Universidad de Montemorelos
Facultad de Ingeniería y Tecnología

Aplicación móvil en la plataforma Android para el Servicio de Alimentos de la Universidad de Montemorelos

Tesis presentada como requisito para el grado de Ingeniería en Tecnología de la Información y Comunicación

Por:

Jairo R. Rocha Mena

Asesores:

Daniel Arturo Gutiérrez Colorado

Ignacio Cruz Domínguez

27/11/15

Montemorelos Nuevo León, México

Capítulo

Aplicación móvil en la plataforma Android para el Servicio de Alimentos de la Universidad de Montemorelos

Jairo Rubén Rocha M. Facultad de Ingeniería y Tecnología. Universidad de Montemorelos

Montemorelos N.L México

I. INTRODUCCIÓN

***Abstract* – En la Universidad de Montemorelos se ha encontrado un problema relacionado a los alimentos. Este problema radica en el desperdicio de estos en el comedor universitario, de aquellos alumnos que deciden no consumir las comidas del día, mientras otros estudiantes carecen de ellas. Se plantea desarrollar una aplicación que solucione dicha problemática y que los alimentos que no se han consumido puedan ser donados a los estudiantes que lo necesitan.**

En este proyecto se quiere facilitar y mejorar el sistema del comedor, continuando con el desarrollo de la aplicación móvil, desarrollándola en la plataforma Android, con el objetivo de ayudar a los alumnos de la Universidad de Montemorelos a donar las comidas/alimentos que no se consumen, ya sea de matrícula a matrícula o aleatoriamente, así como informarles acerca de su consumo en calorías.

Para beneficiar a la mayoría de la sociedad universitaria, se implementará la aplicación en Android, facilitando así a los alumnos la donación de los alimentos en una aplicación segura. También se pretende informar a los alumnos acerca de la comida del día (menú) mediante la aplicación móvil, así como sustituir la página de Facebook del comedor para una mejor administración de información, y evitar que el estudiante desperdicie alimentos; mantenerlo informado acerca de las calorías que debe consumir de acuerdo a sus actividades físicas mediante una tabla calórica.

Palabras claves: Aplicaciones móviles, plataforma Android, dispositivos móviles, teléfono inteligente, sistemas operativos móviles.

ANTECEDENTES

Existen muchas necesidades tanto en nuestro entorno social como en el universitario. Diferentes problemas que a simple vista no se pueden solucionar, pero gracias a la tecnología móvil y a los sistemas operativos que se han ido desarrollando a lo largo de la historia para nuestro beneficio, las incógnitas se vuelven visibles dando un giro totalmente diferente a la forma de solucionar algunos dilemas. Los sistemas operativos móviles aunados a las aplicaciones, han venido a facilitar el trabajo manual, solucionando diferentes problemas.

Durante mucho tiempo, los desarrolladores de dispositivos móviles comprendían un grupo al que se le conocía como grupo de desarrolladores de dispositivos integrados, que más adelante se fueron convirtiendo en desarrolladores de escritorio, y más tarde en desarrolladores web [1]. Los dispositivos integrados, que fueron con los que se inició el desarrollo de dispositivos, no tenían un funcionamiento complejo, por lo tanto no necesitaban sistemas complejos. Los controles remotos de televisores son un ejemplo de dispositivos integrados, estos comprendían un simple chip, que al apretar cualquier botón se traducía la señal en el receptor (televisor). Aquí se puede ver un claro ejemplo de programación básica [1]. Al momento de ir evolucionando la tecnología, la necesidad de sistemas complejos se vieron notoriamente, y es aquí donde nace la oportunidad de crearlos para los primeros móviles. Con esta evolución, los dispositivos integrados complejos, como los primeros PDAs, sistemas de seguridad para el hogar, GPS, los primeros celulares, entre otros, se trasladaron a sistemas operativos estandarizados.

Con el avance de la tecnología, los teléfonos inteligentes y los sistemas operativos móviles, surgió la necesidad de crear aplicaciones móviles.

Al observarse el problema del desperdicio de los alimentos en el comedor universitario de aquellos alumnos que deciden no consumir las comidas del día, mientras otros estudiantes carecen de alimentos, se planteó desarrollar una aplicación que solucione dicha problemática y que los alimentos que no se han consumido puedan ser donados a los estudiantes que lo necesitan, de una manera fácil, sencilla y segura.

El Ingeniero Guerra desarrolló una aplicación móvil como solución a la problemática antes mencionada para la plataforma iOS. Este sistema de donaciones conecta con una

base de datos la cuál es administrada por un servidor virtual llamado XAMPP. Este servidor agiliza y facilita las conexiones entrantes para administrar los datos de la base conforme a las operaciones de la aplicación móvil y un servicio web que es utilizado por los administradores del comedor universitario.

El proyecto cuenta con dos módulos utilizados para que el sistema funcione, el módulo administrativo y el de la aplicación móvil. Estas dos partes son conformadas por una serie de elementos que permiten su funcionamiento.

PROBLEMAS

- Los estudiantes no tienen una aplicación móvil desarrollada en Android en donde puedan donar los alimentos que no consumen, lo cual propicia el desperdicio de las comidas, tampoco un sistema donde informe a los alumnos la información calórica que deben consumir de acuerdo a sus actividades físicas.

- Por normas de la universidad, se bloquea Facebook en el transcurso del día, y por este medio es donde se daba a conocer el menú del comedor. Este desconocimiento provoca también el desperdicio de los alimentos.

- No hay ningún sistema seguro de donaciones de comidas para móviles Android.

A. Declaración del problema

Los alumnos no tienen una aplicación móvil desarrollada en Android, en la que puedan hacer donaciones de comidas, fáciles y seguras. Tampoco hay ningún medio por el cual den a conocer las calorías que se consumen en cada porción o platillo del día, ni tampoco el menú de una manera apropiada, que es donde se deriva el mayor problema. El menú se publica día a día en la página oficial de Facebook, pero dadas las nuevas normas de la universidad, está permanece bloqueada en el transcurso del día.

B. Justificación del problema

Se desarrolló la aplicación con anterioridad para la plataforma iOS con un lenguaje nativo, teniendo en cuenta que Android y este SO móvil son los más usados se ha optado por desarrollar esta aplicación para móviles en la plataforma Android, de igual manera con el lenguaje nativo. Se ha pensado en crear la aplicación del comedor en esta plataforma, donde los alumnos podrán ingresar con su usuario y contraseña usual donando las comidas que no vayan a consumir en el día. Así también dar a conocer el menú del día, ya que esto va ligado a que se desperdicien los alimentos por no conocerlo. Al saber cuál es el platillo, no lo quieren, no lo consumen y no se aprovechan los alimentos. La finalidad de dar a conocer el menú es que con anticipación puedan decidir si consumirán la comida, y así donar los alimentos si es que no se van a consumir, para aprovecharlos y no desperdiciarlos.

En este sistema los usuarios a parte de poder hacer donaciones seguras podrán ver la información calórica que consumirán en los alimentos. Este problema también se deriva, y se quiere solucionar con la app dando a conocer las calorías que necesitan ingerir al día y cuantas porciones deben consumir.

El énfasis en el problema es a la hora de donar las comidas, así que se le dará el mismo énfasis a la aplicación a la hora de hacer la donación. Se implementará seguridad para que no cualquiera pueda entrar y donar comidas, si no que será mediante un token de seguridad donde los administradores del comedor también se les informará con el token, y así asegurar las comidas del estudiante en la app móvil.

C. Objetivos

- Crear el sistema de donaciones en otra plataforma (Android), facilitando a los alumnos la donación de los alimentos.

- Informar a los alumnos de la comida del día mediante la aplicación móvil en dicha plataforma.

- Evitar el desperdicio de alimentos mediante la app y mantener informado al alumno acerca de las calorías que debe consumir para sus actividades físicas.

- Hacer un sistema de donación seguro.

D. Hipótesis

Es posible utilizar el entorno de desarrollo integrado Android Studio para la plataforma Android en la creación de una aplicación móvil que facilite la donación de comidas en el Comedor Universitario.

II. FUNDAMENTOS TEÓRICOS

A. Marco teórico

Actualmente la tecnología nos ha alcanzado, y en muchas ocasiones superado, y más hablando de la tecnología móvil, donde el hardware y las máquinas electrónicas inteligentes se han considerado como productos de compra-venta, donde las empresas (marcas) se han dedicado a competir entre ellas, en buscar desesperadamente la manera de diferenciar su producto con el de sus competidores, y en este caso se habla del diseño y del software.

Casi todo gira en torno a estos dos factores, que son los principales a la hora de enfrentarse a su competencia. Hablando del diseño se entiende por el estilo gráfico que se le da en general, y el software se ocupa de los requisitos de las aplicaciones, se refiere al sistema operativo.

En el mundo de la tecnología móvil existen varias empresas que han desarrollado diferentes sistemas operativos móviles, las cuales han ido recorriendo poco a poco el camino y enfrentándose a los obstáculos de la competencia, mejorando cada vez para no quedarse atrás. Symbian por ejemplo, el cual es un sistema operativo móvil diseñado para teléfonos inteligentes. Sucesor de Nokia series 60, lanzado por primera vez en el 2010, en el teléfono inteligente Nokia N8, que dio un paso muy adelante y se metió a la competencia. Más tarde, Nokia anunció que Symbian sería remplazado por lo que hoy conocemos como Windows Phone, como sistema operativo móvil en todos los futuros teléfonos inteligentes [2]. IOS (antes iPhone OS) fue lanzado en el año 2007, tanto para el iPhone y iPod touch, como para el iPad y el Apple TV. A partir del 2012 app store contenía más de 700.000 aplicaciones de iOS, que en conjunto se descargaron más de 32 mil millones de veces, algo que se hizo notar en gran manera a diferencia de Android[2].

Android es un sistema operativo móvil basado en Linux en conjunto con la Open Handset Alliance. Android, al igual que iOS, tiene una comunidad de desarrolladores que hacen aplicaciones para ampliar las funcionalidades del dispositivo. Estas apps se desarrollan en una versión personalizada de java, las cuales pueden descargarse en la Play Store. A partir del año 2012 contaba con más de 600.000 aplicaciones, que en conjunto se descargaron más de 20 mil millones de veces, empezando a incursionar junto con iOS [2].

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) [3]. 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 [3].

En los últimos años los teléfonos móviles han experimentado una gran evolución, desde los primeros terminales, grandes y pesados, pensados sólo para hablar por teléfono en cualquier parte, a los últimos modelos, con los que el término “medio de comunicación” se queda bastante pequeño [4]. Así nace Android, un sistema operativo desarrollado en Linux para móviles. Esta plataforma de software de Google es la más usada hoy en día, gracias a que está desarrollado en Linux, que es una plataforma de código abierto y permite a los desarrolladores dar mayor funcionalidad a los smartphones. Además Android es un sistema gratuito y multiplataforma; por multiplataforma entendemos que el sistema operativo puede ser usado en distintas plataformas (tablets, teléfonos móviles inteligentes y netbooks), y por plataforma entendemos que es una combinación de hardware y software usada para ejecutar aplicaciones; en su forma más simple consiste únicamente de un sistema operativo, una arquitectura, o una combinación de ambos. Android es gratuito al poder ir instalado gratuitamente en cualquier dispositivo móvil [5].

Existe un problema hoy día con el software que se construye, que se desarrolla. No precisamente de las aplicaciones móviles, si no de cualquier sistema que se escribe. La mayoría de las veces se construye con las mismas técnicas, los mismos métodos de antes. El problema con el software moderno es que hemos estado construyendo nuestros "rascacielos", con los mismos materiales y técnicas que se utilizan para construir chozas [6].

La plataforma Android es una plataforma móvil moderna diseñada para ser verdaderamente libre. Las aplicaciones móviles Android hacen uso del hardware y software avanzado, para brindar innovación y valor a los consumidores [7].

Gracias a Android se pueden reunir en una misma plataforma todos los elementos necesarios que permiten controlar y aprovechar al máximo todas las funcionalidades del dispositivo móvil (llamadas, mensajes de texto, cámara, agenda de contactos, conexión Wi-Fi, Bluetooth, videojuegos, etc.) [8]. Para acceder a las funcionalidades se debe pedir permiso y en el caso de nuestra aplicación solo se pedirán permisos para acceder a la conexión Wi-Fi, ya que es la que conectará con la base de datos de la Administración de Sistemas, que es donde se manejará la donación de alimentos.

Existen diferentes maneras de desarrollar una aplicación, cada una tiene sus ventajas y desventajas. Las aplicaciones Web por ejemplo, están construidas con tres tecnologías básicas que no deben faltar en apps de este tipo, y son HTML (define texto estático e imágenes), CSS (define presentación y estilo) y JavaScript (define las interacciones y las animaciones), Estas aplicaciones son multiplataforma, esto quiere decir que pueden ser ejecutadas en cualquier

dispositivo. Su principal característica es que es responsiva, o sea que se adapta a cualquier dispositivo y es ejecutada mediante un navegador [9]. La principal ventaja de este tipo de aplicaciones es que son compatibles entre plataformas, lo que permite a los usuarios acceder desde cualquier dispositivo o sistema operativo fácilmente y no necesariamente mediante una aplicación móvil nativa. Los navegadores web móviles son bastante estandarizados, lo que hace mucho más fácil su desarrollo, creando una app universal por así decirlo. A diferencia de las aplicaciones nativas, que son ejecutables independientes que interfaz directa con el sistema operativo, aplicaciones web se ejecutan dentro del navegador [10]. Pero hay que tener en cuentas las desventajas, ya que si se quiere acceder a las funciones del dispositivo como cámara, ciertas conexiones entrantes o salientes, agenda de contactos, etc., no es posible, ya que para acceder se necesitan ciertos permisos.

El navegador es en sí mismo una aplicación nativa que tiene acceso directo a la API del sistema operativo, pero sólo un número limitado de estas APIs están expuestos a las aplicaciones web que se ejecutan en su interior. Mientras que las aplicaciones nativas tienen plena el acceso al dispositivo, muchas características sólo están parcialmente disponibles para aplicaciones web o no disponible en absoluto. Aunque se espera que esto cambie en el futuro con los avances en HTML, estas capacidades no están disponibles para los usuarios móviles de hoy en día [9].

A diferencia de las aplicaciones web que se ejecutan desde un navegador, las aplicaciones nativas tienen archivos ejecutables binarios que se descargan directamente en el dispositivo y se almacenan localmente, esto quiere decir que todos los procesos se ejecutan en el dispositivo, no requiere de ninguna conexión para poder correr como las web [10].

Al momento de ser instalada la app, esta interactúa directamente con el sistema operativo sin necesidad de algún intermediario.

Las aplicaciones nativas son libres de acceder a cualquier API que está disponible por el proveedor del sistema operativo, y en muchos casos hay funciones y características exclusivas de ese sistema operativo móvil [10].

Una de las desventajas más grandes con las que se enfrenta un desarrollador a la hora de escribir una app nativa para varias plataformas es la compatibilidad que hay, a diferencia de las web apps, que son multiplataforma, las versiones de las aplicaciones móviles nativas se tiene que desarrollar por separado [9].

Claro que debemos saber que se debe sacar provecho a las aplicaciones nativas, ya que al usar todos los recursos del dispositivo y al ejecutarse completamente local, el rendimiento de la app es superior a la web app, los procesamientos gráficos y de todo el dispositivo son aprovechados al máximo siendo más eficiente [11].

Un IDE (Integrated Development Environment), Ambiente de Desarrollo Integrado por sus siglas, es un entorno de desarrollo, que proporciona al programador herramientas de construcción para que el trabajo se facilite en cierta manera. La mayoría de los IDEs contienen un compilador, un intérprete, como Eclipse, NetBeans, entre otros.

No en todos los IDEs se pueden desarrollar aplicaciones móviles. Eclipse fue el primer IDE a disposición del público para Android y ha estado en uso desde 2008. Antes se requería un proceso de configuración complicada que involucró a la descarga de múltiples piezas para poder instalarlo. Ahora, con el debut de aDtBundle, el proceso es mucho más fácil. Todo lo necesario para construir una aplicación Android en eclipse viene en un solo paquete pre configurado para conseguir ponerlo en marcha en menos de cinco minutos [12].

Para la aplicación SAUM se trabajará con el IDE Android Studio. Una de las ventajas por las cuales se decidió en trabajar en este entorno de desarrollo fue por sus herramientas que facilitan a la hora de desarrollar la aplicación móvil. El emulador es una de ellas, con el cuál se pueden hacer pruebas sin necesidad de un móvil físico. Tiene las mismas características de hardware y software de un dispositivo móvil normal. En la figura 1 podemos ver la pantalla principal de este IDE.

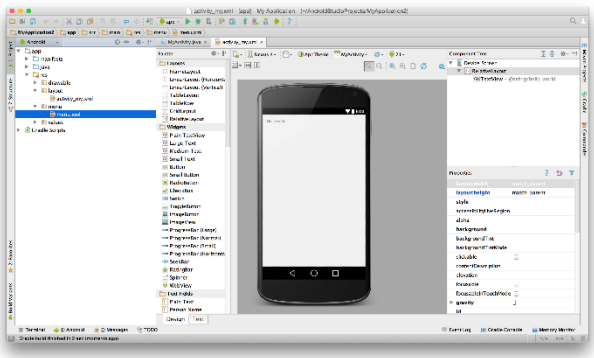

Figura 1. Pantalla principal del IDE Android Studio

Android Studio se basa en el IDE de Java llamado IntelliJ. Parecido a JetBrains (desarrollador de IntelliJ), como RedMine, PyCharm, PhpStorm, WebStorm o AppCode. Todos los productos In-telliJ comparten el mismo IDE cáscara que se puede ver notoriamente al empezar a desarrollar en Androis Studio [13].

La primera pieza importante de software que se debe tener instalada en el equipo para poder empezar a desarrollar en Android es el Android SDK [14]. El SDK de Android contiene un depurador, bibliotecas, un emulador, documentación, código de ejemplo y tutoriales, disponible en la página: http://developer.android.com/sdk/index.html.

Android Studio viene con su propia versión del SDK de Android, que es preconfigurado para utilizar con Android Studio después de la instalación. Si sea tiene un SDK instalado en equipo, Android Studio no utilizará el SDK instalado por defecto.

La herramienta Manager SDK gestiona las versiones del SDK de Android instaladas en el equipo, así que facilita a la hora de elegir cual se utilizará. Cuando se inicia se genera una lista de elementos, y también si están o no instalados. Aquí se descargan las herramientas que se necesitan para el proyecto, y las que no se utilizan pueden descargarse más adelante para que el tiempo ocupado en la descarga no quite tiempo mientras lo necesario se instala, así como se puede ver en la figura 2.

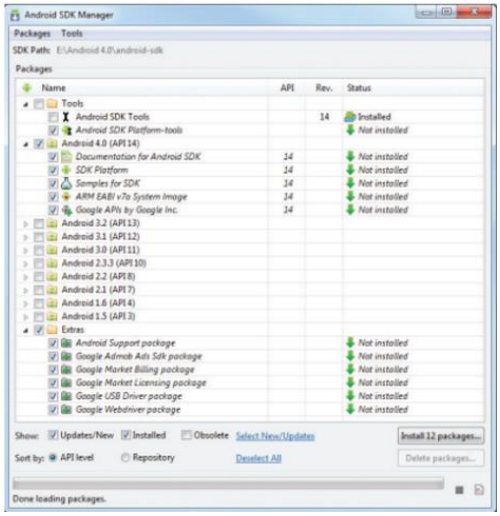

Figura 2. Manager SDK

Este IDE se compone por diferentes paneles que ayudan al desarrollador, y facilitan a la hora de trabajar con diferentes herramientas. Contienen diferentes funciones para ser tan productivo como sea posible en el desarrollo de las aplicaciones Android [14].

Los paneles más importantes en la interfaz de este IDE son el project panel (panel de proyectos) (figura 3.), que es donde se permite navegar a través de la jerarquía de archivos del proyecto y seleccionar, abrir, editar y realizar varias otras acciones en los archivos.

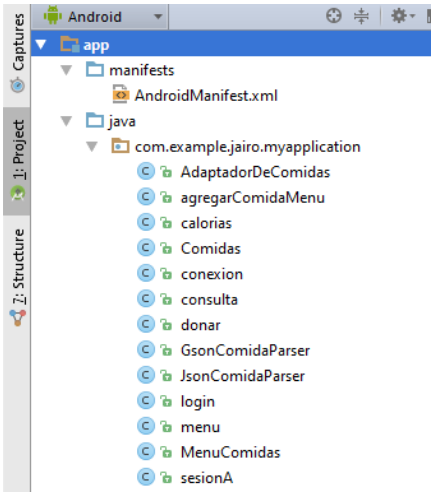

Figura 3. Project Panel

Otros paneles importantes:

- Android Panel
- Messages Panel
- Mave Panel
- Gradle Panel
- Event Log Panel

La zona final, que es de suma importancia, es la barra de estado en la parte inferior de Android de estudio, que se muestra en la Figura 4. Aquí es donde la mayoría de las actualizaciones de estado se producirán en segundo plano mientras se ejecutan los procesos. Algunos de estos procesos en segundo plano incluyen la actualización de los índices de los archivos, Maven o Gradle procesamiento en segundo plano, y los errores de eventos.

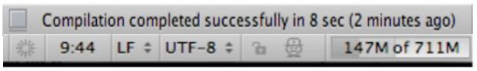

Figura 4. Barra de estado

Una de las ventajas es que el SDK de Android incluye un emulador de dispositivos móviles, como se aprecia en la figura 5, un dispositivo móvil virtual que se ejecuta en el equipo. El emulador permite desarrollar y probar aplicaciones Android sin necesidad de utilizar un dispositivo físico, el emulador tiene las mismas características de hardware y software de un dispositivo móvil normal, proporciona navegación similar a la de un Android y muestra la app que se está desarrollando tal y como se vería en un dispositivo físico [15].

Figura 5. Emulador de Android Studio

B. Estado del arte

La tecnología Java se ha vuelto popular gracias a su versión para dispositivos móviles. En cuanto se dio a luz esta versión, los fabricantes y desarrolladores pronto se interesaron en él. La demanda de estos aparatos fue en aumento, lo que Java ME fue impulsado a crear todo tipo de aplicaciones [16].

En muchos casos las aplicaciones que se desarrollaban en Java ME tenía que ser adaptada a diferentes dispositivos móviles y es donde se creó el problema de compatibilidad denominado fragmentación [17]. Definir qué tipo de aplicación se quiere desarrollar es indispensable.

Hoy en día, hay aplicaciones de ocio, para realizar cierto tipo de tareas, unas de entretenimiento, otras útiles en las labores diarias. Strike [18] comenta que las aplicaciones móviles no solo pueden servir para ocio si no también pueden ser útiles en la vida de un estudiante, útiles si se desea mantener al día con los tiempos y aumentar su proceso de estudio, solo es cuestión de dominio propio, ya que muchas veces los móviles se convierten en distracciones y no en una ayuda. Como sabemos, el desarrollo de estas aplicaciones muchas veces son de ayuda para el desempeño de diversas actividades, ya sean laborales, diarias o escolares, logrando así una eficiencia notoria y fluidez.

Por otro lado los teléfonos inteligentes han venido a sustituir muchas cosas del pasado, como los telegramas, cartas, etc., y se han puesto en el número uno como medios de comunicación. López [19] hace un énfasis en la telefonía celular, que en un principio fue concebida únicamente para voz, pero gracias a la tecnología celular hoy en día es capaz de brindar otro tipo de servicio como datos, audios y video. Gracias a esto, el campo laboral cada vez se ha vuelto más flexible, con el hecho de tener videoconferencias, es posible trabajar en conjunto, y no hablando físicamente, si no a distancia. Conectarse a una red, ya sea de datos o un punto compartido de red, abrir una aplicación para poder contactarse y comunicarse, hoy es tan fácil que no requiere de mucho conocimiento ni esfuerzo, solamente un teléfono inteligente.

Las aplicaciones están dejando de ser un asunto exclusivo para niños, para jugar, para pasar el rato, y se están convirtiendo en una herramienta, como dice Rojas [20], para que muchas empresas gestionen sus operaciones para solucionar ciertos problemas, como el aumento de ventas, la pérdida de clientes, etc. Muchas de ellas están incursionando en la tecnología móvil, aerolíneas, restaurantes, servicios de transporte, servicios de alimentos, etc. Utilizan estos nuevos desarrollos tecnológicos para buscar más canales de contacto con sus clientes. Así como existen aplicaciones para diferentes usos, existen aplicaciones que ayudan al rendimiento físico, ejercicios, rutinas para perder peso, para ganar masa muscular, entre otras cosas. Muchas implementan el control de calorías, y es que hoy en día la sociedad no se preocupa por mantener un control de la ingesta de alimentos, de calorías, ni mucho menos el control de su peso ideal. En 1919 el Laboratorio de Nutrición del Instituto Carnegie de Washington (EE.UU.), publicó una monografía con el título *"A Biometric Study of Basal Metabolism in Man"*. Los autores de esta publicación fueron los fisiólogos-nutricionistas J. Arthur Harris y Francis G. Benedict, y evaluaron parámetros metabólicos de 136 hombres y 106 mujeres. A estos hombres y mujeres se les realizó calorimetría indirecta (obtención del número de calorías emitido a partir del oxígeno consumido y de la producción de dióxido de carbono) y a partir de la evaluación de sus datos con análisis de regresión, se diseñaron fórmulas matemáticas para predecir el gasto energético en reposo, usando como variables la edad, el género, el peso y talla. En

1984, los datos y la fórmula original de Harris y Benedict fueron re-evaluados por Roza y Shizga determinando la masa celular activa total con mediciones de potasio corporal total (Ke). Basándose en sus resultados, propusieron algunas variantes a esa fórmula original de 1919. En 1990, Pellet publicó en el *American Journal of Clinical Nutrition* una completa revisión sobre las fórmulas matemáticas utilizadas para predecir el gasto energético en reposo. En esta publicación aparece una actualización de la fórmula original de Harris y Benedict que es la que se utiliza hoy en día.

Las ecuaciones originales de Harris-Benedict publicados en 1918 y 1919 [21]:

Hombres	TMB = 66,4730 + (13,7516 x peso en kg) + (5,0033 x altura en cm) - (6,7550 x edad en años)

Mujeres	TMB = 655,0955 + (9,5634 x peso en kg) + (1,8449 x altura en cm) - (4,6756 x edad en años)

Las ecuaciones de Harris-Benedict revisadas por Roza y Shizgal en 1984 [22].

Hombres	TMB = 88,362 + (13,397 x peso en kg) + (4,799 x altura en cm) - (5,677 x edad en años)
Mujeres	TMB = 447,593 + (9,247 x peso en kg) + (3,098 x altura en cm) - (4,330 x edad en años)

Las ecuaciones de Harris-Benedict revisadas por Mifflin y St Jeor en 1990 y utilizadas en la actualidad [23].

Hombres	TMB = (10 x peso en kg) + (6,25 × altura en cm) - (5 × edad en años) + 5
Mujeres	TMB = (10 x peso en kg) + (6,25 × altura en cm) - (5 × edad en años) - 161

Es así como diferentes problemas en diferentes lugares han puesto la mira en las aplicaciones móviles como una solución factible y segura. El Ingeniero Guerra [24] fijo como solución la creación de una aplicación móvil para los servicios de alimentos del comedor, en la universidad de Montemorelos, donde los alumnos pueden donar las comidas que no se consumen a otros estudiantes que no tienen las comidas inscritas en el semestre. La aplicación fue desarrollada para la plataforma iOS, utilizando las herramientas X Code, el framework X Code versión 5 y programado en el lenguaje Objective-C. Así también se diseñó una sección donde el alumno podía ver la información calórica, para saber cuántas porciones debe consumir en relación a sus actividades físicas diarias. La seguridad en las aplicaciones es imprescindible, es por eso que se implementará a la aplicación en la plataforma Android con un código seguro (token), cosa que en IOS no se implementó. La administración de datos de la app móvil se hizo un poco complicado para los administradores del comedor de la universidad, integrando a la aplicación un servicio web que se ligaba a esta y desde este servicio se subían los datos. Para arrancarlo se debía encender un servidor independiente (xampp), para poder accesar a los datos de la base, en cambio, lo que se pretende con esta nueva aplicación es desde el mismo móvil conectarse directamente a la base de datos de la universidad trayendo la información necesaria y sin necesidad de ayuda de otros programas.

La aplicación que se está construyendo en este proyecto se está basando en el proyecto Saum app del ingeniero David Guerra, pero para la plataforma Android, mejorando el sistema, y desarrollando la aplicación con cierta seguridad para que el alumno pueda estar seguro que sus comidas nadie las pueda alterar o donar sin su consentimiento. Por otra parte, podemos hacer referencia a que este tipo de aplicaciones nativas utilizan todos sus recursos solamente para la aplicación en uso, quiere decir que su fluidez a la hora de hacer ciertas acciones y conexiones, y así aprovechar los beneficios que da este tipo de aplicaciones nativas.

III. MÉTODOS

A. Procedimientos y métodos a utilizar

El prototipo para la aplicación se creó en Android Studio. Esta herramienta es muy intuitiva para trabajar en el diseño tanto como la lógica. Debido a que existen varios entornos de desarrollo para desarrollar en Android, se ha encontrado este IDE el cual cumple con los requisitos planteados y satisface las necesidades que se tienen. Android Studio es un entorno integrado para la plataforma Android compatible con Microsoft Windows, Mac OS X y GNU/Linux, proporciona asistentes y plantillas que verifican los requisitos del sistema, como el Java Development Kit (JDK) y la memoria RAM disponible, y configurar los ajustes predeterminados, tales como un dispositivo virtual de Android (AVD), emulación predeterminada optimizada e imágenes del sistema actualizados.

Dentro de Android Studio se implementó la interfaz con ayuda de la herramienta editor de diseño, la cual utiliza un estilo llamado Diseño de interfaz de usuario declarativo.

El proyecto maneja archivos XML para poder facilitar la complejidad del diseño de interfaz, declarando la estructura en un archivo definido por elementos intuitivos, a lo que conocemos por objetos. Para esto se utiliza la librería de Parsing XML, que es donde genera automáticamente el código Java a partir de una estructura predefinida de objetos.

En la figura 6 podemos ver la pantalla de inicio de sesión, integrada por 4 elementos, el logo, donde utilizamos un objeto llamado *imageView*, cargando la imagen respectivamente. Esta imagen se muestra en el archivo *login.xml* (pantalla de inicio de sesión). Dos *editText*, user y password, para el usuario y contraseña, donde se introduce

la información necesaria para poder hacer conexión con la base de datos y acceder a la aplicación.

Figura 6. Pantalla de login

Este archivo *xml* es llamado por una clase con el mismo nombre del Layout, haciendo referencia con el siguiente código:

```
setContentView(R.Layout.donar);
```

Como esta pantalla trabaja con un botón, se creó una referencia del mismo para que se pudiera manipularlo y programarlo y que además esté ligado con la parte visual que se declara en XML. Para ello utilizamos el método *findViewById()* que recibe como parámetro una ruta que parte de la referencia a la clase *R.java* que es la que genera los ID's para cada uno de los recursos incluidos en la aplicación (imágenes, layouts, colores, etc.).

```
findViewById(R.id.donarLayout)
```

Las clases asociadas a las pantallas contienen un método que siempre se ejecuta:

onCreate();

En este método se asocia al elemento con el objeto creado en el código como se puede ver en la siguiente figura.

Figura 7. Asociación del objeto con el elemento

Todas las clases que están ligadas con interfaces, como el menú, la pantalla de donar comidas, etc., heredan de la clase *ACTIVITY* ya que se ejecutan ciertas acciones como:

- Animaciones
- La declaración de los objetos visualizados en pantalla
- El llamado de otras clases

Para poder hacer conexión con la base de datos se creó la siguiente clase:

SigninActivity extends AsynTask<>

Consta de tres métodos, los cuales son:

- *onPreExecute()*
- *doInBackground()*
- *onPostExecute()*

Para poder hacer llamado a la clase mencionada mediante el botón de inicio de sesión se le creó una configuración en el archivo xml de este elemento como se muestra a continuación:

```
android:onClick="loginPost"
```

onClick está dando una orden al botón que se debe ejecutar el método en comillas, el cual está programado en la clase *login.java.* El código completo se encuentra en el apéndice B.

Este método es un método intermediario, es el que hace una referencia a la clase *SigninActivity extends AsynTask<>*, pasando dos parámetros como ya se había mencionado, el usuario y contraseña para poder comparar con los datos de la base y crear la conexión mediante el código que se puede ver en el apéndice B.

Para hacer todas las consultas a la base de datos se utilizó un servidor local, del cual ya se hablará más adelante. En este servidor se crearon archivos PHP los cuales contienen QUERYS, que son los que se encargan de realizar todas las operaciones en la base local que se creó para esta aplicación móvil.

Todas las conexiones que se requieren hacer a la base de datos para una operación, la aplicación Android, dependiendo la acción a realizar, llama al servidor y por medio de este al archivo PHP que hará dicha acción. En el caso de inicio de sesión, la clase *SigningActivity*, mediante el método *doInBackground()*, se conecta con el servidor y hace referencia al archivo *index.php* pasándole ciertos parámetros para iniciar sesión y ciertos comandos para ejecutar la acciones, que más adelante se explicará con detalle.

El siguiente método guarda el usuario (Matrícula) en la memoria del dispositivo para poder crear la sesión en curso, como se puede apreciar en la siguiente línea de código.

SharedPreferences.Editor editor =
sharedpreferences*.edit();*

*editor.putString(**"userKey"**, username);*
editor.commit();

Al momento de crear el objeto llamado editor de tipo *SharedPreferences.Editor*, se hace una referencia a que valores se guardarén en la memoria del dispositivo. El método *putString*(); se encarga de crear una variable y permite asignarle un valor. Al momento de llamar este método es necesario introducir dos parámetros, el nombre, en este caso "*userKey*", y el valor, username, que es por medio de una variable que le pasamos el valor.

Esta variable se utiliza como variable global en todas las operaciones que se tengan que hacer por medio del usuario que inició sesión. Para poder obtener este String guardado se utiliza la siguiente línea de código:

SharedPreferences SP =
PreferenceManager.getDefaultSharedPreferences(getBase Context());
String usuario = SP.getString("userKey", "");

El método *getString*(); obtiene las variables guardadas en el dispositivo, obteniéndolas mediante el nombre de estas, en este caso "*userKey*", que es como se le llamo a la variable global que guardó la matricula del usuario que inició sesión.

Al momento de iniciar sesión, como ya se explicó con anterioridad, la clase enlazada con el inició de sesión que ya se mencionó conecta con el archivo *index.php*, el cual recibe dos parámetros, la matrícula y la contraseña mediante las siguientes líneas de código:

$username = $_POST['Matricula'];
$password = $_POST['Contrasena'];

el caracter "$" quiere decir que se está declarando una nueva variable, en este caso se llama username y password, que es el que recibe los dos parámetros enviados.

Los datos que se necesitan para establecer conexión con la base y para poder iniciar sesión en la app se pueden enviar mediante dos tipos de métodos, el método POST y el método GET. Para esta app se utilizó el método POST, como se declara en las líneas de código anteriores, ya que en el método GET la información viaja mediante una URL haciéndose visible, a diferencia del otro que es más seguro, enviando los datos en segundo plano, no visibles para el usuario.

Al momento de tener el valor de la variable de las contraseñas, estas son leídas por el método escritos en el lenguaje PHP, tal y como se puede ver en la siguiente línea:

$password = hash('sha256', $password);

Ya que las contraseñas se han encriptado con tipo sha256, se leen con la línea anterior como método de seguridad.

También se ha asegurado de que el usuario no pueda ingresar caracteres especiales como otro método de seguridad para que no haya inyección de SQL y así de esta manera proteger la base de datos. Las inyecciones de SQL se introducen en los campos de texto para poder manipular la base de datos, borrando, cambiando información u obteniéndola. Al momento de introducir cualquier caracter, los intrusos se aprovechan de las comillas simples y dobles con las que se declaran cadenas de textos y Strings. Se implementó esta seguridad con las siguientes líneas de código

$username = mysql_real_escape_string($username);
$password = mysql_real_escape_string($password);

Este archivo *index.php* tiene dos funciones, crear la conexión e iniciar sesión mediante una sentencia QUERY. Para crear la conexión se agrega la siguiente línea:

$con=mysqli_connect("127.0.0.1","root","","saum");

Se crea la variable "*con*", y por medio del método *mysqli_connect();* le asignamos la dirección de la base de datos, el usuario, en este caso "*root*", la contraseña, y la base de datos.

Para poder inicar sesión se creó un QUERY mediante el método *mysqli_query();* en este método se escribe la consulta que se quiere hacer a la base de datos de acuerdo a lo que se necesite en la aplicación móvil, en este caso con el QUERY "*select*" Matricula, se selecciona la matricula que se le pasó en las líneas anteriores y la contraseña. Si existen, la base de datos devuelve la información para que el usuario inicie sesión, si no, la aplicación indica al usuario que "el usuario o contraseña son incorrectos".

$result *= mysqli_query ($con,"**SELECT** Matricula **FROM** login **where** Matricula=' $username' and Contrasena= ' $password'");*

Al momento de tener correctos los dos campos, automáticamente la app redirecciona al usuario al menú principal.

Para poder redireccionar a otro *Activity* (Pantalla o interfaz) se obtuvo el contexto (clase que contiene la interfaz global) creado anteriormente y se llamó la clase menú, que es el menú principal.

En el directorio *src* se encuentran varios directorios basados en el paquete:

com.example.jairo.myapplication

que se creó el proyecto Saum, lo que significa una ruta específica en la computadora:

com/example/Jairo/myapplication

Al crearse el proyecto se crean juntamente varios archivos por default, pero en el que nos interesa centrarnos a la hora de empezar con la aplicación es el *activityMain.java*. El proyecto lo toma como archivo central y es el que se ejecuta al iniciarse la aplicación móvil.

Para iniciar una o varias actividades, se invocó el método *onCreate()*, como se puede apreciar en las siguientes líneas de código:

```
protected void onCreate(Bundle savedInstanceState) {
   super.onCreate(savedInstanceState);
   setContentView(R.layout.login);

   //permite conexiones con servidores e/s (cada que haga conexion)
   StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
   StrictMode.setThreadPolicy(policy);
```

➔ Código completo puede verse en el apéndice A

Siempre que se usa este método se recibe como parámetro un objeto de la clase *Bundle*, que contiene el estado anterior de la actividad o pantalla en caso de que se haya suspendido. Después de que iniciamos una actividad podemos pausarla o detenerla de forma momentánea si el usuario está realizando otras funciones en el teléfono. Si esto ocurre, la actividad deberá iniciarse nuevamente, y es entonces cuando la información de este *Bundle* nos resulta de utilidad, pero por default se ha configurado para pausar la mayoría de las actividades en la aplicación para tener una mayor eficiencia.

En la figura 8 podemos observar la pantalla principal del menú que tiene acceso a diferentes secciones, como Donar, Tabla de Calorías, Menú del día y a las comidas donadas o disponibles. Cada botón se creó con un elemento llamado *ImageButton*, que se utiliza para acceder a la pantalla deseada.

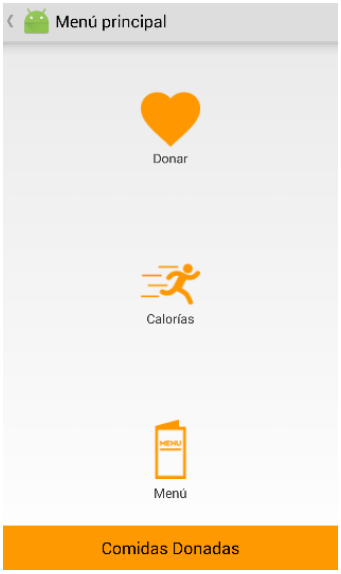

Figura 8. Pantalla del menú principal

Cada *ImageButton* está asociado a un método para poder llamar a otra pantalla o layout mediante la siguiente línea de código:

setOnClickListener()

que es el que se encarga de detectar el click que se hace al botón con ayuda de los siguientes parámetros

***new** View.OnClickListener()*

y con ayuda de otro método

***public void** onClick(View v)*

creando una clase abstracta de tipo *intent* dentro de este para realizar la acción. Cuando se detecta el click del primer método, enseguida se pasa al onClick y se llama al archivo *menucomidas.class*, en este caso, y es cuando se abre el otro Layout declarado en este archivo *.java*.

Dentro de este menú se creó un sub-menú. Se crea un método en el cual se llama al objeto que contiene esta configuración por default mediante otro método llamado *Menu Inflater*, como lo podemos ver en el siguiente código.

```
public boolean onCreateOptionsMenu(Menu menu) {
   // Inflate the menu; this adds items to the action bar if it
is present.
   getMenuInflater().inflate(R.menu.menu_main, menu);
   return true;
```

Mediante el segundo método se agregan los ítems que se insertaron en el archivo *menu_main.xml*. Estos son los que conforman el sub-menú, que es creado en la barra de acciones.

Los ítems del sub-menú son agregados mediante el siguiente código:

```
<item android:id="@+id/agregar_comidamenu"
android:title="@string/agregar_comidamenu"
   android:orderInCategory="100"
app:showAsAction="never" />
<item android:id="@+id/cerrar sesión"
android:title="@string/cerrar sesión
   android:orderInCategory="100"
app:showAsAction="never" />
```

El código @+*id* funciona como identificador del ítem, y @string como se visualizara en la interfaz, o sea en el sub-menú.

El ítem cerrar sesión está ligado a un método llamado *logOut();* Este método lo que hace es borrar todas las variables globales por seguridad y así no estén disponibles hasta que se vuelva iniciar sesión, como se muestra en las siguientes líneas de código:

```
public void logout(View view){
   SharedPreferences sharedpreferences =
getSharedPreferences(this.MyPREFERENCES,
Context.MODE_PRIVATE);
   SharedPreferences.Editor editor =
sharedpreferences.edit();
   editor.clear();
   editor.commit();
}
```

El ítem de agregar comidas solo es para administradores. Este ítem está ligado a una clase llamada agregarComidaMenu{}. En esta sección el administrador puede seleccionar un platillo de los que están cargados en la base de dato y cambiar la fecha en que se va a servir al igual si se servirá en el desayuno, comida o cena como se puede apreciar en la figura 9.

Figura 9. Sección agregar comidas para administradores

Se creó un método dentro de esta clase para poder generar un objeto en el cual se pueda elegir la fecha para no entrar en errores de formatos. Al momento de seleccionar la fecha en el calendario que se muestra en la figura anterior, la fecha automáticamente se carga en una caja de texto. El código que se implementó para crear este objeto de tipo calendario es el siguiente:

```
public void onCalendarButton(View view){
   Calendar c = Calendar.getInstance();
   int mYear = c.get(Calendar.YEAR);
   int mMonth = c.get(Calendar.MONTH);
   int mDay = c.get(Calendar.DAY_OF_MONTH);
   System.out.println("the selected " +  mDay);
   DatePickerDialog dialog = new DatePickerDialog(this,
         this, mYear, mMonth, mDay);
   dialog.show();
}
```

La sección donar cuenta con una pantalla conformada por varios elementos como se puede observar en la figura 9.

Figura 9. Pantalla de donación

Contiene tres elementos de tipo CheckBox, los cuales le permiten al alumno seleccionar la comida que va a donar, ya sea desayuno, comida o cena. Estos elementos fueron declarados con una etiqueta llamada CheckBox como se puede ver en el siguiente código:

```
<CheckBox
   android:id="@+id/checkComida"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Com"
   android:checked="false"
   android:layout_marginTop="23dp"
   android:layout_below="@+id/logo"
   android:layout_centerHorizontal="true" />
```

Android:id es el nombre que se le da a este elemento, como ya se ha explicado ateriormente, en este caso chckComida; esto se hace con cada checkBox y así también con cada elemento de la pantalla donar. Se configura el tamaño mediante las líneas:

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
```

estas líneas declaran que su tamaña será responsivo (adaptable al texto). La posición de cada elemento es mediante los márgenes y en relación a otros objetos, como la líneas de **android:layout_below="@+id/logo",** quiere decir que este elemento está posicionado debajo del elemento con id *logo*. Al momento de elegir si se donará a un alumno en específico la comida se debe introducir en el campo a través de un editText, elemento creado con xml. Al igual que los objetos CheckBox, este se crea con una etiqueta pero llamada <EditText/>. Dentro de esta etiqueta se configura la posición mediante las siguientes líneas

```
android:layout_width="150dp"
android:layout_height="wrap_content"
```

teniendo un ancho de 150 dp (densidad de pixeles) y un alto adaptable. Se configuró el editText para que solamente se puedan ingresar números con la siguiente línea de código **android:inputType="number".**

El elemento checkBox con etiqueta "Solo donar" permite al alumno hacer una donación aleatoria, esto quiere decir que se genera una comida en la base de datos para algún alumno que lo necesite sin necesidad de donarla directamente a su matrícula. Esta parte quedará como aporte a futuro, ya que para que esta sección se necesita de una depuración de matrículas de toda la base de datos de la Universidad de Montemorelos ya que no todos los alumnos que no inscriben las comidas quiere decir que sea por no tener recursos, simplemente por comer fuera de la universidad; este aporte a futuro se explicará más adelante.

Una vez seleccionada la comida que se va a donar y a quien, se creó un botón el cuál sigue los mismos pasos de los objetos anteriores para crearse en el código XML, pero a diferencia de estos, este botón está ligado a un método que llama a una clase similar a la que conecta la app con la base de datos para poder iniciar sesión. Esta clase llama a un archivo PHP donde se crea una sentencia QUERY para poder manipular la base de datos. Se verifica si el alumno que está donando tiene las comidas inscritas para después validar la donación.

La lógica de esta sección se encuentra en el QUERY. Al momento de dar click en el botón donar se llama una clase llamada *DonarComidas.* Pero antes, si el campo de la matrícula a la cual se le donará la comida o las comidas, está vació, o si selecciona el modo aleatorio se genera el siguiente mensaje: *"Ingresa una matrícula. La donación aleatoria no está disponible por este momento"*, mediante la siguiente línea de código:

```
Toast.makeText(this, "Ingresa una matrícula. La donación aleatoria no está disponible por este momento", Toast.LENGTH_SHORT).show();
```

La clase *Toast* contiene el método *makeText();* Este método requiere de dos parámetros, el primero que es el texto a mostrar en el mensaje de diálogo, y el tiempo que tardará mostrándose. En seguida se utiliza el método *show();* para iniciar el cuadro del mensaje.

Al momento de ingresar la matrícula a la cual se le donara la comida seleccionada se ejecuta la clase que se mencionó. En esta clase se pasan 4 parámetros, desayuno, comida, cena, Matricula a la que se le donará, y el ultimó parámetro se obtiene de la sesión en curso, que es la matrícula del donante, mediante las siguientes líneas de código:

```
String usuario = arg0[0];
String donadaMatricula = arg0[1];
String desayuno = arg0[2];
String comida = arg0[3];
String cena = arg0[4];
```

Las variables, como se puede apreciar se pasaron mediante un arreglo llamado arg[]. Se crearon variables para poder identificar cada uno de los datos y así enviarlos al archivo *donarComidas.php*. En este archivo se agregan las comidas donadas en una tabla llamada donaciones. Esta tabla se compone por 6 columnas, como se puede apreciar:

I. Tabla de donaciones (Base de datos Saum).

MatriculaDonante	MatriculaDonado	fecha	desayuno	comida	cena
1140443	1100294	2016-04-03	1	1	1
1077789	1080699	2016-04-03	1	1	0
1150556	965213	2016-04-03	1	1	1
1100294	1032109	2016-04-05	1	1	1

Al momento de donar la o las comidas también se guardan las variables de desayuno, comida y cena. Dependiendo cual haya seleccionado se le asigna un 1 o un 0, si está seleccionado se le asigna valor de 1, de lo contrario un 0 con las siguientes líneas de código:

```
com.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
    if(com.isChecked()) {
      comida = "1";
    }else {
      comida= "0";
    }
  }
});
```

Com es el objeto que se creó en la interfaz como se ya explicó, y lo que se ejecuta en la primera línea es el método para escuchar al momento de dar click sobre él. La línea ***if(com.isChecked())*** comprueba si el elemento esta seleccionado, y si es así a la variable comida se le asigna 1, como ya se explicó. Al momento de tener todos los datos correctos se mandan a la clase *DonarComidas.java*. Se

empaquetan estos para enviarlos al archivo *php* que ya se mencionó.

Como se mencionó, el botón donar crea una instancia de la clase *DonarComidas.java* de la siguiente manera:

```
new DonarComidas(this).execute(usuario, "" +
donadaMatricula, desayuno, comida, cena);
```

Al momento de transferir las variables al archivo php mencionado se obtienen los valores en este archivo:

```
$username = $_POST['Matricula'];
$userDonado = $_POST['MatriculaDonada'];
$desayuno = $_POST['Desayuno'];
$comida = $_POST['Comida'];
$cena = $_POST['Cena'];
```

Enseguida se consulta la tabla login para saber si existe la matrícula a la cual se le esta donando con la siguiente sentencia QUERY:

```
"SELECT * FROM login where Matricula='{$userDonado}'
";
```

Una vez obtenido la consulta como verdadera, se consultaron las comidas que tiene inscrito el usuario que está donando, si es así continúa con la acción de donar las comidas seleccionadas en la app móvil y se guarda un registro en la tabla de donaciones.

Para la sección de calorías se tiene una pantalla que calcula de acuerdo a la edad del usuario en sesión, sexo, peso, estatura, y de acuerdo a las actividades físicas que realiza. Si bien, no todos los usuarios realizan las mismas actividades, algunos son inactivos 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 [23].

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

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

Una vez obtenido el resultado de la fórmula se toma el valor dependiendo la actividad física realizada mostrado en la siguiente tabla:

II. Tabla de kcal de acuerdo a la actividad física realizada

Actividad Física	Kcal x Peso ideal
Poco o ningún ejercicio	Calorías diarias necesarias = TMB x 1,2
Ejercicio ligero	Calorías diarias necesarias = TMB x 1,375
Ejercicio moderado	Calorías diarias necesarias = TMB x 1,55
Ejercicio intenso	Calorías diarias necesarias = TMB x 1,725
Ejercicio muy intenso	Calorías diarias necesarias = TMB x 1,9

Los valores que se necesitan para la fórmula son obtenidos a través de la pantalla presentada en la sección de calorías, como se puede ver en la siguiente figura:

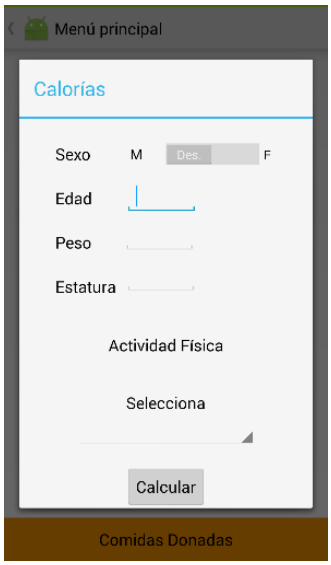

Figura 10. Pantalla de cálculo de calorías

Para crear esta pantalla se añadió un objeto de tipo Switch, que es el que selecciona si es masculino o femenino el usuario. Este elemento se crea en el código XML mediante la etiqueta <Switch/> con sus respectivas configuraciones como ya lo hemos explicado con anterioridad. Al momento de seleccionar M o F se obtiene el valor mediante una condición, preguntado si el switch está encendido, quiere decir que el usuario seleccionó F, si está apagado es M, como se muestra en las siguientes líneas de código:

```
if (switchSex.isChecked()){
   sex="f";
}else {
   sex="m";
}
```

También se obtienen los datos faltantes para poder aplicar la fórmula mediante las siguientes dos líneas de código:

```
int edad = Integer.parseInt(txtEdad.getText().toString());
int estatura =
Integer.parseInt(txtEstatura.getText().toString());
int peso = Integer.parseInt(txtPeso.getText().toString());
```

Al momento de tener todos los valores en una variable se aplica la fórmula sustituyendo automáticamente los valores que se han introducido por el usuario con el siguiente código:

```
if (sex == "f") {
   calorias = Integer.parseInt(peso) * 10 +
Integer.parseInt(estatura) * 6.25 – 5 *
Integer.parseInt(edad) - 5;
} else {
   calorias = Integer.parseInt(peso) * 10 +
Integer.parseInt(estatura) * 6.25 – 5 *
Integer.parseInt(edad) - 161;
}
```

La línea de código **if** (**sex** == **"f"**) es una condición donde se pregunta si la variable creada con el nombre *sex* es igual a f (femenino). Si se cumple esta condición se hacen las operaciones dentro de las llaves {}.

Para poder ingresar a la fórmula la actividad física, se creó un elemento de tipo spinner donde el usuario selecciona el tipo de actividad de acuerdo a la tabla I. ya mencionada con el siguiente código:

```
<Spinner
   android:layout_width="200dp"
   android:layout_height="wrap_content"
   android:id="@+id/spinnerCalorias"
   android:entries="@array/caloriaTable"/>
```

donde las líneas de código **android:layout_width** y **android:layout_height** configuran el ancho con 200 pixeles y una altura adaptable. La línea **android:id** simplemente asigna el nombre al objeto. Donde nos interesa centrarnos es en la última línea, la cual se encarga de direccionar el objeto a los valores que se requieren mostrar en la interfaz de esta pantalla. **android:entries="@array/caloriaTable"/>** Hace referencia a una variable de tipo array llamada *caloriaTable*, que se encuentra ubicada en la carpeta *values*, en un archivo llamado *Strings.* Aquí se declara el array (arreglo de datos) y se ingresan los valores a mostrar:

```
<string-array name="caloriaTable">
   <item>Poco o ningún ejercicio</item>
   <item>Ejercicio ligero</item>
   <item>Ejercicio moderado (3-5 días a la sem)</item>
   <item>Ejercicio intenso (6-7 días a la sem)</item>
   <item>Ejercicio muy intenso</item>
</string-array>
```

Al momento del usuario querer entrar a la calculadora de calorías, el botón hace una consulta automáticamente a la base de datos para verificar si este ya ha hecho el cálculo, con el siguiente QUERY:

"SELECT Calorias FROM login where Matricula='$username'"

donde *SELECT* hace referencia a la columna donde se están registrando las calorías, *FROM* de que tabla se obtienen los resultados en general, y *where* selecciona el usuario en sesión, enviándole la variable llamada *username*, que es la matrícula que identifica al estudiante. Si es así, simplemente manda un diálogo mostrando las calorías que necesita, de lo contrario lo direcciona a la calculadora mostrada en la figura 10 y mediante un archivo php llamado *insertarCalorías*, se guardan en la base de datos de acuerdo al alumno en sesión.

La pantalla de menú del día se creó con un ListView. Este objeto es un grupo que muestra una lista de elementos desplazables, que se insertan automáticamente a la lista utilizando un *adapter* que lanza el contenido de un origen, ya sea una matriz o una consulta a la base de datos, que es el caso con la aplicación móvil Saum, convirtiendo cada elemento que se coloca en la lista como se puede apreciar en la figura 11.

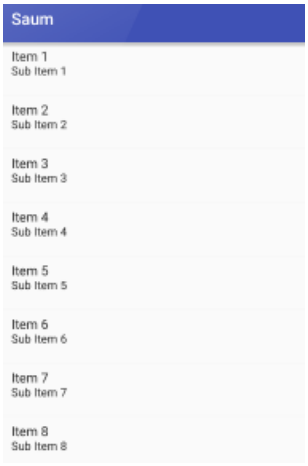

Figura 11. Pantalla de menú

Se creó una clase llamada *MenuComidas*, la cual asigna el View al layout donde se encuentra la lista, que es *menucomidas.xml*. Con un objeto de tipo ListView se hace referencia al elemento en el layout para poder pasarle los datos o valores. Se crea un arreglo para poder meter cada valor y así enviarlos por medio de un *arrayAdapter*.

Estos datos se obtienen desde la base de datos por medio de un archivo PHP. Todas las consultas o peticiones que se le hacen a la base son mediante este tipo de archivos. Dentro de él se hace una consulta a la tabla *menudia*, haciendo una petición de todos los datos dentro de esta tabla con el siguiente QUERY:

"SELECT * FROM menudia"

Los datos son obtenidos en una variable. Teniendo varias columnas en la base, la información llega de una manera desordenada haciendo difícil el manejo de estos datos a la hora de mostrarlos en la pantalla de la aplicación y para esto usamos las herramientas JSON y GSON.

JSON es un formato de intercambio entre clientes basado en la sintaxis de JavaScript para representar estructuras en forma organizada. Facilita el desarrollo y comprensión de intercambio de datos. La librería GSON transforma un formato JSON a un objeto java, que es lo que se hizo con la clase COMIDA para poder mostrar el menú del día. Para poder usar el repositorio de GSON se debe añadir la siguiente dependencia:

compile *'com.google.code.gson:gson:2.3'*

La clase principal de la librería es GSON. Para implementar sus funcionalidades debemos instanciar un nuevo objeto y llamar el método deseado.

Para convertir datos atómicos o no organizados en formato JSON se usó el método *toJson();*.

En el archivo PHP se crea un método para poder obtener la fecha actual, como se muestra en el apéndice D, y en base a esta fecha se crea una sentencia para traer todas las comidas referentes a esta fecha en adelante. El número uno, como podemos ver en el apéndice E, hace referencia al día actual, y de ahí en adelante se muestran las comidas con fechas posteriores.

Se creó un catálogo de platillos, el cual contiene información de estos como el nombre, las calorías y un id de la imagen para poder mostrarse en la interfaz de esta sección.

La última pantalla es la que proporciona al usuario la información de cuantas comidas tiene disponibles, figura 12. Si algún estudiante ha donado alguna comida simplemente le aparecerá en 0 y al alumno que le donó le aparecerá como disponible.

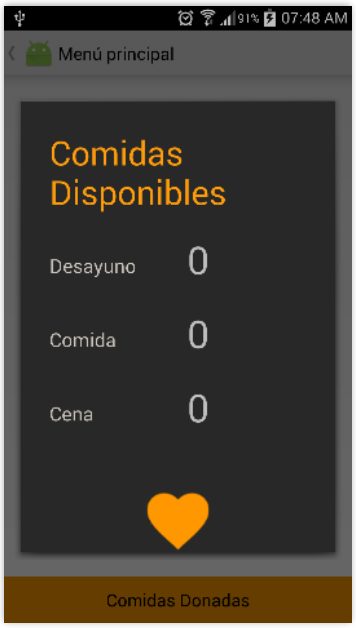

Figura 12. Comidas donadas/disponibles

Cuando se quiere hacer una consulta a la base de datos se utiliza la misma metodología que se utilizó para las demás pantallas que también consultan a la base. Se llama al método ***comidasDisponibles(),*** y este método llama a la clase que conecta el archivo PHP con la base, obteniendo las comidas que se le han donado al usuario.

Todos los archivos XML tienen dos vistas, la vista Design (diseño) y Text (texto o código). En la parte de diseño es de estilo drag and drop (arrastrar y soltar). Para esto se tiene una barra de herramientas (paleta) de ayuda para poder crear el diseño de interfaz dividido en 7 secciones los cuales se han utilizado algunos para crear la interfaz de la aplicación:

.- Layouts: Son contenedores donde los elementos se ordenan y comportan de diferente manera según el tipo de Layout establecido.

.- Widgets: Elementos principales en el diseño de una pantalla.

.- Text Fields: Como su nombre lo dice, son campos de textos con diferentes parámetros y para diferentes usos, por ejemplo para cadenas de caracteres, para cadena de números, multi-texto entre otros.

.- Containers: En esta sección entran algunos widgets contenedores como el group list, radio group

.- Date & Time: Contiene widgets relacionados con la fecha y hora, como reloj, calendario, cronometro, etc.

.- Expert: Contiene algunos widgets más avanzados y algunos contenedores para apps más complejas

.- Custom: Son formatos personalizados, casi no se usa esta sección de la paleta de configuraciones.

A cada vista o pantalla se le personalizó un Layout. Sin este contenedor no se podría añadir ningún elemento de la paleta. Existen 7 Layouts:

- FrameLayout
- LinearLayout (Horizontal)
- LinearLayout (Vertical)
- TableLayout
- TableRow
- GridLayout
- RelativeLayout

Con el que se trabajó en todas las pantallas fue el RelativeLayout. En comparación con los demás, este Layout permite posicionar los elementos hijos en función de los padres o de otros, alinear hacia la izquierda o derecha en relación a otros elementos, por ejemplo:

```
android:Layout_below="@+id/checkComida
android:Layout_alignLeft="@+id/checkComida
```

donde el *editText* se alinea con el elemento anterior (*checkComida*) a este y se alinea del lado izquierdo.

En la carpeta raíz del proyecto hay un archivo XML llamado *AndroidManifest.xml*, en el cual se declaran todos los componentes que forman parte de la aplicación, junto con algunas configuraciones generales. Todas y cada una de las Activity´s (vistas o pantallas) de la app deben ser declaradas en esta sección [25], así como cada de elemento de todos los Layout´s.

En este archivo *manifest* también se declaran los permisos que se concede a la aplicación por parte del dispositivo, como el acceso a los datos, el acceso a la conexión WI-FI, a cualquier tipo de conexiones y a poder descargar información.

Para cada pantalla del diseño de la aplicación es requerido un archivo XML, que como ya hemos mencionado es el que ayuda a definir el diseño de la interfaz, el estilo de los objetos, etc. Cuando se quiere agregar un objeto se puede directamente en el código, o como se dijo, que se usa una librería la cual permite facilitar el diseño, se pueden arrastrar los objetos a la pantalla y automáticamente se crean por default en el código XML.

Cuando se habla de archivos XML, se habla de diseño. En el proyecto, por default se crea una carpeta Layout, que está dentro de la carpeta “res”, que es la carpeta exclusiva de diseño, y en esta carpeta Layout se tienen los archivos XML de las vistas de la aplicación, como por ejemplo la de *login.XML*. Estos archivos se manejan por vistas, si se requiere modificar algún objeto en la interfaz de esta vista hay que acceder al archivo XML Al momento de programar alguna acción en cierto objeto, lo que se hace es acceder a la carpeta JAVA, que es donde están los archivos *.java*, y se crea un archivo con el mismo nombre que él *.xml*, por ejemplo *login.java*. Dentro de este archivo se van declarando los objetos para poder ir trabajando con ellos y programarles acciones.

Los archivos XML se crean con diferentes temas, llamados Activity´s. Existen diferentes tipos como se puede ver en la figura 12.

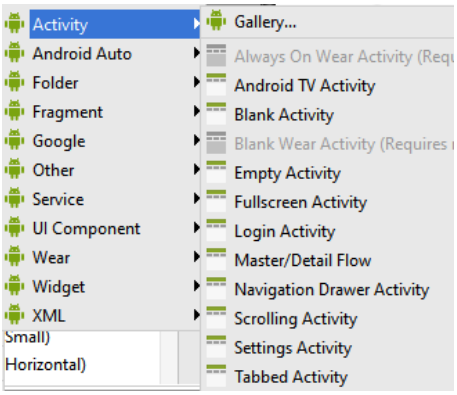

Figura 12. Tipos de Activity´s

La aplicación Saum se ha creado con el tema Blank Activity, ya que es predeterminado o estándar para la mayoría de las aplicaciones móviles y es más fácil de manipular.

El template o tema, tiene un tipo de navegación simple que incluye:

- Barra de título (Barra de acciones en Android 3.0 y posteriores)
- Menú de opciones
- Disposición básica

Para crear la animación de transición entre pantallas se creó un archivo nuevo XML en la carpeta *anim* llamado *fade_in.xml*. Este archivo es cargado en la vista o pantalla donde se quiere tener la transición, se crea el objeto con el código:

```
private Animation animFadein;
```

se carga la animación en el Layout de la pantalla activa y se inicia con el método *.startAnimation* con el siguiente código:

```
menuLayout.startAnimation(animFadein);
```

La base de datos se creó en un servidor local. No se permitió usar la base de daros de la Universidad ya que se tuvieron algunos inconvenientes con los administradores del comedor y de finanzas.

Para montar la base y el servidor se utilizó la herramienta XAMPP, donde se puede administrar la base de datos

mediante una interfaz como se puede apreciar en la figura 13, teniendo una vista preliminar de todas las funciones que se pueden ejecutar en las tablas de la base de datos de la aplicación móvil, agilizando y facilitando la manipulación de esta.

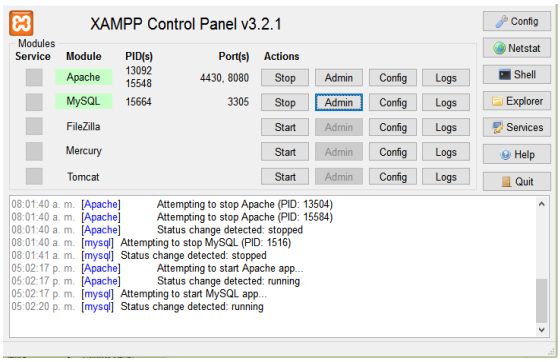

Figura 13. Administrador del servidor y base de Base de datos

Para poder hacer la conexión con la base de datos y poder manipular la información se crearon las diferentes clases ya explicadas que se entrelazan a los archivos PHP guardados en la carpeta del servidor virtual. Los datos que se necesitan para establecer conexión con la base y para poder iniciar sesión en la app son enviados mediante el método POST ya explicado en la pantalla de login como método de seguridad.

La aplicación móvil se creó pensando en los administradores del comedor para facilitar el uso de esta. Al momento de iniciar sesión se verifica si el usuario pertenece a un estudiante o a un administrador. Si es administrador se le conceden privilegios para poder ingresar a las comidas del día, los platillos creados en el catálogo. El administrador puede seleccionar de entre todos los platillos para crear el menú del día y asignarle una fecha, así como se aprecia en la figura 14.

Figura 14. Pantalla para crear menú del día

Al momento de seleccionar el platillo puede elegir si será desayuno, comida o cena, y como el catálogo ya tiene asignado cuantas calorías tiene cada comida ese dato se agregará automáticamente.

Para crear la interfaz se utilizaron algunos elementos que ya se explicaron los pasos para crearlos como la imagen, que se crea con un *imageView*, *TextView* y un botón. En lo que se quiere enfocar en esta interfaz es en el *scrollView*. Este elemento se crea con una etiqueta <*ScrollView*/> como todos los demás objetos. Dentro de este elemento se creó un layout, un contenedor el cual contiene 6 elementos, 3 TextView los cuales informa al administrador donde seleccionar el desayuno, comida, etc., y 3 *spinners*, como se muestra en la figura 15, los cuales cargan el catálogo completo cada uno para así crear el menú del día.

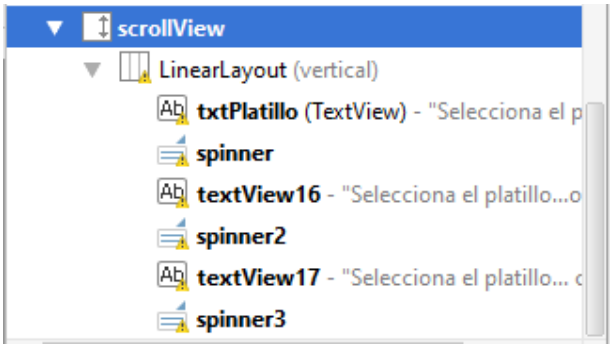

Figura 15. Contenido del elemento *scrollView*

Para poder elegir el platillo para el desayuno, comida y cena, y crear el menú del día se crearon 3 spinner como ya se mencionó, y como se puede ver en la imagen anterior. Los cuales en el código XML se declaran de la siguiente manera:

```
<Spinner
   android:layout_width="150dp"
   android:layout_height="wrap_content"
   android:id="@+id/spinnerDes"
   android:layout_below="@+id/txtPlatillo"
   android:layout_centerHorizontal="true"
   android:layout_marginTop="2dp" />
```

Anteriormente se ha explicado el código XML al momento de crear los objetos, así se omitirá esta parte y avanzaremos a explicar cómo se llena un spinner mediante un adaptador.

El primer paso es recuperar el control *Spinner* del archivo XML a una variable de la clase con el mismo nombre para manipularlo en la actividad, como se ha visto anteiormente. Para eso utilizamos la línea:

```
spinnerDes = (Spinner) findViewById(R.id.spinnerDes);
```

Teniendo la variable *spinnerDes* **se** creó el adaptador, recordando que éste es el que indica al *Spinner* cuáles son las opciones y el aspecto visual que tienen los ítems:

```
ArrayAdapter adapter1 =
ArrayAdapter.createFromResource(
   this, R.array.catalogoPlatillos,
android.R.layout.simple_spinner_item);
```

Para el adaptador se utilizó el método *createFromResource()* que sirve para crear ArrayAdapters utilizando recursos externos. Todos los archivos que se visualizan en el *spinner* se crearon en un archivo XML, en el directorio *res, que* son recursos que podemos mandar a llamar en nuestro proyecto para tener bien ordenadas las cosas. Volviendo al método, éste recibe tres parámetros:

- Contexto. Que es en dónde se utilizó el adaptador, en este caso es dentro de la actividad, por ello se le pasó *this* como valor.
- El ID del recurso del array. En este caso corresponde al archivo que contiene la declaración del arreglo de valores. Se hizo uso del formato *R.array.catálogoPlatillos* que corresponde a la clase *R* que genera todos los identificadores de cada recurso del proyecto, *array* por el tipo de recurso que se llamó, y *catálogoPlatillos* que es el nombre del arreglo de Strings en el archivo *arrays.xml*.
- El ID del recurso que define el aspecto visual de cada una de las opciones. Se utilizó el recurso predefinido por Android.

Después, para definir la forma en la que se desplegó toda la lista de opciones se utilizó el método *setDropDownViewResource()*. Ya teniendo el adaptador listo, se asignó a la variable *Spinner* utilizando la siguiente línea:

```
spinnerDes.setAdapter(adapter);
```

Después se asignó el listener al *Spinner* por medio del método *setOnItemSelectedListener()*. En la inner class *OnItemSelectedListener* se implementaron los métodos *onItemSelected()* para cuando el usuario seleccione una opción del *Spinner* y *onNothingSelected()* cuando no haya seleccionado una opción. Cuando se selecciona una opción estamos indicando que se muestre una notificación *Toast* con el texto.

Al momento de tener seleccionado los platillos estos se guardan en variables para poder insertar los datos en la base mediante el método que se ha implementado en todas las acciones anteriores mediante una clase que conlleva a dicho método.

La interfaz pide al administrador seleccionar una fecha para el menú mediante un *ImageButton*. Para poder seleccionar la fecha, al momento de hacer click en el botón, se muestra un calendario, el cual se genera a partir de las siguientes líneas:

```
public void onCalendarButton(View view) {
   Calendar c = Calendar.getInstance();
   int mYear = c.get(Calendar.YEAR);
   int mMonth = c.get(Calendar.MONTH);
   int mDay = c.get(Calendar.DAY_OF_MONTH);
   DatePickerDialog dialog = new DatePickerDialog(this,
        this, mYear, mMonth, mDay);
   dialog.show();
}
```

Se creó un método *onCalendarButton* en el cual creamos una serie de variables de tipo entero para guardar el año, el mes y el día. En seguida se crea un objeto llamado *dialog* de tipo *DatePickerDialog* que es el que muestra el calendario a través del método *.show();* Una vez guardado todos los datos correctamente se envían llamando la clase *insertarMenu*, enviando de esta clase a través de un método al archivo PHP llamado *crearMenu*. Al momento de insertar el menú, este es mostrado en la sección de menú de la interfaz menú principal.

IV. RESULTADOS

Los resultados del prototipo fueron favorables, ya que se abarcará a la mayoría de los estudiantes de la Universidad de Montemorelos, dado como referencia el uso del sistema operativo Android como el más usado en el medio. La aplicación se conectó a la base de datos exitosamente permitiendo el funcionamiento correcto de esta. El inicio de sesión fue exitoso con ayuda de la conexión como se puede observar en la figura 16.

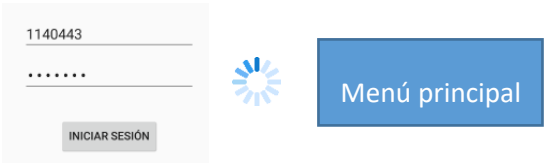

Figura 16. Inicio de sesión

redireccionando correctamente al menú principal, como se aprecia en la figura 16.

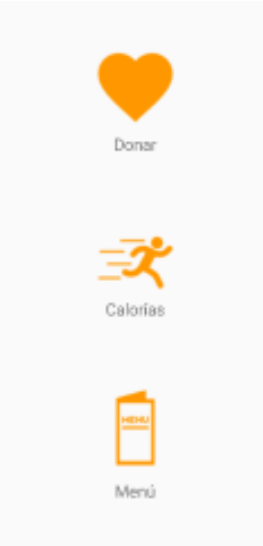

Figura 17. Menú Principal

En dado caso de no tener bien los campos de inicio de sesión se muestra un error notificando al usuario que cualquiera de estos dos datos es erróneo, como se puede observar en la figura 18.

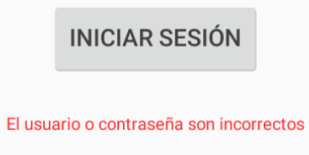

Figura 18. Error de inicio de sesión

Las donaciones fueron exitosas de matrícula a matrícula y aleatoriamente, teniendo un sistema seguro y manipulando correctamente los datos del sistema. Así también el estudiante puede ver las calorías que necesita según sus actividades físicas diarias.

Gracias al menú de comidas mostrado en la sección de menú, al alumno se le permita optar si consumir los alimentos del día o bien, donarlos a algún estudiante que lo necesite, como se muestra en el apéndice E.

Los administradores pueden subir el menú para el alumnado con comidas inscritas mediante la misma aplicación móvil y no con ayuda de aplicaciones terceras. Así también elegir si consumirán o no los alimentos para donarlos, evitando el desperdicio de estos.

También se creó un plan de prueba en el cual se trató de cubrir todas las secciones de la aplicación. Este plan de prueba también es definido como pruebas de caja negra, las cuales consisten en probar el software sin tener en cuenta su funcionamiento interno, simplemente las entradas que recibe y los resultados, o sea las salidas. Se creó una tabla para poder organizar las pruebas. Se seleccionó sección por sección, y a cada prueba realizada a dicha sección se le asignó un ID (nombre). Cada ID tiene una descripción de lo que se hizo en la prueba, y enseguida se describen los resultados esperados. Una vez teniendo estos resultados se describen los resultados actuales en la aplicación móvil, así como se puede apreciar en el apéndice F.

Al igual que las pruebas de caja negra, o el plan de prueba, se realizaron las pruebas de caja blanca. Estas pruebas se centran en los detalles procedimentales del software, lo que quiere decir que está ligado al código fuente. Para estas pruebas se escogieron diferentes valores de entrada para examinar cada uno de los flujos posibles de ejecución del código, de la aplicación, y así cerciorar que se devuelvan los valores de salida adecuados.

Las líneas de código para poder iniciar sesión como se recuerda reciben parámetros para poder compararlos con los datos de la base. Estas 3 líneas obtienen los datos ingresados por el usuario:

```
$username = isset($_REQUEST['Matricula']) ? $_REQUEST['Matricula'] : '';
//$password = isset($_REQUEST['Contrasena']) ? $_REQUEST['Contrasena'] : '';
```

La prueba se ejecutó en el navegador, ingresando valores aleatorios e imprimiendo en consola para poder observar los valores de salida. Al momento de mandar las variables se ejecuta mediante la siguiente línea de código la comparación para poder iniciar sesión:

$result = mysqli_query($con,"SELECT Matricula FROM login where

El resultado, mandando los datos correctos, fue satisfactorio. Si se ingresa el usuario y contraseña tal y cual están en la base de datos, la consola devuelve la matrícula imprimiéndola, tomando como este un resultado positivo, ya que así se programó la prueba de caja negra. En cambio, se ingresó una matrícula correcta pero una contraseña que no existe en la tabla de la base de datos. Para la prueba, se creó una condición, si no existe la matrícula, se debe imprimir en la consola: “contraseña inexistente”.

Al momento de traer cualquier dato de un archivo PHP, se creó una condición en todos estos archivos. Cuando se crea la consulta y selección de datos, y el resultado es correcto, se regresan valores, pero si existe un error se imprime en la consola algún texto que indique que hubo un error, así como se muestra en las líneas de código:

```
if ($result = mysqli_query($con, $sql){
    $resultArray = array();
    $tempArray = array();
```

```
        while($row = $result->fetch_object())
        {
                $tempArray = $row;
           array_push($resultArray, $tempArray);
        }
    }else {
    Echo "error en la consulta"
    }
```

Cuando algún alumno inicia sesión, la matrícula se guarda en el dispositivo, como ya se ha explicado. La prueba de caja blanca se aplica imprimiendo las variables guardadas de cada sesión mediante la siguiente línea:

```
SharedPreferences SP =
PreferenceManager.getDefaultSharedPreferences(getBase
Context());
String usuario = SP.getString("userKey", "");
if(!usuario.isEmpty()){
   System.out.println("usuario en sesión:  "+usuario);
```

Como ya se explicó, se crea un objeto que obtiene la variable guardada con el nombre "userKey", mediante la cuarta línea, con ayuda del método *getString();* guardando en una variable creada con nombre usuario, que es la que se imprime al final de la línea de código de color rojo.

V. DISCUSIÓN

La aplicación móvil creada por D. Guerra [24 no permitía que todos los estudiantes la aprovecharan, debido a que fue desarrollada únicamente para la plataforma iOS, ya que no todos utilizan el mismo sistema operativo móvil. Este prototipo de aplicación móvil permitirá solucionar el problema de poder abarcar a toda la sociedad universitaria, gracias a esta herramienta que ayuda a los estudiantes a poder elegir si consumir o no los alimentos para aprovecharlos, donándolos a otros alumnos mediante un sistema seguro y gracias al menú del día publicado.

La aplicación se creó pensando en los administradores, facilitando el acceso a la aplicación móvil y administrando las comidas de cada día así como el menú completo. Esta función se planteó y se agregó en este proyecto ya que se tuvieron problemas con la app móvil anterior al momento de gestionar la información acerca del menú. Para poder acceder a esta sección, se creó un servicio web externo a la aplicación que se conecta con la misma base de datos, se tenía que iniciar una aplicación la cual contiene dicha base. A los administradores se les hizo un poco complicado realizar dichas tareas, y es por eso que se optó por administrar todo desde la app móvil.

La aplicación puede mejorar en cuestión de manejo, usabilidad, diseño entre otras cosas. Algunos de estos aspectos a mejorar son la usabilidad en modo orientación horizontal. Se implementó correctamente el uso en modo vertical, y para un 100% de usabilidad se puede implementar el otro modo de uso. También en cuestión al diseño se pueden implementar los iconos creados por el Comedor Universitario, los estilos y diseño en general.

Una de las cuestiones por las que no se implementó su diseño fue el hecho del tiempo en que se tardaron para aceptar el proyecto para donar comidas de nuevo, ya que por ciertos motivos ya no se estaba implementando.

Uno de los aportes a futuro como ya se mencionó es el de donaciones aleatorias. No todos los alumnos inscriben las 3 comidas del día como ya se sabe, pero esto no quiere decir que no todos tengan los recursos para pagarlas, si no que algunos no las inscriben para comprar alimentos a fuera. La problemática que se vio en este asunto es que si algún alumno no sabe a quién donar la comida que no va a consumir y la dona aleatoriamente, algunos de los alumnos que no inscribieron comidas las puede tomar, ya que no hay ningún parámetro existente ni establecido para esta situación. Como aporte a futuro se pretende hacer una depuración de matrículas, ya sea con una encuesta, por tipo de plan (industrial, 4 horas, 8 horas, etc.), o por algún método, para tener en cuenta quienes son los alumnos que realmente no tienen para pagar las 3 comidas y que solamente sus matrículas sean las que pueden recibir esas donaciones aleatorias.

VI. CONCLUSIÓN

Los alimentos son aprovechados por los alumnos que no tienen comidas, así como los que tienen se les permite ver el menú del día optando por consumir o donar los alimentos. Mediante la app móvil los estudiantes también pueden tener un control acerca de cuantas calorías necesitan según sus actividades físicas para poder mantener su peso ideal.

Los estudiantes de la Universidad pueden estar seguros de donar sus comidas del día, ya que el sistema de la aplicación cuenta con seguridad para que no cualquiera pueda hacer uso de su matrícula y su contraseña.

Apéndices

APÉNDICE A.- *Código mostrando el método onCreate();*

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.login);

    //permite conexiones con servidores e/s (cada que haga
conexion)
    StrictMode.ThreadPolicy policy = new
StrictMode.ThreadPolicy.Builder().permitAll().build();
    StrictMode.setThreadPolicy(policy);


    userText=(EditText) findViewById(R.id.user);
    passwordText=(EditText)
findViewById(R.id.password);

    loginButton=(Button) findViewById(R.id.Login);
    status = (TextView)findViewById(R.id.textView6);
    sharedpreferences =
getSharedPreferences(MyPREFERENCES,
Context.MODE_PRIVATE);

    /*loginButton.setOnClickListener(new
View.OnClickListener() {
        @Override
        public void onClick(View v) {
            Intent intent = new Intent(getBaseContext(),
MenuPrincipal.class);
            startActivityForResult(intent, 0);
            //finish();
        }
    });*/
    SharedPreferences prefs =
PreferenceManager.getDefaultSharedPreferences(getBase
Context());
    SharedPreferences.Editor editor = prefs.edit();

    editor.putString("urlKey", urlValue);
    editor.commit();

    SharedPreferences SP =
PreferenceManager.getDefaultSharedPreferences(getBase
Context());
    String usuario = SP.getString("userKey", "");
    if(!usuario.isEmpty()){
        //System.out.println("condicion "+usuario);
       // Intent intent = new Intent(getBaseContext(),
MenuPrincipal.class);
        Intent intent = new Intent(this, MenuPrincipal.class);
        //startActivityForResult(intent, 0);
        startActivity(intent);
        this.finish();
    }

}
```

APÉNDICE B.- Código para llamar a la clase que permite hacer la conexión la base de datos. *loginPost();*

```
public void loginPost(View view){
    String username = userText.getText().toString();
    String password = passwordText.getText().toString();

    SharedPreferences.Editor editor =
sharedpreferences.edit();

    editor.putString(User, username);
    editor.putString(Pass,password);
    editor.commit();

    new SigninActivity(this,
status).execute(username,password);
}
```

APÉNDICE C.- Código para hacer conexión a la base de datos

```
import android.content.Context;
import android.content.Intent;
import android.os.AsyncTask;
import android.widget.TextView;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;

public class SigninActivity extends
AsyncTask<String,Void,String>{
    private TextView statusField;
    private Context context;
    private String Matricula="";


    public SigninActivity(Context context, TextView
statusField) {
        this.context = context;
        this.statusField = statusField;
    }

    protected void onPreExecute(){

    }
```

```
    @Override
    protected String doInBackground(String [] arg0) {

            try{
                String username = arg0[0];
                String password = arg0[1];

                String
link="http://192.168.1.137:8080/Saum/index.php";
                String data  =
URLEncoder.encode("Matricula", "UTF-8") + "=" +
URLEncoder.encode(username, "UTF-8");
                Matricula =   URLEncoder.encode(username,
"UTF-8");
                data += "&" +
URLEncoder.encode("Contrasena", "UTF-8") + "=" +
URLEncoder.encode(password, "UTF-8");

                URL url = new URL(link);
                URLConnection conn = url.openConnection();

                conn.setDoOutput(true);

                OutputStreamWriter wr = new
OutputStreamWriter(conn.getOutputStream());

                wr.write( data );
                wr.flush();

                BufferedReader reader = new
BufferedReader(new
InputStreamReader(conn.getInputStream()));

                StringBuilder sb = new StringBuilder();
                String line = "";

                // Read Server Response
                while((line = reader.readLine()) != null)
                {
                    sb.append(line);
                    break;
                }
                return sb.toString();
            }
            catch(Exception e){
                return new String("Exception: " +
e.getMessage());
            }
        }



    @Override
    protected void onPostExecute(String result){
        //System.out.println("resultado "+result);
        //System.out.println(Matricula);

        //Condición para saber si es correcta la matrícula o
no
        if(result.equals(Matricula) && !result.equals("")){
            Intent inten = new Intent(this.context,
MenuPrincipal.class);
            context.startActivity(inten);

        }else {
            this.statusField.setText("El usuario o contraseña
son incorrectos");
        }

    }
}
```

APÉNDICE D.- Código para mostrar el menú del día de acuerdo a la fecha actual

```
<?php

$con=mysqli_connect("127.0.0.1","root","","saum");

if (mysqli_connect_errno($con))

{

  echo "Failed to connect to MySQL: " .
mysqli_connect_error();

}

date_default_timezone_set('Mexico/General');

$fecha = date("Y-m-d");

$sql = "SELECT * FROM menudia WHERE
fecha>='$fecha' ORDER by fecha ASC";



// Check if there are results

if ($result = mysqli_query($con, $sql))

{

        // If so, then create a results array and a temporary
one

        // to hold the data

        $resultArray = array();

        $tempArray = array();

        // Loop through each row in the result set

        while($row = $result->fetch_object())

        {

                // Add each row into our results array

                $tempArray = $row;
```

```
            array_push($resultArray, $tempArray);
        }
        // Finally, encode the array to JSON and output the
results
        echo json_encode($resultArray);
}
mysqli_close($con);
?>
```

APÉNDICE E.- Menú del día

APÉNDICE F.- Pruebas de caja negra/Plan de prueba

Prueba ID	Descripción	Resultados esperados	Resultados actuales
Campos de Login (Usuario)	Se ingresaron diferentes caracteres, comillas simples y dobles, matriculas que no existen en la base de datos, inyección de SQL.	Las variables se han configurado para no recibir ese tipo de valores, así que no se pueden enviar los datos con estos caracteres.	La pantalla no muestra ningún warning, sin embargo no se genera ninguna acción.
Campos de Login (Contraseña)	Se ingresaron diferentes caracteres, comillas simples y dobles, contraseñas inválidas e inyección de SQL.	Ya que se realizaron las mismas pruebas se esperaron los mismos resultados, de no enviar datos con caracteres incorrectos o no permitidos.	La pantalla no muestra ningún warning, sin embargo no se genera ninguna acción.
Donación de comidas	Se ingresaron matrículas inexistentes, matrículas a las cuales ya se les había donado alguna comida del día como prueba. También se intentó donar alguna comida la cual no tenía inscrita el estudiante.	De acuerdo a la prueba, ya sea que no exista la matricula o no tenga comidas inscritas no permite hacer la donación mostrando un mensaje el error.	La aplicación no permite donar si la matricula ya tiene las comidas donadas o si no tiene comidas para donar, mostrando un mensaje de error en un diálogo.
Calorías	Si se ingresan letras o caracteres diferentes a números en los campos para el cálculo de calorías, el resultado sería erróneo, ya que se realizan operaciones matemáticas.	Los usuarios no deben poder insertar textos en los campos para el cálculo de calorías.	Las pruebas de inserción de texto fueron exitosas, ya que al usuario no se le permite ingresar letras, solamente números.
Datos guardados en memoria del dispositivo	Al momento de iniciar sesión se guarda el usuario en memoria para hacer todas las acciones necesarias, y al momento de cerrar sesión se borra del dispositivo para poder crear un Nuevo usuario en sesión al momento de volver a iniciarla.	La variable del usuario se debe imprimir en consola si se inició sesión, y si se Cierra, la variable se debe imprimir sin ningún valor.	Se imprimen las variables en consola. Se imprime la variable de la matrícula y muestra al usuario en sesión, Al momento de cerrarla, se imprime la variable y no se muestra nada en la consola como resultado.

Referencias

1. D. Jerome, “Android”, Mc Graw Hill, New York, USA, 2008.
2. M. Nosrati, R. Karimi, and H. A. Hasanvand, “Mobile computing: Principles, Devices and Operating Systems”, *World Applied Programming*, vol. 2, Issue 7, 2012.
3. D. Howcroft, and B. Bergyall, “Mobile Aplications Development on Apple and Google Platforms”, Manchester Bussiness School, Atlanta, GA, USA, 2011.
4. M. Báez, A. Borrego, J. Cordero, L. Cruz, M. González, F. Hernández, D. Palomero, J. Rodríguez, D. Sanz, M. Saucedo, P. Torralbo, A. Zapata, “Introducción a Android”, E.M.E., Universidad Complutense de Madrid.
5. Gabheran. (2012). Introducción - ¿Qué es Android? [Online]. Disponible: http://histinf.blogs.upv.es/2012/12/14/android/
6. J. Yang. (2015, june 22). The Real Software Security Problem Is Us. [Online]. Disponible: http://www.technologyreview.com/view/538636/the-real-software-security-problem-is-us/
7. Android: “*Security” https://source.android.com/security/.* Último acceso: Enero 2016.
8. C. M. García, “Controles y seguridad bajo entorno Android”, M.S. Tesis, Depto. Informática, Univ. Carlos III Madrid, Leganés, Madrid, 2013.
9. “Mobile Web Apps vs. Mobile Native Apps: How to Make the Right Choice”, Lionbridge, s/a.
10. IBM, “Native, Web or hybrid mobile—app development”, Softwar Group, USA, 2012.
11. M. Power, “Mobile Web apps”, JISC cetis, 2011.
12. K. Grant, C. Haseman, “Beginning Android Programming”, Peachpit Press, USA, 2014.
13. M. Wolfson, “Android Developer Tools Essentials”, O´Reilly Media, 2013.
14. W. Lee, “Beginning Android 4 Application Development”, John Wiley & Sons, Inc, Canada, 2012.
15. V. Matos, *“Android Environment Emulator”,*Cleveland State University, 2009.
16. J. A. Tudela, “Desarrollo de aplicaciones para dispositivos móviles sobre la plataforma Android Google”, M.S. Tesis, Depto. Informática, Univ. Carlos III Madrid, Leganés, Madrid, 2009.
17. Java World, artículo “The fragmentario effect”. http://www.javaworld.com/javaworld/jw-05-2004/jw-0524-fragment.html?page=1. Último acceso: Enero 2016.
18. Lifehack, artículo “25 apps College Students Shouldn´t Live Without”. http://www.lifehack.org/articles/technology/25-apps-college-students-shouldnt-live-without.html Último acceso: Enero 2016.
19. A. López, “Telefonía Móvil Transmisión y redes de Datos”, 2010.
20. El tiempo, artículo “dos de cada diez empresas ya tienen apps propias para sus negocios”. http://www.eltiempo.com/economia/empresas/empresas-con-aplicaciones-moviles/14453536. Último acceso: Enero 2016.
21. J. Arthur Harris and Francis G. Benedict, “A Biometric Study of Human Basal Metabolism”. *Proceedings of the National Academy of Sciences.* Vol. 4, No. 12, pp. 370-373, December 1918.
22. A.M. Roza and H.M. Shizgal, “The Harris Benedict equation reevaluated”. American Journal of Clinical Nutrition. Vol. 40, No. 1, 168-182, July 1984.
23. 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.
24. D. Guerra, “Saum app”, M.S. Tesis, Univ. De Montemorelos, Nvo. León, México, 2015.
25. Android Developers: “*The AndroidManifest.xml File”.* http://developer.android.com/guide/topics/manifest/manifest-intro.html. Ultimo acceso: Noviembre 2015.