CARRERA DE ANÁLISIS DE SISTEMAS

TEMA:

"DESARROLLO DE UNA APP MÓVIL PARA PACIENTES EN TRATAMIENTO DE HEMODIALISIS DE LA CIUDAD DE CUENCA, QUE CONTROLE LOS ELECTROLITOS EN EL CONSUMO DE SUS DIETAS"

AUTOR:

Cristian Leonardo Ochoa Matute

Christian Olmedo Regalado Sarmiento

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

TECNÓLOGO EN ANÁLISIS DE SISTEMAS

TUTORES:

Ing. Juan Pablo Hurtado.

CUENCA – ECUADOR, 2019

INSTITUTO DE TECNOLOGIAS SUDAMERICANO

www.sudamericano.edu.ec

DECLARACIÓN DE AUTORÍA DEL TRABAJO

Yo, **REGALADO SARMIENTO CHRISTIAN OLMEDO**, estudiante del **Instituto Tecnológico Superior Particular Sudamericano** de la ciudad de Cuenca - Ecuador, que cursó la Tecnología en **ANALISIS EN SISTEMAS,** declaro en forma libre y voluntaria que la presente investigación que versa sobre **"DESARROLLO DE UNA APP MÓVIL PARA PACIENTES EN TRATAMIENTO DE HEMODIALISIS DE LA CIUDAD DE CUENCA, QUE CONTROLE LOS ELECTROLITOS EN EL CONSUMO DE SUS DIETAS"** así como las expresiones vertidas en la misma, son autoría de la compareciente, quien ha realizado en base a recopilación bibliográfica, consultas de internet y consultas de campo.

En consecuencia, asumo la responsabilidad de la originalidad de la misma y el cuidado al remitirme a las fuentes bibliográficas respectivas para fundamentar el contenido expuesto.

Atentamente,

REGALADO SARMIENTO CHRISTIAN OLMEDO

Cédula: 010424422-3

DERECHOS DE AUTOR

INSTITUTO DE TECNOLOGIAS SUDAMERICANO

www.sudamericano.edu.ec

DECLARACIÓN DE AUTORÍA DEL TRABAJO

Yo, **OCHOA MATUTE CRISTIAN LEONARDO**, estudiante del **Instituto Tecnológico Superior Particular Sudamericano** de la ciudad de Cuenca - Ecuador, que cursó la Tecnología en **ANALISIS EN SISTEMAS,** declaro en forma libre y voluntaria que la presente investigación que versa sobre **"DESARROLLO DE UNA APP MÓVIL PARA PACIENTES EN TRATAMIENTO DE HEMODIALISIS DE LA CIUDAD DE CUENCA, QUE CONTROLE LOS ELECTROLITOS EN EL CONSUMO DE SUS DIETAS"** así como las expresiones vertidas en la misma, son autoría de la compareciente, quien ha realizado en base a recopilación bibliográfica, consultas de internet y consultas de campo.

En consecuencia, asumo la responsabilidad de la originalidad de la misma y el cuidado al remitirme a las fuentes bibliográficas respectivas para fundamentar el contenido expuesto.

Atentamente,

OCHOA MATUTE CRISTIAN LEONARDO

Cédula: 010418083-1

DERECHOS DE AUTOR

RESUMEN

Este proyecto de tesis consiste en “DESARROLLO DE UNA APP MÓVIL PARA PACIENTES EN TRATAMIENTO DE HEMODIALISIS DE LA CIUDAD DE CUENCA, QUE CONTROLE LOS ELECTROLITOS EN EL CONSUMO DE SUS DIETAS”

Lo que ayudará para que un paciente pueda llevar un mejor control de la ingesta de todos los alimentos asignados por el nutricionista en su dieta diaria, también le ayudará al médico o tratante a conocer de una manera más rápida todo lo que el paciente consume diariamente en alimentos, ya que de una forma automatizada y gráfica la aplicación le entregará todos esos reportes de lo que está comiendo el paciente diariamente.

Las herramientas utilizadas en el desarrollo son: para la base de datos se implementó con FIREBASE, para la programación de la app se uso IONIC conjuntamente con diversos repositorios de iconos, cómo bibliotecas gratuitas para crear los gráficos, manejos de fechas, etc.

ABSTRACT

SUMMARY

This thesis project consists in “DEVELOPMENT OF A MOBILE APP FOR PATIENTS IN TREATMENT OF HEMODIALISIS OF THE CITY OF CUENCA, THAT CONTROLLED ELECTROLYTES IN THE CONSUMPTION OF THEIR DIETS”

What we help a patient to carry out a better control of the intake of all the foods assigned by the nutritionist in his daily diet, we also help the doctor or treating person to know more quickly everything that the patient consumes daily food, since in an automated and graphic way the application will deliver all those reports of what the patient is eating daily.

The tools used in the development are: for the database it is implemented with FIREBASE, for the programming of the application IONIC modifications were used with various icon repositories, free libraries to create the graphics, date handling, etc.

AGRADECIMIENTO

Doy las gracias a mi Padre por permanecer a mi lado siempre!, no hay palabras que definan lo que significa para mí, siempre ha estado en todo momento apoyándome en todos mis proyectos propuestos.

A mis compañeros también por haber estado siempre a mi lado apoyándonos mutuamente para conseguir y cumplir todos nuestros objetivos y metas.

A mi hermano Juan Fernando por su apoyo y cariño incondicional.

Christian Regalado

AGRADECIMIENTO

Quiero empezar dando gracias a Dios por darme la vida y sabiduría para estar cumpliendo esta meta.

También agradecer el apoyo incondicional de mi madre quien supo guiarme en todos mis proyectos a mi hermana Mónica quien fue el pilar fundamental para lograr esta meta y a mi hermana Daysi y esposo quienes me apoyaron diariamente para salir adelante.

Cristian Ochoa.

DEDICATORIA

La presente tesis la dedico a mi padre y a mis hermanos por haber estado siempre a mi lado apoyándome para cumplir mis objetivos y conseguir el éxito.

Christian Regalado.

DEDICATORIA

Esta tesis está dedicada a mi padre que desde el cielo me ha guiado y cuidado en el trayecto de mi vida, también a mis compañeros de diálisis (Guerreros) quien día a día luchan y demuestran que si hay segunda oportunidad en esta vida.

Cristian Ochoa.

Tabla de contenido

INTRODUCCIÓN:

Se crea una app para controlar la ingesta de alimentos para personas en tratamiento de hemodiálisis, el usuario o paciente puede escoger un Doctor o tratante para que le haga el debido seguimiento de lo que va consumiendo diariamente en su dieta propuesta por el nutricionista , el paciente si desea puede escoger un doctor o simplemente puede hacer uso de la aplicación sin tener un tratante ya que la aplicación le visualizará de una manera gráfica los datos de los alimentos que va a consumir y le indicará con gráficas que valores tanto en potasio ,sodio, fosforo y carbohidratos ha consumido diariamente, de ésta manera el paciente sabrá en valores aproximados en que se está excediendo .

El paciente podrá escoger una dieta desacuerdo a cada comida esta será guardada con fecha para llevar los debidos registros diarios o mensuales, la aplicación tendrá información de la enfermedad.

NOTA.

Los alimentos están basados en porciones, que se calcula haciendo una regla de tres siendo la unidad de medida 100mg.

CAPÍTULO I.-

1. PLANTEAMIENTO DEL PROBLEMA:

Un paciente en Diálisis tiene un control de su dieta con un especialista en su centro de diálisis el mismo que es valorado una vez al mes y es enviado con las indicaciones que requiera el paciente de acuerdo con los resultados de los exámenes mensuales, pero un paciente cuando esta fuera del centro de diálisis no puede llevar un control en su dieta ya que el paciente sabe que alimentos no debe ingerir por no conoce los valores de los minerales que contienen cada alimento esto ocasiona que esos valores se eleven y el paciente muestre deterioros en su salud que puede ocasionar hasta la muerte del paciente esto en los perores de los casos, otros problemas pueden ser osteoporosis, presión arterial no controlada entre otras.

En estos tiempos que la tecnología (Apps Móvil) ha avanzado a gran escala no es posible que los pacientes que sufre este tipo de enfermedad no cuenten con la ayuda de la misma ya que con una revisión en las plataformas de descargas de Apps Móviles no hemos encontrado una App Móvil que ayude en este tipo de enfermedad.

1.1.SOLUCIÓN:

Crear una aplicación que va a ayudar al usuario a llevar dicho control de la ingesta de sus alimentos en la dieta entregada por su nutricionista.

A sí mismo la aplicación podrá ser usada por el Doctor o tratante para observar de una manera instantánea todo lo que el paciente ha consumido en las fechas solicitadas, facilitando el control de las dietas a los usuarios.

2. JUSTIFICACIÓN:

La creación de la App Móvil ayudará a controlar la ingesta de alimentos a pacientes con Insuficiencia Renal, esta App tendrá la función de mostrar la cantidad de (mg.) por porción de alimentos que el paciente haya ingerido, estos valores se sumaran de acuerdo como el paciente vaya agregando a su dieta dicho alimento, con estos valores el paciente podrá llevar un control de su alimentación más cercano a lo real, en caso de que los valores ingeridos sean mayores a los valores que debe llevar su dieta la App enviara una notificación al paciente que le indicara que sus valores están elevados y que tiene que cuidarse, los valores podrán ser guardados diariamente estos servirán para llevar un control semanal, quincenal o mensual de su dieta.

La App brindara también información sobre el tratamiento de la Insuficiencia Renal y Nutrición para los pacientes.

3. OBJETIVO GENERAL

Desarrollar una App Móvil que ayude a controlar la ingesta de alimentos en una dieta para pacientes con Insuficiencia Renal.

4. OBJETIVOS ESPECIFICOS

Mediante la App Móvil se pretende:

1. Mostrar los alimentos en la App Móvil con sus respectivos valores de electrolitos (Potasio, Fosforo, Sodio, Carbohidratos).
2. Almacenar los valores ingeridos por el paciente en una base de datos (FireBase) los cuales nos ayudaran a generar los resultados de la alimentación del paciente por día.
3. Emitir alertas cuando los valores superen a los que un paciente en diálisis debe tener.
4. Guardar los valores de ingesta diarios para un control semanal, quincenal o mensual.

CAPÍTULO II.-
MARCO TEORICO

5. **HTML5**

HTML5 nos permite crear documentos WEB (páginas web) en un lenguaje basado en etiquetas de una manera que los diferentes navegadores puedan entender el contenido y mostrar al cliente o usuario, teniendo en cuenta que: (Fernández, 2016)

"HTML5 (HyperText Markup Language) HTML5 es la quinta revisión (mayor) de este estándar. Las principales novedades que trae son nuevas etiquetas para conseguir la Web Semántica (que los elementos o etiquetas aporten significado y no solo contenido) y nuevas APIs para permitir funcionalidades avanzadas de Javascript. (Fernández, 2016)"

6.1 **FIREBASE**

Firebase es una gran herramienta de GOOGLE que nos ayuda en el desarrollo de apps , facilitando la creación o desarrollo de aplicaciones WEB o Móviles , se usa en las plataformas móviles cómo Android , IOS o para la Web, proveyéndonos diferentes ventajas y beneficios para crear de una forma efectiva nuestras aplicaciones y mejorando nuestro negocio, con el uso de Firebase podemos : (Giraldo, 2019)

"-lograr una integración

-lograr una integración dinámica de los usuarios usando Firebase Authentication;

-lograr que nuestras aplicaciones sean visualizadas y utilizadas utilizando la herramienta de compartir o Dynamic Links;

-enviar notificaciones a varias plataformas con Cloud Messaging;

-crear análisis de resultados con Analytics;

-Ahora bien, te voy a dar dos ejemplos sobre lo que podemos hacer con Firebase:

-A la hora de implementar funciones, pueden ser probadas con antelación en un subconjunto de la base de usuarios para verificar si funciona y cómo estos reaccionan ante esta.

-Comunícate con tus usuarios a través de In-app messaging, esto se utiliza para mejorar la experiencia del usuario y fidelizarlo con la app, además, puedes guiarlos y así convertirlos en potenciales clientes

Para hacer crecer nuestras aplicaciones y con esto, nuestro negocio, nos da la posibilidad de:

-enviar notificaciones a los usuarios, con Notifications;

-permite mostrar las aplicaciones de una forma bastante adecuada en los resultados de los motores de búsqueda, con el uso de App indexing;

-proporciona la forma para acceder a la aplicación desde otros links desde otras aplicaciones, mediante Dynamic Links;

-permite hacer publicidad de nuestra aplicación usando AdWords;

-ayuda a monetizar nuestra aplicación mediante la publicidad, utilizando AdMob.” (Giraldo, 2019).

6.2 **CSS**

Es un lenguaje enfocado a definir, construir y mejorar la apariencia de un documento basado en HTML, es decir mejoramos el diseño y apariencia de un documento web, Las siglas **CSS** (*Cascading Style Sheets*) significan «Hojas de estilo en cascada» y parten de un punto primordial: darle estilos (colores ,

diseños , márgenes ,etc…) a los documentos generalmente HTML, algunas reglas, funcionalidades y ventajas de usar CSS: (Rodríguez, 2019)

– Un conjunto de propiedades con valores predeterminados que permitan modificar la presentación del contenido HTML. Ejemplo: quiero que junto al título haya un banner y que tenga un fondo negro.

–También es necesario un selector que se encargará de elegir los elementos afectados por el nuevo valor de la propiedad. Ejemplo: quiero que mi regla CSS aplique para todos los encabezados de mi página.

Se les llama estilos en cascada porque se aplican de arriba a abajo siguiendo un patrón al que se le denomina herencia. En caso de que exista ambigüedad se utilizan una serie de reglas como las que mencionamos arriba.

El CSS se suele trabajar con una separación de presentación y contenido, de esta forma los datos HTML solo incluirán información y datos que se refieren al significado de la información a transmitir. Este proceso se lleva a cabo en dos fases:

1. El navegador convierte HTML y CSS en un DOM (Objeto Documento Modelo) este DOM funciona como un documento para la memoria del ordenador, de forma que puede combinar el contenido del documento con su estilo.

2. El navegador muestra el contenido del DOM.

Ventajas:

– Unificar todo lo referente al diseño visual en un solo documento

– Se pueden hacer modificaciones en un solo lugar sin tener que recurrir a los archivos HTML por separado

– Es menor la probabilidad de que exista duplicación de estilos en diferentes lugares, debido a esto es más fácil de organizar y hacer cambios. A esto debemos añadir que la información a transmitir es considerablemente menor y por tanto las páginas también se descargan más rápido

– La creación de versiones para otros dispositivos: tablets o smartphones se simplifica.” (Rodríguez, 2019)

6.3 **TYPESCRIPT**

Typescript es un lenguaje de Alto Nivel que tiene integrado las estructuras más usadas en la programación orientada a objetos, extrayendo una gran funcionalidad y beneficio a la hora de crear un programa muy grande, quedando escalable cuando queramos aumentar su funcionalidad. Typescript se compila sobre Javascript nativo, pudiendo ser usado en cualquier programa proyecto donde se programe o use Javascript, teniendo en cuenta lo siguiente: (Guerra E. F., 2016)

“TypeScript es lo que se conoce como un "superset" de Javascript, aportando herramientas avanzadas para la programación que traen grandes beneficios a los proyectos.

Qué es un Superset

Es un lenguaje escrito encima de otro lenguaje o mejor dicho, que compila a otro lenguaje. En el caso de TypeScript es un lenguaje que compila a JavaScript, pero que incluye muchas facilidades y ventajas.

Los lenguajes como JavaScript basados en un estándar evolucionan muchas veces más lento que las necesidades de los desarrolladores. Entonces surgen empresas o comunidades que deciden expandir un lenguaje, aportando todas las herramientas que se echan en falta para poder desarrollar con las mejores condiciones.

Los superset compilan en el lenguaje estándar, por lo que el desarrollador programa en aquel lenguaje expandido, pero luego su código es "transpilado" para transformarlo en el lenguaje estándar, capaz de ser entendido en todas las plataformas.

En concreto TypeScript nos ofrece muchas de las utilidades que se necesitan en JavaScript para poder convertirlo en un lenguaje escalable, a la altura de las necesidades más exigentes. TypeScript nos ofrece muchas de las cosas que los desarrolladores de lenguajes más tradicionales vienen usando en su día a día.

Qué necesitamos para usar TypeScript
Básicamente necesitamos descargar dos programas. El primero es NodeJS, no porque necesitemos desarrollar con Node, sino porque el compilador de TypeScript está desarrollado en NodeJS.

Desde el sitio de NodeJS encontramos las opciones para la instalación en nuestro sistema operativo, por medio de un típico instalador con su asistente (siguiente, siguiente…) Más información en el Manual de NodeJS.

Luego necesitarás el TSC (Command-line TypeScript Compiler), la herramienta que nos permite compilar un archivo TypeScript a Javascript nativo. Este software es el que está realizado con NodeJS y su instalación se realiza vía npm con el siguiente comando:
npm install -g typescript" (Guerra E. F., 2016)

6.4 **VISUAL STUDIO CODE**

Es un editor de código multiplataforma utilizable para Windows, MacOS y Linux , posee soporte nativo para Node.js , JavaScript y TypeScript , éste editor muy potente de código puede trabajar con casi cualquier Lenguaje de Programación , Python, Java , C#, etc, algunas generalidades sobre éste espectacular editor de código : (Atareao, 2018)
"EL INTERFAZ DE USUARIO
La interfaz de usuario es muy similar a la de otros editores de código, como puede ser Sublime Text, Atom y similares. Así, como en estos otros editores, la disposición de paneles es parecida. A la izquierda encontrarás un editor de archivos. Mientras que a la derecha, tienes el contenido del archivo que estés editando en ese momento.

Básicamente la interfaz de usuario se divide en cinco áreas,
Editor. Como te puedes imaginar es la parte central y mas importante, y donde editas los archivos. Esta parte central permite hasta tres editores uno al lado de otro. Para abrir un archivo en un segundo o tercer editor, tienes que hacer clic a la vez que pulsas Alt en el explorador de archivos.

Panel lateral. Aquí se emplazan diferentes vistas, como puede ser el explorador de archivos, que te permite trabajar fácilmente con los ficheros de tu proyecto. En el caso del control de versiones, aquí encontrarás aquellos archivos del proyecto que tienen algún cambio con respecto al último commit. También encorarás la vista de búsqueda, depuración y extensiones. Algunas extensiones pueden añadir aquí otras vistas personalizadas.

Barra de estado. Es la barra que se sitúa en la parte inferior de la aplicación. Aquí se muestra diferente información realmente útil. Desde el número de líneas y caracteres del archivo que estas editando, a información relativa con el control de versiones.
Panel de actividades. Es el panel o barra que se encuentra en la parte izquierda de la aplicación.
En este panel, encontrarás cinco botones que te permiten cambiar entre las diferentes vistas. Explorador, búsqueda, control de versiones, depuración y extensiones.
Estas vistas son las que aparecen en el propio panel lateral, mientras que el editor permanecerá inalterado.

Paneles. En la parte inferior del editor se pueden mostrar diferentes paneles donde se recogen información de depuración, errores, avisos o un terminal integrado. Esta posición no es estática, sino que puedes desplazar estos paneles a la derecha para tener más espacio vertical.

Una interesante característica de la aplicación es que si modificas la disposición de las áreas, está queda guardada. De esta forma, la próxima vez que inicies Visual Studio Code, la disposición será igual a la disposición con que cerraste la aplicación.” (Atareao, 2018).

6.5 **ECMASCRIPT(ES6,ES7)**

Ecmascript es Javascript en sus últimas versiones y mejorado, cada año le agregan mejores características, cómo pueden ser funciones nuevas, uso de

clases, uso de nuevas variables, uso de métodos, nuevas formas de asignar a Arrays y a objetos, teniendo en cuenta los siguientes aspectos cómo su evolución y diferentes mejoras que van sacando a medida que pasa el tiempo: (Azaustre, 2018)

“La evolución de JavaScript:

Primero un poco de historia. En 1995 (hace más de 20 años!) Brendan Eich crea un lenguaje llamado **Mocha** cuando trabajaba en **Netscape**. En Septiembre de ese año lo renombra a *LiveScript* hasta que le cambiaron el nombre a *JavaScript* debido a una estrategia de marketing, ya que Netscape fue adquirida por *Sun Microsystems*, propietaria del lenguaje **Java**, muy popular por aquel entonces.

En 1997 se crea un comité (TC39) para estadarizar JavaScript por la *European Computer Manufacturers' Association*, ECMA. Se diseña el estándar del DOM (*Document Object Model*) para evitar incompatibilidades entre navegadores. A partir de entonces los estándares de JavaScript se rigen por ECMAScript.

En 1999 aparece la 3a versión del estándar ECMAScript, que se mantendría vigente hasta hace pocos años. Hubo pequeños intentos de escribir la versión 4, pero hasta 2011 no se aprobó y se estandarizó la versión 5 (ES5) que es la que usamos hoy en día.

En junio de 2013 quedó parado el borrador de la versión 6, pero en Diciembre de 2014 se aprobó al fin y se espera su estandarización a partir de Junio de 2015.” (Azaustre, 2018)

6.6 APACHE CORDOVA

Es un Framework con una multitud de APIS que nos permiten tener acceso a funciones de nuestro dispositivo, tales como el gps, bluetooth , cámara , micrófono, etc… a los cuales nativamente no tenemos acceso , de ésta manera rápida podemos ahorrarnos mucho dinero y tiempo ya que nos permite crear aplicaciones para varias plataformas usando el mismo lenguaje, teniendo en cuenta algunos aspectos importantes : (Guerra Q. F., 2014)

“¿Qué lenguaje se utiliza?
Las aplicaciones con Apache Cordova se hacen con html, css y javascript.

¿Para qué plataformas sirve?
Como se puede ver en su documentación soportan, de diferentes maneras, todas estas plataformas:

- Amazon Fire OS
- Android
- Blackberry 10
- Firefox OS
- iOS
- Ubuntu
- Windows Phone
- Windows 8
- Tizen

¿Mi aplicación funcionará igual de bien/mal en todos los sistemas operativos? No, como veremos más adelante, para cada sistema operativo se genera una aplicación y tienen diferentes características.

¿Y se verá igual en todos los OSs?

Gracias a una de las características más interesantes de Apache Cordova, podremos aplicar cambios concretos para cada aplicación según su sistema operativo, utilizando la carpeta de *merge* tenemos la posibilidad de incluir cualquier archivo (imágenes, css, javascript, etc.) de manera que al compilar nuestra aplicación podemos tener un css para cada plataforma, o un sprite de iconos distinto, sin tener que escribir nada de código.” (Guerra Q. F., 2014)

6.7 **Moment.js**

Es una librería de Javascript con el cual podemos manejar de una manera sólida y estable las fechas, ejemplos básicos: (Álvarez, 2015)

“En este primer ejemplo creamos dos fechas una el 30 de Mayo de 2015 y otra que hace referencia **al momento actual**. La primera fecha la formateamos y la imprimimos por la consola usando diferentes opciones del API de Moment.js. Hecho esto utilizamos la fecha del examen y la fecha actual para restarlas y calcular los días que faltan para un teórico examen (79 días).

30 05 2015

Día de la semana: 6

Mes: 4

Año: 2015

La diferencia en días es 79

Otra de las cosas que también permite Moment.js es formatear las fechas apoyándonos en las diferentes localizaciones (idioma/cultura) que vayamos a utilizar.” (Álvarez, 2015)

6.8 **JDK DE JAVA**

Es el kit de desarrollo de java, siendo un programa que integra todas las herramientas para que nosotros podamos crear programas en Java, está compuesto por : (admin, 2017)

- “Appletviewer.exe: nos permite generar vistas previas visor de applets (pequeñas aplicaciones web). Necesitamos generar vistas previas debido a que al carecer de un método main no se puede ejecutar desde el programa de Java.
- Javac.exe: el compilador que nos permitirá crear el archivo .class para más tarde ejecutarlo con la JVM de JAVA.
- Java.exe: El intérprete de Java.
- Javadoc.exe: nos permite generar la documentación de las clases que contiene un programa en Java.
- JRE al completo: Si instalamos JDK automáticamente tendremos instalado el JRE al completo (JVM, sus bibliotecas Java y sus componentes).” (admin, 2017)

6.9 **JRE DE JAVA**

El JRE nos permite sólo abrir los archivos compilados y creados en Java. para nuestro caso cuando creamos aplicaciones en IONIC lo recomendable es instalar el JDK ya que podremos abrir todo nuestro proyecto en Android para hacer cualquier modificación, (admin, 2017)

“Nosotros instalaremos el JDK ya qué con el JRE solo podríamos ejecutar los programas una vez compilados y con extensión .class

Elementos del JRE

El JRE, está compuesto por la máquina virtual de Java también conocida por JVM o Java Virtual Machine, un conjunto de bibliotecas Java y algunos componentes necesarios para que una aplicación en Java pueda ser ejecutada.

JRE = JVM + bibliotecas Java + componentes” (admin, 2017)

6.10 **CHARTS.JS**

Es una librería de JavaScript que nos da la oportunidad de construir geniales gráficas para nuestro proyecto, las personas cuando ven mucho texto pues simplemente ya no lo leen y pasan ese párrafo rápidamente, observando lo que nos dicen éstos autores en su ejemplo : (admin, 2017)

“La gente normalmente no quiere recorrer una cantidad grande de información presentada a ellos en forma de texto o tablas. Eso es mayormente porque es aburrido, pero más importante, es un poco difícil de procesar números crudos.

Por ejemplo, aquí está una tabla de los diez países más poblados en el mundo:

Nombre del País	Población
China	1,379,302,771
India	1,281,935,911

Nombre del País	Población
Estados Unidos	326,625,791
Indonesia	260,580,739
Brasil	207,353,391
Pakistán	204,924,861
Nigeria	190,632,261
Bangladesh	157,826,578
Rusia	142,257,519
Japón	126,451,398

Con solo diez países en esta tabla, aún hay una muy buena oportunidad de que tu y otros lectores salten la tabla enteramente. Normalmente, la gente solo mira uno o dos países que les interesan. Si los mismos datos hubieran sido presentados en la forma de un gráfico de barras, hubiera tomado muy poco tiempo para que alguien tuviera una idea estimada de la población en estos países.

Además, será mucho más sencillo descifrar tendencias o hechos---por ejemplo, los Estados Unidos tiene el doble de población que Bangladesh, y China tiene alrededor de diez veces más gente que Rusia---solo viendo la longitud de las barras en la gráfica.

Una librería popular que puedes usar para crear diferentes tipos de gráficos es Chart.js. En esta serie, estarás aprendiendo todo sobre los aspectos

importantes de esta librería. Puede ser usada para crear gráficos elegantes y responsivos sobre Canvas HTML5.

La librería te permite mezclar diferentes tipos de gráficos y trazar datos en escalas fecha tiempo, logarítimica, o personalizada con facilidad. La librería también soporta animaciones que pueden ser aplicadas cuando se cambian los datos o se actualizan colores.” (Shokeen, 2017)

6.11 IONIC

Ionices una framework de software libre el cual nos permite realizar aplicaciones hibridas ya sea móviles o de escritorio utilizando tecnologías web (HTML, CSS y JavaScript) (Ionic, 2019)

Ionic se centra en el diseño del front-end ayudando al programador a mejorar el mismo, también se puede implementar con otras librerías fácilmente, Ionic tiene integrado Angular y React. (Ionic, 2019)

6.12 JAVASCRIPT

Javascript es un lenguaje de programación diseñado para creación y diseño de sitios web, este es interpretado por los navegadores. Javascrip tiene una ventaja que puede ser visualizado sin necesidad de tener instalado algún programa. (Mozilla y colaboradores individuales., 2005-2020)

6.13 ANGULAR

Angular nos ayuda a crear aplicaciones web con un front-end de fácil manejo y una presentación y funcionabilidad mejorada, a continuación, su definición:

¨ **Angular es un framework de desarrollo** para JavaScript creado por Google. La finalidad de Angular es facilitarnos el desarrollo de **aplicaciones web SPA** y además darnos herramientas para trabajar con los elementos de una web de una manera más sencilla y óptima. ¨ (Robles, 2020)

6.15 NODEJS

Es un entorno de ejecución en tiempo real que incluyendo todo lo necesario para ejecutar programas de escritorio escritos en javscript, (Lucas, 2019)

¨Tanto JavaScript como **Node.js** se ejecutan en el motor de tiempo de ejecución JavaScript V8 (V8 es el nombre del motor de JavaScript que alimenta Google Chrome. Es lo que toma nuestro JavaScript y lo ejecuta

mientras navega con Chrome). Este motor coge el código JavaScript y lo convierte en un **código de máquina** más rápido. ¨ (Lucas, 2019)

CAPÍTULO III.-

METODOLOGÍA “RUP”

La metodología RUP, abreviatura de Rational Unified Process (o Proceso Unificado Racional), es un proceso de técnicas que deben seguir los miembros del equipo de desarrollo de software con el fin de aumentar su productividad en el proceso de desarrollo

Cuatro fases de la metodología Rup a tener en cuenta para desarrollar un proyecto:

7.1 INICIO

7.2 ELABORACIÓN

7.3 CONSTRUCCIÓN

7.4 TRANSICIÓN

7.1 FASE DE INICIO

Se indica el modelo del negocio y los alcances. Se indican los actores y Casos de Uso, se diseñan los Casos de Uso más esenciales teniendo en cuenta también algunos objetivos:

• Establecer el ámbito del proyecto y sus límites.

• Encontrar los Casos de Uso críticos del sistema, los escenarios básicos que definen la

funcionalidad.

• Dar un coste estimado en recursos y tiempo de todo el proyecto.

• Indicar los riesgos.

7.1.1 CARACTERÍSTICAS PUNTUALES, REQUERIMIENTOS Y RESTRICCIONES DEL PROYECTO

REGISTRO DE MEDICOS

Nosotros cómo administradores registramos a los médicos y les indicamos su usuario y contraseña.

REGISTRO DE USUARIOS

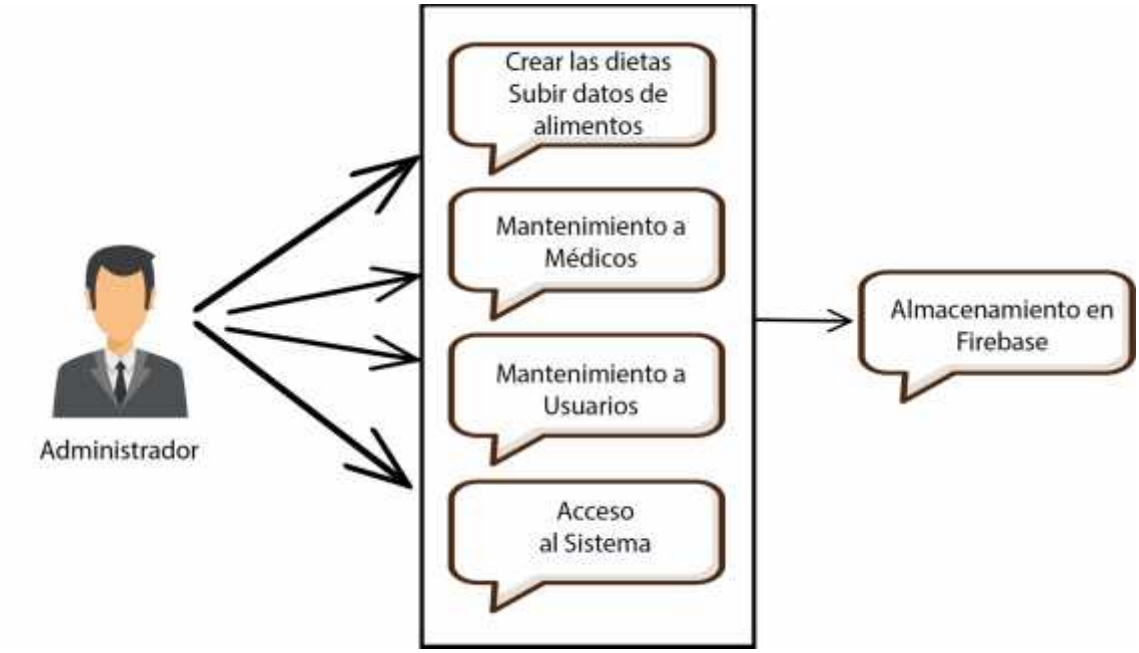
El usuario puede registrarse por sí sólo en la aplicación, si desea puede escoger a un médico, caso contrario puede hacer uso de la aplicación sin el seguimiento de un tratante.

INGRESO DE ALIMENTOS A LA BASE DE DATOS

Sólo Nosotros cómo administradores podemos ingresar los alimentos y la información que va a ser usada por los usuarios y médicos.

7.1.2 CASOS DE USO CONCRETOS.

Administrador, acciones que realizamos cómo administradores, imágenes usadas de la web de recursos gratuitos Freekip.



(Freepik) (freepik) (Autores, 2020)

Caso de Uso #1 Administrador: **Acceso al Sistema**

Actores: Administradores

Secuencia de trabajo:

1. El Administrador accede con sus credenciales
2. El sistema verifica que sus datos concuerden.
3. Si sus credenciales son correctas se le da acceso al sistema.
4. Si son incorrectas no se le da acceso al sistema.

Caso de Uso Administrador: **Mantenimiento a Usuarios**

Actores: Administradores

Secuencia de trabajo:

1. El Administrador puede eliminar a cualquier usuario.
2. Puede también ver todos sus datos y administrarlos.
3. Luego se almacena todo en la Base de Datos Online.

Caso de Uso Administrador: **Mantenimiento a Médicos.**

Actores: Administradores

Secuencia de trabajo:

1. El Administrador crea las credenciales para los médicos.
2. Puede también ver todos sus datos y administrarlos.
3. Puede bloquearlos si no han pagado la cuota mensual. ”función en construcción”.

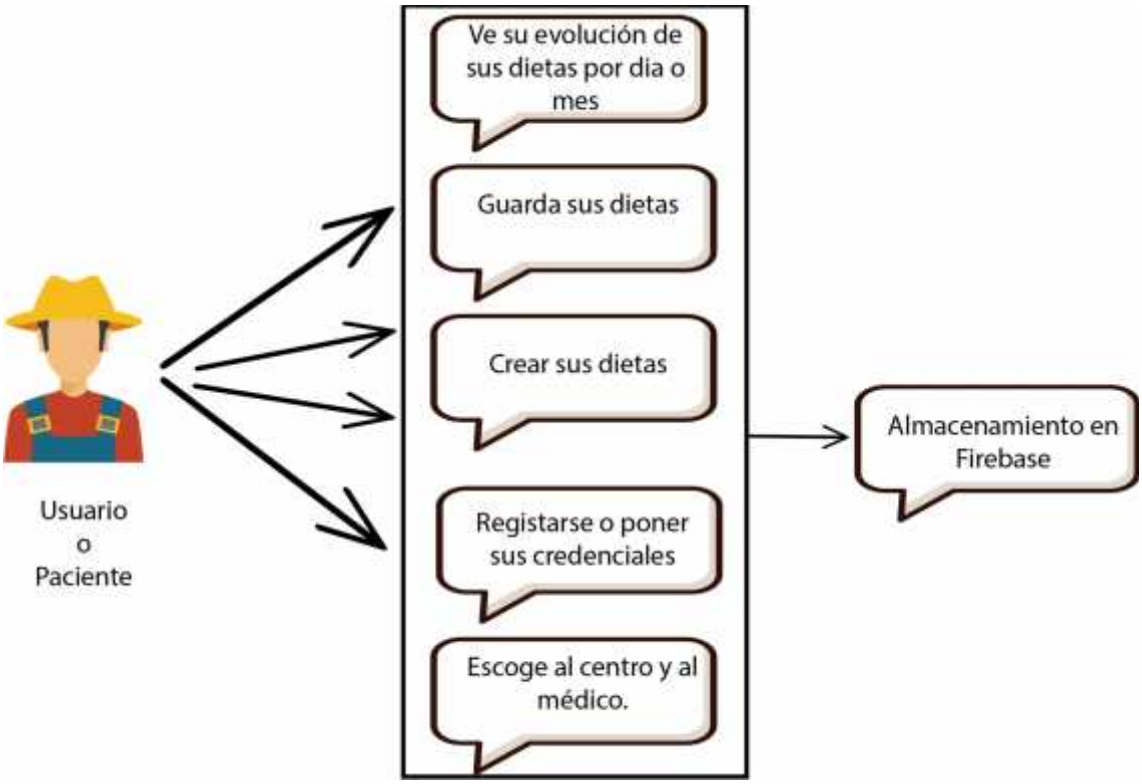
Caso de Uso Administrador: **Crear las dietas y subir datos de los alimentos.**

Actores: Administradores

Secuencia de trabajo:

1. El administrador puede ingresar nuevas dietas.
2. El administrador puede modificar las dietas.
3. El administrador puede eliminar las dietas.

4. El administrador puede hacer el ingreso de toda clase de información.
5. Luego se almacena todo en la Base de Datos Online.



(Freepik) (freepik) (Autores, 2020)

Caso de Uso #2 Cliente o Usuario: **Escoge al Centro y al Médico**

Actores: Usuario o Paciente

Secuencia de trabajo:

1. Abre la aplicación y escoge al centro médico y a su doctor tratante, si lo desea, caso contrario puede usar la aplicación si tener un centro ni un médico tratante.

Caso de Uso #2 Cliente o Usuario: **Registrarse o poner sus credenciales-**

Actores: Usuario o Paciente

Secuencia de trabajo:

2. Si no tiene sus credenciales pues se registra, si ya tiene usuario y contraseña hace el ingreso y el sistema verifica sus datos para darle el acceso.

Caso de Uso #2 Cliente o Usuario: **Crea sus dietas**

Actores: Usuario o Paciente

Secuencia de trabajo:

1. Observa los alimentos ingresados en el programa.
2. Puede ir escogiendo los alimentos que vaya a consumir.
3. Puede ir viendo todos los consejos que le da la aplicación sobre dicho alimento.

Caso de Uso #2 Cliente o Usuario: **Guarda sus dietas.**

Actores: Usuario o Paciente.

Secuencia de trabajo:

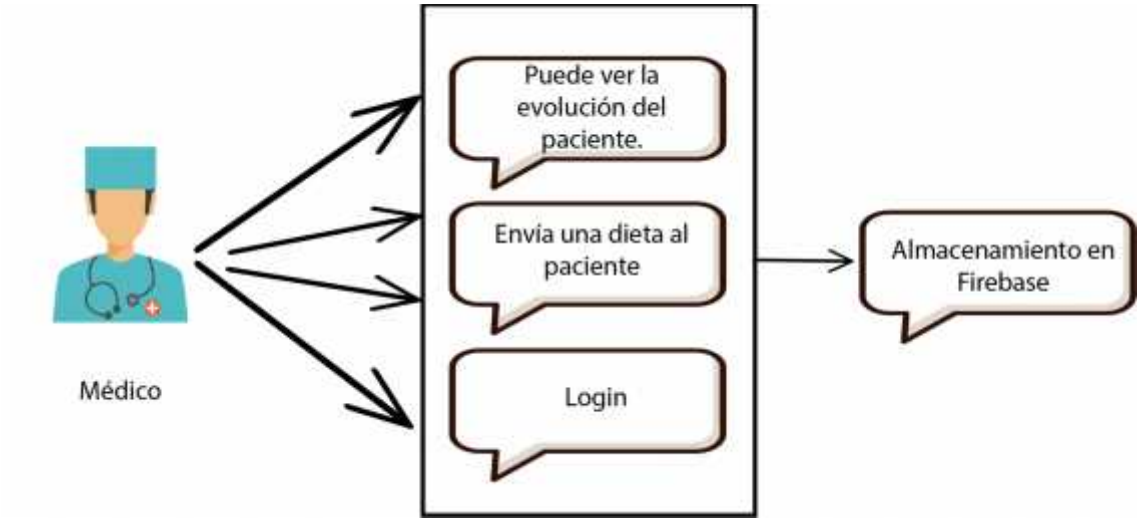
1. Puede ir observando todos los alimentos que vaya a ir consumiendo.
2. El sistema le va a entregar alertas cuando algún alimento se esté excediendo en electrolitos.
3. Ahora cuando el esté seguro de lo que vaya a consumir puede guardar su dieta.

Caso de Uso #2 Cliente o Usuario: **Ver su evolución de su dieta por día o mes.**

Actores: Cliente o Usuario

Secuencia de trabajo:

1. Mediante la función principal de la app el usuario puede observar de una manera gráfica la evolución de su dieta, la cual le muestra los valores consumidos de los electrolitos.



(Freepik) (freepik) (Autores, 2020)

Caso de Uso #3 Médico: **Login.**

Actores: Médico

Secuencia de trabajo:

1. Debe pedir las credenciales a nosotros que somos los creadores de la app.
2. Una vez que dispone de usuario y contraseña hace el ingreso en la aplicación.

Caso de Uso #3 Médico: **Envía una dieta al paciente.**

Actores: Médico

Secuencia de trabajo:

1. Le sugiere una dieta baja en algún electrolito que esté muy alta en ese mes.
2. Esa dieta es seguida por el paciente y lo va almacenando en la base de datos.

Caso de Uso #3 Médico: **Ver su evolución de la dieta del paciente.**

Actores: Médico.

Secuencia de trabajo:

1. Ingresa al sistema, escoge al paciente y puede ver mediante gráficas todo lo que su paciente está consumiendo.

7.1.3 RIESGOS Y PLAN DE CONTINGENCIA

RIESGOS

1. Aplicación no escalable.
2. Vulnerabilidad de la información
3. Desconocimiento del uso de la aplicación.

PLAN DE CONTINGENCIA

1. Se seguirá actualizando la aplicación conforme el framework de ionic y los sistemas operativos se actualicen en el tiempo.
2. Firebase cuenta con su propio sistema de seguridad de registro, si un usuario pierde su contraseña u otra persona está haciendo uso de sus credenciales pues se puede poner en contacto con los administradores para que le cambien de contraseña o eliminen el usuario actual.
3. Dar capacitación para el uso de la aplicación.

7.1.4 PLAN DEL PROYECTO.

La siguiente tabla muestra una la distribución de tiempos y el número de iteraciones de cada fase (para las fases de Construcción y Transición es sólo una aproximación muy preliminar).

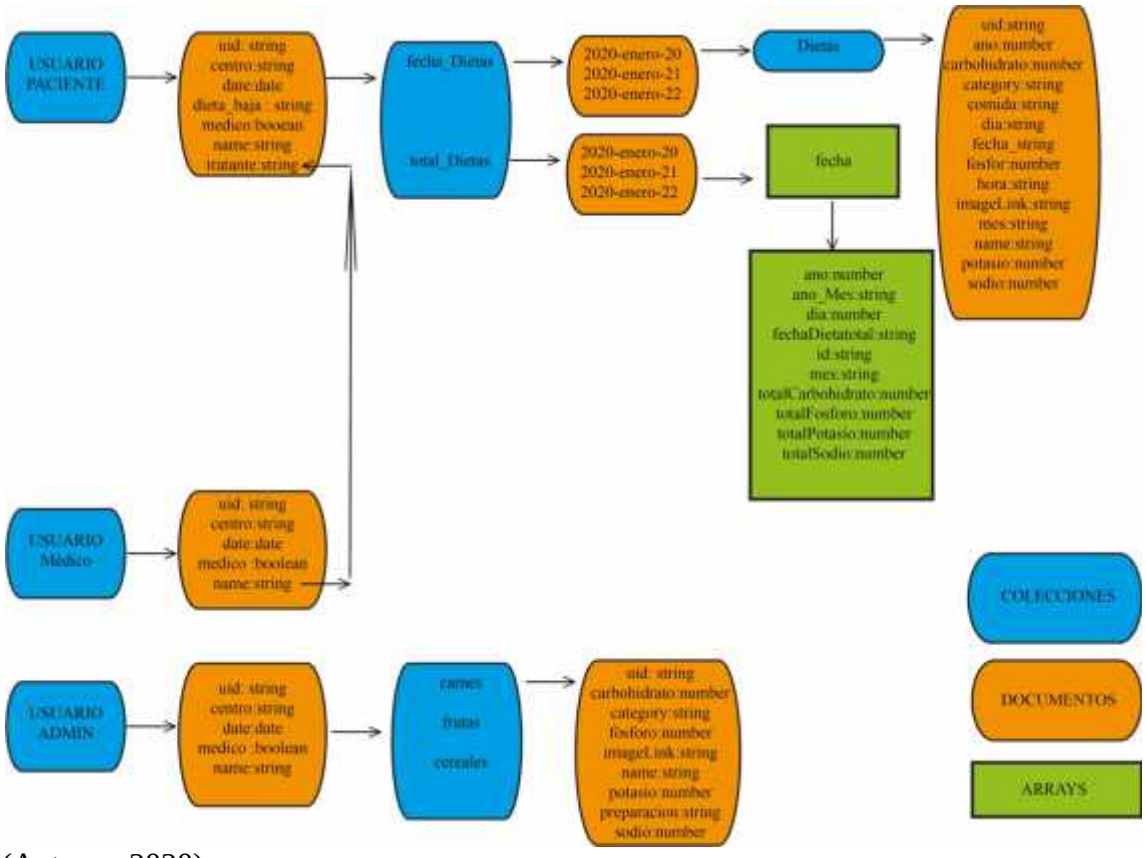
Fases	Nro. Fase Duración Iteraciones	Duración
Inicio	1	3 semanas
Elaboración	2	6 semanas
Construcción	4	10 semanas
Transición	2	3 semanas

7.1.5 Estructura de la Base de Datos en FIREBASE

Firebase es una Base de Datos NOSQL, que vamos a almacenar documentos.

La estructura de la base de datos de FIRESTORAGE está basada en colecciones y documentos, una colección es un conjunto de documentos y un documento son agrupaciones de pares clave valor donde vamos dejar nuestros datos: a continuación el modelo de la Base de Datos NOSQL.

Nota: los gráficos de color azul son colecciones, los de color amarillo son documentos y los de color verde son arrays.



(Autores, 2020)

7.2. ELABORACIÓN

En ésta fase se definen o elaboran las bases de la arquitectura del proyecto, indicamos un buen plan de desarrollo del mismo y mitigamos o eliminamos los riegos más críticos que puedan causar problemas en el desarrollo de la aplicación,

En éste punto se debe crear versiones pequeñas del proyecto o prototipos del mismo, para ir creando dicho sistema e ir comprobando que todo funciona bien , hasta crear el proyecto o aplicación final del mismo , se identifican los casos de uso críticos indicados en la primera fase y también se indican los riesgos que se han mitigado o eliminado.

7.2.1 LOS OBJETIVOS DE ESTA FASE SON:

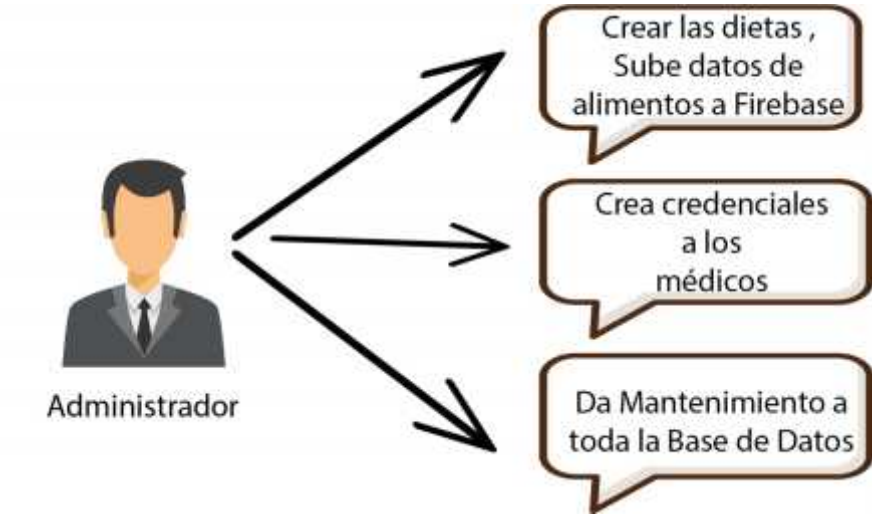
• Indicar las bases de la arquitectura.

• Definir un buen plan en ésta fase , que al pasar el tiempo pueda ser escalable y que evolucione para ir concretando todos sus objetivos.

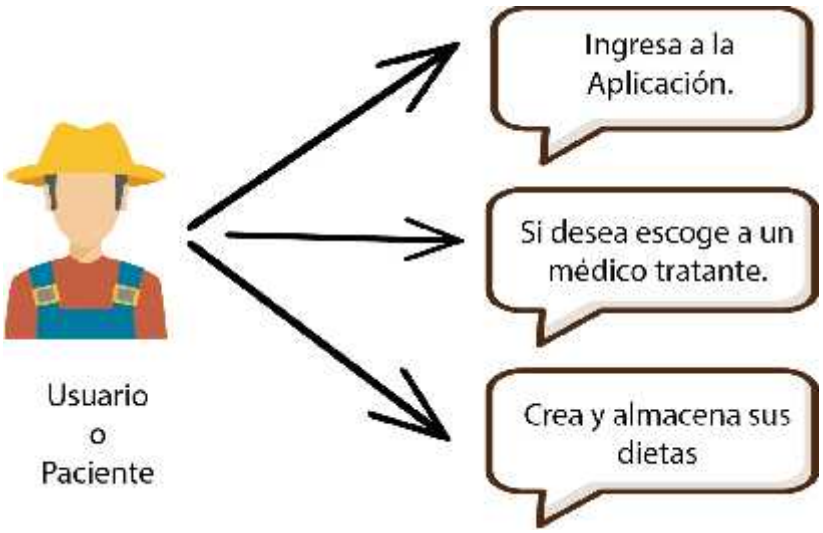
7.2.2 AL DEFINIR ÉSTA FASE SE DEBEN INDICAR LOS SIGUIENTES RESULTADOS:

7.2.2.1 CASOS DE USO Y ACTORES IDENTIFICADOS.

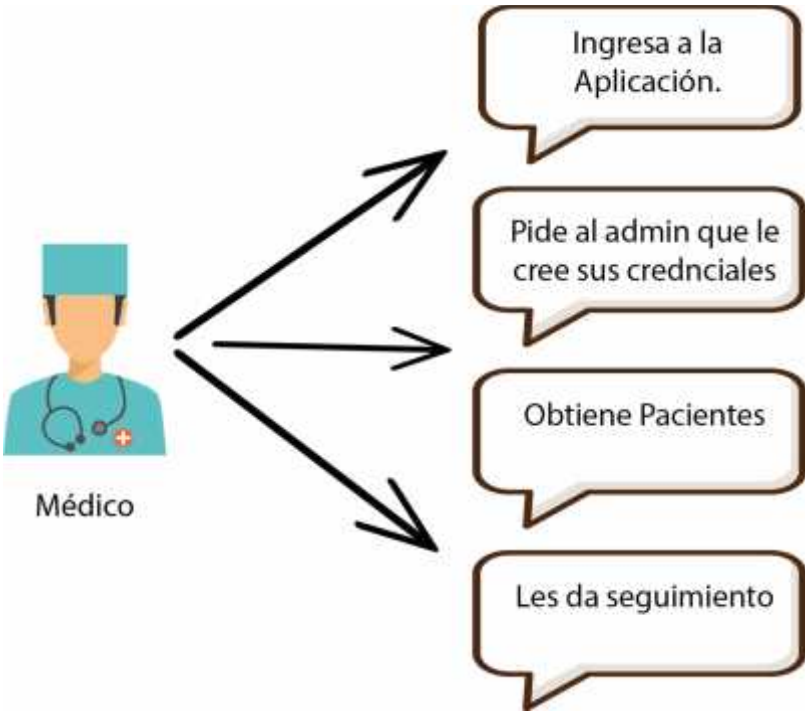
Casos de uso de la función específica que hace cada actor.



(Freepik) (freepik) (Autores, 2020)



(Freepik) (freepik) (Autores, 2020)



(Freepik) (freepik) (Autores, 2020)

DIAGRAMA DE SECUENCIA DE REGISTRO DE NUEVO USUARIO

(Autores, 2020)

Secuencia para Registro de nuevo Usuario

El usuario hace una petición de registro.

El sistema verifica sus datos en Firebase

Luego de verificar todo se crean las credenciales del nuevo usuario.

DIAGRAMA DE SECUENCIA DE REGISTRO DE NUEVO DOCTOR

(Autores, 2020)

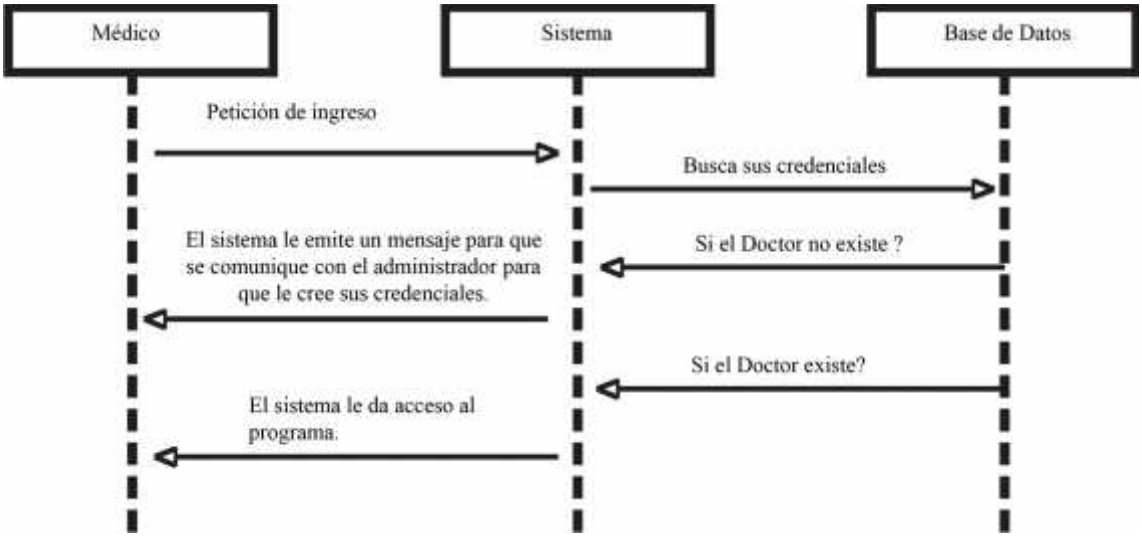
Secuencia para ingreso al sistema de un Médico

El Doctor hace una petición al Administrador para que le de acceso al sistema.

El Administrador verifica sus datos en Firebase.

Luego de hacer la verificación se crean las credenciales para que el doctor tenga acceso al sistema.

DIAGRAMA DE SECUENCIA PARA EL INGRESO DE UN DOCTOR.



(Autores, 2020)

Secuencia para ingreso al sistema de un Médico

El Doctor hace una petición de ingreso al sistema.

El sistema verifica sus datos en Firebase.

Luego de verificar todo, si el doctor existe pues accede al sistema, si no existe se le da un mensaje para que se comunique con el administrador para que se le creen sus credenciales.

DIAGRAMA DE SECUENCIA PARA EL INGRESO DEL ADMINISTRADOR AL SISTEMA.

(Autores, 2020)

Secuencia para ingreso al sistema de un Médico

El Administrador hace una petición de ingreso al sistema.

El sistema verifica sus datos en Firebase.

Luego de verificar todo, si el administrador existe pues accede al sistema, y cómo creador de la app tiene acceso a todos los datos.

7.2.3 PROTOTIPO EJECUTABLE DE LA ARQUITECTURA.

Varias Pantallas tanto de ingreso cómo de consultas ya creadas en la aplicación, usadas por los usuarios, doctores y administradores.

Pantalla del ingreso de los Doctores.

(Autores, 2020)

Gráfico #1, Pantalla del ingreso de los Doctores.

Pantalla del ingreso de los Usuarios.

(Autores, 2020)

Gráfico #2, Pantalla del ingreso de los Usuarios.

Pantalla de agregación de dietas a los Usuarios.

(Autores, 2020)

Gráfico #3, Pantalla de agregación de dietas a los Usuarios.

7.2.4 LISTA DE RIESGOS PLANTEADOS Y CÓMO SE HAN MITIGADO

LISTA DE RIESGOS

1. Aplicación no escalable.
2. Vulnerabilidad de la información
3. Desconocimiento del uso de la aplicación.

Todos los riesgos han sido ya mitigados de la siguiente manera:

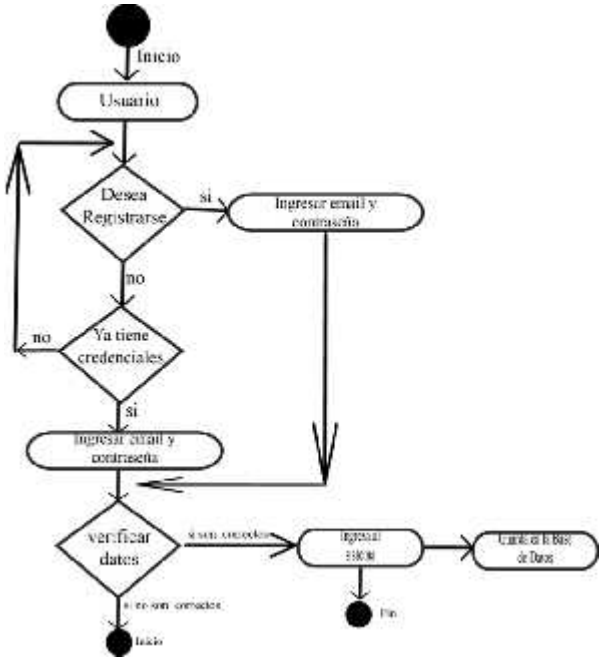
1.-Aplicación no escalable : Estamos usando los frameworks actualmente recomendados para la creación eficiente de apps tanto para móviles cómo para equipos de escritorio, ya que con el mismo código podemos crear la app para : Android , Ios , web , y programa de escritorio, de ésta manera nuestra app está a un paso adelante que cualquier otra persona que use programas cómo Android Studio que solo puede generar la aplicación para un móvil con sistema Android , en cambio nosotros con el mismo lenguaje podemos generar la aplicación para varios sistemas al mismo tiempo.

2-Vulnerabilidad: La seguridad la maneja directamente Firebase, nosotros usamos lo que sus creadores nos recomiendan usar para la seguridad. Igualmente los campos ya están validados para evitar cualquier fallo al ingreso de los datos.

3- En éste punto hemos optado en poner información del uso de la app en la misma aplicación, información tanto en texto cómo un video demostrativo de cómo se usa nuestra aplicación.

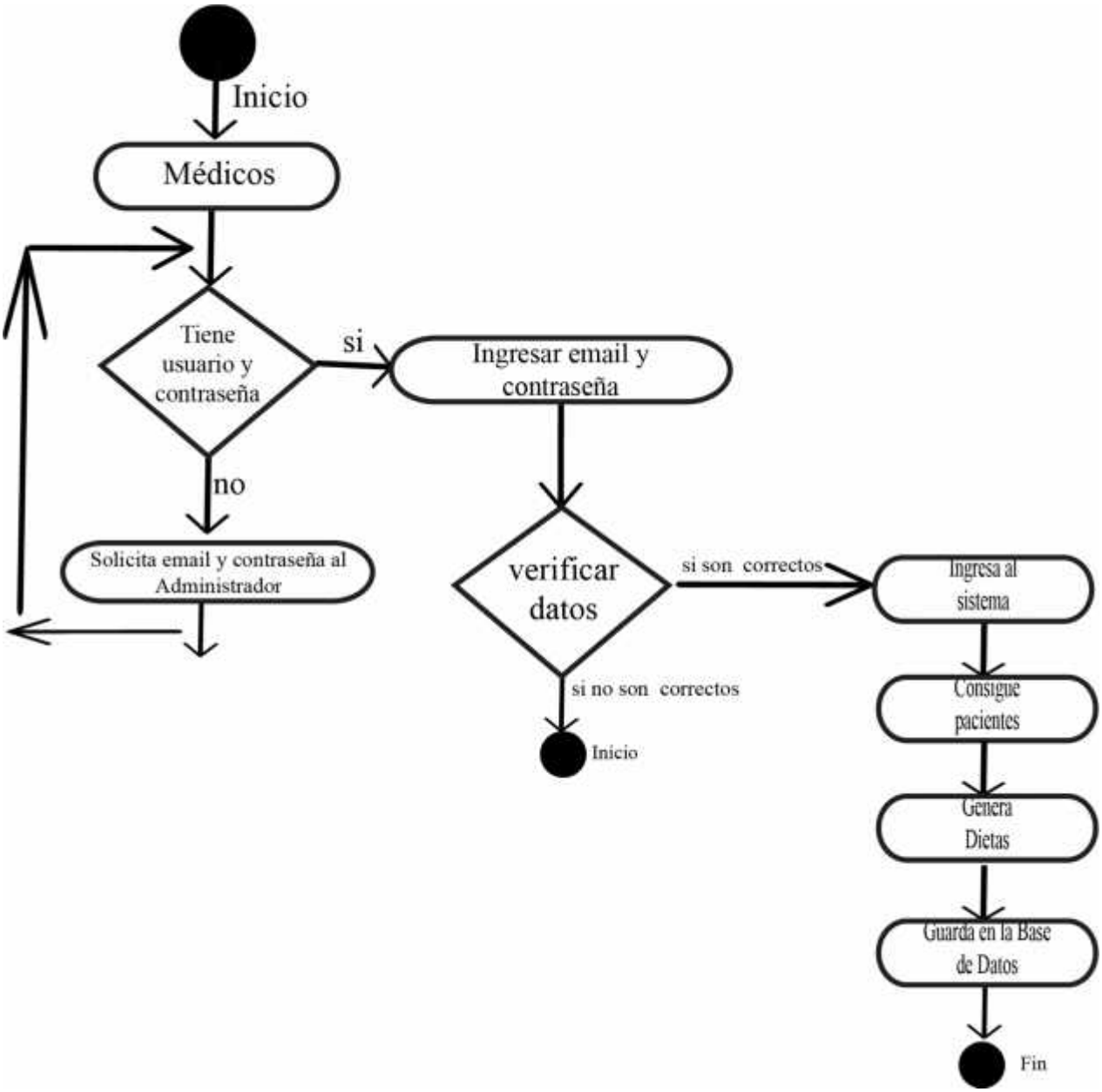
7.2.5 DIAGRAMAS DE ESTADO QUE INDICAN EL FLUJO A SEGUIR DE CADA ACTOR.

DIAGRAMA DE ESTADO, LOGIN DE USUARIO.



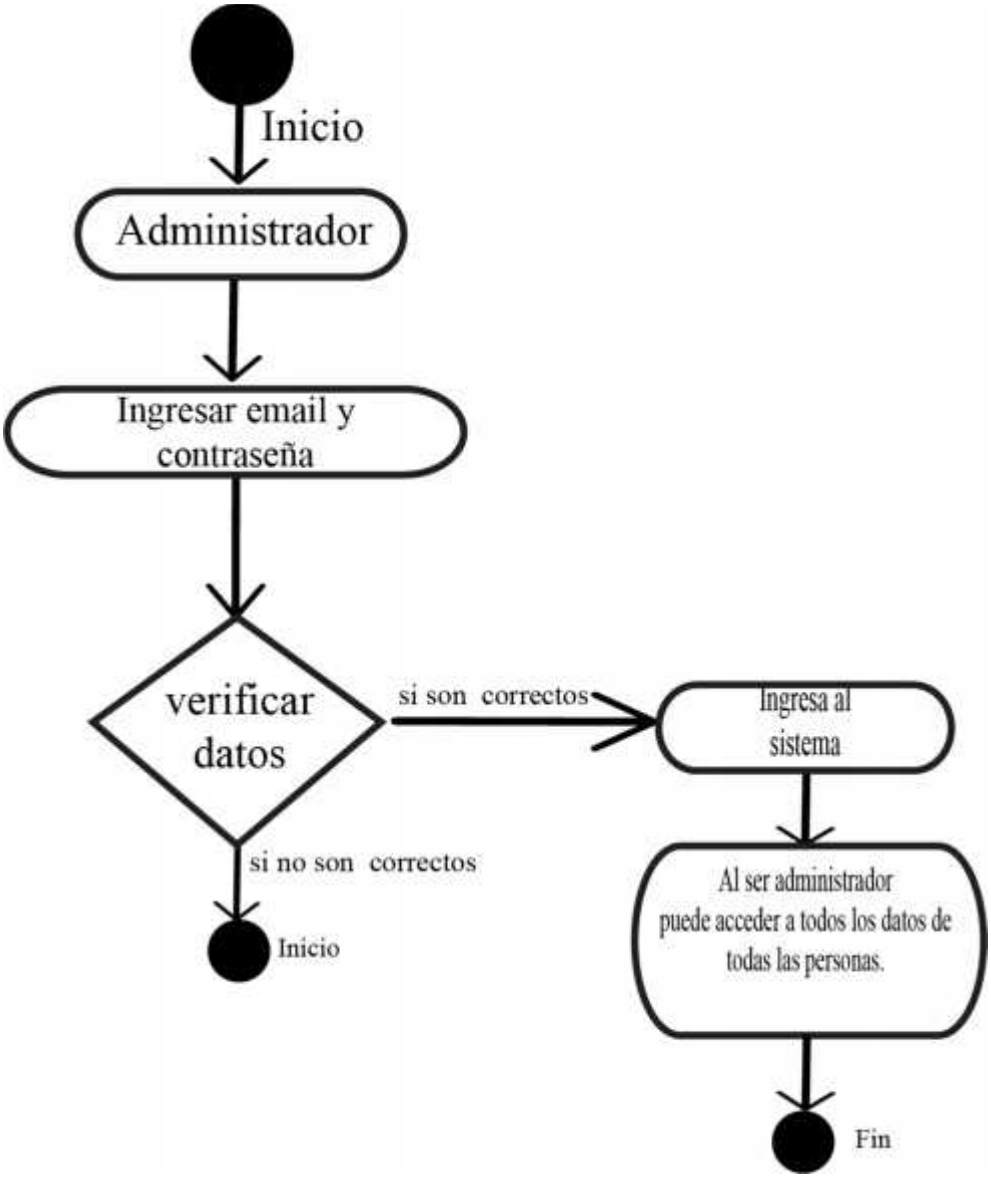
(Autores, 2020)

DIAGRAMA DE ESTADO, LOGIN DE MÉDICOS.



(Autores, 2020)

DIAGRAMA DE ESTADO, LOGIN DE ADMINISTRADOR.



(Autores, 2020)

7.3 CONSTRUCCIÓN

En ésta etapa nosotros cómo programadores deber tener la capacidad de alcanzar que el producto proyecto sea operativo, en una forma secuencial y que vaya evolucionando poco a poco, debemos implementar todas las características, recomendaciones, funciones que durante todas las otras fases y objetivos nos hemos planteado o no han recomendado.

De ésta manera tendríamos poco a poco ya una aplicación sólida y versiones ya estables de nuestro producto.

7.3.1 LOS OBJETIVOS DE ESTA FASE DEBEN CONTENER:

• Investigar bien lo que se desea obtener, para no tener que rehacer o eliminar procedimientos que ya estaban creados.

• Construir lo más rápido la aplicación o proyecto pero no dejando de lado la calidad o funcionalidad del mismo.

• Ir creando versiones estables de nuestra aplicación para ir testeando una a una e ir creando o solidificando una versión ya estable de la misma.

• Analizar todas las propuestas u objetivos que estén faltando de implementar en la aplicación.

• Mientras se crea el proyecto debemos al mismo tiempo ir mitigando los riesgos que ya tengamos y todos los riesgos y problemas que se nos vayan presentando.

• Ir analizando y comparando nuestro proyecto con otros proyectos de la misma índole o creadas por otras personas, para evitar problemas que tengan otros proyectos o para mejorar o dar mejores funcionalidades a nuestro proyecto en mente.

• Si tenemos un director de proyectos debemos en lo mayor posible recabar todas sus recomendaciones para crear un proyecto que sea original e innovador, pero siempre sin dejar de lado el tiempo ya que muchas veces hay proyectos que nunca terminan de concretarse porque quisieron hacer cosas que no pudieron frustrando toda la creación del proyecto.

7.3.2 ARQUITECTURA DEL SISTEMA.

Programación por capas: Se identifica y programa dividiendo por capas, teniendo 3 de las mismas: 1.- capa de presentación, 2.- capa de negocios, 3.- capa de datos.

Capa 1: Interfaz gráfica que observa el usuario para hacer las interacciones o manejo del sistema.

Capa 2: Capa principal, es el cerebro de la aplicación, donde se encuentran todos los procedimientos, funciones. Cálculos, para hacer todas las tareas que definen la aplicación.

Capa 3: En nuestra aplicación es FIREBASE, la base de datos encargada de almacenar toda la información, Firebase maneja sus propias seguridades.

7.3.3 RIESGOS PRESENTADOS Y MITIGADOS

1. Aplicación no escalable.
2. Vulnerabilidad de la información
3. Desconocimiento del uso de la aplicación.

Todos los riesgos han sido ya mitigados en el proceso de creación de la aplicación.

7.3.4 PLAN DEL PROYECTO PARA LA FASE DE TRANSICIÓN.

Ya se han ido creando muchas versiones Beta de las aplicaciones para probarlo en los celulares y en los ordenadores, en los cuales en cada prueba ya se han ido mitigando y eliminando todos los problemas que se nos presentaban en las fases de prueba, igualmente se han ido haciendo todas las pruebas con el almacenamiento de los datos en Firebase.

Igualmente se han hecho pruebas con video tutoriales animados para que los usuarios puedan aprender a manejar el sistema.

7.3.5 MANUAL INICIAL DE USUARIO

Manual De Usuario "DESARROLLO DE UNA APP MÓVIL PARA PACIENTES EN TRATAMIENTO DE HEMODIALISIS PARA EL CONTROL DE LOS MINERALES EN CNSUMO DE ALIMENTOS EN LA CIUDAD DE CUENCA"

REQUERIMIENTOS PARA EL BUEN FUNCIONAMIENTO DEL SISTEMA

a) Hardware:

• Computadora personal."Para el administrador, para que ingrese de mejor manera datos e información al sistema".

• Conexión a Internet. (Aunque la base de datos también funciona en modo offline).

• Dispositivo móvil, (pruebas creadas en sistemas Android.)

b)Software:

• Sistema operativo Windows, Ios, Android, todas las pruebas se han hecho en Android.

Para el funcionamiento de la aplicación se necesita un dispositivo móvil con sistema operativo Android con versiones superiores a la 4.

Para móviles con otros sistemas operativos se harán las pruebas en un futuro.

7.4 TRANSICIÓN

7.4.1 LOS OBJETIVOS QUE DEBE INCLUIR ESTA FASE:

• Prueba de las diferentes versiones Beta.

• Qué los médicos, nutricionistas, usuarios aprendan a manejar la aplicación.

• Seguir mejorando la aplicación y que cumpla todos los objetivos preestablecidos desde el principio.

• Seguir recabando información por parte de los usuarios para seguirle añadiendo mejores funciones para que siga acorde a la tecnología actual y no quede desfasada en el tiempo.

7.4.2 LOS RESULTADOS DE LA FASE DE TRANSICIÓN SON:

Eliminar todos los problemas, quitar los errores por más simples que sean, depurar el programa, investigar todos los inconvenientes que hemos tenido e ir mitigándolos y dejar un proyecto que satisfaga las necesidades del cliente.

7.4.2.1 TODOS LOS PASOS Y SECCIONES DE ÉSTA FASE ESTÁN DIRIGIDAS A CONSEGUIR UNA APLICACIÓN CON 0 ERRORES Y A HACERLA 100% FUNCIONAL.

En ésta etapa nosotros cómo programadores hemos ido haciendo muchas pruebas para ir resolviendo todos los problemas que se nos iban presentando, igualmente se han hecho pruebas con usuarios que manejen la aplicación y nos den su punto

de vista, así mismo la aplicación ha sido manejada por los nutricionistas y médicos y hemos ido poco a poco quitando todas las fallas y poniendo los módulos que nos iban recomendando.

7.4.2.2 CAPTURAS DE PANTALLA DE ALUGNOS SUBMENUS YA OPERATIVOS EN NUESTRA APP.

Creación de dietas por parte del usuario.

El paciente o usuario puede navegar en el sistema e ir observando todos los alimentos que él puede comúnmente ir agregando a su dieta.

(Autores, 2020)

Gráfico #4, Creación de dietas por parte del usuario.

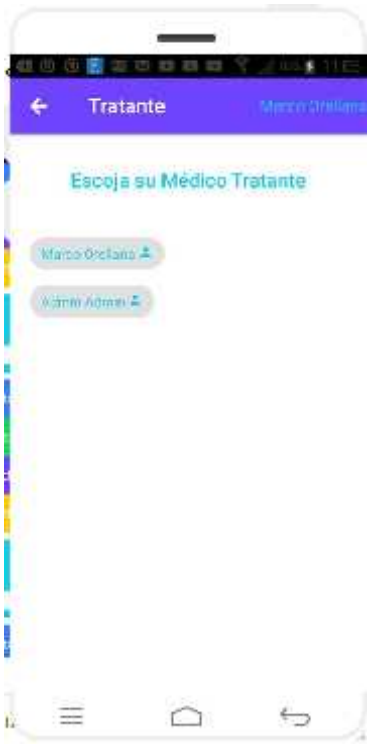
Pantalla donde el usuario puede observar una gráfica de los electrolitos consumidos en su dieta.

(Autores, 2020)

Gráfico #5, Pantalla donde el usuario puede observar una gráfica de los electrolitos consumidos en su dieta.

Pantalla donde el usuario puede cambiarse de médico.

El paciente puede tranquilamente cambiarse de médico, ya que muchas veces un médico puede estar de vacaciones y puede hacerse el tratamiento o seguimiento de su enfermedad o dieta con otro doctor del mismo centro o médico registrado en la aplicación.



(Autores, 2020)

Gráfico #6, Pantalla donde el usuario puede cambiarse de médico.

.Pantalla donde el usuario puede informarse sobre tratamientos para la enfermedad que padece.

(Autores, 2020)

Gráfico #7, Pantalla donde el usuario puede informarse sobre tratamientos para la enfermedad que padece.

7.4.2.3 ELABORACIÓN DE LA BASE DE DATOS.

Para la base de datos se usó FIREBASE, que funciona tanto online cómo offline, de ésta manera cuando el usuario no disponga de acceso a internet podrá usarlo sin ningún inconveniente, cuando tenga internet automáticamente la base de datos que está en la nube se actualiza con los datos guardados en el móvil.

CAPITULO IV

INFORMACIÓN DEL PROGRAMA O SISTEMA.

INGRESO AL PROGRAMA POR PARTE DEL ADMINISTRADOR.

El administrador posee un usuario y contraseña, al ser el administrador o el creador de la aplicación tiene acceso a todos los datos del sistema, puede ver todas las evoluciones de las dietas de todos los usuarios registrados así cómo observar a todos los médicos.

INGRESO DE INFORMACION POR PARTE DEL ADMINISTRADOR.

El administrador puede crear y almacenar toda la información que será requerida por parte de los usuarios, por lo generar el administrador siempre estará subiendo o almacenando dietas, las cuales incluye fotografías y texto que se guardan en el internet.

INGRESO AL PROGRAMA POR PARTE DEL MEDICO.

El médico no tiene credenciales al principio , debe solicitar dichas credenciales al administrador , cuando ya disponga de una credencial de acceso puede ingresar al sistema y tendría que tener pacientes para que les haga el debido seguimiento.

El doctor no agrega pacientes, los pacientes son los que buscan a los médicos registrados en la aplicación, de ésta manera el doctor tendría que pedir a sus pacientes que descarguen la aplicación, se registren y escojan a determinado médico.

INGRESO AL PROGRAMA POR PARTE DEL USUARIO.

El usuario ingresa a la aplicación y debería registrarse para que tenga su usuario y contraseña, si lo desea puede optar por unirse a un médico, caso contrario puede hacer uso de la app sin el seguimiento de un tratante.

CREACION DE DIETAS POR EL USUARIO

En éste apartado, el usuario irá desplegando las diferentes categorías de los alimentos ya creados por el administrador, el usuario podrá ir viendo los valores en los electrolitos que le marca el sistema, luego de observar todos los mensajes de alertas que le da el sistema puede ir guardando y creando sus dieta diaria, para luego poder ir observando toda su evolución.

SEGUIMIENTO DE DIETAS DE UN PACIENTE POR PARTE DEL MEDICO.

Una vez que el médico tenga pacientes y les esté haciendo un seguimiento en sus dietas, el doctor podrá acceder a todos los gráficos o evoluciones de las dietas que está consumiendo el paciente.

INGRESO AL SISTEMA POR PARTE DEL USUARIO.

El usuario pude registrarse en el sistema, si ya tiene sus credenciales pues simplemente da click en login e ingresa al programa.

(Autores, 2020)

Gráfico #8, Ingreso al sistema por parte del usuario.

MENU PRINCIPAL DE LA APLICACIÓN.

El usuario puede navegar entre diferentes opciones del menú, puede cambiarse de tratante, puede crear sus dietas, puede ver las evoluciones de los alimentos que está consumiendo, puede ver recomendaciones e información sobre algunos tratamientos, la evolución de la dieta la puede ver gráficamente y con valores que se hacen mediante los cálculos sobre todos los electrolitos aproximados que contiene cada alimento que de las dietas escogidas.

(Autores, 2020)

Gráfico #9, Figura Menú principal de la aplicación.

Submenú Cambiar Tratante.

El usuario puede cambiar de médico y lo tuviere.

(Autores, 2020)

Gráfico #10, Submenú Cambiar Tratante.

Submenú Crear Dietas.

El usuario escoge los alimentos que está almacenados en la base de datos.

(Autores, 2020)

Gráfico #11, Figura Submenú Crear Dieta.

Submenú Dieta Diaria.

El usuario puede ver por fechas lo que ha estado consumiendo, también puede escoger opciones cómo: “desayuno, almuerzo, cena, etc.”

(Autores, 2020)

Gráfico #12, Figura Submenú Dieta Diaria.

Submenú Evolución Dietas.

El paciente puede escoger el mes y el electrolito a ver la gráfica , de ésta manera puede observar y estar al pendiente si algún electrolito se pasa de su valor máximo y poder dar los resultados a sus medico tratante o a su nutricionista para que observe su evolución.

(Autores, 2020)

Gráfico #13, Figura Submenú Evolución Dietas.

Submenú Gráfico Dietas.

El Usuario puede observar la evolución de los electrolitos consumidos por fechas, dicha información se le despliega en gráficos para su fácil e inmediata interpretación.

(Autores, 2020)

Gráfico #14, Figura Submenú Gráficos Dietas.

Submenú Información Tratamiento.

Información para que el usuario se entere de varias enfermedades, síntomas, causas, complicaciones y prevenciones de algunas enfermedades que están ligadas a personas que se hacen un tratamiento de hemodiálisis.

(Autores, 2020)

Gráfico #15, Figura Submenú Información Tratamiento.

CONCLUSIONES.

El centro médico Baxter con ésta aplicación podrá optimizar tiempo y recursos en la revisión y en el tratamiento de los pacientes, ya que de una manera gráfica podrá observar la evolución de las dietas de sus usuarios o pacientes.

Igualmente por parte del paciente o usuario podrá llevar un registro periódico de lo que consume en sus dietas, esto le ayudará a seguir un régimen estricto de los alimentos para que pueda mejorar en sus salud y a su vez eso registros los podrá observar de una manera fácil, y rápida su médico tratante.

CRONOGRAMA DE ACTIVIDADES

MESES	Septiembre				Octubre				Noviembre				Diciembre				Enero				Febrero			
SEMANAS	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**	**1**	**2**	**3**	**4**
Aprobación del Tema				x																				
Ante Proyecto				x																				
Aprobación				x																				
Investigación					x	x	x	x	x	x	x	x												
Desarrollo							x	x	x	x	x	x	x	x	x	x								
Elaboración de informe final																	x	x	x	x				
Sustentación																							x	

BIBLIOGRAFÍA

Bibliografía

admin. (29 de Agosto de 2017). *javadesde0*. Recuperado el 21 de Enero de 2020, de javadesde0: http://javadesde0.com/jre-jdkque-version-java-instalada-instalo-java-jre/

Álvarez, C. (16 de Marzo de 2015). *genbeta*. Recuperado el 1 de Enero de 2020, de genbeta: https://www.genbeta.com/desarrollo/gestionando-fechas-con-moment-js

Atareao. (18 de Mayo de 2018). *atareao*. Recuperado el Lunes de Enero de 2020, de atareao: https://www.atareao.es/software/programacion/visual-studio-code/

Azaustre, C. (24 de Octubre de 2018). *carlosazaustre*. Recuperado el 20 de Enero de 2020, de carlosazaustre: https://carlosazaustre.es/ecmascript6/

Fernández, A. (09 de 08 de 2016). *Quora*. Recuperado el 20 de Enero de 2020, de Quora: https://es.quora.com/Qu%C3%A9-es-HTML5-y-para-qu%C3%A9-sirve

freepik. (s.f.). *freepik*. Obtenido de freepik: https://www.freepik.es/vector-gratis/conjunto-profesion-plana-avatar_4407653.htm#page=3&query=doctor+icon&position=16

Freepik, h. b. (s.f.). *Freepik*. Recuperado el 23 de Enero de 2020, de Freepik: https://www.freepik.es/vector-gratis/pack-globos-dialogo-dibujados-mano_1167269.htm#page=1&query=bubbles&position=26

Giraldo, V. (16 de Abril de 2019). *rockcontent*. Recuperado el 20 de Enero de 2020, de rockcontent: https://rockcontent.com/es/blog/que-es-firebase/

Guerra, E. F. (02 de junio de 2016). *desarrolloweb*. Recuperado el 20 de Enero de 2020, de desarrolloweb: https://desarrolloweb.com/articulos/introduccion-a-typescript.html

Guerra, Q. F. (14 de Marzo de 2014). *ckgrafico*. Recuperado el 21 de Enero de 2020, de ckgrafico: http://blog.ckgrafico.com/que-es-apache-cordova/

ionic. (12 de 2019 de 2018). *ionic*. Recuperado el 412 de 123 de 123, de ionic: www.ionic.com

Ionic. (11 de 11 de 2019). *Ionic Docs*. Obtenido de https://ionicframework.com/docs/intro

Lucas, J. (04 de Septiembre de 2019). *OpenWebinars S.L.* Obtenido de https://openwebinars.net/blog/que-es-nodejs/

Shokeen, M. (18 de Abril de 2017). *code.tutsplus*. Recuperado el 21 de Enero de 2020, de code.tutsplus: https://code.tutsplus.com/es/tutorials/getting-started-with-chartjs-introduction--cms-28278

ANEXOS

ANEXO 1 CÓDIGO FUENTE DE LA APLICACIÓN.

ANEXO 2 ENCUESTAS

CÓDIGOS DE CADA UNA DE LAS PÁGINAS DE LA APLICACIÓN

ANEXO 1.1 CREAR-DIETA-PACIENTE.PAGE.TS

```
import{ Component, OnInit } from'@angular/core';
import{ UserService } from'../../services/user.service';
import{ AuthService } from'../../services/auth.service';
import{ ModalController, AlertController, ToastController }
from'@ionic/angular';
import{ AlimentoFecha } from'../../models/alimento';
import{ BaseAlimentosService } from'../../services/base-alimentos.service';

@Component({
selector:'app-crear-dieta-paciente',
templateUrl:'./crear-dieta-paciente.page.html',
styleUrls: ['./crear-dieta-paciente.page.scss'],
})
exportclassCrearDietaPacientePageimplementsOnInit {

idUser: string;
comida: string;
category: string;
listaAlimentos: any;
valoresCambio: any;
fecha: any;
time: any;
fechaActual: any;
potasioPush: any[] = [];
dietaBaja: any;
calorias: any;
nutri: any;
proteinas: any;
grasas: any;
```

```
tCarbohidratos: any;
preparacion: string;
textoBuscar = '';


constructor(privateuserService: UserService,
privateauthservice: AuthService,
privatemodal: ModalController,
privatealertCtrl: AlertController,
privatetoastCtrl: ToastController,
privatebaseAlimentos: BaseAlimentosService) { }

ngOnInit() {
this.authservice.verificarUser().subscribe(id=> {
this.idUser = id;
this.userService.getPacienteTratante(this.idUser).subscribe(paciente=> {
paciente.map(p=> {
this.valoresCambio = p
this.dietaBaja = this.valoresCambio.dietaBaja;
console.log('Dieta Paciente:', this.valoresCambio.dietaBaja)
});
    })
  })
 }

asyncpresentToast(message: string) {
consttoast = awaitthis.toastCtrl.create({
message,
duration:3000
  });
toast.present();
 }

verAlimento(event) {
```

```
this.category = event.detail.value;
console.log('Update', this.category);
this.userService.getListaAlimentosPacientes(this.category).subscribe(listas=> {
this.listaAlimentos = listas;
console.log('Listas Alimentos:', listas)
   })
 }

comidadelDia(event) {
this.comida = event.detail.value;
console.log('Comida', this.comida);
}

buscar(event) {
// this.textBuscar = event.detail.value;
this.textoBuscar = event.detail.value;
//console.log('Event', event);
 }

verPreparacion(event) {
this.preparacion = event.detail.value;
console.log('Comida', this.preparacion);
 }

asyncsetDietaDiaria(alimento: AlimentoFecha) {
if (this.valoresCambio.dietaBaja === 'Libre') {
console.log('Dieta Libre')
this.presentAlertDietaLibre(alimento)
   } else {
if (this.valoresCambio.dietaBaja === 'Potasio') {
this.presentAlertDietaPotasio(alimento)
     } else {
if (this.valoresCambio.dietaBaja === 'Fosforo') {
console.log('Es igual a:', this.valoresCambio.dietaBaja)
```

```
this.presentAlertDietaFosforo(alimento)
      } else {
if (this.valoresCambio.dietaBaja === 'Sodio') {
this.presentAlertDietaSodio(alimento)
       } else {
if (this.valoresCambio.dietaBaja === 'Carbohidrato') {
this.presentAlertDietaCarb(alimento)
         }
        }
       }
     }

   }
 }

asyncguardarDieta(alimento: any) {
this.fecha = newDate;
console.log('NEw Date:', this.fecha);
vardia = this.fecha.getDate();
if (dia<= 9) {
dia = '0' + dia
   }
varhoras = this.fecha.getHours();
if (this.fecha.getMinutes() <10) {
this.time = this.fecha.getHours() + ':0' + this.fecha.getMinutes();
console.log('FEcha Actual en minutos;', this.time)
} else {
this.time = this.fecha.getHours() + ':' + this.fecha.getMinutes();
console.log('FEcha Actual en horas;', this.time)
}
varweekday = newArray(7);
weekday[0] = "Domingo";
weekday[1] = "Lunes";
weekday[2] = "Martes";
```

```
weekday[3] = "Miercoles";
weekday[4] = "Jueves";
weekday[5] = "Viernes";
weekday[6] = "Sabado";

varnombreDia = weekday[this.fecha.getDay()];

varmonth = newArray();
month[0] = "enero";
month[1] = "febrero";
month[2] = "marzo";
month[3] = "abril";
month[4] = "mayo";
month[5] = "junio";
month[6] = "julio";
month[7] = "agosto";
month[8] = "semtiembre";
month[9] = "octubre";
month[10] = "noviembre";
month[11] = "diciembre";
varmesNum = this.fecha.getMonth();
varmes = month[this.fecha.getMonth()];

varano = this.fecha.getFullYear();

this.fechaActual = ano + '-' + mes + '-' + dia, this.time;
// console.log('fecha actual gaurdada:', this.fechaActual)
constdieta: AlimentoFecha = {
category:alimento.category,
name:alimento.name,
potasio:alimento.potasio,
fosforo:alimento.fosforo,
sodio:alimento.sodio,
carbohidrato:alimento.carbohidrato,
```

```
imageLink:alimento.imageLink,
fecha:this.fechaActual,
ano:ano,
mes:mes,
dia:nombreDia,
numDia:dia,
hora:this.time,
comida:this.comida,
preparacion:alimento.preparacion
//minutos?:
   }
console.log('Alimeto a guardar con comida:', dieta)
awaitthis.baseAlimentos.addAlimentoDietaDiaria(dieta, this.idUser);
}

asyncpresentAlertVolores(alimento: any, mineral: string) {
constalert = awaitthis.alertCtrl.create({
header:'alerta!',
message:'Usted tiene una dieta baja en <strong>' + mineral + '</strong>!!!
alimento restringido.',
buttons: [
      {
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
this.presentToast('Usted no agregado Alimento a su Dieta...');
}
      }, {
text:'Ok',
handler:async () => {
console.log('Confirm Okay');
this.guardarDieta(alimento);
        }
```

```
        }
      ]
    });
awaitalert.present();
}

asyncpresentAlertDietaLibre(alimento: any) {
constalert = awaitthis.alertCtrl.create({
header:'Aviso!',
message:'Usted tiene una dieta <strong>Libre</strong>!!! y a seleccionado:
<strong>' + alimento.name + '</strong>!!!',
buttons: [
        {
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
console.log('Confirm Cancel: blah');
this.presentToast('Usted no agregado Alimento a su Dieta...');
}
        }, {
text:'Okay',
handler:async () => {
console.log('Confirm Okay');
this.guardarDieta(alimento);
          }
        }
      ]
    });
awaitalert.present();
}

asyncpresentAlertDietaPotasio(alimento: any) {
constalert = awaitthis.alertCtrl.create({
```

```
header:'Aviso ',
subHeader:'Dieta a Agregar: ',
message:'Usted a seleccionado  <strong>' + alimento.name + '</strong>!!!  para
agregar a su dieta.',
buttons: [
      {
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
this.presentToast('Usted no agregado Alimento a su Dieta...');
        }
      }, {
text:'Ok',
handler:async () => {
console.log('Confirm Okay', this.valoresCambio.dietaBaja);
if (alimento.potasio>5) {
this.presentAlertVolores(alimento, this.valoresCambio.dietaBaja)
          }//////////////////////////
else {
this.guardarDieta(alimento);

          }
        }
      }
    ]
  });

awaitalert.present();
}//

asyncpresentAlertDietaFosforo(alimento: any) {
constalert = awaitthis.alertCtrl.create({
header:'Aviso ',
```

```
subHeader:'Dieta a Agregar: ',
message:'Usted a seleccionado  <strong>' + alimento.name + '</strong>!!!  para
agregar a su dieta.',
buttons: [
      {
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
this.presentToast('Usted no agregado Alimento a su Dieta...');
        }
      }, {
text:'Ok',
handler:async () => {
console.log('Confirm Okay', this.valoresCambio.dietaBaja);
if (alimento.fosforo>5) {
this.presentAlertVolores(alimento, this.valoresCambio.dietaBaja)
          }//////////////////////////
else {
this.guardarDieta(alimento)
          }
        }
      }
    ]
  });

awaitalert.present();
}//


asyncpresentAlertDietaSodio(alimento: any) {
constalert = awaitthis.alertCtrl.create({
header:'Aviso ',
subHeader:'Dieta a Agregar: ',
```

```
message:'Usted a seleccionado  <strong>' + alimento.name + '</strong>!!!  para
agregar a su dieta.',
buttons: [
      {
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
this.presentToast('Usted no agregado Alimento a su Dieta...');
        }
      }, {
text:'Ok',
handler:async () => {
console.log('Confirm Okay', this.valoresCambio.dietaBaja);
if (alimento.sodio>4) {
this.presentAlertVolores(alimento, this.valoresCambio.dietaBaja)
          }//////////////////////////
else {
this.guardarDieta(alimento);
          }
        }
      }
    ]
  });

awaitalert.present();
}//

asyncpresentAlertDietaCarb(alimento: any) {
constalert = awaitthis.alertCtrl.create({
header:'Aviso ',
subHeader:'Dieta a Agregar: ',
message:'Usted a seleccionado  <strong>' + alimento.name + '</strong>!!!  para
agregar a su dieta.',
```

```
buttons: [
      {
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
this.presentToast('Usted no agregado Alimento a su Dieta...');
        }
      }, {
text:'Ok',
handler:async () => {
console.log('Confirm Okay', this.valoresCambio.dietaBaja);
if (alimento.carbohidrato>5) {
this.presentAlertVolores(alimento, this.valoresCambio.dietaBaja)
          }//////////////////////////
else {
this.guardarDieta(alimento);
          }
        }
      }
    ]
  });

awaitalert.present();
 }//
}
```

ANEXO 1.2 DIETA-DIARIA-PACIENTE.PAGE.TS

```
import{ Component, OnInit } from'@angular/core';
import*asmomentfrom'moment-timezone';
import{ BaseAlimentosService } from'../../services/base-alimentos.service';
import{ AuthService } from'../../services/auth.service';
import{ TotalDietas } from'../../models/alimento';
```

```
import{ ToastController, AlertController } from'@ionic/angular';
import{ UserService } from'../../services/user.service';

@Component({
selector:'app-dieta-diaria-paciente',
templateUrl:'./dieta-diaria-paciente.page.html',
styleUrls: ['./dieta-diaria-paciente.page.scss'],
})
exportclassDietaDiariaPacientePageimplementsOnInit {

momentjs: any = moment;
idUser: string;
fechaDieta: Date = newDate;
fechaMax: Date = newDate;
fechaMin: any;
fM: any;
fecha: any;
fechaTransformada: any;
totalDietas: any;
totalPotasio: number = 0;
totalFosforo: number = 0;
totalSodio: number = 0;
totalCarbohirato: number = 0;
leght: number;
fechahoy: any;
total: any;
guardarDia: any;
totalLenght: number;
maxSodio: number;
maxCar: number;
guardarmes: any;
guadaranomes: any;
guardarano: any;
filtros: any[] = [];
```

```
comida: string;
totalC: any;
totalF: any;
totalP: any;
totalS: any;
time: any;


constructor(privateauthservice: AuthService,
privatebaseAlimentos: BaseAlimentosService,
privatetoastCtrl: ToastController,
publicalertCtrl: AlertController,
privateuserservice: UserService) { }

ngOnInit() {
this.authservice.verificarUser().subscribe(id=> {
console.log('IdUser dieta diaria:', this.idUser = id);
this.userservice.getPaciente(this.idUser).subscribe(paciente=>{
paciente;
this.fM = paciente
this.fechaMin = this.fM.date
console.log('fecha creacion paciente:', this.fechaMin);


})
   })
  }

asyncpresentToast(message: string) {
consttoast = awaitthis.toastCtrl.create({
message,
duration:3000
   });
toast.present();
  }
```

```
cambiFecha(event) {
this.filtros = [];
this.momentjs.locale('es');
console.log('IonChange:', event.detail.value);
this.fecha = (event.detail.value);
//console.log('FechaDieta:', this.fecha);
this.fechaTransformada = moment(this.fecha).format('YYYY-MMMM-DD');
this.baseAlimentos.getFechasDietas(this.idUser,
this.fechaTransformada).subscribe(dietas=> {
this.totalDietas = dietas;
this.totalPotasio = 0; this.totalFosforo = 0; this.totalSodio = 0;
this.totalCarbohirato = 0;
this.totalLenght = this.totalDietas.length
if (this.totalLenght>= 1) {
console.log('TotalLength', this.totalLenght)
this.totalDietas.forEach(valores=> {
console.log('Valores fechas:', valores.fecha)
//console.log('Valores dietaas:', valores)
this.totalPotasio = this.totalPotasio + valores.potasio;
this.totalFosforo = this.totalFosforo + valores.fosforo;
this.totalSodio = this.totalSodio + valores.sodio;
this.totalCarbohirato = this.totalCarbohirato + valores.carbohidrato;
this.guardarDia = valores.numDia;
this.guardarano = valores.ano;
this.guardarmes = valores.mes
console.log('Guardar', valores)
console.log('GuardarDia', this.guardarDia, this.guardarano, this.guardarmes)
      });
this.fecha = newDate;
this.fechahoy = moment().format('YYYY-MMMM-DD')
vardia = moment().date();
//var mes = moment().month();
varano = moment().year();
```

```
varweekday = newArray(7);
weekday[0] = "Domingo";
weekday[1] = "Lunes";
weekday[2] = "Martes";
weekday[3] = "Miercoles";
weekday[4] = "Jueves";
weekday[5] = "Viernes";
weekday[6] = "Sabado";

varnombreDia = weekday[this.fecha.getDay()];
varmonth = newArray();
month[0] = "enero";
month[1] = "febrero";
month[2] = "marzo";
month[3] = "abril";
month[4] = "mayo";
month[5] = "junio";
month[6] = "julio";
month[7] = "agosto";
month[8] = "semtiembre";
month[9] = "octubre";
month[10] = "noviembre";
month[11] = "diciembre";
varmes = month[this.fecha.getMonth()];
varanomes = ano + '-' + mes
//console.log('Ano', ano, 'Mes:', mes, 'Dia', dia - 1)
console.log('fecha hoy:', this.fechahoy);
constdietaTotal: TotalDietas = {
totalPotasio:this.totalPotasio,
totalFosforo:this.totalFosforo,
totalSodio:this.totalSodio,
totalCarbohidrato:this.totalCarbohirato,
fechaDietaTotal:this.fechaTransformada,
id:this.idUser,
```

```
ano:this.guardarano,
mes:this.guardarmes,
dia:this.guardarDia,
anomes:this.guardarano + '-' + this.guardarmes
      }
console.log('dia guardado', dietaTotal)
this.baseAlimentos.addDiariaTotal(dietaTotal, this.idUser)
    } else {
this.presentToast('No hay Alimentos en esta Fecha...');
console.log('No existe esa fecha')
    }

this.mostrarAlert(this.totalCarbohirato, this.totalFosforo, this.totalPotasio,
this.totalSodio);
console.log('Dietas: ', dietas)
  })//////////////////////////////////////////
//this.fecha.moment.format('YYYY MMMMM DD')

 }
verDesayuno(comida){
this.comida = comida;
this.filtros = [];
this.filtros = this.totalDietas.filter(function (dietas) {
returndietas.comida === comida
  })
this.totalC = 0; this.totalF = 0; this.totalP = 0; this.totalS = 0;
this.filtros.forEach( filtro=>{
this.totalC = this.totalC + filtro.carbohidrato;
this.totalF = this.totalF + filtro.fosforo;
this.totalP = this.totalP + filtro.potasio;
this.totalS = this.totalS + filtro.sodio;
})
console.log('Comida:', this.filtros)
}
```

```
mostrarAlert(mc: number, mf: number, mp: number, ms: number) {
this.maxSodio = 2000;
this.maxCar = 1172;
varmaxfos = 1200;
varmaxpo = 2100;
if (mc>this.maxCar) {
this.presentAlertCar(mc, this.maxCar);
   }
if (mf>maxfos) {
this.presentAlertFos(mf, maxfos);
   }
if (mp>maxpo) {
this.presentAlertPo(mp, maxpo);
   }
if (ms>this.maxSodio) {
this.presentAlertSodio(ms, this.maxSodio);
   }

 }

asyncpresentAlertCar(car: any, valor: number) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'CARBOHIDRATO al Maximo ',
message:'Usted se ha exedido en el consumo de CARBOHIDRATO diario que es
de '+ valor +' mg. controle su dieta, ' + '  ' + 'valor Atual: ' + car + ' ' + 'mg.',
buttons: ['OK']
   });

awaitalert.present();
 }
```

```
asyncpresentAlertFos(fos: any, valor: number) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'FOSFORO al Maximo ',
message:'Usted se ha exedido en el consumo de FOSFORO diario que es de' +
valor + ' mg. controle su dieta, ' + '  ' + 'valor Atual: ' + fos + ' ' + 'mg.',
buttons: ['OK']
    });

awaitalert.present();
  }

asyncpresentAlertPo(po: any, valor: number) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'POTASIO al Maximo ',
message:'Usted se ha exedido en el consumo de POTASIO diario que es de'
+valor + ' mg. controle su dieta, ' + '  ' + 'valor Atual: ' + po + ' ' + 'mg.',
buttons: ['OK']
    });

awaitalert.present();
  }

asyncpresentAlertSodio(so: any, valor: number) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'SODIO al Maximo ',
message:'Usted se ha exedido en el consumo de SODIO diario que es de'+ valor+'
mg. controle su dieta, ' + '  ' + 'valor Atual: ' + so + ' ' + 'mg.',
buttons: ['OK']
    });

awaitalert.present();
```

```
}

}
```

ANEXO 1.3 ALIMENTO-SELECIONADO.PAGE.TS

```
import{ Component, OnInit } from'@angular/core';
import{ UserService } from'../../services/user.service';
import{ AuthService } from'../../services/auth.service';
import{ ModalController, AlertController, ToastController }
from'@ionic/angular';
import{ AlimentoFecha } from'../../models/alimento';
import{ BaseAlimentosService } from'../../services/base-alimentos.service';
import{ Subscription } from'rxjs';

@Component({
selector:'app-alimento-selecionado',
templateUrl:'./alimento-selecionado.page.html',
styleUrls: ['./alimento-selecionado.page.scss'],
})
exportclassAlimentoSelecionadoPageimplementsOnInit {
comida: string;

deleteSubcripcion: Subscription;
dietaSubcription: Subscription;


idUser: string;
category: string;
listaAlimentos: any;
valoresCambio: any;
fecha: any;
```

```
time: any;
fechaActual: any;
potasioPush: any[] = [];
dietaBaja: any;
totalPotasio: any;
totalFosforo: any;
totalSodio: any;
publictotalCarbohidrato: any;
lenghtDietas: number;
publictotalDietas: any;
publictotalDieta: any;
listaComidas: any;
listaComidaslength: number;
tCvalor: number;
tFvalor: number;
tPvalor: number;
tSvalor: number;

constructor(privateuserService: UserService,
privateauthservice: AuthService,
privatealertCtrl: AlertController,
privatetoastCtrl: ToastController,
privatebaseAlimentos: BaseAlimentosService) { }

ngOnInit() {
this.authservice.verificarUser().subscribe(id=> {
this.idUser = id;
this.userService.getPacienteTratante(this.idUser).subscribe(paciente=> {
paciente.map(p=> {
this.valoresCambio = p
this.dietaBaja = this.valoresCambio.dietaBaja;
console.log('Dieta Paciente:', this.valoresCambio.dietaBaja)
      });
```

```
    })
this.baseAlimentos.getAlimentosDietaDiaria(this.idUser).subscribe(alimentos=>
{
this.totalDieta = alimentos;
this.lenghtDietas = alimentos.length;
console.log('Length:', this.lenghtDietas)

    })
   })
 }

asynccrearDietas(comida: string, tC: number, tF: number, tP: number, tS: number)
{
this.comida = comida;
this.tCvalor = tC; this.tFvalor = tF; this.tPvalor = tP; this.tSvalor = tS
this.dietaSubcription = awaitthis.baseAlimentos.getAlimentosComida(this.idUser,
comida).subscribe(comidas=> {
this.listaComidas = comidas;
this.listaComidaslength = comidas.length;
this.totalCarbohidrato = 0;
this.totalFosforo = 0;
this.totalPotasio = 0;
this.totalSodio = 0;
comidas.forEach(co=> {
this.totalCarbohidrato = this.totalCarbohidrato + co.carbohidrato;
this.totalFosforo = this.totalFosforo + co.fosforo;
this.totalPotasio = this.totalPotasio + co.potasio;
this.totalSodio = this.totalSodio + co.sodio;
    })
console.log('Alimentos Comidas:', this.totalCarbohidrato, this.totalFosforo,
this.totalPotasio, this.totalSodio)
this.mostrarAlert(this.totalCarbohidrato, tC, this.totalFosforo, tF, this.totalPotasio,
tP, this.totalSodio, tS)
//this.dietaSubcription.unsubscribe()
```

```
  })

 }

asyncmostrarAlert(car: number, c: number, fo: number, f: number, po: number, p:
number, so: number, s: number) {
if (car>= c) {
awaitthis.presentAlertCar(car, fo, f, po, p, so, s)
  } else {
if (fo>= f) {
awaitthis.presentAlertFosforo(fo, po, p, so, s)
    } else {
if (po>= p) {
awaitthis.presentAlertPotasio(po, so, s)
      } else {
if (so>= s) {
awaitthis.presentAlertSodio(so)
        }
      }
    }
  }
 }


asyncalimentosGuardar() {
this.totalDieta.forEach(asyncdieta=> {
constdataFecha: AlimentoFecha = {
category:dieta.category,
name:dieta.name,
potasio:dieta.potasio,
fosforo:dieta.fosforo,
sodio:dieta.sodio,
carbohidrato:dieta.carbohidrato,
```

```
imageLink:dieta.imageLink,
fecha:dieta.fecha,
ano:dieta.ano,
mes:dieta.mes,
dia:dieta.dia,
numDia:dieta.numDia,
hora:dieta.hora,
comida:this.comida,
preparacion:dieta.preparacion
// minutos: dieta.minutos
     }
awaitthis.baseAlimentos.addDietasxFechas(dataFecha, this.idUser)
console.log('Lista de alimentos a guardar', dataFecha);
   })
this.deleteSubcripcion =
awaitthis.baseAlimentos.getAlimentosDietaDiaria(this.idUser).subscribe(dietasDi
arias=> {
console.log('Lista de alimentos a eliminar', dietasDiarias);
dietasDiarias.forEach(asyncdiarias=> {
awaitthis.baseAlimentos.deleteDietaDiaria(this.idUser, diarias.id);
})
this.deleteSubcripcion.unsubscribe();
   });
 }

asyncpresentAlertCar(car: number, fo: number, f: number, po: number, p: number,
so: number, s: number) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'Carbohidrato al Maximo ',
message:'Usted se ha exedido en el consumo de CARBOHIDRATO diario que es
de 2000 mg. controle su dieta, valor Actual' + '  ' + car + '  ' + 'mg.',
buttons: [{
text:'Cancel',
```

```
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
console.log('Cancelar');
        }
      },
      {
text:'Ok',
role:'ok',
cssClass:'secondary',
handler: (blah) => {
console.log('Ok');
if (fo>f) {
this.presentAlertFosforo(fo, po, p, so, s);
          } else {
if (po>p) {
this.presentAlertPotasio(po, so, s)
            } else {
if (so>s) {
this.presentAlertSodio(so)
              } else {
//this.alimentosGuardar()
              }


            }
          }
        }
      },]
    });

awaitalert.present();
  }
```

```
asyncpresentAlertFosforo(fo: number, po: number, p: number, so: number, s:
number) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'Fosforo al Maximo ',
message:'Usted se ha exedido en el consumo de FOSFORO diario que es de 1200
mg. controle su dieta, valor Actual' + '  ' + fo + '  ' + 'mg.',
buttons: [{
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
console.log('Cancelar');
      }
     },
     {
text:'Ok',
role:'ok',
cssClass:'secondary',
handler: (blah) => {
console.log('Ok');
if (po>p) {
this.presentAlertPotasio(po, so, s);
        } else {
if (so>s) {
this.presentAlertSodio(so)
          } else {
// this.alimentosGuardar()
          }

        }
       }
     },]
   });
```

```
awaitalert.present();
  }

asyncpresentAlertPotasio(po: number, so: number, s: number) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'Potasio al Maximo ',
message:'Usted se ha exedido en el consumo de POTASIO diario que es de 2100
mg. controle su dieta, valor Actual' + '  ' + po + '  ' + 'mg.',
buttons: [{
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
console.log('Cancelar');
        }
      },
      {
text:'Ok',
role:'ok',
cssClass:'secondary',
handler: (blah) => {
console.log('Ok');
if (so>s) {
this.presentAlertSodio(so);
          } else {
//this.alimentosGuardar()
          }
        }
      },]
    });

awaitalert.present();
```

```
  }

asyncpresentAlertSodio(so: number) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'Sodio al Maximo ',
message:'Usted se ha exedido en el consumo de SODIO diario que es de 2000 mg.
controle su dieta, valor Actual' + '  ' + so + '  ' + 'mg.',
buttons: [{
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
console.log('Cancelar');
        }
      },
      {
text:'Ok',
role:'ok',
cssClass:'secondary',
handler: (blah) => {
console.log('Ok');
//this.alimentosGuardar()
        }
      },]
    });

awaitalert.present();
}

guardarDietaDiaria() {
this.presentAlertNotification(this.comida)

  }
```

```
asyncpresentAlertNotification(comida: string) {
constalert = awaitthis.alertCtrl.create({
header:'Aviso',
subHeader:'Guardar Dieta ',
message:'Usted va a agregar estos alimentos a su: ' + '  ' + comida,
buttons: [{
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
console.log('Cancelar');
      }
     },
     {
text:'Ok',
role:'ok',
cssClass:'secondary',
handler:async (blah) => {
console.log('Ok');
this.totalCarbohidrato = 0;
this.totalFosforo = 0;
this.totalPotasio = 0;
this.totalSodio = 0;
this.dietaSubcription =
awaitthis.baseAlimentos.getAlimentosDietaDiaria(this.idUser).subscribe(alimento
s=> {
this.totalDieta = alimentos;
alimentos.forEach(dieta=> {
this.totalCarbohidrato = this.totalCarbohidrato + dieta.carbohidrato;
this.totalFosforo = this.totalFosforo + dieta.fosforo;
this.totalPotasio = this.totalPotasio + dieta.potasio;
this.totalSodio = this.totalSodio + dieta.sodio;
this.totalDietas = dieta
```

```
          });
// console.log('getAlimentos:', this.totalCarbohidrato, this.totalFosforo);
this.mostrarAlert1(this.totalCarbohidrato, this.tCvalor, this.totalFosforo,
this.tFvalor, this.totalPotasio, this.tPvalor, this.totalSodio, this.tSvalor)
this.dietaSubcription.unsubscribe();

        })
//this.alimentosGuardar();

}
    },]
  });

awaitalert.present();
 }

asyncmostrarAlert1(car: number, c: number, fo: number, f: number, po: number,
p: number, so: number, s: number) {
if (car>= c) {
awaitthis.presentAlertCar1(car, fo, f, po, p, so, s)
  } else {
if (fo>= f) {
awaitthis.presentAlertFosforo1(fo, po, p, so, s)
    } else {
if (po>= p) {
awaitthis.presentAlertPotasio1(po, so, s)
      } else {
if (so>= s) {
awaitthis.presentAlertSodio1(so)
        } else {
awaitthis.alimentosGuardar();
        }
      }
    }
```

```
  }
 }

asyncpresentAlertCar1(car: number, fo: number, f: number, po: number, p:
number, so: number, s: number) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'Carbohidrato al Maximo ',
message:'Usted se ha exedido en el consumo de CARBOHIDRATO diario que es
de 2000 mg. controle su dieta, valor Actual' + '  ' + car + '  ' + 'mg.',
buttons: [{
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
console.log('Cancelar');
      }
     },
     {
text:'Ok',
role:'ok',
cssClass:'secondary',
handler: (blah) => {
console.log('Ok');
if (fo>f) {
this.presentAlertFosforo1(fo, po, p, so, s);
        } else {
if (po>p) {
this.presentAlertPotasio1(po, so, s)
          } else {
if (so>s) {
this.presentAlertSodio1(so)
            } else {
this.alimentosGuardar()
```

```
            }

          }
        }
      }
    },]
  });

awaitalert.present();
 }

asyncpresentAlertFosforo1(fo: number, po: number, p: number, so: number, s:
number) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'Fosforo al Maximo ',
message:'Usted se ha exedido en el consumo de FOSFORO diario que es de 2000
mg. controle su dieta, valor Actual' + '  ' + fo + '  ' + 'mg.',
buttons: [{
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
console.log('Cancelar');
      }
    },
    {
text:'Ok',
role:'ok',
cssClass:'secondary',
handler: (blah) => {
console.log('Ok');
if (po>p) {
this.presentAlertPotasio1(po, so, s);
```

```
        } else {
if (so>s) {
this.presentAlertSodio1(so)
          } else {
this.alimentosGuardar()
          }

        }
       }
     },]
   });

awaitalert.present();
  }

asyncpresentAlertPotasio1(po: number, so: number, s: number) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'Potasio al Maximo ',
message:'Usted se ha exedido en el consumo de POTASIO diario que es de 2000
mg. controle su dieta, valor Actual' + '  ' + po + '  ' + 'mg.',
buttons: [{
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
console.log('Cancelar');
       }
     },
     {
text:'Ok',
role:'ok',
cssClass:'secondary',
handler: (blah) => {
```

```
console.log('Ok');
if (so>s) {
this.presentAlertSodio1(so);
        }
else {
this.alimentosGuardar()
        }
       }
     },]
   });

awaitalert.present();
 }

asyncpresentAlertSodio1(so: any) {
constalert = awaitthis.alertCtrl.create({
header:'Alerta',
subHeader:'Sodio al Maximo ',
message:'Usted se ha exedido en el consumo de SODIO diario que es de 2000 mg.
controle su dieta, valor Actual' + '  ' + so + '  ' + 'mg.',
buttons: [{
text:'Cancel',
role:'cancel',
cssClass:'secondary',
handler: (blah) => {
console.log('Cancelar');
       }
     },
     {
text:'Ok',
role:'ok',
cssClass:'secondary',
handler:  (blah) => {
console.log('Ok');
```

```
this.alimentosGuardar()
        }
      },]
    });

awaitalert.present();
  }


asynceliminarDietaDiaria(dieta: any) {
console.log('Alimento para Eliminar:', this.idUser, dieta.id);
awaitthis.baseAlimentos.deleteDietas(this.idUser, dieta.id);
  }
}
```

ANEXO 1.4 EVOLUCION-PACINETE.PAGE.TS

```
import { Component, OnInit } from'@angular/core';
import { AuthService } from'../../services/auth.service';
import { BaseAlimentosService } from'../../services/base-alimentos.service';
import { UserService } from'../../services/user.service';
import { Chart } from'chart.js';
import*asmomentfrom'moment-timezone';
import { Subscription } from'rxjs';

@Component({
selector:'app-evolucion-pacinete',
templateUrl:'./evolucion-pacinete.page.html',
styleUrls: ['./evolucion-pacinete.page.scss'],
})
exportclassEvolucionPacinetePageimplementsOnInit {
cerrarSubscripton: Subscription;
momentjs: any = moment;
fechaDieta: Date = newDate;
fechaMax: Date = newDate;
idUser: string;
```

```
fechasxDietas: any;
fechasPush: any[] = [];
fechasTotales: any;
potasioPush: any[] = [];
fosforoPush: any[] = [];
sodioPush: any[] = [];
carbohidratoPush: any[] = [];
fechaExacta: any[] = [];
filtros: any[] = [];
fecha: any;
fechaTransformada: any;
varCar: any[] = [];
mayor: number;
fechaanomes: string;
fM: any;
fechaMin: any

constructor(privateauthservice: AuthService,
privatebaseService: BaseAlimentosService,
privateuserservice: UserService) { }

ngOnInit() {
this.authservice.verificarUser().subscribe(idUser=> {
this.idUser = idUser;
this.userservice.getPaciente(this.idUser).subscribe(paciente=>{
paciente;
this.fM = paciente
this.fechaMin = moment(this.fM.date).format('YYYY-MM');
console.log('fecha creacion paciente:', this.fechaMin);

    })
  })
 }
```

```
cambiarFecha(event) {
this.filtros = [];
this.fechasPush = [];
//this.carbohidratoPush = [];
this.cerrarSubscripton =
this.baseService.getDietasxFechas(this.idUser).subscribe(fechasDietas=> {
this.momentjs.locale('es');
this.fechaanomes = moment(event.detail.value).format('YYYY-MMMM')
console.log('IonChange:', this.fechaanomes);
fechasDietas.forEach(fechas=> {
this.fechasPush.push(fechas.fecha)
})
this.filtros = this.fechasPush.filter(function (fechas) {
returnfechas.anomes === moment(event.detail.value).format('YYYY-MMMM')
})
console.log('fechas push', this.filtros)
this.cerrarSubscripton.unsubscribe();
   })
 }

verFechas() {
this.fechasPush.forEach(totales=> {
this.fechaExacta.push(totales.anomes)
})
console.log('fechas ano mes', this.fechaExacta)

 }

verCarbohidrato() {
varmcarbo = 'Carbohidrato';
this.varCar = [];
this.carbohidratoPush = [];
this.filtros.forEach(totales=> {
this.carbohidratoPush.push(totales.totalCarbohidrato)
```

```
this.varCar.push(totales.dia)
   })
this.mayor = 0;
varcarLength = this.carbohidratoPush.length
console.log('Car length:', carLength)
for (vari = 0; i<carLength; i++) {
if (this.carbohidratoPush[i] >this.mayor) {
this.mayor = this.carbohidratoPush[i]
     }
//console.log('Numeros:', this.carbohidratoPush[i])
}
console.log('el mayor es::', this.mayor)
console.log('Totales Carbodrato', this.carbohidratoPush)
console.log('Totales Carbodrato Dias', this.varCar)
varctx = (<any>document.getElementById('myChart')).getContext('2d');
varmyChart = newChart(ctx, {
type:'line',
data: {
labels:this.varCar,
datasets: [{
label: ['Valor Max Carbohidrato x dia 2000 mg.'],
data:this.carbohidratoPush,

backgroundColor: [
'rgba(255, 99, 132, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(255, 99, 132, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
```

```
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(255, 99, 132, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(255, 99, 132, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(255, 99, 132, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(255, 99, 132, 0.2)',
'rgba(54, 162, 235, 0.2)',


        ],
borderColor: [
'rgba(255, 99, 132, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(255, 99, 132, 1)',
'rgba(54, 162, 235, 1)',
```

```
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(255, 99, 132, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(255, 99, 132, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(255, 99, 132, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(255, 99, 132, 1)',
'rgba(54, 162, 235, 1)',

],
borderWidth:3
    }]
   },
options: {
scales: {
yAxes: [{
ticks: {
max:this.mayor + 100,
```

```
min:0,
stepSize:100
          }
        }]
      },
title: {
display:true,
text:'Valores Mesuales: ' + ' ' + mcarbo + ' ' + this.fechaanomes,
fontColor:'#9966cc',
fontSize:20
      },
animation: {
duration:3000// general animation time
      },
hover: {
animationDuration:3000// duration of animations when hovering an item
      },
responsiveAnimationDuration:3000,
layout: {
padding: {
left:10,
right:10,
top:5,
bottom:5
        }
      },
legend: {
display:true,
labels: {
fontColor:'#9966cc',
        }
      }
    }
  });
```

```
  }

verfosforo() {
varmfosforo = 'Fosforo';
this.varCar = [];
this.fosforoPush = [];
varlegnthFosforo = this.fechasPush.length;
console.log('lenght Fosforo', legnthFosforo)
this.fosforoPush = [];
this.filtros.forEach(totales=> {
this.fosforoPush.push(totales.totalFosforo)
this.varCar.push(totales.dia)
   })
this.mayor = 0;
varcarLength = this.fosforoPush.length
console.log('Car length:', carLength)
for (vari = 0; i<carLength; i++) {
if (this.fosforoPush[i] >this.mayor) {
this.mayor = this.fosforoPush[i]
     }
//console.log('Numeros:', this.carbohidratoPush[i])
}
console.log('el mayor es::', this.mayor)
console.log('Totales Fosforo', this.fosforoPush)
varctx = (<any>document.getElementById('myChart')).getContext('2d');
varmyChart = newChart(ctx, {
type:'line',
data: {
//labels: ['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sabado', 'Domingo'],
labels:this.varCar,
datasets: [{
label: ['Valor Max Fosforo x dia 2000 mg. '],
data:this.fosforoPush,
```

```
backgroundColor: [
'rgba(0,204,255, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(0,204,255, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(0,204,255, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(0,204,255, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(0,204,255, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(0,204,255, 0.2)',
'rgba(54, 162, 235, 0.2)',
```

```
],
borderColor: [
'rgba(51, 204, 204, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(51, 204, 204, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(51, 204, 204, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(51, 204, 204, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(51, 204, 204, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(51, 204, 204, 1)',
'rgba(54, 162, 235, 1)',
```

```
          ],
borderWidth:3
        }]
      },
options: {
scales: {
yAxes: [{
ticks: {
max:this.mayor + 100,
min:0,
stepSize:100
           }

         }]
        },
title: {
display:true,
text:'Valores Mesuales: ' + ' ' + mfosforo + ' ' + this.fechaanomes,
fontColor:'#9966cc',
fontSize:20
        },
animation: {
duration:5000// general animation time
        },
hover: {
animationDuration:5000// duration of animations when hovering an item
        },
responsiveAnimationDuration:5000,
layout: {
padding: {
left:10,
right:10,
top:5,
```

```
bottom:5
        }
      },
legend: {
display:true,
labels: {
fontColor:'#9966cc',
        }
      },
    }
  });
 }

verPotasio() {
varmpotasio = 'Potasio';
this.varCar = [];
this.potasioPush = [];
this.filtros.forEach(totales=> {
this.potasioPush.push(totales.totalPotasio)
this.varCar.push(totales.dia)
  })
this.mayor = 0;
varcarLength = this.potasioPush.length
console.log('Car length:', carLength)
for (vari = 0; i<carLength; i++) {
if (this.potasioPush[i] >this.mayor) {
this.mayor = this.potasioPush[i]
    }
//console.log('Numeros:', this.carbohidratoPush[i])
}
console.log('el mayor es::', this.mayor)
console.log('Totales Potasio', this.potasioPush)
varctx = (<any>document.getElementById('myChart')).getContext('2d');
varmyChart = newChart(ctx, {
```

```
type:'line',
data: {
//labels: ['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sabado', 'Domingo'],
labels:this.varCar,
datasets: [{
label: ['Valor Max Potasio x dia 2000 mg.'],
data:this.potasioPush,

backgroundColor: [
'rgba(255, 204, 153, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(255, 204, 153, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(255, 204, 153, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(255, 204, 153, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(255, 204, 153, 0.2)',
```

```
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(255, 204, 153, 0.2)',
'rgba(54, 162, 235, 0.2)',

        ],
borderColor: [
'rgba(255, 204, 0, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(255, 204, 0, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(255, 204, 0, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(255, 204, 0, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
```

```
'rgba(255, 204, 0, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(255, 204, 0, 1)',
'rgba(54, 162, 235, 1)',

],
borderWidth:3
      }]
    },
options: {
scales: {
yAxes: [{
ticks: {
max:this.mayor + 100,
min:0,
stepSize:100
        }
       }]
      },
title: {
display:true,
text:'Valores Mesuales:' + ' ' + mpotasio + ' ' + this.fechaanomes,
fontColor:'#9966cc',
fontSize:20
      },
animation: {
duration:5000// general animation time
      },
hover: {
animationDuration:5000// duration of animations when hovering an item
```

```
        },
responsiveAnimationDuration:5000,
layout: {
padding: {
left:10,
right:10,
top:5,
bottom:5
          }
        },
legend: {
display:true,
labels: {
fontColor:'#9966cc',
          }
        }
      }
    });
  }

verSodio() {
varmsodio = 'Sodio';
this.varCar = [];
this.sodioPush = [];
this.filtros.forEach(totales=> {
this.sodioPush.push(totales.totalSodio)
this.varCar.push(totales.dia)
    })
this.mayor = 0;
varcarLength = this.sodioPush.length
console.log('Car length:', carLength)
for (vari = 0; i<carLength; i++) {
if (this.sodioPush[i] >this.mayor) {
this.mayor = this.sodioPush[i]
```

```
    }
//console.log('Numeros:', this.carbohidratoPush[i])
}
console.log('el mayor es::', this.mayor)
console.log('Totales Sodio', this.sodioPush)
varctx = (<any>document.getElementById('myChart')).getContext('2d');
varmyChart = newChart(ctx, {
type:'line',
data: {
//labels: ['Lunes', 'Martes', 'Miercoles', 'Jueves', 'Viernes', 'Sabado', 'Domingo'],
labels:this.varCar,
datasets: [{
label: ['Valor Max Sodio x dia 2000 mg. '],
data:this.sodioPush,

backgroundColor: [
'rgba(51, 153, 102, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(255, 159, 64, 0.2)',
'rgba(51, 153, 102, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(51, 153, 102, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(51, 153, 102, 0.2)',
'rgba(54, 162, 235, 0.2)',
```

```
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(51, 153, 102, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
'rgba(51, 153, 102, 0.2)',
'rgba(54, 162, 235, 0.2)',
'rgba(255, 206, 86, 0.2)',
'rgba(75, 192, 192, 0.2)',
'rgba(153, 102, 255, 0.2)',
        ],
borderColor: [
'rgba(0, 128, 0, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(255, 159, 64, 1)',
'rgba(0, 128, 0, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(0, 128, 0, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(0, 128, 0, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
```

```
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(0, 128, 0, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
'rgba(0, 128, 0, 1)',
'rgba(54, 162, 235, 1)',
'rgba(255, 206, 86, 1)',
'rgba(75, 192, 192, 1)',
'rgba(153, 102, 255, 1)',
          ],
borderWidth:3
        }]
      },
options: {
scales: {
yAxes: [{
ticks: {
max:this.mayor + 100,
min:0,
stepSize:100
            }
          }]
        },
title: {
display:true,
text:'Valores Mesuales:' + ' ' + msodio + ' ' + this.fechaanomes,
fontColor:'#9966cc',
fontSize:20
        },
animation: {
duration:5000// general animation time
```

```
        },
hover: {
animationDuration:5000// duration of animations when hovering an item
},
responsiveAnimationDuration:5000,
layout: {
padding: {
left:10,
right:10,
top:5,
bottom:5
          }
        },
legend: {
display:true,
labels: {
fontColor:'#9966cc',
          }
        }
      }
    });
  }


}
```

ANEXO 2: ENCUESTAS

1. ¿Usted dispone de teléfono celular personal?

   Si

   No

2. ¿Usted sabe manipular aplicaciones móviles?

   Si

   No

3. En caso de haber respondido en la pregunta anterior NO. ¿Usted tiene quien le ayude a manipular una aplicación?

   Si

   No

4. ¿Usted sabe o conoce de alguna aplicación para el control de alimentos en pacientes con insuficiencia renal?

   Si

   No

5. ¿Usted en su control mensual con el médico tratante en los últimos 3 meses ha tenido valores elevados en algún electrolito (Carbohidrato, Fosforo, Potasio, Sodio)?

   Si

   No

6. En caso de existir una aplicación que ayude a controlar su dieta, ¿Usted la utilizaría?

   Si

   No

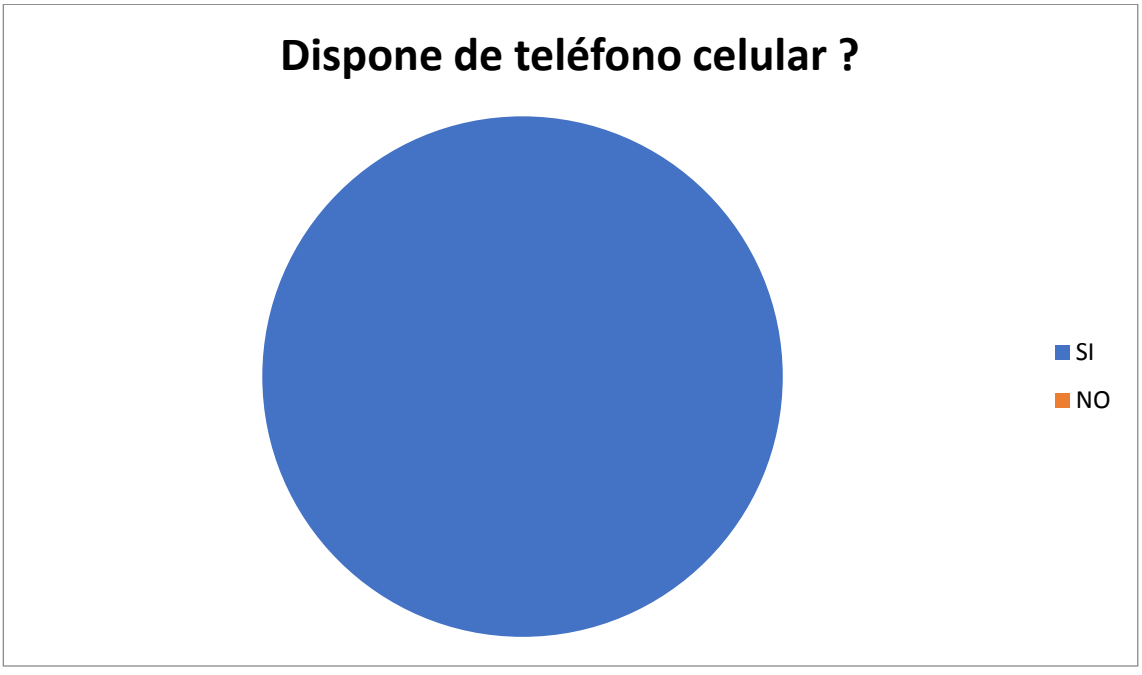
Encuesta

El numero de encuestados fue de 100 pacientes de la unidad renal Baxter Cuenca S.A. las respuestas de acuerdo a cada pregunta fueron las siguientes.

1. ¿Usted dispone de teléfono celular personal?

RESPUESTA	FRECUENCIA	PORCENTAJE
SI	100	100%
NO	0	0%
TOTAL	100	100%

Tabulación de la Pregunta 1.



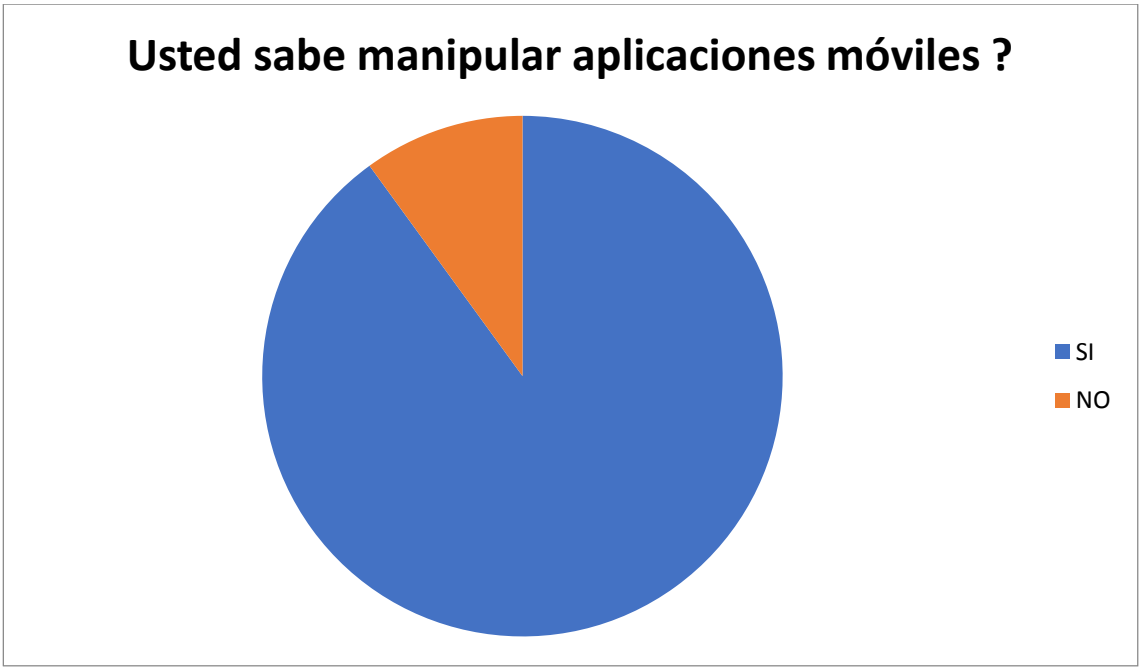
(Autores, 2020)

Gráfico, pregunta #1 de la encuesta.

2. ¿Usted sabe manipular aplicaciones móviles?

RESPUESTA	FRECUENCIA	PORCENTAJE
SI	90	90%
NO	10	10%
TOTAL	100	100%

Tabulación de la Pregunta 2.



(Autores, 2020)

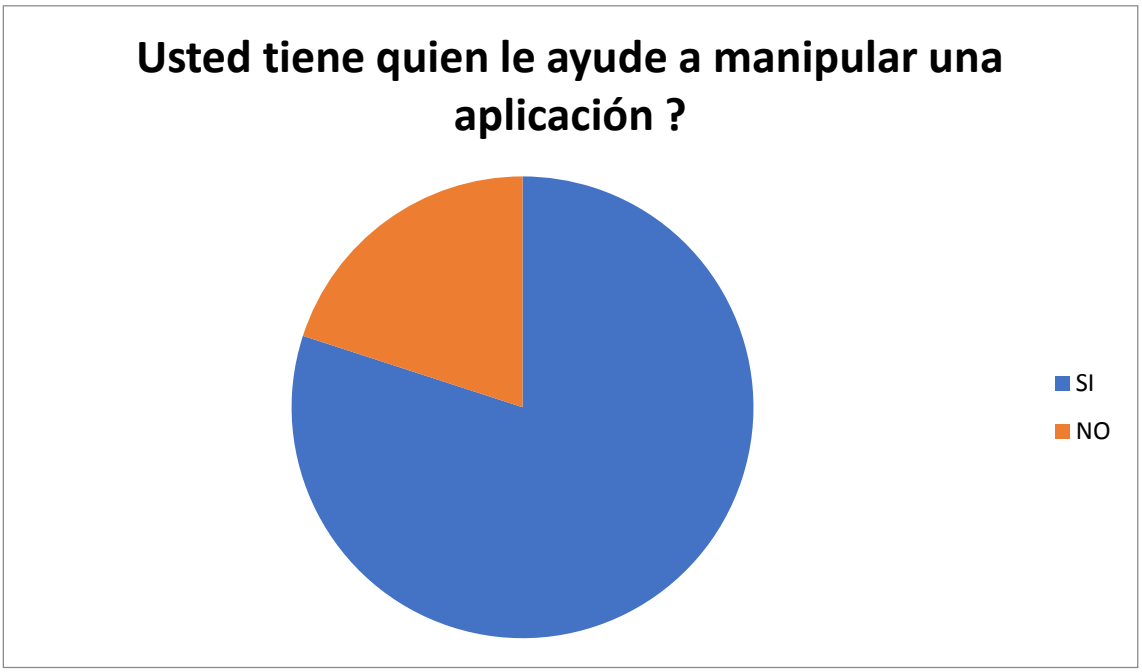
Gráfico, pregunta #2 de la encuesta.

3. En caso de haber respondido en la pregunta anterior NO. ¿Usted tiene quien le ayude a manipular una aplicación?

RESPUESTA	FRECUENCIA	PORCENTAJE

SI	80	80%
NO	20	20%
TOTAL	100	100%

Tabulación de la Pregunta 3.



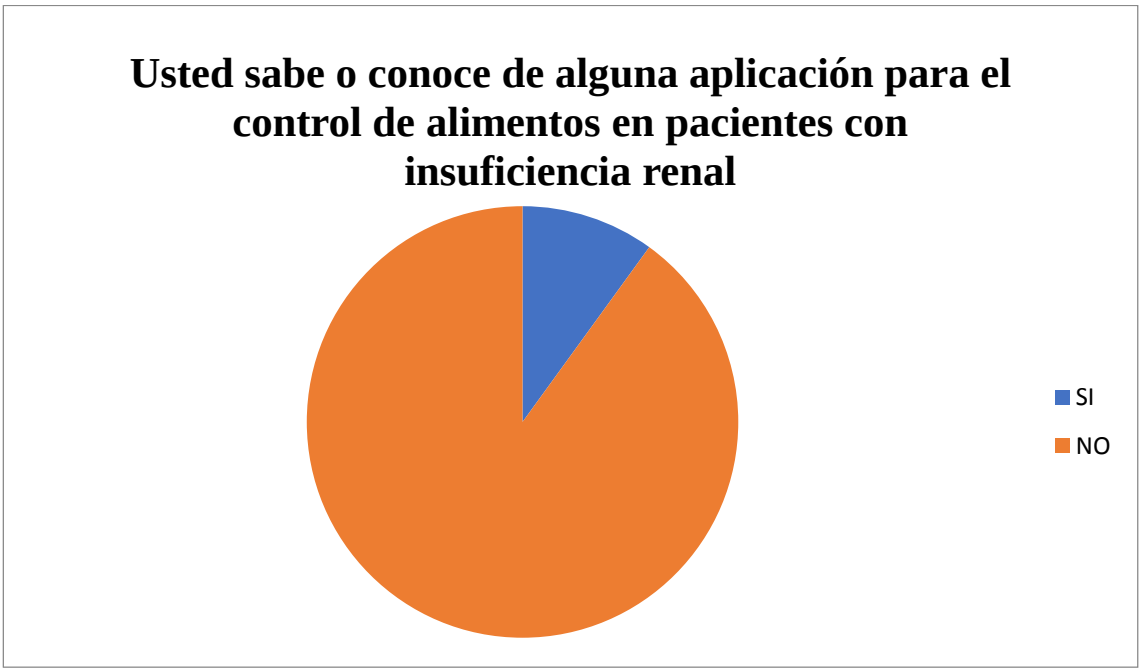
(Autores, 2020)

Gráfico, pregunta #3 de la encuesta.

4. Usted sabe o conoce de alguna aplicación para el control de alimentos en pacientes con insuficiencia renal?

RESPUESTA	FRECUENCIA	PORCENTAJE
SI	10	10%
NO	90	90%
TOTAL	100	100%

Tabulación de la Pregunta 4.



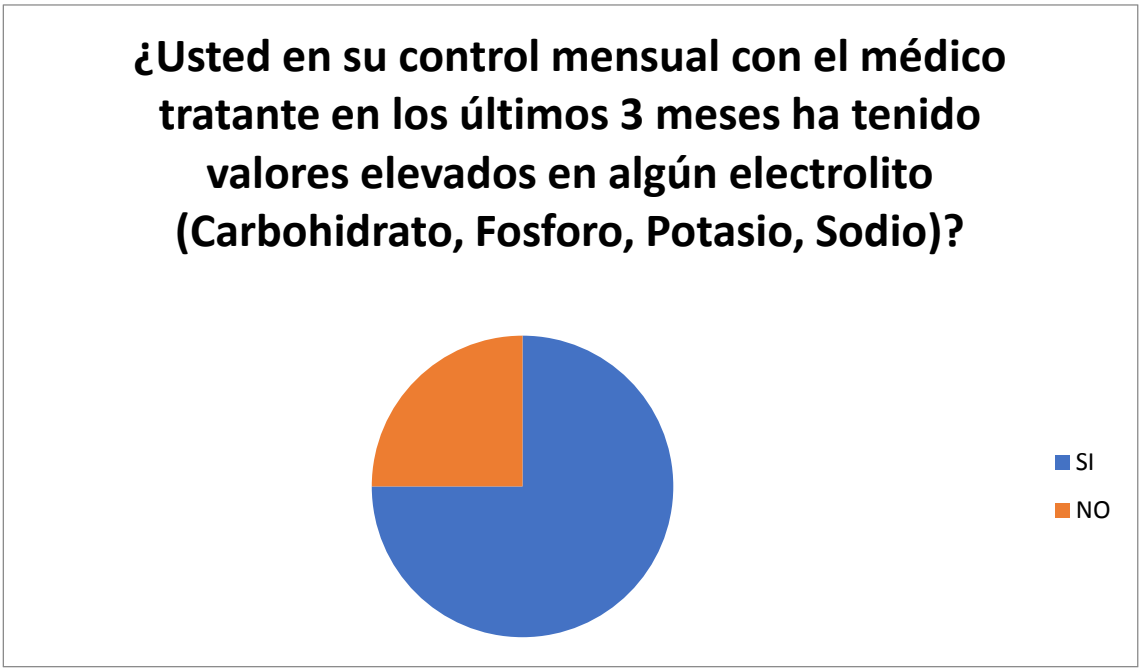
(Autores, 2020)

Gráfico, pregunta #4 de la encuesta.

5 ¿Usted en su control mensual con el médico tratante en los últimos 3 meses ha tenido valores elevados en algún electrolito (Carbohidrato, Fosforo, Potasio, Sodio)?

RESPUESTA	FRECUENCIA	PORCENTAJE
SI	75	75%
NO	25	25%
TOTAL	100	100%

Tabulación de la Pregunta 5.



(Autores, 2020)

Gráfico, pregunta #5 de la encuesta.

6 En caso de existir una aplicación que ayude a controlar su dieta, ¿Usted la utilizaría?

RESPUESTA	FRECUENCIA	PORCENTAJE
SI	95	95%
NO	5	5%
TOTAL	100	100%

Tabulación de la Pregunta 6.

(Autores, 2020)

Gráfico, pregunta #6 de la encuesta.