Universidad de Montemorelos
Facultad de Ingeniería y Tecnología

SAUM app

Tesis
Presentada en cumplimiento parcial de los requisitos para el grado de Ingenieria y Tecnología

Por:

David Enoc Guerra Cahuich

Daniel Arturo Gutiérrez Colorado

SAUM app

Proyecto presentado en
cumplimiento parcial de los
requisitos para el grado de
Ingeniería en Sistemas
Computacionales.

Por:

David Enoc Guerra Cahuich

APROBADO POR LA COMISIÓN:

Asesor Principal: Daniel Arturo Gutiérrez Colorado

Director: Alejandro Walterio García Mendoza

Evaluador: María Tolentino Hernández

Coordinador: Jair Arody del Valle López

Evaluador: Carlos Emilio Hernández Rentería

8/05/15

Fecha de aprobación

DECLARACIÓN DE INTEGRACIÓN DE LA FE

Cuando la sabiduría entrare en tu corazón, Y la ciencia fuere grata a tu alma,La discreción te guardará; Te preservará la inteligencia, Para librarte del mal camino, De los hombres que hablan perversidades, Prov. 2: 10-12

SAUM app

David Enoc Guerra Cahuich
Facultad de Ingeniería y Tecnología
Universidad de Montemorelos
Montemorelos, México
degc437@gmal.com

Daniel Arturo Gutiérrez Colorado
Facultad de Ingeniería y Tecnología
Universidad de Montemorelos
Montemorelos, México
daniel.gutierrez@um.edu.mx

***Abstract*— En este trabajo se explica el desarrollo de una aplicación móvil, que tiene como propósito ayudar a los alumnos a mejorar su condición física, además, evitar el desperdicio de alimentos. Gracias a las tecnologías móviles se intenta obtener que los alumnos tengan acceso a toda la información referente a su consumo de calorías, teniendo así, un lugar donde ellos puedan estar informados de la cantidad de calorías necesarias para desempeñar sus actividades diarias. Además se propone en este trabajo un sistema de donación de alimentos que le permita saber al alumno por anticipado la comida de ese día y su horario, para que pueda decidir si tomar el alimento o simplemente dejarlo a donación anónima, o mejor aún, donar sus alimentos a alguien que no tenga comidas específicamente. Este sistema propuesto espera ser lo más seguro posible para evitar problemas de donaciones no deseadas o malos entendidos.**

Palabras Clave- Tecnologías de la información, Dispositivos móviles, Smartphone, Aplicaciones Móviles, Índice de masa corporal y Calorías.

INTRODUCCIÓN

A. Antecedentes y estado del arte

Desde a mediados de los 80's la industria de telecomunicaciones se ha convertido en uno de los núcleos de la economía global, y una parte importante de esta industria se le denomina a la telefonía móvil [1]. Cuando el primer teléfono móvil fue presentado a finales de los 70's, este buscaba impresionar a las grandes masas del mercado global [1], ya que este era un teléfono que podía recibir llamadas telefónicas en cualquier lugar del mundo, siempre y cuando se dispusiera de señal. Con el tiempo la tecnología móvil se convirtió en algo más familiar, y su objetivo inicial aun continua innovando y evolucionando con teléfonos cada vez más livianos y con mayores utilidades [1]. Como cualquier otra tecnología emergente, es difícil predecir cómo un dispositivo puede ser adoptado y usado, de ahí la especulación generalizada acerca de sus efectos e implicaciones. Por ejemplo, pocos pudieron predecir que la mensajería de texto llegaría a ser usada en los teléfonos móviles, o mejor aún, que fueran adaptados hasta llegar a tener un teléfono multifuncional con capacidades de uso similares a los de un ordenador común, por ejemplo, teléfonos con recepción y envió de correos electrónicos, capacidad de compartir fotos y videos con amigos y familiares, recibir video llamadas, o revisar las redes sociales [1].

En México la introducción de los teléfonos celulares inteligentes ha generado un cambio drástico, según la Asociación Mexicana de Internet (AMIPCI) [3] reporta que en el año 2012, el 58% de los usuarios que accedían a internet lo hacían desde sus dispositivos móviles, esto quiere decir, que 23.5 millones de habitantes acceden a internet de este modo. Según la encuesta online realizada por AMIPCI [3] a 2 mil 329 personas con un nivel de confianza del 95% y un margen de error del .04%, revela que el 80% de los usuarios encuestados utilizan sus dispositivos para enviar y recibir correo electrónico, 77% accede a redes sociales, 71% accede a buscar información, 55% envían y reciben mensajes instantáneos, 44% accede

a la banca en línea, 29% compra en línea, 29% busca empleo en internet y 16% accede o crea sitios propios.

Otros estudios realizados por IAB México [4] revelan que en el 2013 la penetración de teléfonos inteligentes es del 39%, incrementando 17 puntos con respecto al año anterior que contaba con el 22%, esto muestra que la posesión de teléfonos celulares no inteligentes pasó de 87% a 78% indicando que los usuarios han migrado de sus dispositivos básicos a otros más avanzados. Por otro lado estos estudios [4] demuestran que 87% de los usuarios encuestados siempre salen de casa con sus dispositivos móviles, además no conformándose con eso el 60% declara nunca apagar sus dispositivos móviles, debido a que los usuarios encuentran cada vez más ventajas en los dispositivos móviles, ya que el 72% utiliza sus dispositivos como medio de comunicación , el 41% los utiliza como herramienta laboral, 36% para acceder a internet, 34% para entretenimiento y el 25% para hacer la vida más práctica.

Las estadísticas anteriores demuestran que la telefonía celular ha ido evolucionando, llegando a producir celulares con menor tamaño y peso, que llegan a ser más aceptados por la población mexicana, y gran parte de esto se debe a la llegada del iPhone en el año 2007 que cambio por completo la tecnología celular, presentando un teléfono celular sin teclado, con solo controles táctiles; fue tanto el impacto que las empresas que dominaban el 75% del mercado de telefonía móvil (Sony Erikson, Nokia, Siemens, Motorola y Samsung) empezaron a enfrentarse con graves problemas, reportando los peores resultados financieros[1], debido que a pesar de que ya existían teléfonos móviles con utilidades parecidas a las de un ordenador, o dispositivos con tecnología táctil, aún no tenían una buena sensibilidad táctil o no contaban con una interfaz de usuario más amigable [2]. Podría decirse también que el iPhone fue (como su campaña promocional decía) "un teléfono móvil revolucionario, un iPod de pantalla completa y control táctil, un dispositivo orientado al internet con correo electrónico, navegador de internet y mapas" [1].

El iPhone ha sido popular desde su creación, ya que permitió a los usuarios no solo realizar diferentes actividades, más allá de las que un teléfono celular podía hacer (recibir llamadas o mandar mensajes de texto), sino que además de eso, el iPhone permite reproducir música, videos, navegar por internet, capturar imagen y video, y hacer uso del sistema de posicionamiento global [5]. Este teléfono inteligente fue el invento que revolucionó en Apple después de su iPod en el 2001, debido a que además de ser un teléfono móvil con características similares a las de un ordenador común, podía conectarse directamente a su propia tienda virtual sin necesidad de estar conectado a una computadora, para poder comprar música, películas, videos, aplicaciones, etc.[1]. Fue gracias al auge de este teléfono que en el 2008 Google anuncio la salida de un sistema operativo *"open source"(de código abierto)* orientado a dispositivos móviles, llamado Android con la intención de hacer competencia con iPhone [1].

El sistema operativo Android fue desarrollado inicialmente por una pequeña compañía llamada Android.inc en Palo Alto California, con un sistema operativo en base Linux orientado a los dispositivos móviles. Este sistema fue comprado más adelante por Google en el 2005, y en el 2008, Google anunció su consolidación con empresas de telefonía móvil (Samsung, Motorola, LG, Vodafone e Intel por ejemplo) [1]. Esta unión hizo que diversas compañías impulsaran el desarrollo de nuevos teléfonos móviles con el sistema operativo Android, a diferencia de Apple con su iPhone, produciendo mayor competitividad, debido a nuevos teléfonos más baratos, con similitudes a las de un iPhone [1].

Debido al auge de los dispositivos móviles, fue que se decidió permitir que nuevos desarrolladores ajenos a los desarrolladores oficiales pudieran crear aplicaciones a sus sistemas operativos [5], por ejemplo, Apple hizo uso en sus inicios de aplicaciones Web, con su plataforma de desarrollo *"DashcodeProject",* pero tres años más tarde, Apple subió un poco más de rango, debido a que las aplicaciones de este tipo no tenían muy buen rendimiento, y permitió el desarrollo de aplicaciones nativas con su herramienta *"X Code"* [2]. Pero aun así todavía se siguen utilizando aplicaciones de tipo Web, debido a que pueden ser utilizadas en diferentes plataformas al mismo tiempo, por ejemplo, si se le pide al desarrollador una aplicación para IOS y Android, se tomaría más tiempo crear una aplicación para cada una de estas plataformas si se llegara a hacerse de manera nativa, pero, si se llegara a hacer una sola aplicación Web, podría lograrse que la aplicación se pueda utilizar con los dos sistemas operativos, Android y IOS; pero por otro lado, si se intenta desarrollar aplicaciones con tecnología 3D, el desempeño de una aplicación Web no sería lo suficiente bueno como lo es una aplicación nativa, pero aun así resulta más fácil de desarrollar una aplicación Web a la hora de programar aplicaciones administrativas, o aplicaciones de texto [2].

De acuerdo con [6], en el 2012 existían 566,165 aplicaciones móviles, desarrolladas por 101,764 programadores activos en la App store (tienda de aplicaciones móviles de Apple), aproximadamente 775 aplicaciones son subidas a esta tienda cada año, y hasta el año 2012 se habían descargado alrededor de 15 billones de aplicaciones de la App store en los pasados 3 años. La categoría más popular en el año 2012 fue la de juegos, que disponía de 74,379 aplicaciones, seguido por libros, entretenimiento y educación, y para esta última categoría en el 2011 los educadores habían elegido ocasionalmente 40,653 aplicaciones educacionales para sus trabajos [6].

Como se mencionó anteriormente, el desarrollo de aplicaciones para dispositivos móviles con iOS tuvo sus

inicios con aplicaciones de tipo web, con lo que se conocía como *"Dashcode Project"* pero tres años más tarde el desarrollo para esta plataforma cambio a uno nativo, con el lenguaje de programación de Apple *"Objective C"*[2]. Las herramientas básicas utilizadas para el desarrollo de aplicaciones móviles basado en Objective C son Xcode IDE, iOS sdk y iOS Simulator [5] que se explicaran más adelante.

Xcode IDE es el entorno de desarrollo integrado que provee de todas las herramientas para codificar, crear y depurar aplicaciones desarrolladas para los sistemas operativos Mac OS X y iOS; está integrado con Cocoa, que es de ambiente orientado a objetos, y sus librerías, llamadas Coca Frameworks [5].

iOS SDK es el kit de desarrollo de aplicaciones para los dispositivos móviles iPhone, iPod touch y iPad; contiene el código, la información y las herramientas de desarrollo que funcionan de la mano, y se acoplan perfectamente con Xcode para crear aplicaciones nativas para dichos dispositivos [5]. Cuando se trabaja con iOS SDK, las mismas herramientas se acomodan a las necesidades del dispositivo para el cual se va a programar, utilizando diferentes frameworks, como el llamado Coca Touch Framework [5].

iOS Simulator se encuentra incluido en el iOS SDK; es una herramienta que ayuda a los desarrolladores a realizar pruebas de funcionamiento de las aplicaciones creadas; ejecuta la aplicación como si se estuviera haciendo en el dispositivo real y simula los eventos táctiles con la ayuda del mouse, la rotación de la pantalla y advertencia de memoria baja, entre otras [5].

El framework del software de Android corre en Linux bajo la máquina virtual Java, [8]. Al ser un sistema operativo que corre bajo Java, este dispone de la portabilidad para ser implementado (gracias a su Framework Android), en diferentes dispositivos, por ejemplo Samsung, LG, Motorola y HTC [7].

Android es un framework multitarea, es decir, cada proceso corre en una máquina virtual que es implementado por procesos que corren en una plataforma de Linux, y cada proceso es una pequeña aplicación java [8]. Ser un lenguaje implementado en Java ayuda grandemente al desarrollo de aplicaciones o modificaciones, debido a que ya existen muchos desarrolladores maduros en cuanto a este lenguaje, y al final, lo único que tienen que hacer, es simplemente estudiar e implementar el framework de Android [7].

Java es un lenguaje multiplataforma (en cuanto ordenadores de escritorio), y cuenta con muchas herramientas gratuitas en diferentes sistemas operativos, tales como Eclipse, y Android App Inventor para desarrollo móvil [7].

El desarrollo web multiplataforma fue utilizado en los inicios del desarrollo de aplicaciones móviles por Apple, pero más tarde se optó por una desarrollo nativo, provocando así dos líneas de pensamiento, primero, construir diferentes aplicaciones por cada plataforma es realmente caro, además se tendría que dar soporte a cada una de las plataformas en las que se lanzara la aplicación, segundo, el rendimiento de las aplicaciones no siempre es el más rápido en aplicaciones nativas, a menos que sea una aplicación 3D, pero para el resto de aplicaciones como aplicaciones de texto o aplicaciones informativas el desempeño y velocidad de ellas es realmente alta [2].

La herramienta para el desarrollo de aplicaciones móviles multiplataforma más utilizado es PhoneGap, que es un framework gratuito que le permite al desarrollador manejar ambientes para crear aplicaciones en HTML, CSS, y JavaScript y aun así llamar características nativas con la API JS [2]. El framework PhoneGap contiene unas partes de código nativo, dependiendo del sistema operativo y pasa información de regreso a JavaScript que se encuentra en un contenedor WebView; en otras palabras, es una aplicación que utiliza de manera nativa el contenedor WebView para mostrar aplicaciones en JavaScript [2].

B. Problema

El problema radica en cinco problemas principales:

- El sistema actual del comedor universitario no dispone de una plataforma electrónica que permita al alumno donar sus alimentos.
- Actualmente se utiliza la red social de Facebook para informar a los alumnos sobre los alimentos del día, pero esta red social ha sido bloqueada por la universidad.
- Los alumnos no tienen un control de las calorías que ingieren en el comedor universitario.
- Los alumnos desconocen actualmente la cantidad de calorías que contienen los alimentos proporcionados por el comedor universitario.
- La administración del comedor universitario permite la donación de alimentos, pero para esto el alumno tiene que ir personalmente a donar el alimento, el problema es que no todos los alumnos tienen tiempo para realizar el proceso.

i. Declaración del problema

Los alumnos no tienen un sistema en el que puedan tener acceso completo sobre las calorías o información nutrimental de los alimentos que se proporcionan en el comedor universitario. Además, los alumnos no tienen un control sobre las calorías que requieren y las calorías que ingieren en el comedor universitario. Por otro lado muchos estudiantes no permiten que los alimentos que no van a consumir sean aprovechados por otros estudiantes que realmente los necesiten.

ii. Justificación del problema

Hace unos años un grupo de estudiante empezó a trabajar dentro del comedor universitario, específicamente en el área de losa (lavando los platos y charolas después de ser usados); al poco tiempo, notaron que se desperdiciaba mucha comida, que muchos estudiantes que llegaban se servían toda la comida y así, con la charola intacta, tiraban la comida a la basura. Esta situación preocupó mucho a los estudiantes que trabajaban en la losa, y fue entonces que se empezó a publicar en un grupo de Facebook la comida que se iba a servir con una hora de anticipación. Los resultados fueron notorios, el número de desecho de charolas intactas disminuyó grandemente.

Este sistema fue adoptado por el comedor de la Universidad de Montemorelos y actualmente se maneja en Facebook, una interfaz que provee no solo de la información referente a la comida del día, sino que además, se le permite al estudiante recibir una notificación cada vez que el menú del día sea servido. El sistema funciona, y sustituirlo con una aplicación oficial del comedor podría parecer algo innecesario, pero este sistema no toma en consideración las siguientes situaciones: ¿qué sucede con esa comida que no se consume?, es una comida pagada que no se consume y que podría servirle a un estudiante que lo necesite.

La problemática incluye además que los estudiantes que consumen alimentos en el comedor son estudiantes con una variedad de actividades diarias, mientras que unos estudiantes no realizan alguna actividad más que el estudio, hay quienes realizan más de una actividad física. Definir específicamente la cantidad de comida necesaria o calorías necesarias para las actividades de cada estudiante es un reto para el personal del comedor, ya que no tiene un registro exacto de las actividades de cada uno de sus estudiantes que consumen sus alimentos en el comedor de la Universidad de Montemorelos.

La aplicación SAUM App promete crear un sistema de notificaciones para los estudiantes, que permitiría que los alumnos sean notificados de cada una las comidas del día, junto con la información calórica de cada porción de comida que se sirva al estudiante, y gracias a una calculadora de calorías se le informará al estudiante la cantidad de calorías que requiere para sus actividades diarias para poder llevar un control de lo que se consume. En caso de que el estudiante no pueda o quiera consumir ese alimento tendrá la opción de donarlo (por medio de un usuario y contraseña) a otro estudiante que no tenga comidas.

C. Objetivos

- Sustituir la página del comedor universitario en Facebook.
- Informar a los alumnos sobre la comida del día.
- Hacer una interfaz amigable, con toda la información necesaria para realizar las actividades diarias.
- Facilitar a los alumnos la donación de alimentos.
- Hacer un sistema de donación segura.
- Mantener al estudiante con las calorías requeridas para actividad física.
- Evitar que el estudiante desperdicie alimentos.
- Desarrollar un prototipo movil que sea la base para crear un sistema para multiples sistemas operativos moviles.

D. Preguntas de investigación.

- ¿Qué lenguajes de programación son los adecuados para cada sistema operativo existente?
- ¿Qué herramientas o frameworks son los adecuados para el desarrollo de una aplicación movil y el sistema administrativo de la aplicación?
- ¿Cuáles son las fórmulas para calcular las calorías necesarias para la actividad diaria del estudiante?
- ¿Cómo realizar conexión entre una base de datos y una aplicación movil?

MÉTODOS

El proyecto cuenta dos módulos utilizados para que el sistema funcione, el módulo administrativo y el de la aplicación movil. Estas dos partes son conformadas por una serie de elementos que permiten el funcionamiento. A continuación se mostrarán los algoritmos y módulos utilizados para la creación del prototipo de la herramienta.

A. Aplicación iOS

El prototipo para la aplicación se creó en X Code, debido a que esta herramienta presenta una manera más fácil e intuitiva para crear conexiones y vistas dentro del diseño de la aplicación. La aplicación fue programada con el lenguaje de programación Objective-C, utilizando el framework X Code versión 5.

Para crear una aplicación en X Code son necesarios una serie de archivos, desde la parte gráfica hasta la parte lógica, siendo los más importantes: “Main.storyboard” (que en general se encarga de la parte gráfica de la aplicación), “viewController.h”, “viewController.m” (el nombre puede variar pero no la extensión en ambos casos), AppDelegate.h y AppDelegate.m [14].

El archivo “Main.storyboard”, es donde se encuentran las relaciones entre cada una de las vistas, es el lugar en donde se le pueden asignar botones, “label”, imágenes, texto, etc… [14] a cada una de las vistas, es aquí donde el desarrollador puede designar la cantidad de vistas tendrá la aplicación, y la manera en la cual se comunicarán entre ellas [14]. En el caso del proyecto actual, el archivo “Main.storyboard” contiene 3 vistas y un “navigationController”. El “navigatorController” es el encargado de permitir que se pueda navegar entre las vistas. Sin este controlador se podría acceder a las vistas pero no se podría regresar a la vista anterior de la aplicación, por este motivo se le asignó a este controlador la vista principal [14]. Las 3 vistas tienen las siguientes funciones: calcular calorías, mostrar la información del menú del día y permitir la donación del alimento del día en curso.

Cada vista tiene elementos padres, y en este caso son los archivos “viewController.h” y “viewController.m”. Juntos estos archivos permiten el funcionamiento a la vista en el “main.storyboard”. Como se mencionaba anteriormente en el archivo “main.storyboard” es donde se crean las vistas y se le agregan a estas vistas los elementos gráficos tales como botones, cajas de texto, imágenes, etc. Dentro del archivo

"viewController.h" se declaran estos elementos, y se indican las funciones que pueden tener, ya sea "outlet" (elemento de entrada o salida de información) o "action" (elemento que genera una función o método en el archivo "viewController.m"). Ya que cada elemento de la vista ha sido registrado en el archivo "viewController.h", se puede editar el archivo "viewController.m", en este archivo se genera el manejo de datos de la vista, es el lugar donde se procesará la información [14].

A continuación se explicarán las funciones y algoritmos creados en las tres vistas de la aplicación iOS.

1. Vista "home".

i. Vista "home"

La primera vista, es identificada dentro del archivo "main.storyboard" como "home", con sus archivos padres "home.h" y "home.m". Esta vista tiene como objetivo principal mostrar al usuario la comida del día en curso mediante una foto del platillo, junto con su información calórica. El comedor universitario de la Universidad de Montemorelos maneja dentro de su menú varios elementos para tener una comida completamente nutritiva, y constan de: platillo principal, acompañamiento, vegetales y sopa. Lo que se necesita en la vista "home" es que el usuario pueda reconocer la cantidad de calorías que su cuerpo necesita y la cantidad de calorías que actualmente está ingiriendo en esa comida. Para esto la vista contiene un visor de imágenes, 9 etiquetas (4 para el nombre del elemento del menú, que puede ser acompañamiento, plato principal, sopa y vegetales; 4 para las calorías de cada elemento del menú, y una para ver las calorías agregadas y restantes), 4 elementos de tipo *switch* (que se encargan de la suma de calorías para el etiqueta de calorías restantes), y por ultimo dos botones que re-dirigen a las otras vistas (donación y configuración). Cada uno de los elementos mencionados anteriormente, (con excepción de las 4 etiquetas para el nombre y los botones), son declarados en el archivo "home.h" de la siguiente manera: las etiquetas de las calorías son declaradas como "outlets", mientras que los elementos tipo switch son declarados como acción para crear las funciones en el archivo "home.m".

El archivo "home.m" tiene como función principal procesar cada uno de los elementos de la vista, pero dentro de cada archivo con extensión ".m" se tienen por defecto algunas funciones previamente generadas por X Code, una de esas funciones es la función "viewDidLoad". Esta función es la que se inicia justo cuando la vista se ha seleccionado o en este caso, cargado [14]. Dentro de esta función se verifica (por medio de un arreglo configurado en el archivo "appDelegate.m"), si la aplicación ha sido abierta por primera vez, y en caso de que así sea, se cambiará a la vista de la calculadora.

Anteriormente se había mencionado que el sistema tiene dos módulos principales, el módulo de la aplicación y la el módulo de la administración de la aplicación . En la función "viewDidLoad" se crea una conexión entre ambos módulos, la conexión tiene como propósito recibir de la parte administrativa las calorías del menú publicado. El funcionamiento es muy simple, se creó una conexión entre la aplicación iOS y el servicio web "calconsulter.php", este servicio web le envía un archivo "json", inmediatamente este archivo es convertido a un NSArray (así se le llama a los arreglos en Objective-C [15]), y una vez obtenido el arreglo, se convierte a un "NSDictionary" por medio de un ciclo "for", que pasa los elementos del "NSArray" como elementos "json". Un "NSDictionary" es otro arreglo pero que tiene dos dimensiones [16], la primera, contiene el nombre de cada uno de los elementos (plato principal, acompañamiento, vegetales, y sopa, de acuerdo con su respectivas abreviaciones en el código), y la segunda dimensión contiene la cantidad de calorías asignadas para el menú alimenticio de ese día. Dentro de un ciclo "for" se buscan cada uno de los elementos por su abreviatura en el arreglo "json", se toma el valor asignado para esa abreviatura (las calorías), se convierte a "NSString", y por último, se le asigna ese valor a la etiqueta de la vista "home" con el respectivo nombre o abreviatura. Para poder asignarle valores a una etiqueta existen dos formas: la primera, es asignar un valor fijo desde el archivo "main.storyboard", y la segunda, y en este caso más recomendada, es por medio de la función ".text", [14] que trabaja con la sintaxis en Objective-C (1).

nombreDelLabel.text=@"texto asignado"

(1)

A continuación se explicará la manera en la cual se realiza la suma de los valores. Los archivos AppDelegate.m y AppDelegate.h tienen la misma extensión de los archivos viewController, y ambos, tienen más o menos la misma función que los viewController. En AppDelegate.h se crean las variables públicas y en AppDelegate.m se manipulan, pero, a diferencia de ellos, el nombre de estos archivos no puede variar, no son ligados a una sola vista, y no se ejecutan. Por consecuencia, estos archivos se ejecutan desde antes que la aplicación se abra [14]. La función de estos archivos en una aplicación iOS es demasiado extensa, puede ir desde guardar

la configuración de una aplicación hasta realizar las notificaciones cuando estas son requeridas [14].

La vista tiene que mostrar al usuario la cantidad de calorías que requiere para su estilo de vida, y la cantidad de calorías que está consumiendo, es aquí donde se utilizan los archivos "AppDelegate". Dentro de los archivos es muy recomendado que se declaren cierto tipo de elementos, en este caso, los "NSUserDefaults" [18].

Las aplicaciones para dispositivos móviles, procesan informacion en el momento de ejecucion del programa, pero hay cierta información que el usuario guarda para la próxima vez que se ejecute su programa, esta información comúnmente es conocida como ajustes del programa. En este caso, la función de los "NSUserDefaults", es guardar información que el usuario va a necesitar después de haber ejecutado su aplicación [18]. En este caso se utilizó para que la aplicación guarde la cantidad de calorías obtenidas por medio de la vista de configuración. De ahí la importancia de que el usuario llene los datos al ejecutar por primera vez su aplicación, en la vista de la calculadora, para que una vez que se calcule la cantidad de calorías necesarias para las actividades diarias del estudiante, la vista "home" pueda consultar estos datos y de esta manera se presenten estas calorías en la novena etiqueta que contiene una salida parecida a (2).

"calorías tomadas/calorías necesarias"

(2)

Ya que se ha explicado la manera en la que se trabaja la obtención de datos, es momento de que se explique la parte del procesamiento, en otras palabras, darle vida a la vista, darle vida a el noveno "label". En el archivo "home.h" se declararon 4 elementos tipo switch, no como "outlet", si no como "action". Al declarar un elemento como "action", dentro del archivo "home.m" (en este caso), se crea una función que se va a ejecutar cada vez que el elemento sea presionado por el usuario [19]. La razón por la cual se hace esto con los 4 switch es debido a que es necesario que la suma de calorías sea realizada a la hora de que se presione el switch y se actualice el noveno "label" con las calorías totales señaladas. Como cada uno de los switch hace lo mismo solamente se explicará el proceso en general. Cada vez que el usuario active el switch, se le sumarán las calorías al "label", y cada vez que se desactive, el valor disminuye, esto es posible gracias a la función "isOn" que el lenguaje Objective-C tiene con su switch [20], y tiene una sintaxis parecida a (3), esto devuelve un valor positivo o negativo, ofreciendo que se puede implementar una función if/else, si devuelve positivo, se realiza la suma, si no (else), realiza una resta. Para el manejo de datos, se han declarado fuera de las funciones dos elementos: "cal" y "calor", el primer es el valor de las calorías que se van a sumar o restar, dependiendo el caso, de tipo float, y el segundo valor, será el total de calorías, de tipo "NSNumber". Para asignarle valores al elemento cal, fue necesario obtener el texto de las etiquetas que muestran las calorias, para esto fue necesario utilizar la funcion ".text" Esta función obtiene los valores de tipo NSString [17], pero, en este caso para poder realizar las sumas y restas entre "cal" y "calor" fue necesario convertir ambos valores a tipo "float" para realizar la suma y resta entre ambos valores esto fue posible gracias a la función de Objective-C "floatValue" [21], que tiene una sintaxis parecida a (4) para el caso de "cal", en el caso del elemento "calor" se utilizó función llamada "NSNumber numberWithFloat" [21], con la sintaxis (5), se sustituyeron los valores en la función y se le asignó el resultado a "calor", de esta manera se obtuvo el valor de "calorias tomadas".

"nombreDeSwitch.isOn"
(3)

"[valorString floatValue]"
(4)

"[NSNumber numberWhitFloat: valorA+ valorB]"
(5)

Por último, para actualizar la novena etiqueta se utilizaron dos valores "calorías tomadas" y "calorías necesarias". Hasta ahora se tienen los valores de "calorías tomadas" pero aún faltan las "calorías necesarias", que como se había mencionado anteriormente se encuentran en "AppDelegate.h" dentro de un "NSUserDefaults". Para obtener los datos, primero se declara un elemento "AppDelegate" con a siguiente estructura (6).

"AppDelegate *app= (AppDelegate *)[[UIAplication sharedApplication] delegate];"

(6)

Una vez declarado esto, se pueden obtener las variables de los elementos que tienen los archivos "AppDelegate", para eso se creó una variable de tipo "NSString" con nombre "requeridas" y se le asignó el valor de las calorías requeridas de la siguiente manera (7).

" NSString *requeridas=[app.nombreDelNSDefaults objectForKey:@"calorias"];"

(7)

Una vez que se obtiene el resultado de la suma y el valor de la suma o resta, se asignan los valores a la novena etiqueta con el siguiente formato (2), para eso Objective-C tiene la función "stringWithFormat" que en este caso quedaría como se muestra en (8).

"novenoLabel.text = [NSString stringWithFormat: @"%@ %@ %@ %@ (se agrega un %@ por cada string a agregar en la siguiente parte)", @"calorías totales", calor, @"/", requeridas];"
(8)

ii. Vista "calculadora"

La vista "calculadora" ("fig. 3"), se encuentra dentro del archivo "main.storyboard" con el id "P1" y cuenta como "viewController" con los archivos "calculadoraC.m" y "calculadoraC.h". Esta vista tiene como función principal realizar un cálculo de las calorías que el usuario debe de consumir dependiendo de sus actividades diarias. Si bien es cierto, no todos los usuarios realizan las mismas actividades,

algunos son sedentarios, y otros al contrario son extremadamente activos, es por eso que la vista calculadora intenta hacer que el usuario pueda revisar la cantidad de calorías que debe consumir para mantener su peso. Para realizar este cálculo se utilizó la ecuación Harris-Benedict [22].

La ecuación [22] hace uso de la siguiente información del usuario: peso (kilogramos), estatura (centímetros), sexo y edad (años), en base a estos datos genera un valor nombrado tasa metabólica basal [22], como se ve en la formula (9).

$$\text{Hombres} - \text{TMB} = (10 \times \text{peso en kg}) + (6{,}25 \times \text{altura en cm}) - (5 \times \text{edad en años}) + 5$$

$$\text{Mujeres} - (10 \times \text{peso en kg}) + (6{,}25 \times \text{altura en cm}) - (5 \times \text{edad en años}) - 161 \quad (9)$$

Una vez que se obtiene el valor de la tasa metabólica basal se procede a multiplicar este valor con un valor establecido en el estudio [22], este valor depende de la actividad física del usuario tal como se ve en "fig. 2", de esta multiplicación entre los dos valores se obtiene la cantidad de calorías requeridas para mantener el peso [22].

I. TABLA DE ACTIVIDAD FÍSICA

Actividad	*Fórmula*
Poco o ningún ejercicio	Calorías diarias necesarias = TMB x 1,2
Ejercicio ligero (1-3 días a la semana)	Calorías diarias necesarias = TMB x 1,375
Ejercicio moderado (3-5 días a la semana)	Calorías diarias necesarias = TMB x 1,55
Ejercicio fuerte (6-7 días a la semana)	Calorías diarias necesarias = TMB x 1,725
Ejercicio muy fuerte (dos veces al día, entrenamientos muy duros)	Calorías diarias necesarias = TMB x 1,9

2. Tabla de actividad física.

3. Vista Calculadora.

La vista calculadora ("fig. 3") está conformada por un switch, 4 "text field" (campos de texto), un "picker view" y un botón de tipo "action", cada uno de estos elementos son declarados en el archivo "calculadora.h". Los 4 "text field" son usados para obtener los datos de edad, peso y estatura, y el switch es utilizado para obtener el sexo. Si el switch está activado, es tomado como sexo femenino si esta desactivado, es tomado como sexo masculino, esta vez, a diferencia de la vista "home", el switch no es declarado de tipo "action", sino de tipo "outlet", debido a que no se realizará ninguna acción hasta que se presione el botón calcular.

En esta ocasión dentro de la aplicación se muestra un elemento nuevo, que no habla por sí solo como lo hace el "text field" (campo de texto), este elemento es el "picker view", que en Objective-C es conocido como "UIPickerView" [23]. Este elemento muestra una serie de opciones para que el usuario seleccione la opción de su preferencia [23], es por eso que fue utilizado este "picker view" para las opciones de actividad física que la ecuación Harris-Benedict requiere [22].

Para introducir los elementos que muestra el "picker view" y para obtener el valor seleccionado por el usuario, fue necesario que se implementaran una serie de funciones dentro archivo "calculadoraC.m" que se explicarán a continuación.

El elemento "UIPickerView" requiere de una serie de datos que mostrar dentro de su menú, es por eso que dentro de la implementación del archivo "calculadoraC.m" (se entiende por implementación el lugar donde se declaran los elementos que pueden ser compartidos dentro de todas las funciones que se lleguen a declarar dentro de este archivo [14]), se declaró un "NSArray" con nombre "activ" y un "NSString" llamado "actest", este último es donde se guardó la selección del usuario.

Dentro de la función "viewDidLoad" se le asignan los valores de la formula al arreglo "activ" (selecciona, poco ejercicio, ejercicio ligero, ejercicio moderado, ejercicio fuerte y ejercicio muy fuerte).

Se creó una función llamada "numberOfComponentsInPickerView", esta función está definida dentro de la documentación de "UIPickerView" en

Objective-C [23], esta función tiene como propósito, regresar un valor de tipo int (entero), que indicará a la vista la cantidad de elementos que el usuario podrá seleccionar dentro del "picker view" [23], en este caso se le indica que regrese solo un elemento.

Se declara a continuación la función llamada "numberOfRowsInComponent", que igual que la función anterior, regresa un valor de tipo int (entero), pero en este caso, el número que regrese esta función fue utilizado para indicarle al "picker view" la cantidad de elementos u opciones que tendrá que mostrar al usuario [23], para esto, Objective-C tiene preparado en la documentación de "NSArray" una función llamada ".count", que regresa la cantidad de elementos disponibles en el arreglo (el tamaño del arreglo) [15], de esta manera se informó al "picker view" la cantidad de elementos que mostrara en su lista.

Para asignar los datos al elemento de tipo "picker view", fue necesario iniciar la función "titleForRow", que tiene como propósito principal insertar los datos que se mostrarán al usuario en el "picker view" devolviendo un "NSString" [23]. En este caso, se tomaron los datos del "NSArray" "activ", para esto, a la hora de crear la función se declaró un valor de tipo NSInteger llamado "row", la función hace que en el momento que se active la vista se inicie un conteo desde 0 hasta el valor devuelto por la función "numberOfRowInComponent", esto permite que los valores del NSArray "activ" sean leídos uno por uno y devueltos de la misma manera al "picker view". La sintaxis en este caso quedaría como se muestra en (10).

"return activ[row];"
(10)

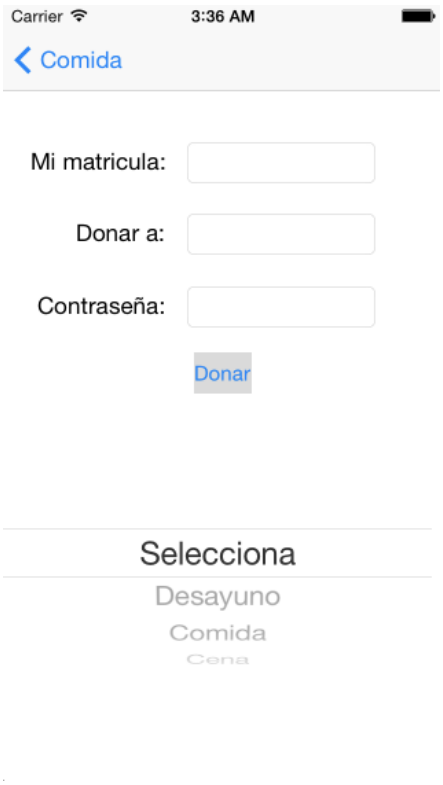
Para obtener la selección del usuario, "UIPickerView" tiene una función llamada "didSelectRow", que no regresa ningún valor en especial, pero se activa en el momento en que el usuario selecciona un valor dentro del "picker view" [23]. Al igual que la función "titleForRow", se declaró al momento de crear esta función un valor de tipo "NSInteger" llamado "row". En este caso, el valor "row", no inicia un conteo desde 0, si no que devuelve el id de la selección del usuario [23]. Dentro de esta función e hizo uso del "NSString" con nombre "actest" que se declaró en la implementación para guardar la selección del usuario. Para hacer esto, se tomó el valor del id que contiene en "row" y se pidió el valor de ese id en el "NSArray" "activ", y la función quedó como se muestra en (11).

"actest=[[NSString alloc]
initWithFormat:@"%@",activ[row]];"
(11)

Debe ser señalado que en el momento de seleccionar un "text field" aparece el teclado, pero no desaparece al terminar de escribir, por lo que dentro del archivo "calculadoraC.m" se creó una función llamada "touchesBegan" [24]. Esta función tiene como propósito que cada vez que se toque fuera del text field (campo de texto), el teclado se desaparezca, para esto se declararon los "text field" dentro de esta función [24], con una sintaxis parecida a (12).

[self.nombreDelTextField resignFirstResponder];
(12)

Anteriormente se explicó que era necesario implementar la ecuación Harris-Benedict [22], para esto se hizo uso de la función del botón "calcular" en el archivo "calculadoraC.m". Para implementar la ecuación Harris-Benedict [22] se necesita primero validar que todos los campos estén completados, es por eso que dentro de la función se implementó un condicional de tipo "if", el cual revisa que los valores que contienen los campos de textos no tengan un valor "float" igual a 0, o que "actest" sea diferente a "selecciona". En caso de ser así, se le muestra una alerta al usuario indicando que llene todos los campos de texto. En caso de que la condición "if" no se cumpla (que todos los campos estén llenos), se empiezan a procesar los datos conforme a la ecuación Harris-Benedict [22]. Al obtener los datos, se procede a guardar el resultado (calorías) dentro del "NSUserDefaults" "datos" con el id "calorias" para que la vista "home" pueda mostrar estos datos como en (2), y además se le muestra al usuario una alerta con las calorías que debe de consumir.



4. Vista "Donacion de Alimentos".

ii. Vista "Donación de Alimentos"

Por último, para terminar con la sub-sección de la aplicación iOS, se explicará la última vista, la vista correspondiente de a la donación de alimentos ("fig.4"), esta vista tiene como objetivo realizar una donación de alimentos (como su nombre lo dice), a algún alumno en específico por su número de matrícula, l. La donación solo puede hacerse para una o las 3 tres comidas del día en curso. Para hacer la donación posible se le pide al usuario que inserte su número de matrícula, el número de matrícula del estudiante al que el usuario desea donar su alimento, la contraseña del usuario (la contraseña del sistema académico de la Universidad de

Montemorelos), y la comida que desea donar (desayuno, comida o cena).

La vista está enlazada a los archivos “viewController” “DonarA.h” y “DonarA.m”, y está compuesta por 3 etiquetas, 3 “text field" (campos de texto), un “picker view" y un botón (donar) (“fig. 4”). Las 3 etiquetas son utilizadas para mostrar el nombre de los campos de texto a llenar, por lo tanto esas etiquetas son estáticas y no es necesario declararlos en el archivo “DonarA.h”, y el resto de los elementos es declarado de la siguiente manera: los 3 “text field" como “outlet”, “picker view" como “outlet” y el botón “donar” como “action”.

Como se puede ver, se hace uso de un “picker view", que se utilizará para mostrarle al usuario las comidas que puede donar (desayuno, comida y cena) en el día en curso.

El funcionamiento de esta vista es muy básico, todo lo realiza en la función del botón, lo único que hace es tomar los valores de los 3 “text fieds" (número de matrícula del usuario, matricula a donar y contraseña), y el valor seleccionado por el usuario en el “picker view", revisa por medio de un condicional “if” si el tamaño de las matriculas son mayores a 7 (el tamaño normal de una matrícula de la Universidad de Montemorelos), y que además, la selección sea diferente a “selecciona”, en caso de que no se cumpla la condición, se muestra una alerta indicando que se llenen todos los campos. En caso de cumplir la condición, se genera una conexión con un servicio web llamado “donara.php”, que realiza el proceso de registro de donación, verifica que el usuario y la contraseña sean correctos, que el usuario tenga comidas y en caso de ser correcto todo esto, se registra la donación en la base de datos del comedor universitario.

B. Administración

En esta sub-sección se hablará de la manera en la cual la aplicación realiza la conexión con la base de datos, la manera en que se realizan las entradas y salidas de los datos, y por último, la manera en que el administrador de la aplicación sube la información.

Objective c tiene un problema, no puede manejar una base de datos de manera nativa, por lo que es necesario que se conecte a un servicio web por medio de una operación de tipo “post” que se encargue del manejo de datos. Como se mencionó en la sub-sección anterior, las vistas de “home” y “donación de alimentos” realizan una conexión a archivos php que se encargaban de la inserción y/o la consulta de datos. En esta sección se explicarán estos archivos, y además se explicará la manera en la cual el administrador maneja los datos que son introducidos y/o consultados en la base de datos.

Como base de datos se utilizó MySQL, y para el manejo de datos se utilizó PHP en su versión 5.5.18. La razón por la cual fue manejada la base de datos con php se debe a que este lenguaje tiene una serie de servicios nativos que permiten tener un mejor manejo de la base de datos, por ejemplo, permite realizar una conexión con la base de datos y ademas permite comprobar que la conexión se establezca bien, permite un cambio rápido entre base de datos, recibe la respuesta de la base de datos y permite insertar datos de una manera más eficiente [25].

Para la consulta que realiza la aplicación en la vista “home” con la base de datos se utiliza el archivo “calconsulter.php”, que al ser consultado en el servidor web regresa un archivo de tipo “json" (un arreglo), que permite a la aplicación mostrar los datos, y para realizar esto el archivo tiene que establecer una conexión con la base de datos, y realizar la consulta de la tabla que contiene los datos [25]. Con el objetivo de lograr el funcionamiento descrito, previamente se creó un usuario con permiso de solo “consulta”, una base de datos especial para el comedor y una tabla con las calorías (plato principal, acompañamiento, vegetales y sopa) e imagen de la comida en curso, utilizando “phpAdmin” (administrador gráfico de la base de datos en MySQL).

El primer paso es establecer una conexión con la base de datos [25] para esto, php hace uso de las librerías de mysqli, que están disponibles en la nueva versión de php. Para lograr la conexión entre php y mysql se tiene que crear una variable de tipo mysqli con la estructura que se muestra en (13).

```
$nombreDeLaConexion = new mysqli(“direccionBD”,
“usuario”, “contraseña”, “baseDeDatos”);
```
(13)

Después de declarar esta variable de tipo mysqli, se verificó que la conexión se hubiera realizado correctamente, para esto se creó una condicional de tipo “if”, que revisa si hay una conexión exitosa, en caso de que la conexión no se establezca se devuelve un mensaje al usuario con el error que no permitió la conexión entre php y mysql. Para realizar esta consulta, las librerías mysqli tienen una función llamada “mysqli_connect_errno()” [25], que se introduce dentro de la condicional “if” (la función no regresa valores positivos o negativos, esta función regresa valores de texto, entonces lo que realmente sucede en php a la hora de implementar estas funciones en las condicionales es revisar si la función devuelve alguna clase de datos [25]), en caso de cumplirse (no lograr la conexión), dentro del mensaje de salida al usuario se concatena esta función, y la función inmediatamente muestra la razón por la cual el servidor MySQL rechazó la conexión con el servicio php.

Si la conexión se realiza correctamente se puede continuar con la consulta a la base de datos, para esto se selecciona la base de datos del comedor (previamente creada en MySQL) por medio de la función “select_db” [25] que tiene la siguiente estructura que se muestra en (14).

```
$nombreDeLaConexion-
>select_db(nombreDeLaBaseDeDatos);
```
(14)

Una vez que se realizaron los ajustes preliminares, ya se puede realizar una consulta [25], y para esto se declararon dos variables: la variable “$consulta” y la variable “$resultado”. La primera de ellas guarda el código MySQL, que puede ser creación de tablas, usuarios, dar permisos, etc., [25], y en este caso contiene la consulta a la tabla que contiene los datos de las calorías del día, esta consulta es parecida a lo que se muestra en (15):

```
$consulta = "SELECT * FROM calactu “;
```
(15)

En (15) se muestra el valor “calactu”, este es el nombre de la tabla que contiene la cantidad de calorías del menú de ese día. Una vez hecho esto se pasó al variable $resultado, esta variable es de tipo mysqli_query, la declaración de la variable de este tipo es parecida a lo que se muestra en (16).

```
$resultado=mysqli_query($nombreDeLaConexión,
$consulta);
```
(16)

Con la linea de arriba (16) se obtienen los datos que se consultan a la base de datos dentro de la variable $consulta. Para que php revise si hay alguna respuesta por parte del servidor, se crea una condicional “if”, y se introduce la linea anterior dentro de la condicional, y en caso de obtener respuesta se crearon dos arreglos, un arreglo para obtener los datos temporales ($tempArray) y un arreglo para guardar todos los datos ($resultArray), dentro de la condicional “if” se creó una condicional de tipo “while” con la siguiente estructura que se muestra en (17)

```
while($row = $resultado -> fetch_object())
```
(17)

El ciclo “while” (17) crea una variable $row, a la cual se le asigna una fila del conjunto de datos que se obtuvo en el $resultado. Esto es posible gracias a la función fetch_object (), de tal manera que se inicie el ciclo cada vez que se lea una nueva fila. Dentro del ciclo “while” se le asigna el valor de “$row” al arreglo “$tempArray”, y una vez hecho esto se le agrega al final del arreglo “$resultArray”, el arreglo temporal. Lo anterior da como resultado, un arreglo con todos los datos consultados en “$resultado”, pero este arreglo no se puede compartir con la aplicaión iOS, es por eso que se tuvo que convertir a un archivo tipo “json” [26], y para esto se implementó la función (18).

```
echo json_encode($resultArray);
```
(18)

Esta función (18) toma todo el arreglo y lo convierte a un archivo de tipo “json”, que puede ser compartido con lenguajes tales como Javascript, Java y Objective C[26].

Por último se cierra la conexión con la base de datos por medio de la función [25] (19):

```
mysql_close($nombre de la conexión)
```
(19)

El otro archivo al que se hace referencia, o que la aplicación en iOS necesita es el archivo “donarA.php”, que es el encargado de recibir la información del usuario, para que se realice la donación (matricula del usuario, contraseña, matricula del estudiante al que se le va a donar, y la comida). Esta información es recibida en el momento en que se consulta este archivo por medio del sistema “post” y además se crea una variable de tipo “date” (fecha), que es el lugar donde se consulta al sistema la fecha actual [28] con la función de PHP (20).

```
$dat= date(“dmy”);
```
(20)

En la linea (20), se puede ver la declaración de una variable llamada “$dat”, a la cual se le asigna el valor que devuelva la función “date”. La función “date()” es una función que realiza una consulta al sistema operativo del servidor, y este servidor le regresa la fecha [28]. Para obtener el formato deseado, la función “date()” tiene las siglas en ingles “d” (day), “m” (mont) y “y” (year), entre otros, dependiendo de la manera en que se acomoden las siglas la función date regresará la fecha con el formato deseado [28].

La tabla donde se guardan las donaciones ha sido diseñada para guardar los siguientes tamaños de datos: los 7 dígitos que contienen los números de matrícula (donante, receptor), 6 dígitos para la fecha (ddmmyy), y por ultimo un dígito para registrar la comida que se va donar. La comida seleccionada que se recibe desde la aplicación tiene el siguiente formato: “Desayuno”, “Comida” “Cena”, pero, la base de datos no tiene la cantidad de caracteres para guardar estos datos, debido a que los caracteres no tienen un valor fijo dentro de la tabla, es por eso que se le asignaron los valores 1, 2 y 3, dependiendo la comida que el usuario seleccione, para hacer esto, el valor de comida que se recibe de la aplicación es comparada, y en caso que coincida con el valor que se tiene registrado en el script php, se modifica el valor de la comida por los números previamente mencionados (dependiendo el caso 1, 2 y 3). Para hacer esta comparación posible se utilizó la función nativa de PHP (21).

```
strcmp($a, $b)
```
(21)

Esta función (21) se encarga de verificar que los valores introducidos dentro del paréntesis sean iguales (valor $a y valor $b), y en caso de que sean valores iguales devolverá un valor igual a 0, y si son diferentes regresa un valor positivo o negativo, pero nunca igual a 0 [29].

Una vez adaptados los datos para la base de datos, es momento de crear una conexión con el sistema de la universidad, el cual recibe los datos de usuario y contraseña, los verifica, y por último regresa un valor indicando si los datos son correctos, y qué tiene permitido donar. El valor que regresa es 1 en caso de que sean correctos y 0 en caso de que sean falsos.

Una vez que se verifican los datos, en caso de ser positivo, se realiza una conexión con la base de datos del comedor, y se inserta la donación a la tabla “donarA” por medio de un usuario que solo tiene acceso a escritura. El código MySQL utilizado para insertar datos en una tabla es (22).

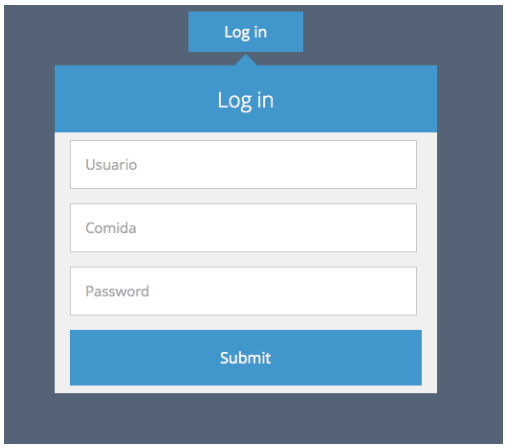
```
"INSERT INTO donaciona (usuario, receptor, fecha,
comida) VALUES ('{$usuario}', '{$receptor}',
‘{$fecha}’,’{$comida}')"
```
(22)

En caso de que la inserción de datos sea exitosa se devolverá el siguiente mensaje: “donación exitosa”, en caso contrario se devolverá el siguiente mensaje: “no se registró tu donación” y se le agregará la razón por la cual no se pudo realizar la donación por medio de la función mysql_error () [25]. Por último se cierra la conexión.

Para el administrador de la aplicación se ha hecho uso de una interfaz gráfica por medio de HTML, CSS, y Javascript,

que se conecta directamente con un web service en php. La interfaz gráfica tiene como propósito revisar las donaciones, y subir el menú del día al servidor MySQL, para que la aplicación pueda agregar las calorías y el menú del día en su vista “home”.

La interfaz gráfica para el administrador fue diseñada por Miro Karilahti, y cuenta con 3 archivos, index.htm, index.js y style.css. Los últimos dos archivos son los encargados de generar cajas de texto, botones, fondo, letra…etc, personalizados que pueden ser implementados en cualquier archivo HTML. Para adaptar estos archivo a los servicios web en PHP, fue necesario modificar el archivo index.html, y en base a este archivo crear 3 nuevas interfaces, la interfaz para inicio de sesión, consulta de donaciones y por último, inserción de datos a la base del menú del día.



5. Inicio de Sesión

Para el inicio de sesión se modificó el archivo index.html y se creó un archivo nuevo, llamado login.htm (“fig. 5”), el cual muestra al usuario tres campos de texto a llenar: usuario, comida y contraseña. En el campo de comida el usuario tiene que insertar la comida que se va a consultar de la siguiente manera: Desayuno, Comida y Cena. La interfaz cuenta con un botón con el id de “submit”, el cual toma los datos de los campos de texto y los envía por medio de “post” a un archivo php llamado “login.php”. Este archivo al recibir los datos verifica que el dato de “comida” cumpla con las características requeridas tal y como se verifica en el archivo “donarA.php”, en caso de no cumplir con los requisitos, el script PHP mostrará el siguiente mensaje: “comida no válida” e inmediatamente se regresará a la página de inicio de sesión. Ya que se revisó lo anterior, y todos los datos son correctos, se inicia sesión en MySQL y se verifica que la conexión se realice exitosamente, en caso de lo contrario se devolverá el mensaje: “no se logró la conexión” y se le agregará la razón por la cual no se logró la conexión e inmediatamente se regresará a la página de inicio de sesión. En caso de que la conexión sea exitosa se abrirá la página de consulta de donaciones.

La página de consulta de donaciones se encuentra en el archivo “consulta.html”, este archivo a diferencia al archivo “login.html” tiene un script en javascript, que permite al usuario ver la respuesta de un servicio web en php sin necesidad de recargar la página, este script hace uso de una conexión por medio de “ajax” [30]. “Ajax” es una función de jQuery (una librería de javascript), que permite enviar y recibir datos desde servicios web, y esto permite al usuario consultar cierta información desde una página web sin tener que salir de la misma [30]. Para hacer uso de “Ajax” se importó jQuery por medio de la línea de código (23).

<script type="text/javascript" src=“http://ajax.googleapis.com/ajax/libs/jquery/1.11.2/jquery.min.js”></script>
(23)

6. Consulta

Una vez importado jQuery, al archivo “consulta.html” (“fig. 6”), se agregó un área de texto, 3 botones y un label, el área de texto para introducir la matricula del estudiante al que se le donó la comida, el botón para consultar un script php, y el “label” para mostrar al usuario la respuesta del servicio web, el primer botón “consultar”, para consultar la donación, el segundo botón para acceder al servicio servicio de inserción de datos del menú del día, y el último para cerrar sesión.

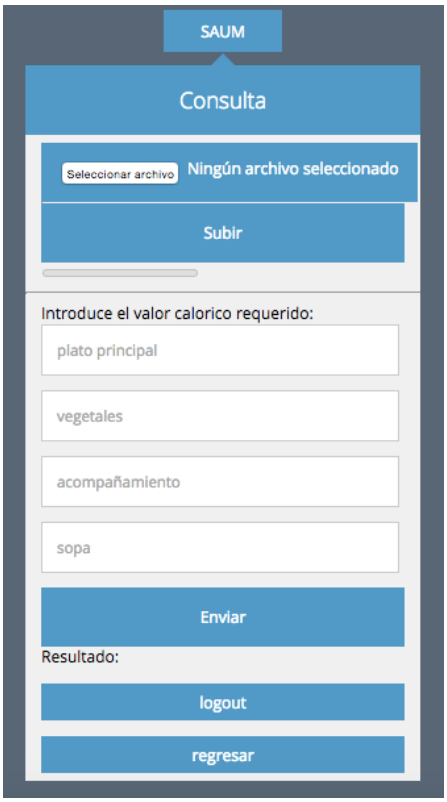
Por medio de “ajax” [30] se creó un script dentro del archivo “consulta.html” con una función en javascript que hace una consulta cada vez que se presiona el botón “consultar” o se presiona la tecla “enter”, esta consulta se hace al servicio web en php “registroAct.php”.El servicio web “registroAct.php” es el encargado de consultar con la base de datos de donaciones, si el número de matrícula que ha recibido desde “consulta.html” está registrado. Para realizar esto, el script “registroAct.php” se conecta a la base de datos del comedor en MySQL, y realiza la consulta en la tabla “donaciona”, con los datos: “comida”, “fecha” (es autogenerada), “matricula” por medio de la linea de código (24).

SELECT * FROM donaciona WHERE dto='{$matricula}' and meal='{$comida}' and dat=‘{$fecha}’
(24)

La línea (24) le solicita a mysql que revise dentro de su tabla “donaciona”, específicamente en la fila donde se encuentran los datos recibidos desde “consulta.html”. En caso de que encuentre algo, gracias a la función

"mysqli_num_rows()" [25] se obtiene la cantidad de filas que contienen estos datos, en caso de que el numero devuelto sea mayor a 0 (quiere decir que se ha hecho una donación), se le pide el número de matrícula del donador a MySQL y se imprime este número de matrícula, en caso de lo contrario se imprime el siguiente mensaje "no tiene comida".

El tercer botón "cerrar sesión" se encarga de ejecutar el servicio web en php "logout.php", que básicamente lo único que hace es cerrar la conexión con mysql.



7. Página de Publicación del Menú

El segundo botón re-dirige a una página "pubMen.html" ("fig. 7"), esta página se encarga de publicar el menú del día en mysql. El funcionamiento es parecido a lo que se realiza en la página de consulta, maneja envío de datos a un servicio web en php por medio de "Ajax", pero la diferencia es que ahora contiene más campos de texto (sopa, plato principal, vegetales y acompañamiento), y además permite subir fotos a un servidor ftp (file transfer protocol), para mostrar el platillo del día. Para la selección del archivo se utilizó la siguiente función en javascript (25)

```
document.getElementById(sTargetID).value =
oFileInput.value;
```
(25)

Esta función regresa un valor de tipo "path" con la dirección del servidor donde se encuentra el archivo que se seleccionará. Este "path" [31], se utiliza para la inserción de datos al servidor ftp que se explicará más adelante. Para poder seleccionar un archivo se utilizó una entrada de tipo "file" en html, el cual muestra un botón que a la hora de seleccionarlo despliega un navegador de archivos que permite seleccionar el documento [32], en este caso, la foto del alimento que se va a publicar. Una vez seleccionado este archivo la función anterior busca el path y lo convierte en texto para más tarde enviarlo a el servicio web en php.

El archivo encargado de publicar los datos del menú del día en php es pubMen.php. Este archivo recibe los 5 datos desde el archivo pubMen.html (calorías del plato principal, sopa, vegetales, acompañamiento y el "path" del archivo a subir), y en el caso de los datos de las calorías, se publican directamente en la tabla "calactu". En el caso del archivo, se toma el path y se sube el archivo al servidor ftp. Esto es posible gracias a la siguiente función nativa en php (26).

```
ftp_put($conexion, $archivoRemoto, $archivo, FTP_ASCII)
```
(26)

Para establecer conexión entre el servicio web php y el servidor ftp se utiliza la función "ftp_connect()" que es declarada dentro de una variable, dentro de esta función se introduce la dirección ip o dominio, donde se encuentra alojado el servidor ftp [33]. La variable conexión es declarada de la siguiente forma (27).

```
$conexion= ftp_connect(ftp://direccionFtp.com);
```
(27)

Se creó la variable $archivo, para recibir desde el archivo "pubMen.html" el path de la foto a subir, una vez recibido el path es subido al servidor ftp por medio de la función php "ftp_put" [33] (28), y se guardó ahora la dirección dentro del servidor ftp donde se quedó almacenado el archivo. Una vez listos todos los datos, se suben al servidor mysql para que la aplicación en iOS los muestre al estudiante.

```
ftp_put ( resource $ftp_stream , string $remote_file , string
$local_file , int $mode [, int $startpos = 0 ] )
```
(28)

Para fijar el costo del sistema prototipo se ha considerado la cantidad de horas invertidas en el proyecto, lo cual fueron 2 horas por día, 5 días a la semana por 8 semanas, dando un total de 80 horas. El costo por hora para un desarrollador es valorado en el mercado por $150 mxn, lo que dá un total de $12,000 mxn, sin considerar los impuestos.

RESULTADOS

El prototipo creado tuvo como resultado las siguientes aplicaciones:

- Aplicación que permite la donación de alimentos a estudiantes sin comidas en día en curso.
- El estudiante podrá saber las calorías necesarias para mantener su peso.
- El estudiante podrá ver las calorías que contiene el menú del día.
- El administrador dispone de una herramienta de consulta para donaciones.
- El administrador dispone de una herramienta para subir el menú del día con las calorías que posee.
- El sistema de donación es seguro.

CONCLUSIONES

A. Discusión de los resultados

El prototipo creado permitirá solucionar los problemas planteados al inicio del artículo, debido a que ya se tiene una herramienta que publica el menú del día y además permite la donación de alimentos de una manera segura, evitando que el estudiante done más que su comida del día en curso, y además evita que el cualquier persona con acceso a la aplicación realice donaciones sin antes tener el número de matrícula del estudiante al que se le va a donar, el número de matrícula del donador y su contraseña del sistema.

El prototipo además muestra un sistema de conteo de calorías, esto permitirá al alumno tener un control personalizado sobre lo que ingiere. La herramienta del conteo de calorías permitirá ser una tecnología que disminuya la cantidad de alumnos de la Universidad de Montemorelos con obesidad.

B. Conclusiones de la investigación

Las dos partes principales del prototipo fueron creados exitosamente, tanto la parte de la aplicación iOS, como el desarrollo del sistema web. Ambos permitirán trabajar en sincronía para solucionar los problemas previamente mencionados.

El prototipo creado podrá ser utilizado en la reducción del desaprovechamiento de los alimentos del comedor dentro de la Universidad de Montemorelos. Los avances obtenidos durante el desarrollo de este prototipo son las bases para un mejor sistema, en el caso de la aplicación móvil en iOS, es la base para crear una aplicación para el sistema operativo Android, y por otro lado, el sistema para la administración de la aplicación desarrollado muestra las bases para crear un sistema más seguro.

C. Reflexión

El desarrollo de esta herramienta, en lo personal, ha sido una experiencia que me permitió crecer de manera profesional, gran parte del proyecto era solo una idea que tenía que ser desarrollada, normalmente sin conocimiento de las herramientas y funciones en código. Cada función creada en este sistema fue un reto personal, era como enfrentar un gigante sin nada más que una navaja suiza cada vez que se agregaba alguna función en el sistema. Durante el desarrollo se puede ver que no hay más limite que la imaginación y la persistencia que se tenga a la hora de buscar respuestas a cada una de las dudas que surgen a lo largo del desarrollo de un sistema. Muchas veces no es tanto la respuesta a las dudas que surgen, sino tener las preguntas correctas para buscar lo que realmente se necesita, a la hora de crear una herramienta cada vez más fina. Dentro de este artículo no basta con solo poner las referencias de cada una de las herramientas que fueron usadas para desarrollar este sistema, porque en lo personal sé que aún faltan aquellas respuestas que me hicieron llegar hasta estas referencias.

El desarrollo de este sistema me ha mostrado la vida real de un desarrollador de software, desde buscar un problema, proponer una solución hasta desarrollar esta solución, y a decir verdad, hace que me guste más mi trabajo.

D. Recomendaciones

Se sugiere que el sistema sea llevado a otras plataformas móviles, debido a que no todos los estudiantes poseen un dispositivo con iOS, es sobre todo recomendable que el sistema sea llevado a una aplicación para la plataforma Android.

El sistema para el administrador de la aplicación no es del todo seguro, presenta serias fallas que podrían ser solucionadas si fueran llevadas a otro lenguaje o framework más seguro, o mejor dicho especializado en seguridad.

E. Futuros aportes

Durante el desarrollo del prototipo se observaron una serie de fallas en el sistema actual del comedor universitario, pero la que realmente hace falta solucionar es la pérdida de conexión entre el ordenador de administración y el sistema o base de datos de la Universidad de Montemorelos. La solución que se podría aplicar en este caso, es crear una base de datos local en el ordenador de la administración del comedor, y crear un nuevo sistema que clone la base de datos de la institución a la hora de iniciar sesión, y a la hora de cerrar sesión, se envié a la base de datos de la Universidad todas las entradas.

Otro aspecto que se puede agregar dentro del sistema es un contador de pasos, esto permitirá a la aplicación saber el tipo de actividad correspondiente en la fórmula de Harris-Benedict.

El prototipo solo muestra el menú fijo para una sola comida, esto se debe a que cada comida tiene diferentes componentes, es por eso que se recomienda crear una tabla para cada comida dentro de la base de datos del comedor.

REFERENCIAS

1. D. Howcroft, and B. Bergyall, “Mobile Aplications Development on Apple and Google Platforms”, Manchester Bussiness School, Atlanta, GA, USA, 2011.

2. A. Charland, and B. Leroux, “Mobile Aplication Development: Web vs Native”, *Communications of the ACM*, vol. 5, pp. 49-53, 2011.
3. R. Juárez, “Hábitos de los Usuarios de Internet en México”, Asociación Mexicana de Internet, Guadalajara, JL, México, 2012.
4. D. Saavedra, “Los Dispositivos Móviles Están Transformando la Manera en la que los Mexicanos se Relacionan con el Mundo”, Interactive Advertising Buereau México, México, D.F., 2013
5. K. M. Rojas, A. C. Alarcón, and J. E. Roa, “Desarrollo de Aplicaciones Móviles Bajo la Plataforma de Iphone”, *Revista Facultad de Ingeniería, vol. 20, pp. 77-91, Noviembre 2011.*
6. H. Walker, “Evaluating the Effectiveness of Apps for Mobile Devices”, *Journal of Special Education Technology*, submitted for publication.
7. J. Berthiaume, and J. Balton, “iOS vsAndroid in the Enterprise”, *Network World,* pp.20-22, May 2012.
8. B. Wong, “Google’s Android vs Apple’s iOS and the Winner is?”, *Engineering Feature,* pp. 34-40, November 2010.
9. M. Maisto, “Mobile Phone’s History in 10 Industry-Changing Devices”, January 2014, [Online]. Available: http://www.eweek.com/mobile/slideshows/mobile-phones-history-in-10-industry-changing-devices.html.
10. L. Gunwoong, “Determinants of Mobile Apps’ Success: Evidence from the App Store Market”, *Journal of Management Information Systems,* vol. 31, pp. 133-170, october 2014.
11. D. Riehle, “Role Model Based Framework Design and Integration”, *Proceedings of the 1998 Conference on Object-Oriented Programming Systems, Languages, and Applications*, 1998, pp. 117-133.
12. M. Siegler, “Analist: There’s a great future in iPhone apps”, *Venture Beat News*, June 2008, [Online]. Available: http://venturebeat.com/2008/06/11/analyst-theres-a-great-future-in-iphone-apps/.
13. B. Woodcock, A. Middleton, and A. Nortcliffe, “Considering the Smartphone Learner: an investigation into student interest in the use of personal technology to enhance their learning”, *Student Engagement and Experience Journal,* pp. 2017-9476, 2012.
14. *UIViewController Class Reference,* Apple Inc., Cupertino, CA, USA, 2013.
15. NSArray *Class Reference,* Apple Inc., Cupertino, CA, USA, 2013.
16. *NSDictionary Class Reference,* Apple Inc., Cupertino, CA, USA, 2013.
17. *UILabel Class Reference,* Apple Inc., Cupertino, CA, USA, 2013.
18. *NSUserDefaults Class Reference,* Apple Inc., Cupertino, CA, USA, 2013.
19. *Concepts in Objective-C Programming,* Apple Inc., Cupertino, CA, USA, 2013.
20. *UISwitch Class Reference,* Apple Inc., Cupertino, CA, USA, 2013.
21. *NSNumber Class Reference,* Apple Inc., Cupertino, CA, USA, 2013.
22. J. A Harris and F. G. Benedict, “A new predictive equation for resting energy expenditure in healthy individuals”, *American Journal of Clinical Nutrition,* vol. 51, pp. 241-247, February 1990.
23. *UIPickerView Class Reference,* Apple Inc., Cupertino, CA, USA, 2013.
24. *UIGestureRecognizer Class Reference*, Apple Inc., Cupertino, CA, USA, 2013.
25. *Extensión MySql Mejorada,* The php group, 2015.
26. *JavaScript Object Notation*, The php group, 2015.
27. *Variables predefinidas*, The php group, 2015.
28. *Funciones de Fecha/Hora*, The php group, 2015.
29. *Strcmp*, The php group, 2015.
30. *jQuery.ajax()*, The jQuery Foundation, 2015.
31. *Document,* Mozilla developer Network and Individual Contributors, 2015.
32. *Using Files From Web Applications*, Mozilla developer Network and Individual Contributors, 2015.
33. *Funciones de FTP*, The php group, 2015.