descrita en su correspondiente apartado.

3.6.1.4 Campo de resultados

Este campo de texto será el encardo de permitir al usuario visualizar los resultados arrojados por el algoritmo de búsqueda (en caso de haberlos). Este campo deberá poder desplegar cada uno de los resultados obtenidos después de haber sido realizada una búsqueda, resaltando además las coincidencias existentes entre el o los criterios de búsqueda proporcionados por el usuario y los resultados de la búsqueda.

Para poder hacer notorias o distinguibles dichas coincidencias, se requiere un componente de texto que soporte formato de texto para poder proporcionar un resalte a las coincidencias de búsqueda. Mediante esto se podrán resaltar dichas partes del texto de resultados, cambiando las coincidencias por un texto con un formato distinto que resalte sobre el resto del texto. Se ha optado por emplear un elemento textView.

El siguiente es un ejemplo visual de otros formatos de texto en un textView. Se puede apreciar la diferencia entre resalte que

proporciona cada uno. Unos resultan más fáciles de distinguir del resto del texto que otros.

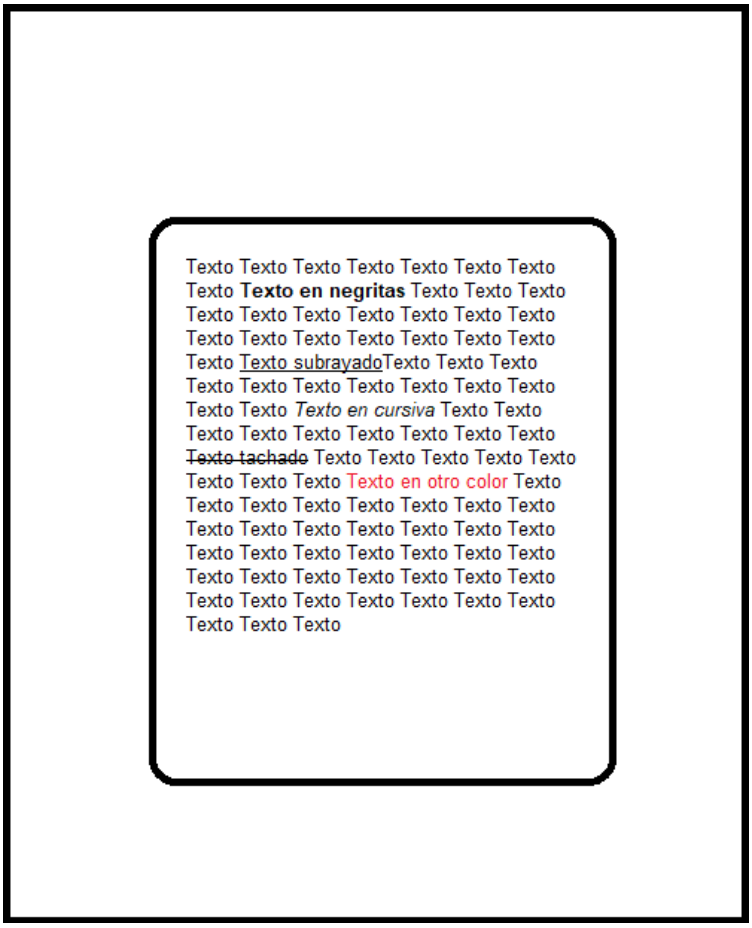

Figura 3.x.x Ejemplo de otros formatos de texto en un textView.

Apegado a proporcionar una mejor experiencia de usuario, se analizará la opción más conveniente para escoger el tipo de resalte que se utilizará. Primeramente, como opciones de formato propuestas, se tienen el subrayado, las cursivas, las negritas (boldface) o el cambio de color en el texto. Cabe mencionar que se han descartado de entrada otras opciones como el cambio de tamaño de la fuente a uno mayor, al considerar que desperdicia espacio en pantalla, a diferencia de las otras. Otra opción desechada fue el tachado, al considerar que ésta quita o disminuye la legibilidad del texto remarcado, haciendo incómoda la lectura y afectando la experiencia de usuario.

Para este caso, se optará por emplear un cambio de color en el texto coincidente. Esto proporcionará una opción mucho más llamativa y más

cómoda visualizar que otras de las opciones, por ejemplo, la cursiva, donde el texto no resulta tan apreciable o discernible del resto del texto, a simple vista. Para los otros casos (negritas y subrayado), en algunos dispositivos estos formatos pueden resultar molestos de visualizar, dado el pequeño tamaño de la pantalla o la poca resolución de algunos dispositivos. Considerando buscar una mayor cantidad de dispositivos compatibles, se han descartado también estas opciones.

3.6.1.5 Despliegue de resultados

Para poder resaltar las coincidencias de búsqueda, se ha optado por hacer uso de etiquetas HTML para modificar tales partes del texto que contiene los resultados. Añadiendo etiquetas al texto que se quiere resaltar, permitirá resaltarlas las coincidencias de búsqueda dentro del textView para poder ser apreciadas por el usuario. Se agregará el código pertinente para poder visualizar estas propiedades HTML dentro del textView durante el desarrollo del mismo.

Para realizar esta tarea, se desarrollará una función que permita agregar dichas etiquetas dentro del texto de resultados de búsqueda. Una vez añadidas tales etiquetas, se procederá a enviar el texto de resultados de búsqueda al textView donde se podrán interpretar mediante HTML y pasarán a ser reemplazadas por el texto resaltado (en este caso, texto en otro color).

3.6.1.6 Botón y función de limpieza de campos

Pensando en proporcionar más comodidad al hacer uso de los elementos de texto en la aplicación (textView y editText), se empleará un botón asociado a una función de limpieza de texto. Éste se encargará de vaciar el contenido de ambos componentes de la aplicación, eliminando cualquier texto que se encuentren mostrando tales. Esto hará más rápida y cómoda la siguiente búsqueda que se realice en la aplicación al evitar tener que borrar los campos manualmente.

Se desarrollará una función asociada a un evento del botón “limpiar” la cual se encargará de suprimir el texto contenido en ambos componentes. Dicha función no requerirá argumentos para ser llamada.

3.6.1.7 Mensajes al usuario

Para poder indicar al usuario de las acciones que se espera realice, el resultado de las acciones realizó o indicarle algún error que pudiera llegar a ocurrir, se requiere el tener una correcta comunicación entre la aplicación y el usuario. Para esto se ha

decidido por emplear una manera de enviar mensajes al usuario con el fin de informarlo sobre los eventos sobre la aplicación que le sean relevantes a nivel usuario, así como indicaciones sobre acciones que se espera que realice.

Para esta aplicación se hará uso del toast de Android para realizar esta tarea. El toast proporciona la posibilidad de mostrar notificaciones simples de duración ajustable, dentro de la aplicación misma, empleando un diseño minimalista y fácil de leer para el usuario.

Para permitir una reutilización del código fuente, se ha optado por desarrollar una función que sea llamada dentro del código cuando se requiera mostrar una notificación al usuario. Dicha función recibirá como argumentos el texto con el mensaje que se desea mostrar y la misma de encargará de iniciar el toast para mostrarlo en pantalla con el mensaje adecuado.

3.6.1.8 Función de copiado de texto al portapapeles

Como ya se ha mencionado, considerando la experiencia del usuario en el sentido de la comodidad, se agregará esta función a la aplicación móvil.

Se pretende crear una manera de permitir recolectar el texto mostrado en el textView de la aplicación y traspasarlo al portapapeles del sistema operativo para su posterior uso por parte del usuario. Dado que el portapapeles del sistema operativo Android no almacena formato alguno, no será necesario almacenar el formato que se la ha dado al texto dentro del textView, por lo tanto, se despreciara éste.

Para esto se desarrollará una función que obtenga el texto contenido por el textView, lo almacene de forma temporal en la aplicación para después sea enviado al portapapeles del sistema operativo. Dicha función no requerirá argumentos, sin embargo, enviará una notificación al usuario que le permita saber que el texto fue copiado al portapapeles con éxito.

3.7 Layout

Considerando proporcionar una interfaz de usuario que, además de ser cómoda, sea de un tamaño adecuado al dispositivo que la muestre, se ha decidido hacer uso de una interfaz responsiva. Este tipo de interfaz permitirá que los componentes de la misma adapten su tamaño y disposición en función del tamaño de la pantalla del dispositivo actual del usuario.

Con la finalidad de realizar esto, se utilizará un ConstraintLayout. Este layout permite colocar y redimensionar elementos en una vista de forma responsiva.

Capítulo IV. Desarrollo

Durante este capítulo se procederá a convertir el boceto realizado durante el diseño de software, en el código correspondiente con el fin de obtener una aplicación funcional como la que se ha propuesto durante el análisis hecho previamente. Se explicará lo más detalladamente posible el código a realizar y su respectiva justificación en el proyecto. Además, de nuevo se mencionarán y se tomarán en cuenta las consideraciones que se crean necesarias durante la codificación con el objetivo de mejorar la experiencia de usuario en dicha aplicación móvil y obtener un producto que proporcione un mejor resultado.

4.1 Ícono de la aplicación

Tratando de mejorar la experiencia del usuario proporcionando un icono más vistoso y distintivo que el que Android Studio proporciona por defecto, se ha seleccionado otro icono que identifique a la aplicación móvil. Dado el ámbito de la app y pensando en los potenciales usuarios de esta misma, se ha decido usar el logotipo de la Sociedad Mexicana de Oncología (SMeO).

Figura 4.1. Logotipo de la SMeO.

4.2 Archivos XML

Mediante los archivos XML del proyecto en Android Studio, se harán las modificaciones necesarias a los componentes de la aplicación. Para este caso se emplearán principalmente para configurar la interfaz de usuario y para el manejo de las cadenas de texto en la aplicación. A continuación, se explicarán dos de los principales archivos XML que se emplearán en esta aplicación.

4.2.1 activity_main.xml

Dentro de este archivo XML se manejan los componentes de la interfaz gráfica. En éste se harán las modificaciones necesarias para el diseño de los componentes gráficos de la aplicación (dimensiones y posiciones), propiedades de los componentes, además de la creación de eventos asociados a los mismos, los cuales estarán encargados de iniciar las funciones de la aplicación.

En la siguiente captura de pantalla se muestra una parte del IDE de Android Studio. En dicha pantalla se pueden visualizar los resultados de las modificaciones realizadas al archivo activity_main.xml, reflejados en la interfaz de usuario.

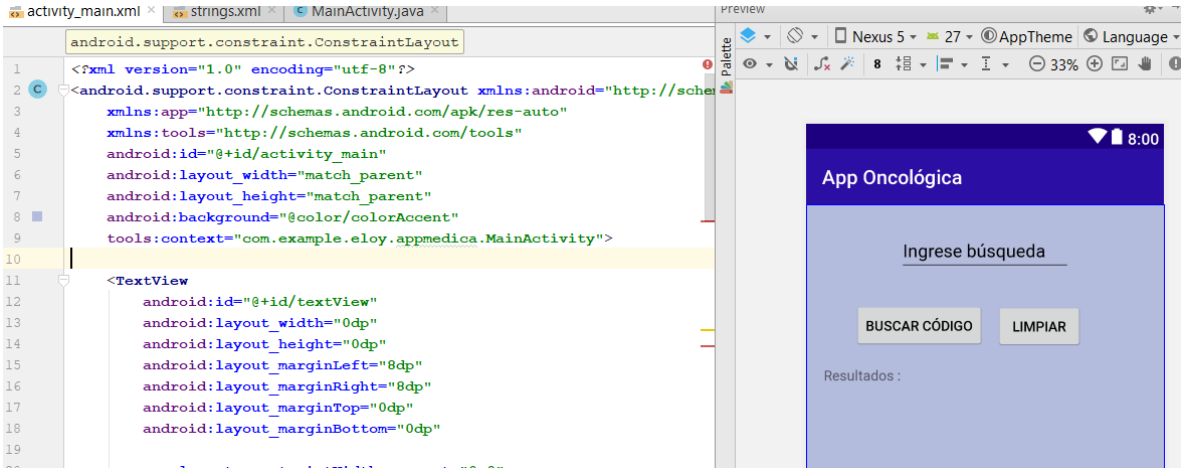

Figura 4.2.1. Parte del archivo activity_main.xml.

4.2.2 strings.xml

Para facilitar el manejo y modificación de cadenas de texto (strings) dentro del proyecto, se colocarán en el archivo "strings.xml" las cadenas que serán utilizadas por el código fuente de la aplicación. Esto incluye el texto mostrado por los componentes de la interfaz gráfica. En la imagen siguiente se aprecian las cadenas de texto

dentro de la aplicación dentro del archivo XML. Desde aquí se puede a acceder a ellas más fácilmente dentro del proyecto.

```xml
<resources>
    <string name="app_name">App Oncológica</string>
    <string name="nombreButton">Buscar código</string>
    <string name="nombreBoton2">Limpiar</string>
    <string name="critBusq">Ingrese búsqueda</string>
    <string name="resultados">\nResultados :\n</string>
    <string name="buscar">buscar</string>
    <string name="limpiar">limpiar</string>
</resources>
```

Figura 4.2.2. Archivo strings.xml.

4.3 Componentes de la interfaz de usuario

Como ya se mencionó durante el diseño de la aplicación, se hará uso de varios componentes en la interfaz de usuario tales como botones y campos de texto para el uso de la misma. Tales componentes serán agregados a la vista actual en la que se trabaja. Estos serán:

- *ConstraintLayout* para el layout sobre el que se colocarán los demás componentes
- *button* para el botón "Buscar"
- *button* para el botón "Limpiar"
- *editText* para el campo de búsqueda
- *textView* para el campo de resultados

Los componentes de la interfaz serán colocados en la pestaña de diseño del IDE. Serán acomodados como se mostró en el boceto del capítulo anterior, como se aprecia en la siguiente imagen de la vista de la aplicación:

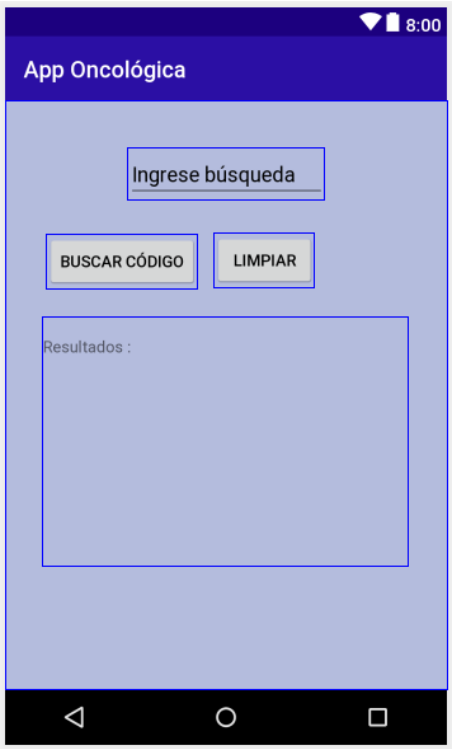

Figura 4.3. Primera disposición de los componentes de la interfaz de usuario.

4.3.1 Eventos asociados a los componentes

Una vez colocados los componentes de la interfaz gráfica en la vista, se procederá a asignar los correspondientes eventos que permitirán iniciar las funciones propias de la aplicación las cuales permiten el funcionamiento de la misma. Se agregarán a los botones los eventos de click que llamarán a las correspondientes funciones que se les asociarán. Esto se realizará desde el archivo xml, activity_main.xml añadiendo el evento onClick para iniciar la función asociada al botón "buscar". Se realizará de manera similar en el botón "limpiar".

```
android:layout_marginBottom="8dp"
android:layout_marginLeft="36dp"
android:layout_marginTop="8dp"
android:layout_weight="1"
android:onClick="buscar"
android:text="Buscar código"
app:layout_constraintBottom_toBottomOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
```

Figura 4.3.1. Añadiendo el evento onClick.

Nuevamente considerando en una interfaz minimalista que no requiera agregar más elementos a los que ya se añadieron, se ha decidido que la función de copiado de texto al portapapeles sea iniciada desde el mismo textView. Para realizar esto, se agregará un evento onClick a dicho componente con el fin de no añadir más botones, tener una interfaz sencilla e intuitiva y así mejor la experiencia de usuario.

```
android:onClick="copiaTextView"
android:scrollbars="vertical"
android:text=" Resultados : "
android:textIsSelectable="true"
app:layout constraintLeft toLeftOf="parent"
```

Figura 4.3.1a. Agregando el evento onClick al textView.

Mediante esto, el usuario podrá copiar al portapapeles de Android con un solo toque al textView haciendo más cómoda y rápida dicha tarea.

4.4 Responsividad en la interfaz gráfica

Buscando desarrollar una interfaz de usuario cuyos componentes se distribuyan y adapten correctamente al dispositivo en el que se ejecuta la aplicación móvil, se hará uso de la responsividad en el layout. Se requiere para esto que los componentes se posicionen con relación a los márgenes y a la posición de los demás componentes. Además, ciertos componentes (como es el caso del textView) requerirán dimensionarse con relación a la pantalla del dispositivo, esto con el fin de evitar de que dichos componentes quedan fuera de la pantalla y no resulten visibles al usuario. Para esto se requiere que los componentes ocupen un porcentaje constante de la pantalla en la que se visualizan.

Para lograr esto, se editarán los atributos de los componentes de la interfaz necesarios desde el archivo activity-main.xml. Primeramente, se mantendrá el porcentaje que ocupa el textView en la pantalla del dispositivo. Se añadirán los atributos layout_constraintWidth_percent y layout_constraintHeight_percent para cambiar su tamaño dinámicamente con base a la resolución del dispositivo y su orientación actual. Se optará por que estos siempre ocupen un noventa y cincuenta por ciento de la pantalla, respectivamente.

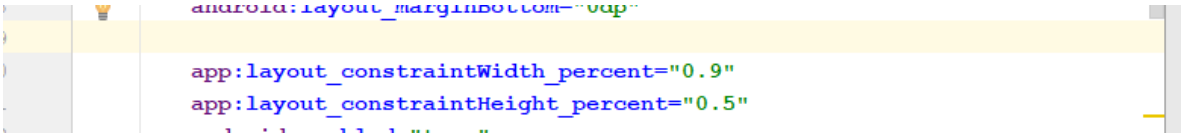

Figura 4.4a. Estableciendo el porcentaje del textView en pantalla.

Considerando que los otros componentes (botones y editText) poseen un tamaño adecuado para ser utilizados, se optará por dejarlos con un tamaño estático, esto con el fin de que no se vuelvan muy grandes en algunos dispositivos. Dicho esto, para evitar que los componentes mencionados se sobrepongan unos con otros, se añadirán algunos constraints para darles una posición relativa, es decir, que se organicen y acomoden dependiendo de la posición de los otros. Se añadirán dichas restricciones en los dos botones, en el textView y en el editText.

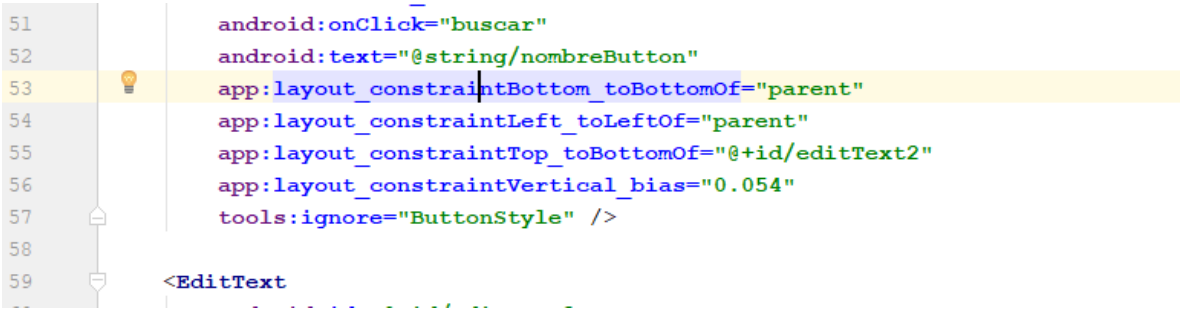

Figura 4.4b. Colocando constraints en el botón "buscar".

4.5 Main activity

Mediante el archivo MainActivity.java se controlará el funcionamiento de la aplicación móvil haciendo uso del lenguaje de programación Java. En este documento se escribirá el código fuente necesario para lograr desarrollar las funcionalidades diseñadas en el capítulo anterior. Más aún, se agregarán (de ser requeridas) las correcciones de código que sean necesarias para optimizar o corregir el funcionamiento del programa, tanto a nivel gráfico como a nivel software.

A continuación, se describirá el código de las funciones que serán contenidas en el archivo ya mencionado. Se indicarán sus argumentos, sus variables, los archivos que emplee, su detallado funcionamiento y el valor de retorno que les corresponda a cada una.

Primeramente, se agregarán en este mismo archivo algunas variables globales las cuales serán empleadas por las funciones que más adelante se mencionarán. Esto se realizará con el fin de acceder a ellas más fácilmente y ahorrar el pasarlas como parámetros entre funciones.

```
public class MainActivity extends AppCompatActivity {
    /*Variables globales*/
    TextView tv;
    EditText editText;
    InputStream inputStream;
    BufferedReader reader;
    String[] cvss;
    @Override
```

Figura 4.5. Variables globales de la app.

4.5.1 Función "buscar()"

Esta función, la cual es iniciada al presionar el botón "buscar" en la app, es la principal y una de las más importantes dentro de MainAvctivity.java. En esta función se ejecutan las instrucciones que permitirán buscar las coincidencias de búsqueda dentro de la base de datos de los códigos y descripciones, hacer las validaciones pertinentes e incluso manipular algunas características del textView según se requiera. El código que la estructura se explica a continuación.

Al iniciarse esta función, se almacenará el criterio de búsqueda dado por el usuario en una variable y se creará otra variable boleana ("encontrado") la cual indicará si hubo coincidencias en la búsqueda solicitada (iniciada por omisión como false). Para evitar iniciar una búsqueda no suficientemente específica y que despliega una larga cantidad de resultados, se verificará de primera instancia que la cadena de texto ingresada por el usuario posea determinada longitud mínima para poder realizar una consulta.

En caso de que la cadena no satisfaga esta condición, se le enviará una notificación al usuario para corregir su criterio de búsqueda, terminará la función con una sentencia de retorno y la aplicación esperará la siguiente instrucción del usuario. En otro caso, la función seguirá ejecutando las instrucciones siguientes del código.

```
vate void buscar(){//función asociada al evento de click al botón "buscar c
 String codigo=editText.getText().toString();//Criterios de búsqueda del us
 boolean encontrado=false;//Booleano que indica si se encontraron coinciden
 if(codigo.equals("")||codigo.length()<2){//si no hay suficientes criterioa
     tv.setText("");
     muestra("Ingrese un código válido a buscar (3 caracteres al menos).");
     return;
 }
 try{//reader para csv
```

Figura 4.5.1a. Validando longitud de la búsqueda.

Una vez validada la búsqueda ingresada por el usuario, se comenzará a buscar las coincidencias existentes entre esta entrada y los campos del archivo CSV. Para ello se crearán algunas variables temporales para manejar cadenas que no se requerirán almacenar posterior a la lectura del archivo.

Considerando esto, se crearán las variables tipo String "lineaCsv" y "aux", además de un arreglo de cadenas llamado "critBusq". Este último almacenará el o los criterios de búsqueda ingresado en el textView en cadenas separadas por espacios, lo cual se logrará mediante el método Split de la clase String. Además, para manejar las posibles excepciones que ocurran durante la búsqueda del archivo con la base de datos, se encerrarán estas instrucciones en bloque try-catch. En caso de caso de encontrar error en el archivo, la aplicación manejará la excepción como corresponde. Estas variables y bloque try-catch se observan a continuación:

```
    return;
}
try{//reader para csv
    //Reemplazar caracteres (),{},[],*,¿? para evitar crasheos!!!
    String lineaCsv;
    String aux="";
    tv.setText("");
    String [] critBusq=codigo.split(" ");//Separar en palabras el criterio de b
    }
    catch(IOException ex){//Excepción en lectura del archivo
        throw new RuntimeException("Error de lectura: "+ex);
    }
```

Figura 4.5.1b. Bloque de lectura del CSV.

Hecho esto, se iniciará un doble ciclo en esta misma función para recorrer cada línea de la base de datos y, a su vez, recorrer cada uno de los criterios ingresados por el usuario, los cuales fueron almacenados en el arreglo "critBusq". El ciclo while se encargará de leer el archivo CSV mientras siga encontrando líneas, es decir, recorrerá el archivo línea por línea desde la primera hasta la última que encuentre. Por otra parte, el ciclo interno for se encargará de recorrer el arreglo de cadenas "critBusq". Este ciclo se ejecutará al menos una vez, tomando en cuenta que ya se validó que la entrada no sea una cadena vacía. Una vez más, se hará uso de un bloque try-catch para manejar excepciones, esta vez encerrando al ciclo for. En la figura de abajo se observan estos ciclos anidados.

```
while((lineaCsv=reader.readLine())!=null){
    cvss=lineaCsv.split("%");//separa códigos y descripciones //Se puede mover
    try{
        for(int i=0;i<critBusq.length;i++){



        }
    }catch(Exception e) {
        tv.setText("Error :s :" + e.toString());          //Nunca debe entrar aqu
    }
```

Figura 4.5.1c. Ciclos anidados para comparar cadenas.

Como se aprecia antes de entrar al bloque try-catch, se almacena la línea leída en el arreglo "cvss" para poder trabajar por separado con códigos y descripciones. Una vez más se recurrirá al método split de la clase String. En la siguiente imagen se pueden visualizar de manera más gráfica los recorridos que hacen los ciclos anidados mencionados:

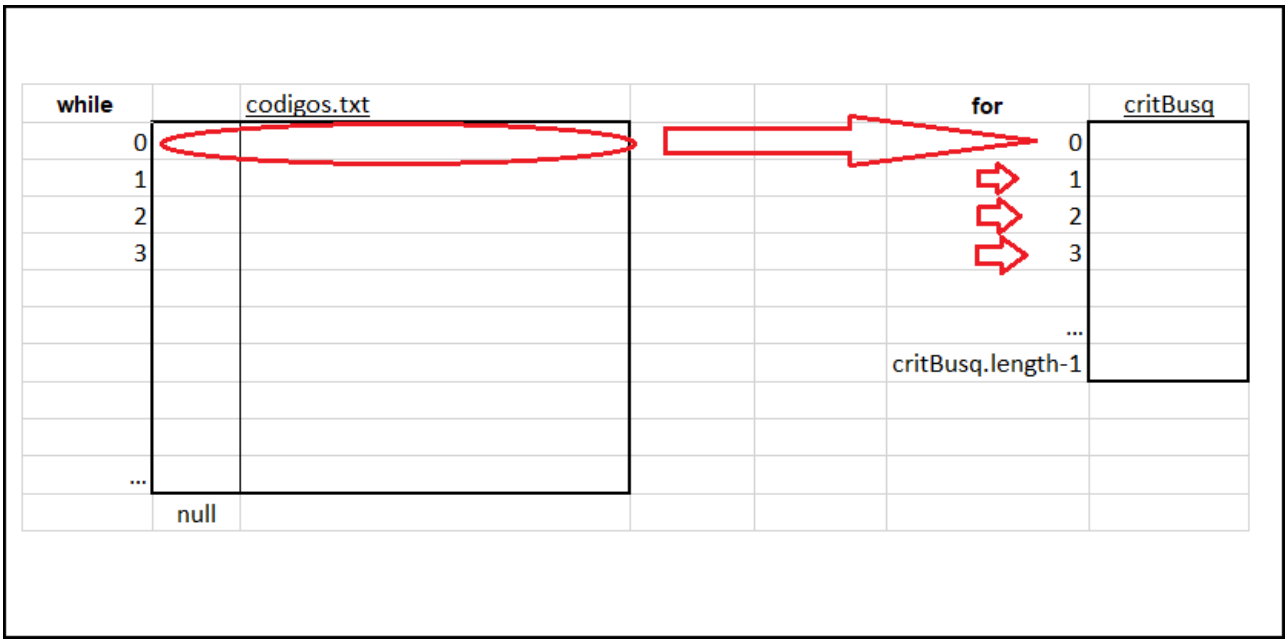

Figura 4.5.1d. Diagrama del recorrido de la base de datos.

Para poder hacer más cómodas las consultas y que se desplieguen resultados independientemente de sí las palabras están o no acentuadas adecuadamente en la búsqueda ingresada por el usuario, se hará empleo de la clase Normalizer de Java. Normalizando tanto la cadena leída del CSV como los criterios de búsqueda del usuario, se permitirá hacer comparación entre ambas cadenas, despreciando las posibles acentuaciones existentes en alguna de las dos. Por ejemplo, una búsqueda "cancer" [sic] será igual a ingresar "cáncer", haciendo más cómodas las consultas en la aplicación.

En este caso se almacenarán ambas cadenas normalizadas en variables temporales tipo String. Estas variables serán "normalizedAux" y "normalizedAux2".

```
String normalizedAux=Normalizer.normalize(lineaCsv, Normalizer.Form.NFD);//C
String normalizedAux2=Normalizer.normalize(critBusq[i],Normalizer.Form.NFD);
```

Figura 4.5.1e. Cadenas normalizadas.

Con la finalidad de descartar los caracteres que no aporten nada a la búsqueda o que puedan causar errores al no ser reconocidos adecuadamente (tales como los caracteres que son signos de

acentuación o signos propios de algún idioma), se ha optado por eliminarlos de la cadena. Se considera que resultará más eficiente descartarlos en las búsquedas realizadas.

Para lograr esto se hará uso del método replaceAll() de la clase String. Para expresar de manera general los caracteres que se espera reemplazar en la cadena, se hará uso de una expresión regular (REGEX) como primer argumento y como segundo argumento una cadena vacía. Los caracteres a reemplazar serán los que no pertenezcan al conjunto ASCII. Para esto se usará la expresión regular [^\p{ASCII}]. Serán sustituidos por cadenas vacías. Los reemplazamientos se aplicarán las dos cadenas normalizadas anteriormente y se almacenarán en dos nuevas variables String.

```
String lineaCsvNorm=normalizedAux.replaceAll("[^\\p{ASCII}]", "");
String critBusqNorm=normalizedAux2.replaceAll("[^\\p{ASCII}]", "");
```

Figura 4.5.1f. Reemplazando caracteres mediante una REGEX.

Hecho esto, el código entrará a un bloque condicional donde se verificará que los criterios ingresados a la aplicación estén o no contenidos en alguno de los registros de la base de datos local. Si todos los criterios están contenidos en la línea actual, se entiende que la línea del archivo CSV cumple con el o los criterios ingresados por el usuario y se le considera una coincidencia de búsqueda.

En caso de ser cierta la validación, la línea actual del CSV será concatenada en la cadena "aux", agregando unas etiquetas HTML(las cuales serán explicadas más adelante), a la variable booleana "encontrado" se le asignará el valor true. En caso contrario, el ciclo for se dentendrá con la instrucción break y se buscará en la siguiente línea del archivo CSV. El siguiente código se encargará de esta tarea:

```
if(lineaCsvNorm.toLowerCase().contains(critBusqNorm.toLowerCase()))
    ;
else
    break;
if(i==critBusq.length-1){
    aux=aux.concat("<b>"+cvss[0]+"</b>----"+cvss[1]+"<br/>");//Guard
    encontrado=true;
}
```

Figura 4.5.1g. Almacenando los resultados encontrados.

En la siguiente parte del código, dependiendo de si se encontraron o no coincidencias de búsqueda en la base de datos, se mostrará el mensaje pertinente al usurario. En ambos casos se hará uso de la función "muestra()". Para ambos casos, se colocará en el textView la cadena "aux" con los resultados encontrados (en caso de no haber encontrado ninguna coincidencia, "aux" será una cadena vacía).

```
if (encontrado)//Si se encuentran coincidencias en la búsqueda,
    muestra("Resultados para \""+codigo+"\":");
else
    muestra("No se encontraron resultados para esa búsqueda.");
tv.setText(aux);
```

Figura 4.5.1h. Indicando el resultado de la búsqueda al usuario.

Como se indicó en capítulos anteriores, se precisa resaltar las coincidencias encontradas en el textView. Para cumplir este requerimiento y ubicar estas coincidencias dentro de los resultados, se comenzará haciendo uso de la clase Collator de Java. Primero se creará un objeto tipo Collator. Con este objeto se puede modificar el Locale y para localizar nuestro Collator al español, además de establecer el nivel de diferenciación a primario.

Se almacenarán los resultados de la búsqueda en una cadena temporal para trabajar con ellos. Además, se crearán otras variables auxiliares las cuales se explicarán conforme se haga uso de ellas.

```
//Resaltar coincidencias
Collator tertiaryCollator = Collator.getInstance(new Locale("es"));
tertiaryCollator.setStrength(Collator.PRIMARY);
String origString=tv.getText().toString();
int i,j;
String aux2;
String palabra;
```

Figura 4.5.1i. Iniciando el Collator y otras variables auxiliares.

En este punto, se iniciará un ciclo anidado triple para recorrer cada palabra del criterio de búsqueda ingresado por el usuario, carácter a carácter, y compararlo con los caracteres del textView para determinar las coincidencias. Este recorrido y los ciclos correspondientes se aprecian en el siguiente diagrama:

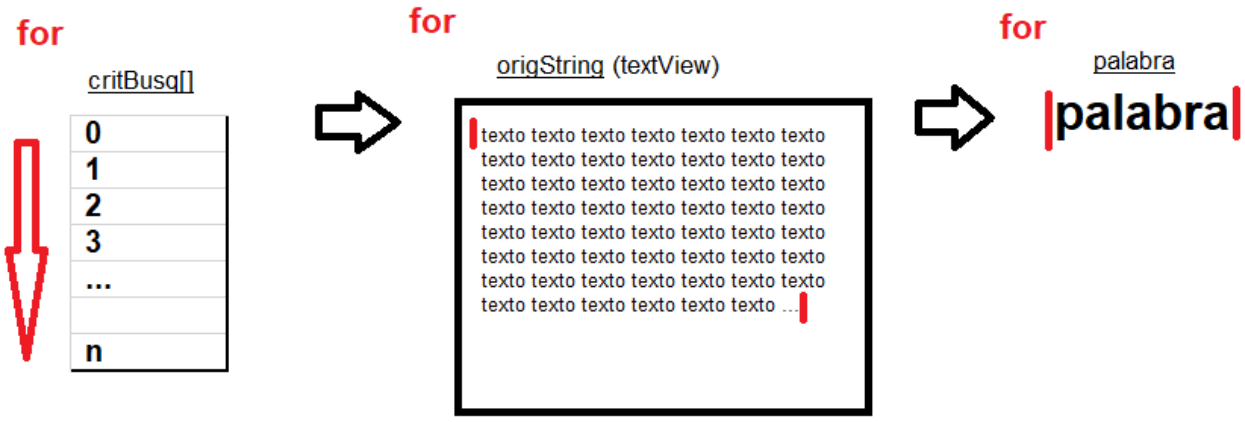

Figura 4.5.1j. Diagrama de recorrido de los ciclos anidados.

Como se indica, el primer ciclo recorrerá cada una de las posiciones del arreglo que almacena los criterios de búsqueda. en cada iteración del mismo, se almacenará el criterio en la iteración actual en la variable auxiliar "palabra".

El siguiente ciclo se encargará de recorrer el contenido del textView (previamente almacenado en la variable "orgString" con el fin de ahorrar tiempo de acceso a dicho componente). El último ciclo recorrerá carácter por carácter el criterio de búsqueda actual, es decir, la variable "palabra". Los ciclos for que hacen esto se muestran a continuación:

```
String palabra;
for(int a=0;a<critBusq.length;a++){
    palabra=critBusq[a];
    for(i=0;i<origString.length()-palabra.length();i++){
        for(j=0;j<palabra.length();j++){



        }
    }
}
```

Figura 4.5.1k. Ciclos triples anidados.

Dentro del ciclo que recorre la cadena "palabra" se realiza una comparación carácter por carácter contenido en dicha variable con el carácter del textView que se encuentra en la posición actual del ciclo que lo recorre. Si la comparación (la cual se realiza empleando el Collator) resulta en caracteres "idénticos", se entra a una sentencia vacía y el ciclo continua para comparar el siguiente carácter de "palabra". Si no hay igualdad de caracteres, se rompe la iteración actual y se salta al siguiente carácter del textView.

Una vez que el contador "j", el cual recorre la variable "palabra", llega a la última posición, se entiende que se ha encontrado una coincidencia de palabras. Se reemplaza dicha palabra encontrada dentro del texto del textView por la misma palabra, pero con unas etiquetas HTML para cambiar posteriormente dicha palabra por texto en rojo. Realizado esto, el contador del ciclo que recorre el textView salta hasta la posición del siguiente carácter después de la palabra encontrada. Los ciclos continúan hasta recorrer todo el texto dentro del textView y todas las palabras dentro del arreglo "critBusq".

```
//Aqui debe colocar
if(tertiaryCollator.compare(origString.substring(i+j,i+j+1),palabra.substring(j,j+1))==0)
    ;
else
    break;
if(j==(palabra.length()-1)){
    aux2=origString.substring(i,i+palabra.length());//se guarda la palabra encontrada
    // Reemplazar coincidencias en el text view por texto resaltado
    origString=origString.substring(0,i)+"<font color='red'>"+aux2+"</font>"+origString.substring(i+aux2.length());
    i=i+18;
}
```

Figura 4.5.1l. Comparación y sustitución de cadenas.

Concluido el recorrido de los ciclos, simplemente se procederá a establecer el texto contenido en la variable "origString" dentro del textView, pero empleando la clase Html para poder hacer uso de las etiquetas HTML añadidas. Finalmente, se reiniciarán los objetos empleados para leer la base de datos y así poder iniciar nuevas búsquedas.

```
    }
}
tv.setText(Html.fromHtml(origString));//Se muestra el text view
//Se reinicia el reader para poder iniciar otra búsqueda
inputStream.close();
reader.close();
inputStream=getResources().openRawResource(R.raw.codigos);
reader=new BufferedReader(new InputStreamReader(inputStream));
```

Figura 4.5.1m. Estableciendo texto resaltado y reiniciando el reader.

Nombre de la función: buscar()
Modificador de acceso: private
Parámetros: ninguno
Tipo de retorno: void

4.5.2 Función "limpiar()"

Apelando a la necesidad de agilizar la tarea de vaciar los contenidos de texto de la interfaz, se hará uso de la función llamada "limpiar()". Esta función será iniciada al presionar el botón "limpiar" en la aplicación.

Dentro de esta parte del código se requiere eliminar el contenido de texto tanto en el textView como en el editText, independientemente del contenido (incluso si no hay texto alguno). Para lograr esto simplemente se reemplazará el texto en ambos widgets por cadenas vacías o texto nulo. Se hará empleo entonces del método setText() de dichas clases para ambos casos.

```
    private void limpiar(){//función asociada al ev
        editText.setText("");
        tv.setText("");
        return;
```

Figura 4.5.2. Reemplazando el texto en ambos componentes por cadenas vacías.

Al no requerir ningún tipo de información o variable adicional, no se requerirán parámetros para iniciarla. Además, al no requerir regresar algún valor, tendrá un retorno tipo void.

Nombre de la función: limpiar()
Modificador de acceso: private
Parámetros: ninguno
Tipo de retorno: void

4.5.3 Función "copiaTextView()"

Como se mencionó anteriormente, se desarrollará una función que, al ser llamada por el usuario, permita recolectar el contenido actual del textView y lo traspase al portapapeles del sistema operativo del dispositivo. Cabe agregar que el texto almacenado en el portapapeles no se perderá al emplear el botón "limpiar", en caso de que una supuesta nueva búsqueda no genere resultados del interés del usuario.

Primeramente, se recolectará el texto almacenado en el momento en que se presione el textView en la aplicación. Dicho texto será convertido a un tipo de dato String, con el fin de evitar incompatibilidad de tipos de dato. Se usarán los métodos getText() y toString() respectivamente para estas tareas.

```
}rivate void copiaTextView(){
    String textoTv=tv.getText().toString();
    ClipboardManager portaPape = (ClipboardManager) getSystemServ
```

Figura 4.5.3a. Almacenando resultados en un String.

Realizado esto, se emplearán las clases necesarias para el manejo del portapapeles de Android. Se crearán dos objetos, uno ClipboardManager y otro ClipData para permitir disponer del portapapeles. Se enviará la variable "textoTv" (la cual almacena el texto recogido del textView) al objeto "clip". Éste después será

guardado en el portapapeles del sistema operativo. Esto se visualiza en el siguiente código.

```
String textoTv=tv.getText().toString();
ClipboardManager portaPape = (ClipboardManager) getSystemService(Context.CLIPBOARD_SERVICE);
ClipData clip = ClipData.newPlainText(textoTv,textoTv);
portaPape.setPrimaryClip(clip);
muestra("Resultados copiados al portapapeles ");
```

Figura 4.5.3b. Código para enviar texto al portapapeles de Android.

Finalmente, se la función llamará a la función "muestra()" (cuyo funcionamiento se explicará en el siguiente punto) para informar al usuario que el copiado de los resultados de la búsqueda han sido enviados al portapapeles.

Nombre de la función: copiaTextView()
Modificador de acceso: private
Parámetros: ninguno
Tipo de retorno: void

4.5.4 Función "muestra()"

Para mostrar los mensajes necesarios al usuario, se empleará esta función la cual simplemente iniciará un toast con un mensaje corto en la parte inferior de la pantalla. Ésta cambiará el mensaje a mostrar según el texto que reciba.

Esta función será iniciada dentro de las demás funciones cuando sea requerido mostrar algún mensaje al usuario. Al ser llamada, recibirá como parámetro una variable String con el texto que se desea mostrar. Al no requerir regresar valor alguno, la función será de retorno tipo void.

Para el uso del toast de Android, se requiere crear la instancia de clase Toast y emplear su método makeText(). Dicho método requiere como argumentos:

- El contexto de la aplicación.
- El mensaje que se desea mostar en el toast(el cual en este caso será la cadena "texto" recibida como argumento por la función "muestra()".
- La duración del toast (para este caso LENGTH_SHORT, dado el pequeño tamaño de los mensajes que se mostrarán).

Finalmente, se necesita mostrarla haciendo empleo del método show() de la misma clase. Se realizará esto en una sólo línea de código para hacerlo más legible y compacto.

```
Toast.makeText(this,texto,Toast.LENGTH_SHORT).show();
```

Figura 4.5.4. Haciendo uso de un toast para mostrar los mensajes al usuario.

Nombre de la función: muestra()
Modificador de acceso: private
Parámetros: String texto
Tipo de retorno: void

Capítulo V. Prueba e Instalación

En el siguiente capítulo se hablará sobre las pruebas de software planeadas y realizadas sobre la aplicación móvil. Dichas pruebas permitirán verificar que el software producido se desempeña de la manera que se esperaba, sin errores y cumpliendo cada uno de los requerimientos que han sido durante el análisis del problema.

Además, se describirá el modo de usar la aplicación por parte del usuario con el fin de asegurar que el mismo pueda conocer y disponer de todas las características del producto que está utilizando. Complementando esto, se indicará cómo se podrá disponer de dicha aplicación por parte de los usuarios finales.

5.1 Primera versión completa

Debido a la metodología de desarrollo similar a Scrum que se empleó, después de haber obtenido varias entregas parciales, se ha obtenido una "versión completa", es decir, una que fue desarrollada considerando cumplir todos los requerimientos solicitados (una versión similar a la entrega final). Una vez considerados estos requerimientos y corregidos los bugs que impiden compilar la aplicación, se obtuvo esta "versión completa".

Figura 5.1. Interfaz de la aplicación

Como se observa en la imagen anterior, todos los componentes de la interfaz gráfica de usuario aparecen correctamente. A continuación, se procederá a realizar las pruebas requeridas en la aplicación, para distintos dispositivos y versiones del sistema operativo empleado.

5.2 Casos de prueba

Para verificar la completa funcionalidad de la aplicación móvil, se realizarán algunas pruebas de software. Se realizará esto haciendo uso de algunos dispositivos móviles diferentes, así como de distintas versiones del sistema operativo Android.

5.2.1 Dispositivos y versiones de Android utilizadas

Primeramente, se indicará cuáles fueron los diferentes dispositivos y versiones del sistema operativo que se seleccionaron para realizar estas pruebas. Para el caso de los dispositivos, se eligió arbitrariamente entre algunos modelos de teléfonos celulares y un dispositivo tipo Tablet. Para el caso de las versiones, se tomaron

en cuenta algunas de las más populares y la versión estable más reciente (al momento de realizar las pruebas).

En la siguiente tabla se pueden observar las especificaciones de los dispositivos seleccionados para las pruebas:

Dispositivo	Marca	Modelo	Versión de Android instalada
Tablet	Acer	Iconia Tab 10 A3-A20	4.4.2 KitKat
Smartphone	Motorola	Moto E	4.4.2 KitKat
Smartphone	Huawei	Ale L23	5.0.1 Lollipop
Smartphone	Motorola	Moto G4 Plus	7.0 Nougat
Smartphone	Samsung	Galaxy S7 Flat	8.0 Oreo

Figura 5.2.1. Dispositivos y sus especificaciones.

5.2.2 Pruebas de escritorio

Para comprobar el funcionamiento de la aplicación, se ha decido por realizar algunas pruebas de escritorio en cada módulo de la misma. Esto facilitará encontrar errores dentro de la app y poder corregirlos posteriormente. Los módulos de prueba se mencionan a continuación.

5.2.2.1 Interfaz gráfica

Durante las pruebas realizadas a la interfaz gráfica de usuario, se verificó que los componentes de la misma se mostraran correctamente y en su totalidad, es decir, que los widgets no se desacomodaran o se salieran de la pantalla. En la siguiente imagen se aprecian los resultados en dos dispositivos de diferentes resoluciones, mostrados en ambas orientaciones de pantalla.

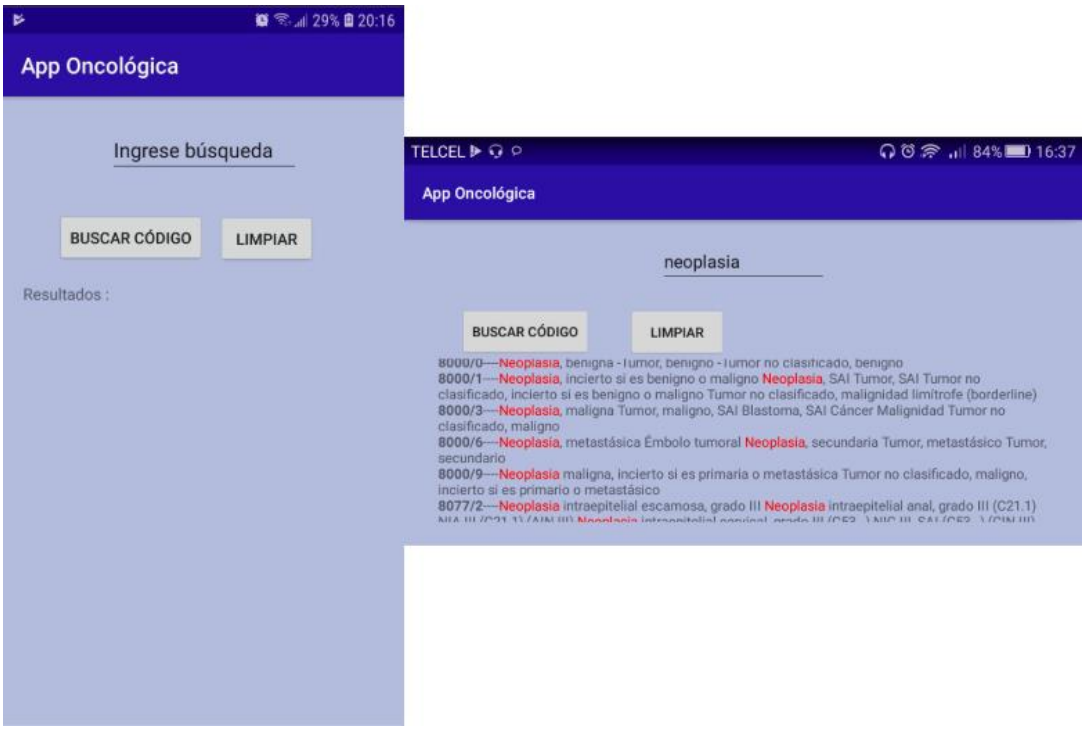

Figura 5.2.2.1. Comprobando la GUI.

Como se aprecia, los componentes no salen de la pantalla y se muestran de la manera en la que se esperaba durante el desarrollo. En el caso del textView, este se desplaza (scroll) correctamente y es completamente visible dentro de la pantalla.

Además, se observa como las coincidencias encontradas durante la búsqueda son resaltadas en rojo dentro del textView.

5.2.2.2 Botones

Para las siguientes pruebas, se comprobará que los botones cumplan con las funcionalidades que le fueron asignadas y no causen fallos en la app. El botón "limpiar" realiza correctamente el vaciado de los widgets de texto, despejando cualquier texto contenido por los mismos. Por otro lado, el botón "buscar código" inicia las consultas válidas solicitadas por el usuario, le indica si su búsqueda es inválida de ser el caso, e inicia los mensajes pertinentes mediante el toast implementado.

Nuevamente, se muestran los resultados de las pruebas realizadas en diferentes dispositivos.

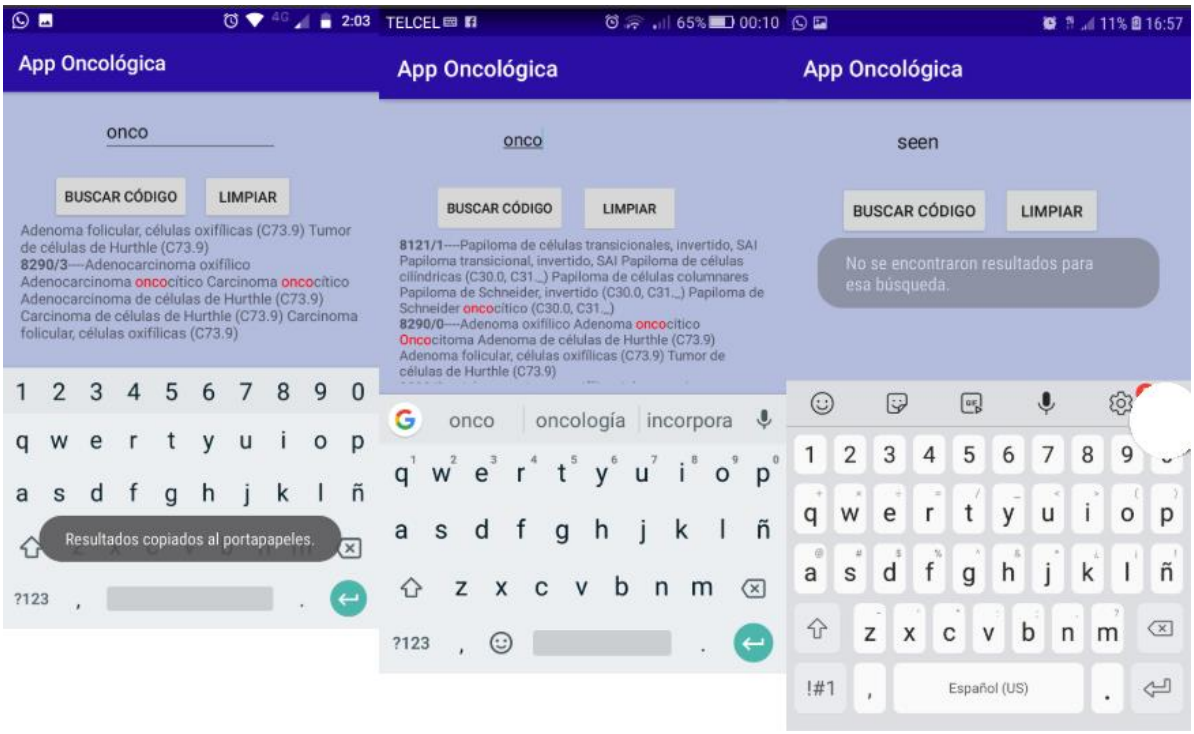

Figura 5.2.2.2. Probando los botones en la aplicación.

5.2.2.3 Entrada del editText

Como se mencionó previamente, se requiere validar la entrada ingresada por el usuario en el editText. Se verificará esta funcionalidad ingresando distintas cadenas que cumplan los distintos casos posibles.

- ✓ Búsqueda válida
- ✓ Búsqueda invalida (vacía o sin la longitud correcta)
- ✓ Búsqueda con caracteres no válidos

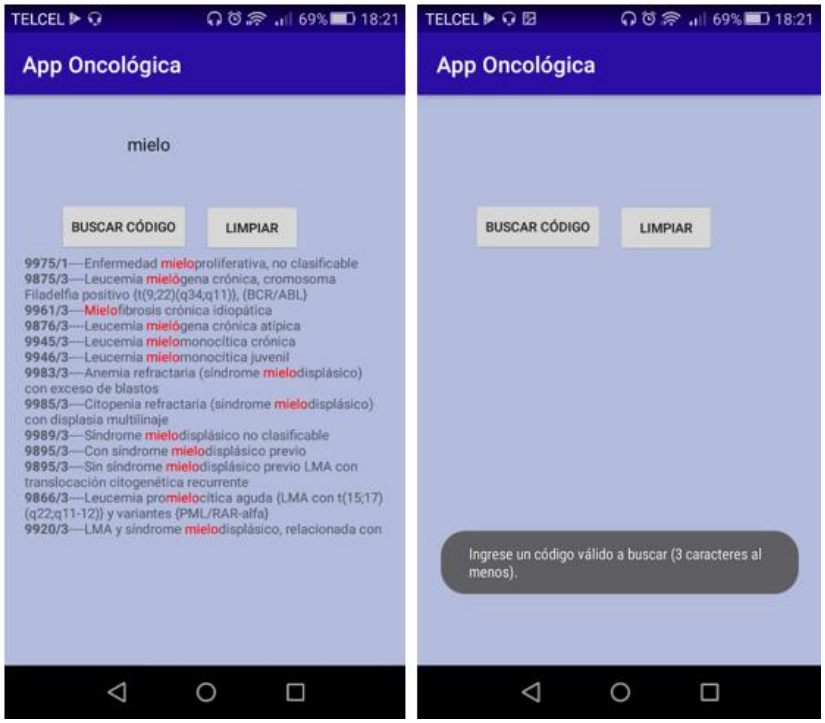

Figura 5.2.2.3. Resultados de algunas de las pruebas.

Para los tres casos se obtienen los resultados esperados. Las búsquedas válidas permiten iniciar la búsqueda con normalidad, mientras que las inválidas muestran el mensaje indicado al usuario para hacérselo notar.

Por otra parte, cuando se ingresan a la búsqueda caracteres inválidos, estos son eliminados dentro del código de la aplicación para evitar conflictos durante la ejecución del algoritmo de búsqueda.

5.2.2.4 Portapapeles

Para verificar la funcionalidad de copiado del texto almacenado en el textView al portapapeles del dispositivo, se probará con distintas longitudes de texto.

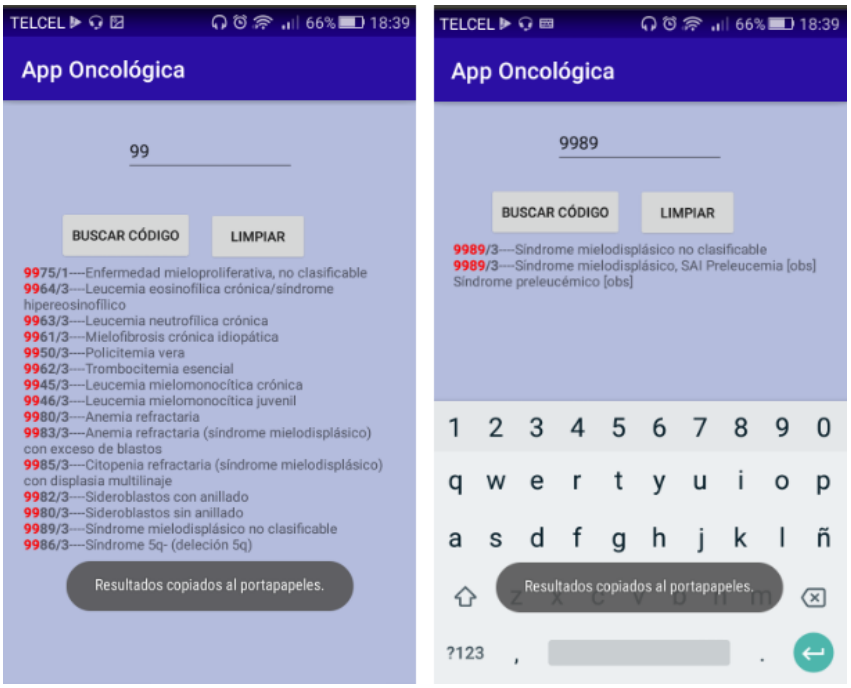

Figura 5.2.2.4a. Verificando la función de copiado al portapapeles.

El texto es copiado independientemente de su longitud, con un solo toque en el textview. El texto puede ser pegado en otras aplicaciones sin problemas como se aprecia en la imagen de abajo.

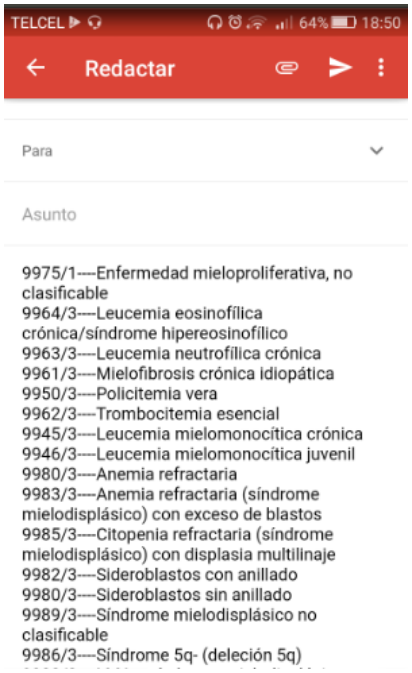

Figura 5.2.2.4b. Texto copiado a la app de Gmail.

5.3 Correcciones

A continuación, se mencionarán las correcciones realizadas a lo largo del desarrollo y prueba de la aplicación. Todas estas comprenden los errores encontrados y solucionados durante la elaboración de este software.

5.3.1 Texto perdido en el textView

Durante las pruebas realizadas se encontró que el contenido del textView se perdía al momento de realizar un cambio en la orientación de la pantalla, de horizontal a vertical o viceversa.

Dicho problema fue solucionado guardando la instancia dentro del método onCreate(). De este modo al cambiar la orientación de la pantalla, los valores son conservados.

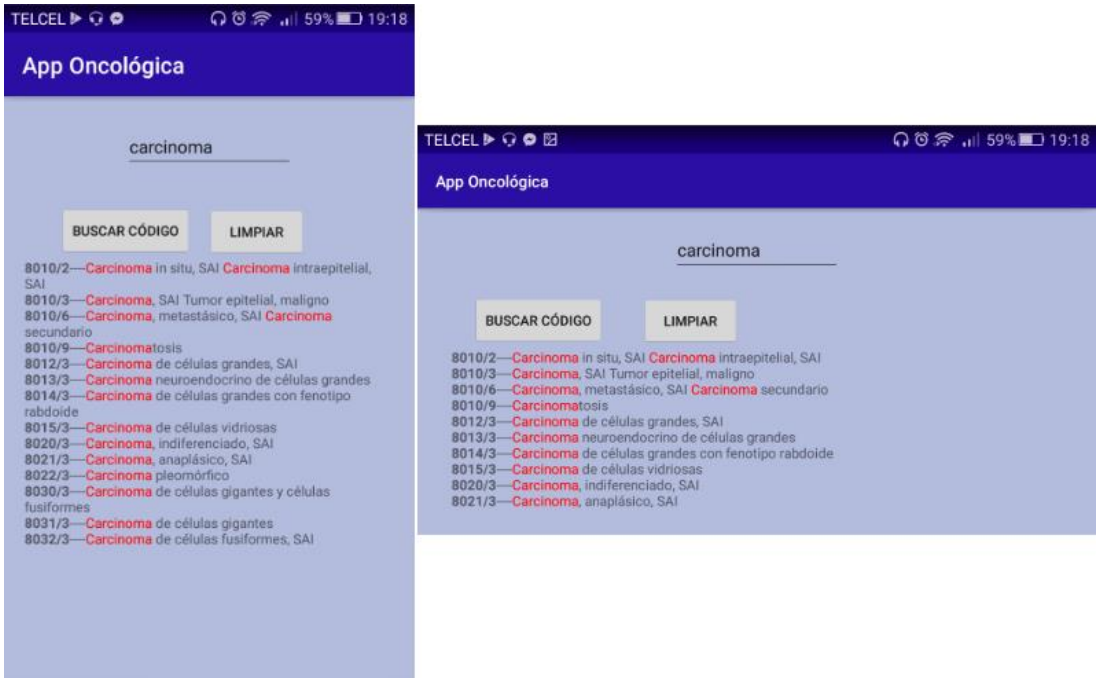

Figura 5.3.1. El textView no se pierde al cambiar rotar la pantalla.

5.3.2 Componentes fuera de pantalla

Otro de los errores encontrados durante las pruebas de la aplicación, fue el posicionamiento fuera de pantalla en algunos dispositivos con

distinta resolución, especialmente en el textView, el cual salía de la pantalla y no era completamente visible. Esto se presentó únicamente con la orientación horizontal de dichos dispositivos. Esto fue corregido ajustando los valores estáticos de los constraints dentro de los atributos de dichos widgets.

Figura 5.3.2. Los componentes no salen de pantalla.

Como se aprecia, el textView se adapta a la resolución y orientación de la pantalla, permitiendo visualizar completamente dicho componente.

5.3.3 Entradas inválidas en editText

Se detectó que algunas de las entradas ingresadas al editText, hacían que al presionar el botón "buscar código" la aplicación colapsara y se detuviera abruptamente. Dichos fallos ocurrían en ciertos casos solamente.

Se encontró que dichos "crasheos" se ocurrían al emplear ciertos caracteres los cuales, a su vez, causaban conflicto con las expresiones regulares empleadas dentro del código de la aplicación. Esto se resolvió ignorando dichos caracteres al ser leídos del editText, los cuales no aportan nada a la búsqueda.

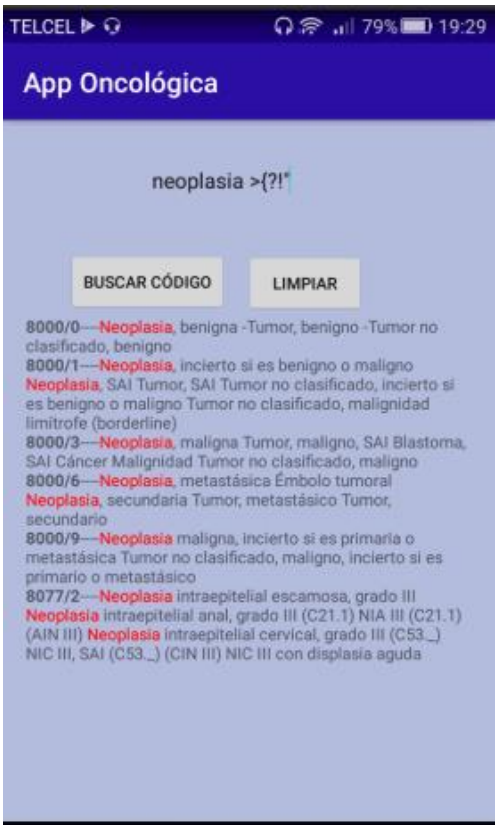

Figura 5.3.3. Caracteres no comunes no causan fallos.

5.4 Manual de usuario

En este apartado se procederá a mencionar el manual de usuario de la aplicación móvil desarrollada. Se mencionarán los requerimientos de software y hardware, la forma de instalar dicha app, y la forma de operar la misma.

5.4.1 Requerimientos

Esta aplicación requiere un dispositivo táctil portátil (tales como smartphones o tablets) que posea una versión del sistema operativo Android. Se recomienda emplear un dispositivo con una resolución de 768 x 1280 o mayor para más comodidad.

Esta app está diseñada y probada para ser compatible con las versiones de Android desde 4.4.2 "KitKat" hasta la versión 8.0 "Oreo".

5.4.2 Instalación

Desde el archivo apk:

1) Una vez localizado el archivo apk dentro de los archivos de su dispositivo, se debe abrir el mismo.

2) Se deben proporcionar los permisos del dispositivo que se soliciten.

3) Se procede a seleccionar la opción "instalar".

4) Si se muestra el mensaje de "aplicación instalada", la instalación de la app fue completada con éxito.

5) Ahora puede comenzar a utilizar su aplicación.

5.4.3 Modo de uso

Una vez iniciada la aplicación, se despliega la interfaz de usuario que se muestra a continuación:

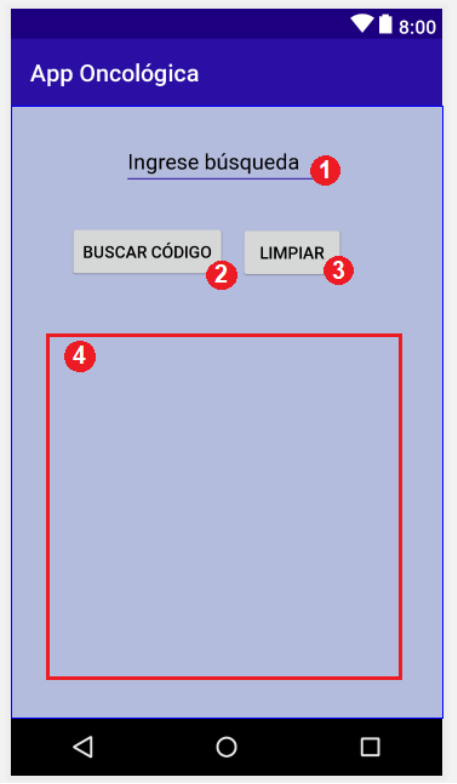

Figura 5.4.3. Interfaz de usuario.

Los elementos que la conforman son los siguientes:

1) Campo de búsqueda. En este campo usted ingresará los códigos o palabras clave que desee buscar.

2) Botón de búsqueda. Con este botón se iniciará la búsqueda de las palabras que haya ingresado en el campo de búsqueda.

3) Botón de limpiar. Este botón vaciará el contenido del campo de búsqueda y el de resultados (4) para poder realizar una nueva búsqueda desde cero.

4) Campo de resultados. En este espacio se mostrarán los resultados de la búsqueda realizada.

Para utilizar la aplicación móvil, basta con seguir los siguientes pasos:

1. Se debe ingresar en el campo de búsqueda al menos un código o palabra clave (la palabra ingresada debe contener al menos 3 caracteres). Se recomienda colocar varias palabras clave para obtener un resultado más específico.

2. Una vez ingresadas las palabras o códigos a buscar, se debe presionar el botón buscar para comenzar la búsqueda.

3. En caso de haber resultados, en el campo de resultados se desplegarán tales, resaltando en rojo las palabras que coincidan con su búsqueda. De no haber resultados, se le indicará mediante una notificación.

4. Para copiar los resultados obtenidos al portapapeles de su dispositivo, sólo debe presionar el campo de resultados y todo el texto será copiado.

5. Para iniciar una nueva búsqueda, basta presionar el botón limpiar el cual eliminará el texto en pantalla y repetir los pasos anteriores.

6. Para salir de la aplicación, solamente presione una vez el botón regresar (la flecha hacia atrás) de su dispositivo móvil.

5.5 Disponibilidad en Google Play Store

Para facilitar el acceso a esta aplicación móvil a los usuarios finales, se ha decido cargar la misma a la plataforma de distribución de aplicaciones para Android oficial de Google: Google Play Store. En esta plataforma se podrá buscar y obtener esta aplicación por parte de cualquier usuario final interesado en esta herramienta.

La aplicación estará disponible sin costo alguno en la plataforma digital y podrá ser descargada por cualquier usuario. En este mismo portal se cargarán las versiones nuevas de la app que salgan.

Capítulo VI. Resultados

En este capítulo se hablará sobre el producto final obtenido y sus resultados. Se describirá el desempeño y comportamiento de la app una vez que ha sido utilizada y probada por los usuarios finales. Se mencionará cuál fue la aceptación que tuvo por parte de dichos usuarios y la percepción en cuanto a diseño, uso intuitivo y funcionalidad por parte de los mismos.

6.1 Desempeño en los distintos dispositivos

A continuación, se describirá el comportamiento de la aplicación en los diferentes dispositivos móviles en los que fue probado como usuario (pruebas de usabilidad).

6.1.1 Interfaz de usuario

La interfaz se adaptó correctamente a los distintos dispositivos en los que se probó la versión final de la aplicación. Los botones presentaron un tamaño cómodo y de fácil alcance para los usuarios.

Más aún, el textView permitío deslizar y visualizar cada uno de los resultados de búsqueda de manera cómoda en los dispositivos de menor resolución (smartphones de pantalla pequeña). La propiedad de deslizar el texto permite que el usuario visualice fácilmente el texto sin importar la orientación o la poca resolución de su dispositivo.

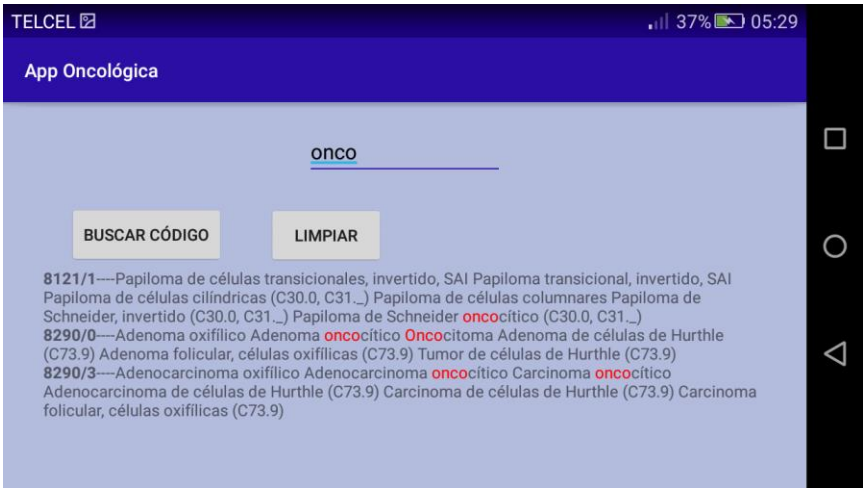

Figura 6.1.1a. Interfaz de uuario en un smartphone.

Por otra parte, las pruebas en dispositivos con mayor resolución (para estas pruebas, tablets), el textView ocupaba un mayor espacio, lo que mejoraba la visualización del texto de resultados en pantalla.

Figura 6.1.1b. Interfaz de usuario en una tablet.

6.1.2 Distinto hardware

Para observar, verificar y comparar el desempeño de la aplicación en distinto hardware, se hicieron pruebas de rendimiento en tiempo de respuesta en diferentes dispositivos móviles. Los dispositivos elegidos para estas pruebas de desempeño se muestran a continuación:

Dispositivo	Versión de Android	Procesador
Acer Iconia Tab 10 A3-A20	4.4.2 KitKat	Quad-core Mediatek MT8127 1.5GHz
MotorolaMoto E	4.4.2 KitKat	Quad-core 1.2GHz Qualcomm Snapdragon 410
HuaweiAle L23	5.0.1 Lollipop	Octa-core 1.2 GHz Cortex-A53
Samsung Galaxy S7 Flat	8.0 Oreo	Exynos 8890 Octa 2.3GHz / Snapdragon 820 2.15GHz

Figura 6.1.2a. Dispositivos empleados para las pruebas de rendimiento.

Para realizar las pruebas se decidió utilizar la funcionalidad de búsqueda al ser ésta la que más exige al procesador del dispositivo. Para disponer de un tiempo de comparación más notorio, se empleó un criterio de búsqueda que despliegue bastantes resultados y haga recorrer el archivo varias veces usando varios criterios de búsqueda. Para este caso, se ingresó como criterio de búsqueda la cadena de texto "99 99 99 99 99 99 99 99 99 99 99 99".

Los resultados de promedio de tiempos obtenidos se aprecian en la tabla siguiente.

Dispositivo	Procesador	Tiempo de respuesta promedio [s]
Acer Iconia Tab 10 A3-A20	Quad-core Mediatek MT8127 1.5GHz	3.948
MotorolaMoto E	Quad-core 1.2GHz Qualcomm Snapdragon 410	4.198
HuaweiAle L23	Octa-core 1.2 GHz Cortex-A53	2.804
Samsung Galaxy S7 Flat	Exynos 8890 Octa 2.3GHz / Snapdragon 820 2.15GHz	2.006

Figura 6.1.2b. Resultados de las pruebas.

Se puede apreciar que el tiempo de respuesta en los dispositivos gama media utilizados estuvo cerca de los 4 segundos. Mientras tanto, en un dispositivo más moderno y de procesador más potente, el tiempo disminuyó hasta los dos segundos en promedio.

Se considera aquí que el tiempo de respuesta para una búsqueda "pesada" en un dispositivo moderno es aceptable. Se espera que el tiempo de respuesta disminuya utilizando procesadores más veloces en dispositivos futuros.

6.2 Comodidad

Para los usuarios finales (los médicos que utilizaron la aplicación) la interfaz de usuario de la aplicación resultó bastante práctica y

agradable. El hecho de no requerir navegar a través de menús y tener a la vista todas las funcionalidades de la app resultó del gusto de los usuarios.

Funcionalidades como la limpieza y búsqueda resultaron bastante prácticas de realizar para los usuarios. Más aún, la función de copiado aprovecho el espacio del textView para realizar esta actividad con un solo toque en el texto sin requerir desplegar más menús o vistas en la aplicación.

Se interpreta que la interfaz gráfica de usuario fue en general cómoda y proporciona una experiencia de usuario agradable.

6.3 Usabilidad

En cuanto a la usabilidad de la app, se observó bastante intuición al momento de probar la misma. Con pocas indicaciones sobre la forma de usar la aplicación móvil, los médicos fueron capaces de hacer uso de las distintas funcionalidades disponibles en la app.

La función de los botones resultaba clara una vez utilizados por primera vez. Además, el uso de los mensajes a modo de notificaciones ayudó a guiar a los usuarios sobre que pasaba al ejecutar un comando o si había algún error al momento de usar la app.

Aunado a esto, se añade también un manual de usuario. Dicho manual complementa el modo de uso de la aplicación realizada lo cual evita ambigüedades o dudas sobre la operación del software realizado.

Figura 6.3. Vista general de la interfaz.

Se aprecia que el uso de la aplicación fue bastante intuitivo y provee una buena usabilidad para los usuarios finales. Se tiene una vez más una buena experiencia de usuario en cuanto a esta característica.

6.4 Aceptación de los usuarios finales

Se apreció que en general la aceptación hacia la aplicación móvil fue positiva. Los médicos seleccionados para las pruebas de usuario final mostraron buena aceptación con las funcionalidades y uso en general de la app.

Como se mencionó, el uso fue bastante intuitivo y no requirió demasiadas explicaciones para utilizar el software. Los usuarios no se confundieron con menús complejos o cambios de vistas que no requerirían.

Finalmente, las observaciones hechas con respecto a las primeras versiones de la aplicación que se probaron fueron corregidas en su

totalidad. Esto resultó en un producto que satisfizo las necesidades de los usuarios finales y se adaptó mejor a sus preferencias.

Conclusiones

El marco teórico elaborado ayudó a entender mejor la situación a tratar en este escrito. Ayudó a comprender las cuestiones a considerar para poder analizar y luego resolver el problema en cuestión. Más aún, facilitó la elección de las herramientas y metodologías que después se utilizaron aquí.

Después de explicar y analizar el problema planteado junto con las condiciones que se tenían, se optó como opción más viable para resolver la problemática, el desarrollo de una aplicación para dispositivos móviles bajo el sistema operativo Android. Esto se escogió como una alternativa que satisfaría cada uno de los aspectos que se consideraron en el análisis realizado.

Antes de codificar algo, y apegado a la ingeniería de software, se realizó un diseño de lo que se necesitaba programar. Se diseñó dicha aplicación logrando no solamente cumplir los requerimientos de software planteados, sino también proporcionar una buena experiencia de usuario (UX) para quienes hicieran uso de la herramienta de software con el fin de crear una aplicación más agradable.

Empleando la metodología de desarrollo ágil de software mencionada, sumado al diseño previamente realizado, se pudo desarrollar una app que satisfizo los requerimientos de una manera estructurada y ordenada. El uso del marco de desarrollo permitió trabajar adecuadamente con un proyecto del tamaño del realizado aquí el cual agregó, además, algunos requerimientos adicionales.

Concluida la codificación necesaria, se procedió a realizar las pruebas de software pertinentes para comprobar que la aplicación móvil elaborada funcionaba tal y como se esperaba. Espero logró encontrar y eventualmente depurar varios errores cometidos durante el diseño y desarrollo. Esto mostró la importancia de realizar comprobaciones en el software para mejorar la calidad del producto final.

Como se mencionó, la finalidad del diseño y desarrollo de esta aplicación móvil fue proporcionar una herramienta sencilla, portátil e intuitiva para los usuarios finales. El resultado de este trabajo fue una aplicación móvil que resultó fácil de utilizar y del agrado de los usuarios que probaron la misma. La aceptación de la app una vez realizadas las correcciones y modificaciones requeridas, fue en general positiva.

La aplicación resultante ayudará a los oncólogos a poder consultar de manera más práctica y sencilla las descripciones y códigos pertenecientes a la Clasificación Internacional de Oncología, en su

versión más reciente. Esto presenta muchas ventajas sobre la consulta convencional en libros impresos.

Sin lugar a duda, la tecnología en la computación ha sido algo que llegó para quedarse. Resulta difícil imaginarse ahora la vida sin la ayuda que todos estos avances han proporcionado a lo largo de las últimas décadas. El hecho de que cada vez más personas dispongan de algún dispositivo computacional es clara muestra de la necesidad en que se ha convertido para ayudarnos en nuestras actividades cotidianas. Aprovechar este alcance de la tecnología moderna para resolver los problemas modernos es, sin dudarlo, una labor de gran importancia para los ingenieros en la actualidad y de generaciones venideras.

Glosario

CIE:
Clasificación Internacional de Enfermedades, es el estándar de la Organización Mundial de la Salud encargado de clasificar y codificar información sobre distintas enfermedades, síntomas y causas, a nivel internacional.

CSV:
Comma-Separated Values (Valores Separados por Comas), formato de archivo para almacenar datos en forma de tablas también visible como texto plano.

IDE:
Integrated Development Enviroment (Entorno de Desarrollo Integrado), conjunto de aplicaciones y herramientas conjuntas utilizadas para el diseño, desarrollo y depuración de software.

Interfaz de usuario:
medio que permite al usuario interactuar con un dispositivo.

Localizar:
procedimiento que consiste en adecuar software a cierto idioma traduciendo texto o adaptándolo a cierto alfabeto.

Expresión regular:
secuencia de caracteres que se utilizan para describir a un lenguaje.

Responsividad:
capacidad de las aplicaciones o sitios web de adaptar sus componentes dependiendo del dispositivo en el que son visualizados.

Toast (Android):
utilidad del sistema operativo Android que permite desplegar pequeños mensajes (popups) al usuario.

Usabilidad:
expresión que describe el grado de la facilidad o intuitividad con la que se puede emplear una herramienta por una persona.

XML:
eXtensible Markup Language (Lenguaje de Marcado Extensible, es un metalenguaje de marcado usado para representar, compartir y guardar información mediante etiquetas.

Referencias

Tullis, T., Albert, B. (2013). Measuring the User Experience: Collecting, Analyzing, and Presenting Usability Metrics (2a ed.). Massachussetts: Elsevier Inc.

Mendoza, A. (2014). Mobile User Experience: Patterns to Make Sense of it All. Massachussetts: Elsevier Inc.

Fritz, A., Percy, C., Jack, A., Shanmugaratnam, K., Sobin, L., Perkin, M., Whelan, S. (2003). Clasificación Internacional de Enfermedades para Oncología. (3a ed.) Washington, D.C.: Organización Mundial de la Salud.

NO INDICA (NO INDICA). Algunas Tecnologías de la Informática. Facultad de Ingeniería, Universidad Nacional Autónoma de México. Recuperado el 28 de marzo de 2018, de http://profesores.fi-b.unam.mx/heriolg/Infdist9.pdf

Gonzáles, Gabriela (2016). Finalmente pasó: el uso de Internet en todo el mundo es mayor en móviles y tablets. Madrid, España. Recuperado el 7 de marzo de 2018, de https://www.genbeta.com/actualidad/finalmente-paso-el-uso-de-internet-en-todo-el-mundo-es-mayor-en-moviles-y-tablets

NO INDICA (NO INDICA). Sistemas Operativos. Universidad Nacional Autónoma de México. Recuperado el 5 de marzo de 2018, de http://www.bunam.unam.mx/intComputacion/Unidad_2/c02u2t02p01.html

NO INDICA (NO INDICA). Sociedad Mexicana de Oncología, A.C. SMEO. Recuperado el 5 de marzo de 2018, de http://www.smeo.org.mx

NO INDICA (NO INDICA) Class Collator. Java Platform Standard Ed. Recuperado el 5 de marzo de 2018, de https://docs.oracle.com/javase/8/docs/api/java/text/Collator.html

NO INDICA (2018). Diseños. Android Developers. Recuperado el 8 de mayo de 2018, de https://developer.android.com/guide/topics/ui/declaring-layout?hl=es

NO INDICA (2018). ConstraintLayout. Android Developers. Recuperado el 15 de julio de 2018, de https://developer.android.com/reference/android/support/constraint/ConstraintLayout

Macedo Reza, Yasmine (NO INDICA). Arquitecturas cliente/servidor. Facultad de Ingeniería, Universidad Nacional Autónoma de México. Recuperado el 5 de marzo de 2018, de http://profesores.fi-b.unam.mx/yasmine/tema1_1.pdf

Garret, Jesse James (2000). The Elements of User Experience. UX Design. Recuperado el 22 de marzo de 2018, de http://uxdesign.com/assets/Elements-of-User-Experience.pdf

Sterling, Greg (2012). Analyst: Mobile To Overtake PC For Local Search By 2015. Search Engine Land. Recuperado el 18 de marzo de 2018, de https://searchengineland.com/analyst-mobile-to-overtake-pc-for-local-search-by-2015-119148

Mendiola Zuriarrain, José (2017). Android ya es el sistema operativo más usado del mundo. España. Recuperado el 16 de abril de 2018, de https://elpais.com/tecnologia/2017/04/04/actualidad/1491296467_396232.html

NO INDICA (NO INDICA). Ingeniería de Software, Análisis y Diseño. Universidad de las Américas Puebla. Recuperado el 22 de abril de 2018, de http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/fuentes_k_jf/capitulo2.pdf

NO INDICA (2015). International Classification of Diseases for Oncology, 3rd Edition (ICD-O-3). World Health Organization. Recuperado el 23 de abril de 2018, de http://www.who.int/classifications/icd/adaptations/oncology/en/

Minato, Charlie (2012). ComScore: Mobile Will Force Desktop Into Its Twilight In 2014. Australia. Recuperado 7 de marzo de 2018, de https://www.businessinsider.com.au/mobile-will-eclipse-desktop-by-2014-2012-6

Pastrana, Ophelia (2014). 5 beneficios de aplicar metodologías ágiles en el desarrollo de software. Intelligence to Business. Recuperado el 15 de mayo de 2018, de http://www.i2btech.com/blog-i2b/tech-deployment/5-beneficios-de-aplicar-metodologias-agiles-en-el-desarrollo-de-software/

NO INDICA. (2018). Mobile Operating System Market Share Worldwide. Statcounter: GlobalStats. Recuperado el 15 de mayo de 2018, de http://gs.statcounter.com/os-market-share/mobile/worldwide#monthly-200908-201810

Anexos

Código fuente:

```
package com.example.eloy.appmedica;

import android.content.ClipData;
import android.content.ClipboardManager;
import android.content.Context;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.text.Html;
import android.text.method.ScrollingMovementMethod;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.text.Collator;
import java.text.Normalizer;
import java.util.Locale;

public class MainActivity extends AppCompatActivity {
    /*Variables globales*/
    TextView tv;
    EditText editText;
    InputStream inputStream;
    BufferedReader reader;
    String[] cvss;
    @Override
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        editText=(EditText) findViewById(R.id.editText2);
        Button button=(Button)findViewById(R.id.button);
        Button botonLimpiar=(Button)findViewById(R.id.button2);
        inputStream=getResources().openRawResource(R.raw.codigos);
        reader=new BufferedReader(new InputStreamReader(inputStream));
        tv=(TextView)findViewById(R.id.textView);
        tv.setMovementMethod(new ScrollingMovementMethod());
        //Eventos de click a los botones
        button.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v) {
                buscar();
            }
        }
        );
        botonLimpiar.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v){
                limpiar();
            }
        }
        );
        tv.setOnClickListener(new View.OnClickListener(){
            @Override
            public void onClick(View v){
                copiaTextView();
            }
        }
        );
    }
```

```
    private void buscar(){//Función asociada al evento de click al botón "buscar código"
        String codigo=editText.getText().toString();//Criterios de búsqueda del usuario
        boolean encontrado=false;//Booleano que indica si se encontraron coincidencias de
búsqueda
        if(codigo.equals("")||codigo.length()<2){//si no hay suficientes criterioa de
búsqueda
            tv.setText("");
            muestra("Ingrese un código válido a buscar (3 caracteres al menos).");
            return;
        }
        try{//reader para csv
            //Reemplazar caracteres (),{},[],*,¿? para evitar crasheos!!!
            String lineaCsv;
            String aux="";
            tv.setText("");
            String [] critBusq=codigo.split(" ");//Separar en palabras el criterio de
búsqueda
            while((lineaCsv=reader.readLine())!=null){
                cvss=lineaCsv.split("%");//separa códigos y descripciones //Se puede mover
esto para no hacerlo cuando no se requiera
                try{
                    for(int i=0;i<critBusq.length;i++){
                        String normalizedAux=Normalizer.normalize(lineaCsv,
Normalizer.Form.NFD);//Cadenas auxiliares para normalizar las cadenas de búsqueda
                        String
normalizedAux2=Normalizer.normalize(critBusq[i],Normalizer.Form.NFD);
                        String lineaCsvNorm=normalizedAux.replaceAll("[^\\p{ASCII}]", "");
                        String critBusqNorm=normalizedAux2.replaceAll("[^\\p{ASCII}]", "");
                        if(lineaCsvNorm.toLowerCase().contains(critBusqNorm.toLowerCase()))
                            ;
                        else
                            break;
                        if(i==critBusq.length-1){
                            aux=aux.concat("<b>"+cvss[0]+"</b>----
"+cvss[1]+"<br/>");//Guardar resultados obtenidos en un variable y cargar al textView al
final
                            encontrado=true;
                        }
                    }
                }catch(Exception e) {
                    tv.setText("Error equisde :s :" + e.toString());//Nunca debe entrar aquí
(en teoría :P)
                }
            }
            if (encontrado)//Si se encuentran coincidencias en la búsqueda, despliega
resultados
                muestra("Resultados para \""+codigo+"\":");
            else
                muestra("No se encontraron resultados para esa búsqueda.");
            tv.setText(aux);
            //Resaltar coincidencias
            Collator tertiaryCollator = Collator.getInstance(new Locale("es"));//Collator
para signos diacríticos y mayúsculas/minúsculas
            tertiaryCollator.setStrength(Collator.PRIMARY);
            String origString=tv.getText().toString();
            int i,j;
            String aux2;
            String palabra;
            for(int a=0;a<critBusq.length;a++){
                palabra=critBusq[a];
                for(i=0;i<origString.length()-palabra.length();i++){
                    for(j=0;j<palabra.length();j++){
                        //Aquí usar collator

if(tertiaryCollator.compare(origString.substring(i+j,i+j+1),palabra.substring(j,j+1))==0)
                            ;
                        else
                            break;
                        if(j==(palabra.length()-1)){
                            aux2=origString.substring(i,i+palabra.length());//se guarda la
palabra encontrada
```

```
                            // Reemplazar coincidencias en el text view por texto resaltado
                            origString=origString.substring(0,i)+"<font
color='red'>"+aux2+"</font>"+origString.substring(i+aux2.length());
                            i=i+18;
                        }
                    }
                }
            }
            tv.setText(Html.fromHtml(origString));//Se muestra el text view con formato html
para resaltar coincidencias
            //Se reinicia el reader para poder iniciar otra búsqueda
            inputStream.close();
            reader.close();
            inputStream=getResources().openRawResource(R.raw.codigos);
            reader=new BufferedReader(new InputStreamReader(inputStream));
        }
        catch(IOException ex){//Excepción en lectura del archivo
            throw new RuntimeException("Error de lectura: "+ex);
        }
        return;
    }
    private void limpiar(){//Función asociada al evento de click al botón limpiar
        editText.setText("");
        tv.setText("");
        return;
    }
    private void copiaTextView(){
        String textoTv=tv.getText().toString();
        ClipboardManager portaPape = (ClipboardManager)
getSystemService(Context.CLIPBOARD_SERVICE);
        ClipData clip = ClipData.newPlainText(textoTv,textoTv);
        portaPape.setPrimaryClip(clip);
        muestra("Resultados copiados al portapapeles.");
        return;
    }
    private void muestra(String texto){//Toast para mostar mensajes al usuario
        Toast.makeText(this,texto,Toast.LENGTH_SHORT).show();
        return;
    }
}
```