UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Sistema de apoyo gerencial y control de comandas para el restaurante Antic

Trabajo de graduación en modalidad de Megaproyecto presentado por:
Claudia María Grajeda Díaz,
Manuel Alejandro Rojas Flores,
Gustavo Andrés Sánchez Cardona y
Jennifer Pamela Valdez Cabrera
para optar al grado académico de Licenciados en Mecatrónica

Guatemala
2014

Sistema de apoyo gerencial y control de comandas para el restaurante

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Sistema de apoyo gerencial y control de comandas para el restaurante Antic

Trabajo de graduación en modalidad de Megaproyecto presentado por:
Claudia María Grajeda Díaz,
Manuel Alejandro Rojas Flores,
Gustavo Andrés Sánchez Cardona y
Jennifer Pamela Valdez Cabrera
para optar al grado académico de Licenciados en Mecatrónica

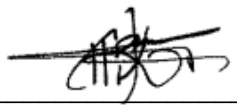
Guatemala
2014

Vo. Bo. :

(f) 

(Kurt Emmanuel Kellner Juarez)

Tribunal examinador:

(f) 

(MSc. Carlos Alberto Esquit Hernandez)

(f) 

(Roberto Enrique Saravia Fernandez)

(f) 

(Kurt Emmanuel Kellner Juarez)

Fecha de aprobación: Guatemala, 25 de noviembre de 2014

ÍNDICE

LISTA DE CUADROS

Página

LISTA DE FIGURAS

Página

RESUMEN

La forma en que los sistemas se comunican ha cambiado la manera en la que los seres humanos interactúan con su entorno, haciendo cada vez más sencilla la conexión entre diferentes dispositivos. Los desarrolladores de estos se han enfocado en simplificar la comunicación permitiendo tener una unidad centralizada que interactúa con todos sus periféricos de forma transparente.

Este megaproyecto busca automatizar los procesos de toma de órdenes y organización de restaurantes para que le permita al negocio ser más eficiente, así como obtener estadísticas de las ventas y modificar su menú con promociones o nuevos productos de manera sencilla. Para alcanzar esto se realizaron 5 módulos: una aplicación de meseros para el ingreso de órdenes, una aplicación para el despliegue de órdenes en cocina, un dispositivo de notificaciones para meseros, una página web para el uso de los gerentes y una aplicación de clientes.

Trabajando en conjunto los módulos forman un sistema que le permite al restaurante apoyarse en la tecnología para la toma de órdenes. Al mismo tiempo se recopila información importante sobre las actividades del restaurante y se obtiene realimentación de los clientes. Se obtuvo el apoyo del restaurante Antic para poder basar el proyecto en sus necesidades y sus productos. Se tuvieron reuniones con los gerentes para especificar los términos de la colaboración.

I. INTRODUCCIÓN

Este trabajo de megaproyecto tiene como objetivo implementar un sistema de apoyo gerencial y de control de las comandas del restaurante Antic. El control de las comandas se basa en que los meseros del restaurante cuentan con una *tablet* por medio de la cual realizan el pedido de la orden de las mesas. Al finalizar de pedir la orden se envía la comanda de manera automática a bar o a cocina según lo pedido por los clientes; el mesero ya no debe ir a notificar de la comanda ya que se envía de manera inalámbrica. En el bar o en cocina se muestran las órdenes pendientes junto con las modificaciones de cada plato. Al tenerla lista solamente se debe presionar un botón para notificar a los meseros. Ellos cuentan con un dispositivo de notificaciones inalámbrico tipo brazalete que, cuando una orden ya está lista, vibra y muestra en una matriz de LEDs si debe ir a cocina o a bar. De esta manera el mesero no se dirige al bar o la cocina de manera innecesaria y la mayor parte de su tiempo se encuentra atendiendo a los clientes. El apoyo gerencial se basa en la implementación de una página web en la cual los gerentes del restaurante puedan modificar el menú ya sea cambiando o agregando un plato o una categoría y que las aplicaciones de los meseros y de los clientes se actualizan. La página web también presenta a los gerentes las estadísticas que deseen observar ya que los datos al finalizar la orden son almacenados en la base de datos. Con el objetivo de brindar un mejor servicio a los clientes de Antic se desarrolló una aplicación con sistema operativo Android donde se puede observar el menú y genera una vía de comunicación fácil con el restaurante ya que presenta la opción de llamar, enviar correo, etc.

Para alcanzar el objetivo el proyecto se dividió en cuatro módulos: el módulo del diseño y creación de un dispositivo de notificaciones inalámbrico tipo brazalete para los meseros del restaurante, el módulo de implementación de una página web y una aplicación de bar o cocina que recibe las ordenes que deben preparar; el módulo de implementación de la aplicación de los meseros y la aplicación para los clientes, y el módulo de interconexión de módulos del sistema.

El desarrollo de las aplicaciones se apoyó en el material del curso en línea de la universidad de Maryland llamado “Programación de aplicaciones móviles para sistemas portátiles Android”, de Coursera, impartido por Adam Porter durante abril y mayo de 2014. Además se utilizó material adicional obtenido de diversos tutoriales para generar códigos sencillos. El desarrollo de las aplicaciones se restringió a la versión 22.6.4 del paquete de *software* para el desarrollo de Android (SDK) y además que se tuviera la versión de la interfaz de programación de aplicaciones (API) 18 por razones técnicas.

Uno de los resultados del proyecto consiste en una aplicación para la generación y envío automático de comandas y el envío de información requerida por el restaurante Antic a la base de datos para ser almacenada. También se desarrolló una aplicación para los clientes donde se puede ver el menú actualizado

del restaurante. Esta permite realizar llamadas al restaurante, enviar correos electrónicos o entrar a su Facebook y Twitter. Además los clientes pueden realizar encuestas y enviar comentarios al restaurante. Otro resultado fue una aplicación para bar y cocina que despliega las órdenes pendientes con las modificaciones pedidas por los clientes y en la cual se presiona un botón para que el mesero llegue lo antes posible a traer la orden. Se realizó también una página web en donde el gerente puede realizar cambios al menú y ver las estadísticas que desee. Adicionalmente puede ver los resultados a las encuestas realizadas por los clientes y los comentarios de los mismos. Se realizó un dispositivo inalámbrico tipo brazalete para notificar a los meseros cuando una orden ya está lista de bar o cocina. Asimismo obtuvo una base de datos y una librería de Java que permite comunicarse con la base de datos por los demás módulos.

II. OBJETIVOS

A. GENERAL

1. Crear y diseñar un sistema de apoyo gerencial y control de comandas para el restaurante Antic.

B. ESPECÍFICOS

1. Diseñar y desarrollar una página web para administración del menú y control de personal de servicio del restaurante Antic y una aplicación en Android para el despliegue de órdenes en cocina.

2. Diseñar y desarrollar una aplicación para *tablets* con sistema operativo Android para generación de comandas, y una aplicación para teléfonos inteligentes con sistema operativo Android que permita a los clientes obtener información de los servicios del restaurante Antic.

3. Diseñar y construir un dispositivo tipo brazalete que reciba notificaciones de texto por parte de un dispositivo Android vía Bluetooth, indicándole al usuario el lugar al cual dirigirse.

4. Interfazar módulos independientes en un sistema de automatización de restaurantes utilizando sistemas de bajo costo y consumo energético.

III. JUSTIFICACIÓN

Con el paso del tiempo la comunicación entre dispositivos electrónicos se ha hecho más estrecha y frecuente. Al contar con dispositivos conectados a internet se tiene la capacidad de tener acceso a nuestra información en todo momento. Teniendo esto en cuenta han aparecido conceptos como “El internet de las cosas” y ciencias como la domótica que presentan computadoras comunicándose con dispositivos inesperados como lavadoras, microondas o incluso ropa para poder hacer algunas tareas más simples o realizarlas enteramente para el ser humano.

Los restaurantes son el punto de reunión de muchas personas para hacer negocios, compartir con amigos o familiares. Por ello esta experiencia debe ser de agrado para los clientes al igual que para los meseros. Podemos mejorar esta experiencia haciendo más eficiente la comunicación entre el mesero y la cocina y viceversa, mejorando así tiempos de atención. También el poder separar la orden en diferentes cuentas de forma en que ni los meseros o clientes deben realizar operaciones matemáticas, obtener realimentación de los clientes, generar estadísticas instantáneamente, obtener información de los platos que más se consumen y que la recepcionista pueda observar desde la entrada la cantidad y el lugar de las mesas vacías y limpias.

Este megaproyecto consiste en la identificación de problemas que se dan en un restaurante solucionándolo con aplicaciones y dispositivos fáciles de usar. Estas serán creadas por un grupo de estudiantes de Ingeniería Mecatrónica de la Universidad del Valle, adaptándolas a las necesidades y deseos del restaurante.

IV. MARCO TEÓRICO

A. PÁGINA WEB

La Web (o World Wide Web) es un sistema de documentos de hipertextos interconectados contenidos en Internet. (Wikipedians) En 1989, Tim Bernes-Lee empezó a escribir una propuesta para la web, buscado resolver la necesidad de compartir información alrededor del mundo. Como resultado, escribió el primer navegador Web, el servidor y la página Web y las primeras especificaciones de URL (Uniform Resource Locator), HTTP (HyperText Transfer Protocol) y HTML (HypertText Markup Language) para 1990. Con un buscador web se pueden observar páginas web que contienen textos, imágenes, videos, y otros objetos multimedia por medio de hipervínculos. Muchos estándares y especificaciones técnicas que han sido definidas por el Consorcio "World Wide Web" (W3C por sus siglas en inglés), que es una comunidad internacional donde se desarrollan estándares opcionales para la Web. La misión del Consorcio es llevar a la Web a su mayor potencial, desarrollando protocolos y pautas que aseguren su crecimiento. (About W3C, 2014) Los estándares también están definidos por la organización "Internet Engineering Task Force (IETF) y otras. (HELP AND FAQ, 2014)

Los estándares recomiendan los leguajes HTML y XHTML para definir la estructura y la interpretación de documentos con hipertextos. Así, se propone que las hojas de estilo estén escritas en CSS (Cascading style sheets), cuya función es definir cómo desplegar los elementos HTML. El lenguaje más común para el código que no necesita ser pre-procesada o compilada antes de correr el programa es JavaScript, desarrollado por Ecma, que puede modificar todos los elementos HTML. Con la implementación de este lenguaje se puede hacer que la página sea más dinámica, junto con la ayuda de AJAX (Asynchronous JavaScript and XML). (W3C, JavaScript Web APIS, 2014) Existe un subconjunto de notaciones literarias de objetos de JavaScript, llamado JSON, éste es usado como formato para la serialización de datos. (Berjon, 2014) Para guardar datos en bases de datos, se puede utilizar un API, denominado mySQL, el cual es una variante de SQL (Structured Query Language). (W3C, Web SQL Database, 2010) PHP es un procesador de hipertexto, ejecutado en el servidor. Un archivo PHP puede contener código en HTML, CSS, JavaScript y PHP (del cual se hablará posteriormente). (w3Schools)

Existen diferentes APIs de JavaScript que se pueden incluir en la página Web que se está desarrollando, dado que los autores han otorgado una licencia que permiten distribuir y modificar, entre otras cosas, a cualquier persona. Algunas de estas son Choosen.js que permite darle un estilo diferente a la opción "select" y "multiple select", Fullpage.js que da formato a el menú de la página y como desplazarse entre las opciones y Chart.js que es una librería que da formato a diferentes gráficos, de barras, líneas, circular ente otros.

B. APLICACIONES MÓVILES

En los últimos años gracias al avance que han tenido los teléfonos inteligentes, mejor conocidos como *smartphones*, se han vuelto muy populares las aplicaciones móviles (app). Una app es una aplicación de software que se instala en dispositivos móviles o *tablets* con el objetivo de facilitar a los usuarios diferentes tipos de tareas, ya sean éstas de carácter profesional, comercial o de entretenimiento, entre otros. La palabra app viene de la palabra en inglés *application*, y a partir del 2008 se empezó a utilizar como un término para aplicaciones móviles debido a que en dicho año fue el lanzamiento del App Store de Apple, el primer SDK para Android y la inauguración del Android Market (Vilela, 2012).

Las apps son muy utilizadas hoy en día, según la empresa de investigación *ABI Research* en el 2010 hubo casi 8000 millones de descargas de apps en todo el mundo, y debido a su éxito varias empresas han implementado apps para que los usuarios los conozcan o para facilitar sus ventas, etc.; sin embargo la creación de una app puede ser complicada y costosa ya que existen varios tipos de sistemas operativos entre los cuales se puede mencionar Android, Mac iOS, Windows Mobile, Symbian y RIM; y una app realizada para un sistema operativo no funcionará en otro, así que hay que programarlo para varios sistemas si se desea llegar a más usuario (Vilela, 2012) (Lanacion, 2011).

La primera app que se reconoció fue para un ordenador portátil de Psion que utilizaba el sistema operativo EPOC; este evento ocurrió en los años 90s en donde las máquinas que desarrolló EPOC permitían aplicaciones como procesador de textos, base de datos, hojas de cálculo y el diario para los usuarios. Luego hubo modelos más nuevos con sistemas operativos de 32 bits y permitían a los usuarios agregar otras aplicaciones a través de paquetes descargables; además EPOC desarrollaba sus aplicaciones en *Open Programming Language* (OPM), y esto permitía a sus usuarios a crear sus propias aplicaciones. Estos avances dieron origen al sistema operativo Symbian el cual fue el sistema utilizado anteriormente por Psion, Ericsson, Motorola y Nokia; incluso hubo ciertos dispositivos Samsung y LG que utilizaron dicho sistema operativo.

Posteriormente se creó las aplicaciones con conexión a exploradores de internet utilizando un protocolo conocido como *Wireless Application Protocol* (WAP), dichas aplicaciones estaban en lenguaje *Wireless Markup Language* (WML) el cual estuvo basado en el lenguaje XML[1]. También Java Micro Edition proporcionó una plataforma para el desarrollo de apps, y se volvió tan popular que desarrolló varias normas para su uso en dispositivos móviles, una de las cuales es MIDP que fue diseñada para teléfonos móviles e incluye una interfaz gráfica de usuario, una API para almacenamiento de datos y una API para juegos en dos dimensiones. Sin embargo hubo diferentes plataformas, y cada empresa utilizando su propia API

[1] De las siglas en inglés de *eXtensible Markup Languaje*, que es un lenguaje de marcas desarrollado para almacenar datos en forma legible.

significó una gran variedad de posibilidades de implementación pero poca posibilidad de mercado para aplicaciones así que se llegó a crear el software libre. En el 2012 Nokia cambió su sistema operativo a Windows Phone, y luego las demás empresas fueron cambiando al sistema estándar Android el cual cuenta con una plataforma única haciendo posible la implementación de una aplicación para cualquier dispositivo con sistema operativo Android. (Bates, A History of Mobile Application Development, 2014)

C. SISTEMA OPERATIVO ANDROID

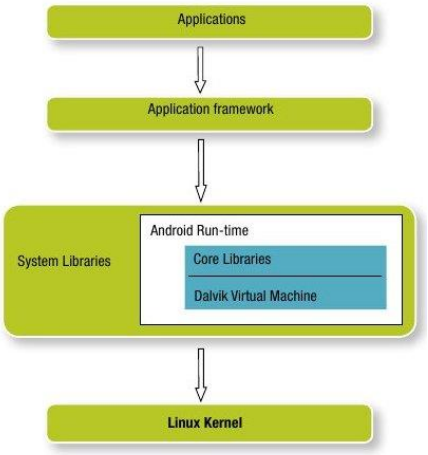
1. Definiendo qué es Android.

Android es un sistema operativo diseñado para dispositivos móviles que está basado en Linux, el cual es un núcleo de sistema operativo libre, gratuito y compatible con múltiples plataformas. Android permite que los desarrolladores programen empleando una variación de Java llamada Dalvik, y además proporciona las interfaces requeridas para desarrollar aplicaciones que accedan a funciones de los dispositivos móviles como por ejemplo llamadas, uso del GPS, etc. (Sacristrán & Fernández, 2013)

Las aplicaciones para Android se programan en el lenguaje de programación Java, y se ejecutan en un *framework*[2] especializado en aplicaciones orientadas a objetos en una máquina virtual Dalvik que compila en el tiempo de ejecución de la aplicación. Todas las apps están comprimidas en un archivo de tipo APK[3] que permite su instalación desde cualquier explorador de archivos. Además las bibliotecas están en el lenguaje de programación C e incluyen una interfaz gráfica, para dichos aspectos gráficos utiliza una interfaz para el renderizado de gráficos en dos dimensiones y en tres dimensiones. También contiene un motor DE renderizado de páginas Web llamado WebKit; además contiene bibliotecas estándar para su desarrollo. (Sacristrán & Fernández, 2013) (Roldayan, 2013) .

[2] Se define como una estructura de soporte definido, es la organización y la base del desarrollo del *software*.
[3] Ver definición más detallada en la sección de glosario

Figura 1. Base del funcionamiento de Android



2. Historia de Android. La empresa Android Inc. fue fundada en octubre de 2003 por Andy Rubin, Rich Miner, Nick Sears y Chris White, y tenía como objetivo el desarrollo de aplicaciones para dispositivos móviles. En junio de 2005 Android Inc. fue comprada por Google, y uno de los cofundadores de la empresa, Andy Rubin, fue el director de la división de plataformas móviles de Google. El 5 de noviembre de 2007 se anunció la creación de *Open Handset Alliance* (OHA); una organización encargada de la difusión de la plataforma móvil Android que actualmente contiene a 84 empresas. Android fue el primer sistema operativo abierto para móviles, es decir que su funcionamiento no está vinculado a una marca o a un dispositivo específico. El 12 de noviembre de 2007 Google lanzó un kit de desarrollo del software SDK (por sus siglas en inglés de *Software Development Kit*) que permitió a los desarrolladores a generar sus propias aplicaciones en dicha plataforma. (Roldayan, 2013) (García, Hernández, & Berlingen, 2010) (Pérez, 2012)

El 21 de octubre de 2008 Google y OHA publicaron el proyecto *OpenSource*[4] conocido como Android cuyo desarrollo fue en mayor parte por Google, y que se distribuye bajo la licencia Apache 2.0. El mismo año se lanzó *Android Market*, que actualmente se conoce como *Play Store*; la cual es una plataforma que Android utiliza para distribuir sus aplicaciones ya sean gratuitas o pagadas. (Pécheron, 2012) (Pérez, 2012) (Tomás, 2013)

Actualmente Android es uno de los sistemas operativos más utilizados. Para el año 2012 la cifra de terminales activas era de 250 millones con 700.000 activaciones diarias totalizando en 11 mil millones de apps descargadas hasta entonces. (Pérez, 2012) (Tomás, 2013)

[4] Es el concepto del software desarrollado y distribuido libremente.

3. Versiones de Android. En septiembre de 2008 surgió la primera versión de la plataforma Android, la versión 1.0; luego en febrero de 2009 se lanzó la segunda versión 1.1; esta última versión fue la que primero se instaló en un *smartphone* Android, y fue comercializada en España en marzo del mismo año. En abril de 2009 salió la versión 1.5 que incluyó acelerómetro, grabación de video y App Widgets[5]. Posteriormente en septiembre se lanzó la versión 1.6 que soportó distintas resoluciones y tamaños de pantalla. En octubre del mismo año salió la versión 2.0 que tenía soporte para HTML5 y Exchange[6]. En la versión 2.2 se incluyó la funcionalidad de acceso WiFi. En el 2010 se lanzó la versión 2.3 que incluyó NFC[7]. Con cada versión Android ha ido mejorando la eficiencia de su sistema operativo y ha ido agregando funcionalidades para facilitar las tareas de los usuarios. Las versiones de Android se pueden observar en la Cuadro 1. Versiones de Android. (Pécheron, 2012)

Cuadro 1. Versiones de Android

Año	*Plataforma*
2008	Android 1.0 – Apple Pie
2009	Android 1.1 – Banana Bread
	Android 1.5 – Cupcake
	Android 1.6 – Donut
	Android 2.0/2.1 – Éclair
2010	Android 2.2 – Froyo
	Android 2.3 – Gingerbread
2011	Android 3.0/3.1/3.2 – Honeycomb
	Android 4.0 – Ice Cream Sandwich
2012	Android 4.1/4.2/4.3 – Jelly Bean
2013	Android 4.4 – KitKat

Los nombres de las versiones de Android están en orden alfabético (Roldayan, 2013)

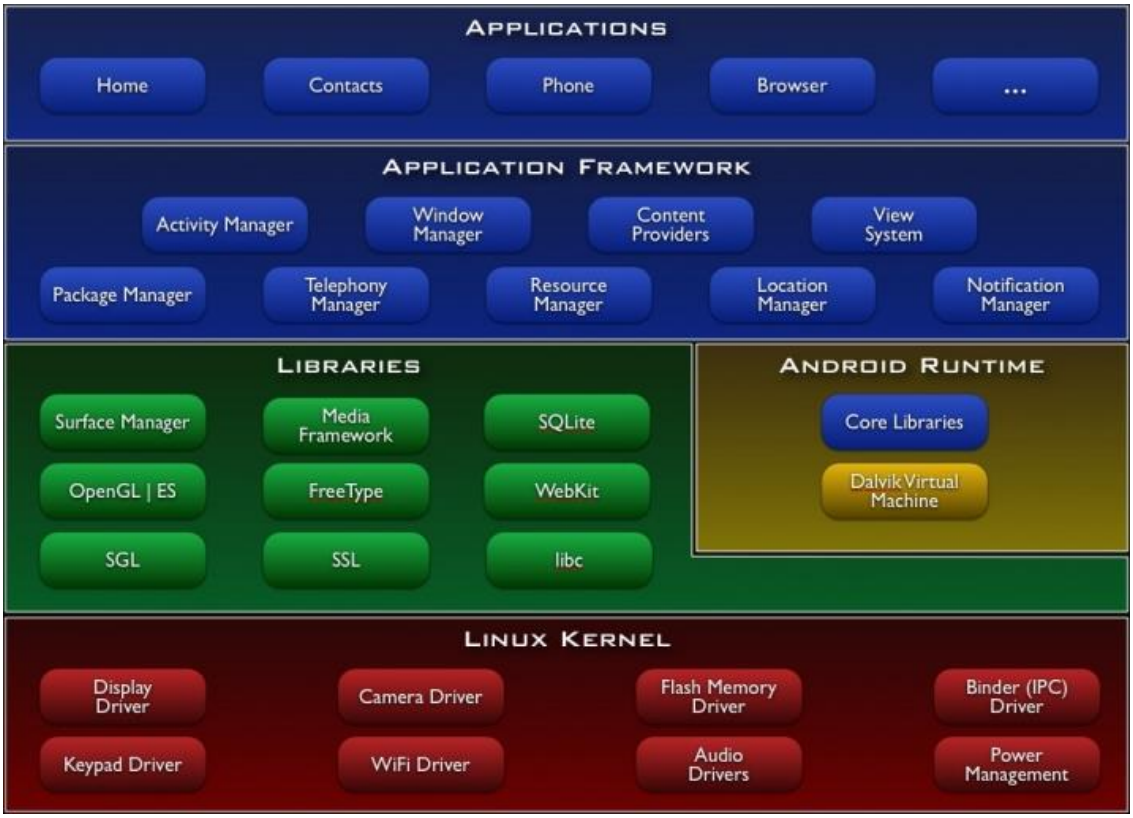
4. Arquitectura. La forma en la que se encuentra estructurado un sistema operativo se conoce como su arquitectura, y en el caso de Android se organiza en varias capas para facilitar la programación de aplicaciones. Este tipo de arquitectura de varias capas permite a los desarrolladores hacer uso de los componentes de *hardware* de los dispositivos sin necesidad de programar a bajo nivel[8] sino que solamente

[5] Ver definición de App Widget es la sección de glosario.
[6] Microsoft Exhange es el correo electrónico implementado para empresas.
[7] Siglas de Near Field Communication, que es un protocolo de comunicación inalámbrico.
[8] La programación en bajo nivel es el que

se debe hacer uso de las librerías previamente programadas, pero si un desarrollador desea modificar una capa inferior programando en bajo nivel puede realizarlo. (Vico, 2011)

El tipo de arquitectura que utiliza el sistema operativo Android se conoce como pila ya que cada una de las capas utiliza funciones o elementos de la capa inferior para su funcionamiento. A continuación se muestra una figura que permite visualizar la organización de las capas de Android, y se explica cada capa de la arquitectura. (Vico, 2011)

Figura 2. Arquitectura del sistema operativo Android (Vico, 2011)



a. Kernel de Linux (Linux Kernel). Como se mencionó anteriormente Android está basado en el kernel de Linux versión 2.6, sin embargo el kernel fue adaptado para ser compatible con el *hardware* en el que se iba a utilizar, en este caso con los dispositivos móviles. Esta capa es la inferior en la arquitectura, y eso es debido a que el desarrollador no accede directamente a ésta sino que lo hace a través de bibliotecas de capas superiores. Esta capa tiene como objetivo realizar una conexión entre el *hardware* y el resto de capas, y así los desarrolladores no deben preocuparse por las características de cada teléfono. Para cada componente del dispositivo móvil existe un controlador, mejor conocido como *driver*, en el kernel que permite utilizar dicho componente desde el software; así que el kernel se encarga de proveer

acceso y uso de los componentes físicos de manera independiente al dispositivo. Por ejemplo, se permite el uso de acelerómetro en cualquier teléfono que tenga uno, por medio de las bibliotecas de Android. Además el kernel también se encarga del manejo de los recursos del dispositivo móvil como por ejemplo la energía, memoria, etc. y también del sistema operativo en sí como por ejemplo elementos de comunicación. (Vico, 2011)

b. Librerías (*Libraries*). La siguiente capa arriba del kernel son las librerías que permiten acceder al *hardware* de los dispositivos por medio de *software* utilizando los *drivers* del kernel. Las librerías están escritas en el lenguaje de programación C o en C++, y son desarrolladas e instaladas por los fabricantes de los dispositivos móviles antes de venderlo. (Vico, 2011)

c. Entorno de ejecución (*Android Runtime*). El entorno de ejecución se encuentra al mismo nivel que las librerías debido a que también está formado a base de librerías que son las *Core Libreries*[9]. Todas las aplicaciones de Android son ejecutadas en una máquina virtual llamada Dalvik, luego de ser codificadas en Java. Esto genera la ventaja que los programas se compilen solamente una vez, y luego ya pueden ejecutarse en cualquier dispositivo con sistema operativo Android que contenga la versión mínima requerida por la aplicación. (Vico, 2011)

Es importante mencionar que la máquina virtual Dalvik fue creada para ejecutar archivos en formato .dex (*Dalvik Excecutable*) que son generados por el SDK de Android, y no puede ejecutar otra aplicación de Java ya que producen un archivo en *bytecode* que es apto para ejecución en las máquinas virtuales de Java (*Java Virtual Machines*). (Vico, 2011)

d. *Framework* de aplicaciones (*Application Framework*). Esta capa es la encargada de brindar facilidades a los desarrolladores, está diseñada con el objetivo de reutilizar los componentes del sistema. Esta capa se compone por librerías que acceden a los recursos de las capas inferiores por medio de la máquina virtual Dalvik. En esta capa se pueden mencionar doce librerías para el uso en las aplicaciones: Activity Manager, Windows Manager, Content Provider, Views, Notification Manager, Package Manager, Telephony Manager, Resource Manager, Location Manager, Cámara y Multimedia. Con todas las librerías mencionadas los desarrolladores pueden utilizar los componentes del dispositivo móvil. (Pérez, 2012) (Vico, 2011)

[9] Son las librerías utilizadas por varias porciones del JDK.

e. Aplicaciones (*Applications*). Esta es la capa superior que contiene todas las aplicaciones de los dispositivos incluyendo las de interfaz de usuario, las nativas[10] y las administrativas[11]. Se incluyen las aplicaciones instaladas por los fabricantes como el arranque del equipo o la barra de herramientas, y las aplicaciones que ha instalado o programado el usuario. (Vico, 2011)

D. HERRAMIENTAS DE DESARROLLO DE APLICACIONES DE ANDROID

El desarrollo de aplicaciones para dispositivos móviles debe considerar las restricciones que el dispositivo impone sobre la memoria disponible, el tamaño de la pantalla, e incluso el tamaño del teclado, entre otros. Debido a esto se motivó la creación de módulos y herramientas específicos para el desarrollo de apps, compatibles con IDEs populares, por sus siglas en inglés *Integrated Developmento Environment*; estos entornos se utilizan en los ordenadores para programar aplicaciones para dispositivos Android. (García, Hernández, & Berlingen, 2010)

El desarrollo de la mayoría de las aplicaciones para Android se realiza en lenguaje Java con el SDK[12] proporcionado por Google gratuitamente; sin embargo los desarrolladores pueden programar las aplicaciones en HTML CSS3, Javascript y en Android NDK (del cual se hablará posteriormente). (Roldayan, 2013)

1. IDEs. Como se mencionó anteriormente un IDE es un entorno de desarrollo integrado con la finalidad de programar aplicaciones para Android desde un ordenador. Actualmente existen varios IDEs siendo el más utilizado Eclipse, y el segundo NetBeans; en ambos se puede programar la aplicación deseada utilizando el lenguaje de programación Java. También existe el IDE llamado MIT App Inventor que permite a los desarrolladores que no poseen conocimiento en Java a producir una aplicación. Sin embargo este método no provee flexibilidad con el uso de los componentes del dispositivo, y no permite generar una interfaz gráfica personalizada ya que se genera automáticamente al agregar un componente. Además, en vez de Java se utiliza programación por bloques, que es una metodología de programación orientada a desarrolladores con poca experiencia técnica, aunque limitante comparada con la programación habitual. (García, Hernández, & Berlingen, 2010)

[10] Las aplicaciones nativas son implementadas en lenguaje nativo del propio terminal.
[11] Son las aplicaciones que se encargan de la administración del dispositivo.
[12] Siglas del inglés *Software Development Kit* que significa un kit de desarrollo de *software*.

Ya que Elipse es el más utilizado se recomienda utilizar dicho IDE ya que se encuentra más información y tutoriales en internet acerca de este entorno, además presenta varios paquetes o *plugins*[13]. Para la programación de aplicaciones de Android se utiliza el entorno Eclipse con el SDK de Android y el *plugin* de herramientas de desarrollo de Android (ADT por sus siglas en inglés *Android Development Tools*). (García, Hernández, & Berlingen, 2010)

Para obtener el entorno de Eclipse con el SDK y el ADT de Android se puede descargar directamente desde la página de desarrolladores de Android en http://developer.android.com/sdk/index.html. En la Figura 3 se puede observar la página para la descarga del IDE Eclipse.

Figura 3. Página para obtener Eclipse ADT con Android SDK para Windows

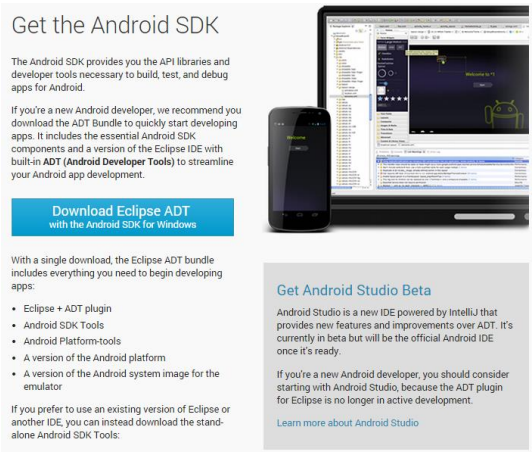


Imagen tomada de la página de desarrolladores de Android

Para utilizar el IDE se debe cumplir con los siguientes requisitos mínimos:

a. Sistema operativo
   1) Windows XP (32 bit), Vista (32 o 64 bits) o Windows 7 (32 o 64 bits)
   2) Mac OS X 10.5.8 o posterior
   3) Linux con la versión de Ubuntu

b. Herramientas de desarrollo
   1) JDK 6, excepto para Linux que requiere un JDK 4
   2) Apache Ant 1.8 o posterior

(Mednieks, 2012)

[13] Es un complemento que relaciona una aplicación con otra para generar una nueva función generalmente muy especializada.

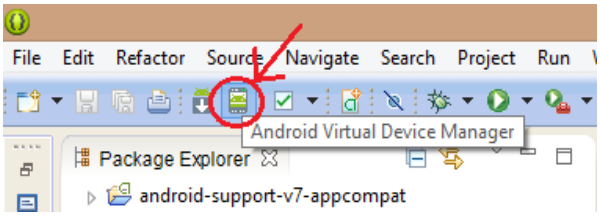
2. Emuladores. Un emulador es un software que imita todo el *hardware* de un dispositivo. Los emuladores proporcionan varias funcionalidades de un dispositivo real como por ejemplo el teclado, el menú, navegador de aplicaciones, etc. Sin embargo el usar un emulador presenta ciertas limitaciones como falta de:

- soporte para el GPS
- conexiones USB ni *Bluetooth*
- cámara de captura de fotos o videos
- auriculares
- simulación de insertar o exultas una tarjeta SD
- determinar el nivel de carga.

(García, Hernández, & Berlingen, 2010)

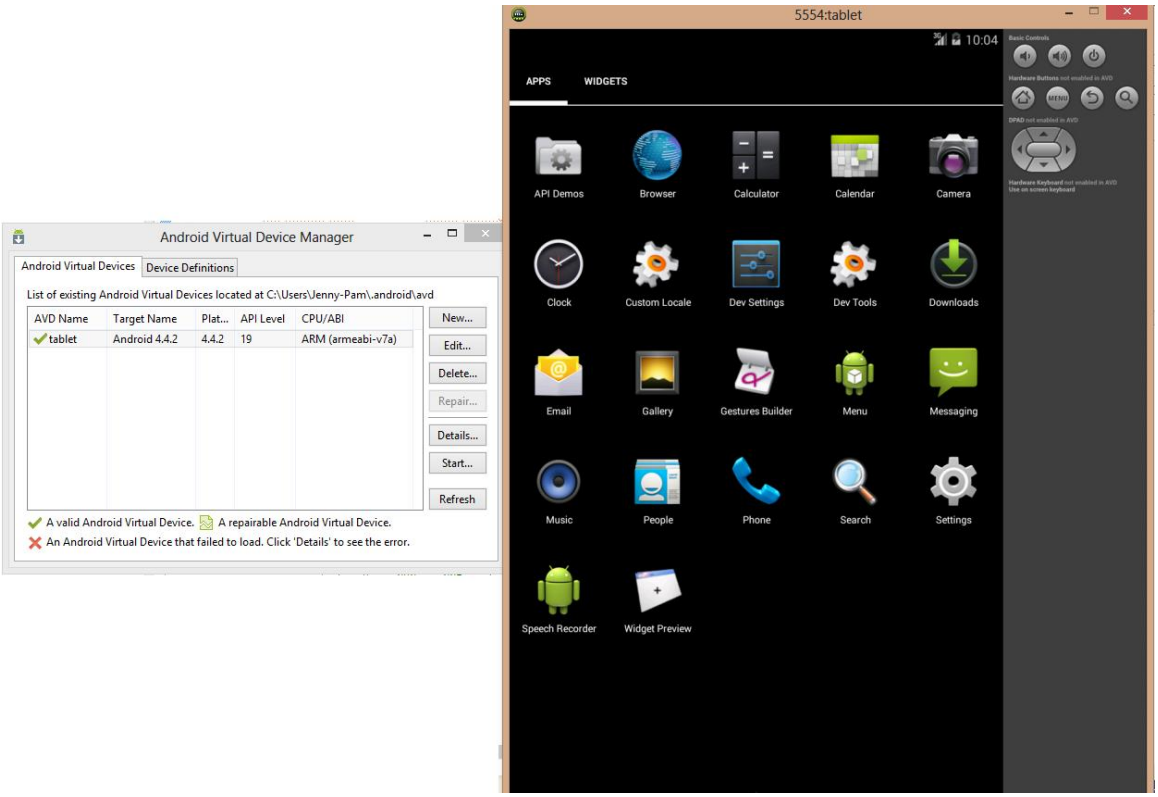
Para utilizar los emuladores el SDK tiene la capacidad de generar un dispositivo virtual de Android (AVD de las siglas en inglés de *Android Virtual Device*). Para trabajar con un AVD se debe presionar el botón de *Android Virtual Device Manager* que se encuentra en el menú superior de Eclipse como se muestra en la Figura 4. (Sacristrán & Fernández, 2013)

Figura 4. Botón para generar un AVD



Una vez abierto el Android Virtual Devide Manager se selecciona el dispositivo que se quiere simular, junto con la versión del sistema operativo que se tiene, el nivel del API, la capacidad de RAM, etc. En la Figura 5 se observa del lado izquierdo la tabla donde se especificaron todas las características del dispositivo simulado. Al lado derecho se observa la simulación del dispositivo donde se puede navegar por las aplicaciones, y se pueden ejecutar los programas que se estén desarrollando. (Sacristrán & Fernández, 2013)

Figura 5. Muestra de un AVD



3. Kit de desarrollo nativo (NDK). En NDK es un conjunto de herramientas que permiten a los desarrolladores programar ciertas partes de la aplicación en código nativo en las apps de Android, es decir que permite a los programadores implementar código en C y C++, que les permite acceder a capas inferiores en la arquitectura y modificar ciertas librerías de Android para generar un código más eficiente según la aplicación que desee el desarrollador. (Ratabouil, 2012)

E. COMPONENTES DE UNA APLICACIÓN ANDROID

Todos los componentes de Android son objetos por lo tanto todos son una clase que se compone de estado y de comportamiento; es decir, posee datos almacenados que son sus propiedades (variables) y que definen por ejemplo su color, tamaño, etc. Por otro lado presenta servicios que periten realizar actividades como mostrar un mensaje, cambiar de estado, etc. (García, Hernández, & Berlingen, 2010)

El desarrollo de aplicaciones Android se basa mediante bloques esenciales de componentes, hay cinco tipos de componentes básicos para una aplicación que se presentan a continuación:

1. Activity. Cada pantalla es una Activity que se presenta en una aplicación. Este componente muestra una interfaz gráfica en vistas, las cuales son unas áreas rectangulares en la pantalla, responsables de la elaboración y gestión de eventos. Todos los componentes visuales se definen dentro de una vista

llamada *View*; además las actividades también responden a eventos que pueden o no estar relacionados con los componentes gráficos de la *View*. Al desarrollar una aplicación Android el desarrollador crea sus propias actividades, obteniendo todas las funcionalidades de la Activity provista por Android por medio de la técnica de herencia, del paradigma e orientación a objetos. Con esto consigue escribir actividades con funcionalidades modificadas o añadidas acorde a los fines de la aplicación. Una aplicación móvil generalmente contiene varias actividades y éstas se llaman entre sí utilizando *Intent*[14] (García, Hernández, & Berlingen, 2010). (Pérez, 2012)

Cada actividad posee un ciclo de vida, y una aplicación Android puede manejar varios ciclos de vida de distintas actividades; cuando una actividad es iniciada la anterior es pausada y almacenada en una pila de actividades para su futura reanudación. Una actividad puede estar en cuatro estados distintos que se explican a continuación, y en la Figura 6 se puede observar qué métodos llevan de un estado al otro.

a. Activa. Es cuando la actividad se encuentra en ejecución como la pantalla principal de la aplicación, y por lo tanto se encuentra hasta arriba en la pila de actividades. (García, Hernández, & Berlingen, 2010)

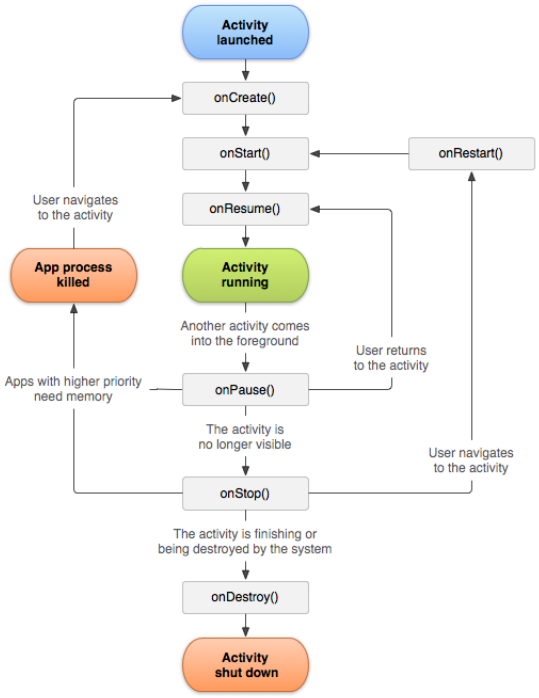
b. Pausada. Este estado se alcanza cuando otra actividad es llamada, efectivamente reemplazando la pantalla principal y reclamando prioridad de ejecución, sin embargo aún es visible ya que la otra actividad no ocupa toda la pantalla o es traslúcida. Una actividad en pausa sigue ejecutándose, conserva su estado pero tiene la posibilidad de ser finalizada por el sistema en caso de que falte algún recurso como por ejemplo memoria. (García, Hernández, & Berlingen, 2010)

c. Detenida. Cuando una actividad llama a otra que sí ocupa toda la pantalla, la actividad inicial pasa a ya no estar visible y por lo tanto pasa a estar detenida. Cuando una actividad está en este estado es muy probable que el sistema operativo del dispositivo móvil la cierra con el objetivo de liberar memoria y recursos. La actividad debe asegurarse de guardar sus propiedades actuales para que al volver a estar activa recupere su estado original; para dicha acción existen dos métodos que son *onSaveInstanceState()* y *onRestoreInstanceState()* que son encargados de guardar y de restaurar el estado de la actividad antes de que pueda ser finalizada. (García, Hernández, & Berlingen, 2010)

d. Finalizada. Cuando una actividad es cerrada entonces se finaliza y libera todos los recursos que estaba utilizando terminando su ciclo de vida. Si en dado caso se volviera a iniciar la actividad se iniciaría como una nueva y no se obtiene el estado anterior a diferencia de cuando es detenida. (García, Hernández, & Berlingen, 2010)

[14] Ver definición abajo

Figura 6. Ciclo de vida de una Activity (Pozo, 2012)



Los métodos *onCreate(), onStart(), onResume(), onPause(), onStop(), onRestart()* y *onDestroy()* son los métodos que se ejecutan cuando una actividad cambia de estado, y en la Figura 6 se puede observar en qué momento se ejecuta cada método y qué estado genera. Si se desea guardar los datos al pausar la actividad o generar eventos o ejecutar instrucciones al crearse la actividad se deben sobrescribir dichos métodos en el código. (Pozo, 2012)

2. Intent. Un *Intent* se puede describir como el deseo de que la aplicación realice alguna acción; por ejemplo si se necesita realizar una llamada se crea un objeto *Intent*, con la acción *CALL* y como datos del *Uniform Resource Identifier*[15] (URI) el teléfono al cual se desea llamar. Al realizar un *Intent* el sistema busca qué aplicaciones pueden ejecutar la acción requerida, y delega la tarea a otra aplicación. Cuando un *Intent* es lanzado se muestra al usuario la lista de las aplicaciones que pueden ejecutar la tarea deseada, sin embargo se puede indicar explícitamente el destinatario del *Intent*, en ese caso se ejecuta la acción siempre en una aplicación previamente escogida. Se podría interpretar como que los *Intents* son una interfaz de comunicación entre aplicaciones. Un aspecto muy útil de estos componentes es que no es necesario conocer ningún detalle sobre la aplicación que resolverá la acción que se desea ejecutar. (García, Hernández, & Berlingen, 2010)

[15] Ver definición en glosario

El objeto *Intent* está compuesto de una acción que indica la tarea a realizar; por ejemplo la acción *CALL* para llamar, la acción *SEND* para enviar un correo, etc. También está compuesto de una categoría que son los metadatos[16] referidos al *Intent,* y los datos definidos como un objeto URI. Al llamar a un *Intent* se pueden generar tres componentes: puede ser una *Activity*, un *BroadcastReceiver* o puede ser un *Service.* (García, Hernández, & Berlingen, 2010)

3. Service. Un servicio es un proceso que carece de interfaz gráfica y cuyo tiempo de ejecución generalmente es significativamente largo. Tiene la capacidad que aunque se ejecuten otras actividades un servicio puede seguirse ejecutando en un segundo plano. Un ejemplo puede la aplicación reproductora de música que aunque el usuario se salga de la aplicación que la inicio y se inicien otras aplicaciones la música sigue sonando. Aunque los servicios carezcan de interfaz gráfica son capaces de mostrar notificaciones a los usuarios. Un aspecto importante es que los servicios pueden ser locales o remotos. Un servicio local es cuando dicho servicio solamente puede accederse desde la aplicación que lo contiene, en cambio un servicio remoto puede ser accedido por medio de otra aplicación. (García, Hernández, & Berlingen, 2010)

4. Content Provider. El componente *Content Provider* brinda la capacidad a cualquier aplicación de Android de almacenar datos definiendo una interfaz con métodos especializados. Cualquier aplicación puede acceder a los datos sin necesidad de conocer la implementación del almacenamiento; lo único necesario para utilizarlo es conocer los métodos de la API. (Pérez, 2012)

5. Broacast Recievers. Este componente es utilizado para realizar la ejecución de una actividad dentro de otra aplicación activa cuando un evento es producido, por ejemplo cuando la batería de un dispositivo está baja se lanza una alerta informando que se debe cargar el dispositivo sin necesidad de estar en una actividad que maneje la energía del mismo. Estos componentes son similares a los servicios ya que se ejecutan en segundo plano y carecen de interfaz gráfica, pero son activados por medio de un *Intent*, y su función es realizar una tarea a partir de un evento estando en cualquier otra aplicación. (García, Hernández, & Berlingen, 2010) (Pérez, 2012)

F. ESTRUCTURA DE UN PROYECTO ANDROID

1. Android Manifest. El archivo *AndroidManifest.xml* es el que contiene la configuración principal de la aplicación, así como su nombre, versión, ubicación del icono, etc.; contiene también los componentes como las actividades, servicios, etc. y además contiene los permisos necesarios para su ejecución.

[16] Los metadatos son datos que describen otros datos, pueden documentar atributos como nombre, tipo de dato, etc.

En la Figura 7 se observa el archivo *AndroidManifest.xml*. Las partes importantes a notar son los recuadros; el cuadro anaranjado son los permisos agregados por el desarrollador para permitir que la aplicación ejecute ciertas acciones; en este caso acceso a llamadas telefónicas, a internet y acceso al estado de *network*; el cuadro verde es un ejemplo de cómo agregar otra actividad a la aplicación distinta al MainActivity; se debe especificar el nombre de la clase que hereda de *Activity*, y además se debe especificar el nombre que se mostrará en pantalla; si esto no se agrega al *manifest* la aplicación dejará de responder al *Intent* que llama a la *Activity*, no se ejecutará y ya no responderá la aplicación. (Ostrander, 2012)

Figura 7. Archivo XML del Android Manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="example.restauranteantic"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="19" />

    <uses-permission android:name="android.permission.CALL_PHONE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="example.restauranteantic.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

        <activity
            android:name="example.restauranteantic.MenuCategorias"
            android:label="@string/app_name_menu" >
        </activity>

    </application>

</manifest>
```

2. Layout. Los *layouts* son contenedores de vistas (*ViewGroup*), y se crea un archivo por *Activity*. Estos archivos fueron creados con el objetivo de facilitar el diseño de la interfaz gráfica según la distribución que desee el programador de los componentes a colocar; al arrastrar un componente gráfico se genera automáticamente el código en el respectivo *layout*. (García, Hernández, & Berlingen, 2010)

En los archivos *layouts* existen contenedores que agrupan y posicionan a los *widgets*[17], dentro de ellos. Entre los paneles más utilizados se encuentran los siguientes:

a. LinearLayout. Posiciona a los *widgets* en una fila o columna dependiendo si se define la orientación de la pantalla como vertical u horizontal; cada componente gráfico que se agrega se coloca ya sea abajo y a la derecha, respectivamente, del componente gráfico previamente colocado. (Sacristrán & Fernández, 2013) (Carrero)

b. RelativeLayout. Este panel permite posicionar a cada *widget* de forma relativa a otro *widget* o a su contenedor padre; así que si se posiciona con respecto a otro componente, y si éste se mueve se moverán todos los componentes dependientes de su posición. (Sacristrán & Fernández, 2013)

c. AbsoluteLayout. Al utilizar este tipo de layout se permite posicionar a los componentes gráficos en una posición absoluta, es decir, especificar en qué pixel de la pantalla inicia el componente; una desventaja de este contenedor es que no se ajusta a distintos tipos de tamaños de pantallas. (Carrero)

d. TableLayout. Este panel organiza a los *widgets* como si fuera una tabla, es decir, se organiza en filas y columnas. Se define en una forma similar a HTML[18] el cual se indica las filas (*TableRow*) y luego las columnas de cada fila. (Sacristrán & Fernández, 2013)

e. FrameLayout. Este panel permite superponer en la pantalla varios componentes gráficos; generalmente cuando se utiliza este contenedor se utilizan pocos *widgets* ya que al poseer varios puede generar confusión al sobreponer tantos elementos. (Sacristrán & Fernández, 2013)

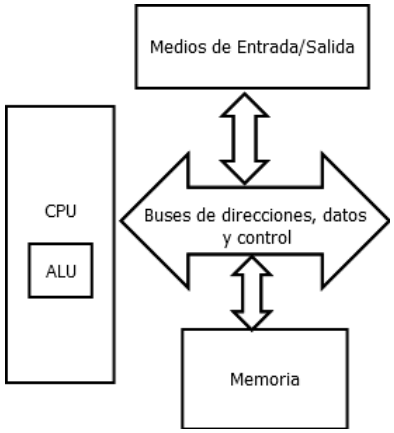
G. MICRO-CONTROLADORES

Un micro-controlador se puede describir como la combinación de una unidad central de procesamiento (conocido por sus siglas en inglés como CPU), memoria de almacenamiento y medios de entrada y salida, todos estos elementos encapsulados dentro en un chip. (Valdés Pérez & Pallas Areny, 2007)

[17] Un *widget* es un componente de la interfaz gráfica que permite o no interacción con el usuario.

[18] Es un lenguaje de programación para páginas web; viene de sus siglas en inglés de *HyperText Markup Language.*

Figura 8. Diagrama de bloques simple para un microcontrolador (Steiner, 2005)



La CPU realiza la obtención de las instrucciones almacenadas en memoria, luego decodifica las instrucciones y las ejecuta. Por medio de circuitos internos, la CPU es capaz de realizar operaciones lógicas y aritméticas básicas con datos binarios, los circuitos internos que permiten al CPU realizar lo antes mencionado es conocido como ALU (Arithmetic Logic Unit). (Valdés Pérez & Pallas Areny, 2007)

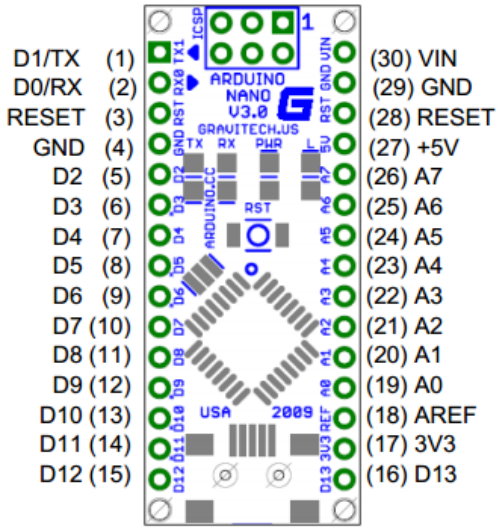
En la memoria es donde se encuentran almacenadas las instrucciones del código de programación y datos que se manipulan con el programa. La sección de memoria se encuentra dividida en dos tipos: memoria RAM (Random Access Memory) y memoria ROM (Reas Only Memory). La primera es conocida como memoria de lectura y escritura, esta memoria pierde los datos y la información almacenada en el momento que es desconectada de la alimentación, por este motivo es llamada memoria volátil. El segundo tipo es memoria de solo lectura y no es una memoria volátil. Ambas memoria son de acceso aleatorio, ya que el tiempo requerido para encontrar un dato no depende de la localidad en la cual fue almacenado. (Valdés Pérez & Pallas Areny, 2007) (Steiner, 2005)

Los medios de entrada y salida permiten al microcontrolador comunicarse e interactuar con dispositivos externos para controlar diferentes tareas. (Valdés Pérez & Pallas Areny, 2007)

1. Arduino Nano. Arduino es una plataforma para desarrollo de electrónica bajo la iniciativa Open-Source. Dispone de hardware y software de uso fácil para que cualquiera que se encuentre interesado en el desarrollo de proyectos de electrónica sea capaz de utilizarlo, el lenguaje de programación es compatible con C++. La plataforma es capaz del control de datos, manejos de puertos de entrada/salida, interacción con variedad de sensores, control de motores entre otros. (Arduino , 2014)

Arduino Nano es una tarjeta electrónica compuestas por un microcontrolador ATmega328P de la familia AVR perteneciente a la compañía Atmel, el diagrama de identificación de pines se observa en Figura 9. La tarjeta de Arduino Nano dispone con 30 pines, 14 de los pines son para entrada/salida digital de los cuales 6 pines pueden configurarse independientemente como salida PWM (Pulse Width Modulation), el resto de la distribución de pines se observa en Cuadro 2. (Arduino , 2014)

Figura 9. Diagrama de distribución de pines Arduino Nano (v 3.0), vista superior (Arduino , 2014)



Cuadro 2. Distribución de pines de tarjeta Arduino Nano (v 3.0) (Arduino , 2014)

Pin	Identificador	Tipo	Descripción
1,2,5-16	D0-D13	Entrada/Salida	Pines de entrada/salida digital
3,28	RESET	Entrada	Reinicio (se activa en bajo)
4,29	GND	Alimentación	Tierra pin de alimentación
17	3V3	Salida	Salida de 3.3 V regulado por FTDI
18	AREF	Entrada	Referencia para ADC
19-26	A0-A7	Entrada	Entradas analógicas
27	+5V	Salida o Entrada	+5V de salida del regulador o +5V de entrada por alimentación externa
30	VIN	Alimentación	Voltaje de entrada pin de alimentación

La tarjeta de Arduino Nano puede alimentarse por medio de un conector Mini-B USB, con alimentación externa no regulado de 7-12V en el pin 30 o con alimentación externa regulada de 5V en el pin 27. La fuente de alimentación es automáticamente seleccionada a la fuente con mayor valor de voltaje. El chip

FTDI FT232RL que provee 3.3V regulados en el Arduino Nano es funcional únicamente si la tarjeta es alimentada mediante el conector USB. (Arduino , 2014)

El microcontrolador ATmega32 cuenta con 32 KB de memoria, 2 KB son utilizados para almacenar el bootloader. Bootloader es un código pre-programado en el Arduino para permitir la programación sin necesidad de un hardware externo. Cuenta también con 2 KB de memoria SRAM y 1 KB de memoria EEPROM. (Arduino , 2014)

Arduino Nano cuenta con varias formas de comunicación con la computadora o bien con otros dispositivos, entre ellos se encuentra la comunicación serial UART TTL, disponible en los pines 0 para TX y 1 para RX. También dispone de comunicación I2C y SPI. (Arduino , 2014)

H. COMUNICACIÓN SERIAL

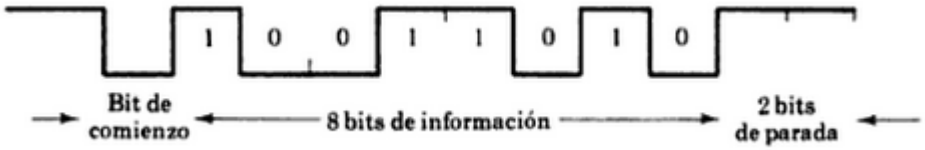
La comunicación serial es la transmisión de datos por una sola línea física. La transferencia de datos es secuencial, se envía un bit tras otro, según se indique en el protocolo establecido. La comunicación serial necesita de convertidores paralelo-serial y serial-paralelo, por lo que la implementación es más compleja a la comunicación en paralelo. (Sanchís, 2002)

Ya que la transferencia de datos se realizar por una única línea física, se tiene una línea de transmisión conocida como TX y una línea de recepción conocida como RX. Tanto la línea de transmisión como la de recepción disponen de una entrada de reloj para sincronizar la razón de envío o recepción de bits. Si la señal de reloj es compartida para los dispositivos que se están comunicando de forma serial, se conoce como comunicación síncrona pero si los dispositivos en comunicación no comparten la misma señal de reloj entonces se conoce como comunicación asíncrona. (Morris Mano, 1982)

Para la comunicación serial se debe indicar el comienzo de transmisión de un nuevo dato, con el modo síncrono a veces se utilizar un carácter de sincronización, pero para el modo asíncrono como mínimo se envían dos bits extra a los bits de datos, el bit de inicio y el bit de parada. (Morris Mano, 1982)

Los tipos de bit enviados en la transmisión de datos asíncrona en forma serial son: Bit de inicio, es el primer cambio de nivel alto a nivel bajo seguido por un tiempo en estado de parada. Bit de datos, son los bits que llevan la información de los datos. Bit de paridad, es un bit opcional que sirve cuando se dispone de una fase de detección de errores. Bit de parada, indica el fin de la transmisión. La distribución grafica de los bits de transmisión se pueden observar en Figura 10. (Sanchís, 2002)

Figura 10. Distribución de bits de transmisión asíncrona en serie (Morris Mano, 1982)

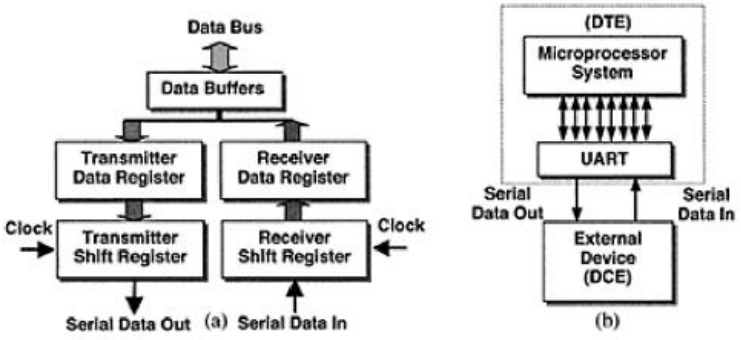


1. UART (Universal Asynchronous Receiver-Transmitter). El Transmisor-Receptor Asíncrono Universal (por sus siglas en inglés, UART), es un circuito integrado que contiene todos los registros y componentes de sincronización necesarios para recibir datos en forma serial para convertir y transmitir en forma paralela y de manera inversa de paralelo a serial. (Anand Kumar, 2009)

El UART es utilizado para transmisión asíncrona de datos en forma serial entre equipo de termina de datos (por sus siglas en inglés, DTE) y equipo de comunicación de datos (por sus siglas en inglés, DCE). Las tareas que realiza el UART para la comunicación entre DTE y DCE son: la conversión de datos serial-paralelo y paralelo-serial, comprobación de bit de paridad y detección de error, inserción y detección de bits de inicio y parada. (Tomasi, 2003)

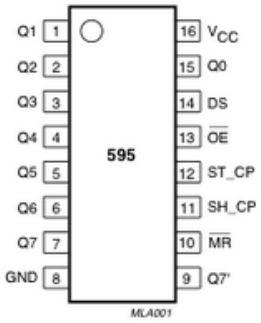
Dentro de los dispositivos DTE y computadoras se realiza la transmisión en niveles TTL, pero en líneas de comunicación los niveles son de -14 a +14 voltios. Además las señales dentro del DTE o la computadora se transmiten de forma paralela, por lo que se hace necesario un intermediario, en este caso el UART. Por esta misma razón el UART dispone de una unidad de control y un adaptador de nivel de voltaje. El funcionamiento interno del UART y la interacción entre DTE y DCE se pueden observar en la Figura 11. (Bai, 2006)

Figura 11. (a) Diagrama de bloques de la estructura interna del UART (b) Diagrama de bloques de comunicación serial entre DTE y DCE (Bai, 2006)



a. Corredor de bit. Un corredor de bits es un tipo de convertidor de serial a paralelo, con un solo pin de datos se pueden obtener varias salidas, como en el caso del circuito integrado (por sus siglas en inglés, IC) 74HC595. Con este IC es posible tener un solo pin de transmisión de datos y 8 pines de salida, por medio de una señal de reloj y un pin de control para determinar en qué momento leer datos de su entrada es posible enviar una señal de forma serial y obtener su igual en 8 bits individuales. En la Figura 12 se puede observar la distribución de pines del 74HC595, además en el Cuadro 3 se puede observar la descripción de cada pin. (Arduino, 2014)

Figura 12. Diagrama de distribución de pines para 74HC595 (Arduino, 2014)



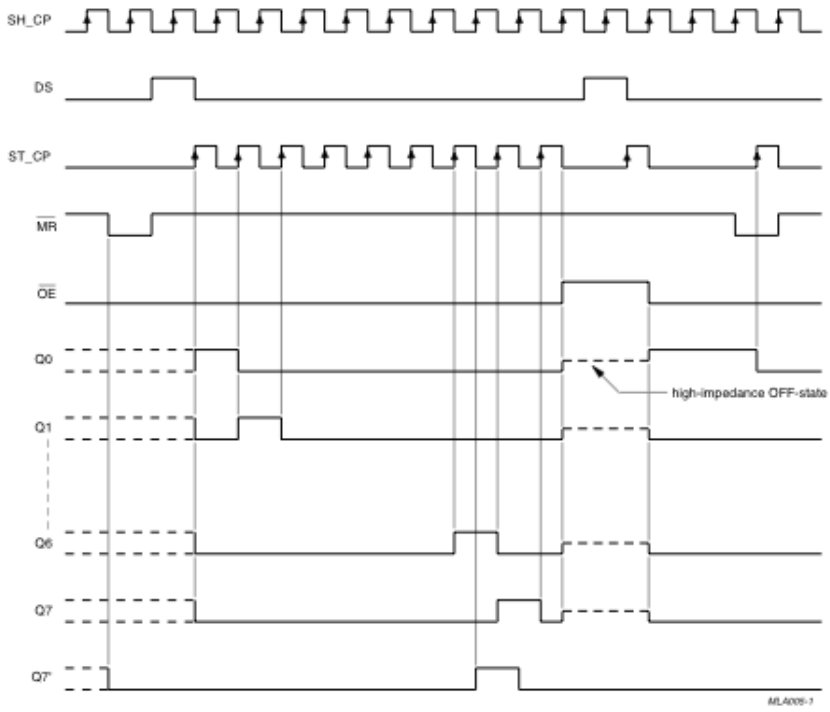
Cuadro 3. Distribución y descripción de pines de 74HC595

Pin	Identificador	Descripción
1-7,15	Q1-Q7,Q0	Salidas en niveles TTL
8	GND	Tierra pin de alimentación
9	Q7´	Salida serial
10	MR	Borrado maestro
11	SH_CP	Pin de entrada de reloj de corrimiento
12	ST_CP	Pin de almacenamiento de dato
13	OE	Habilitador de salida, se activa en nivel bajo
14	DS	Entrada serial
16	Vcc	Voltaje de alimentación

(Arduino, 2014)

El 74HC595 toma el valor que posee el pin de datos cuando existe un cambio de nivel bajo-alto, el valor del pin de datos es asignado a Q0 si esto se realiza 7 veces más el primer valor ingresado en el pin de datos se encontrara en Q7, el diagrama de tiempo se puede observar en Figura 13. (Arduino, 2014)

Figura 13. Diagrama de tiempos para funcionamiento de 74HC595 (Arduino, 2014)

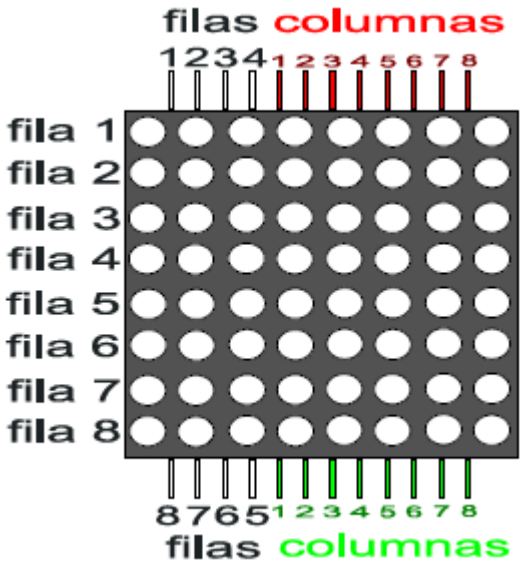


I. LED´s

Diodo emisor de luz (por sus siglas en inglés, LED), al igual que todo diodo el LED se obtiene de la unión de material con carga negativa y un material con carga positiva, conocido como unión P-N. La corriente es capaz de viajar desde la zona P hacia la zona N, pero no es capaz de viajar en forma inversa. Al tener un diferencial de voltaje los electrones viajan a los espacios libres en la zona P, "agujeros", cuando un electrón llega a un "agujero" este entra a un nivel de energía bajo y libera su energía en forma de fotón. (Winder, 2008)

1. Matriz de LED. Es un arreglo de LED´s dispuestos en forma de filas y columnas, interconectados de tal forma que cada uno de los LED´s puede activarse individualmente. En la Figura 14 se puede observar la distribución de pines de una matriz RG de 8x8. (Gallardo, 2014)

Figura 14. Matriz RG de LED´s, de 8x8 (Gallardo, 2014)



J. BLUETOOTH

Aparte del WI-FI, el Bluetooth es un estándar de comunicación inalámbrica, la tecnología Bluetooth fue inventada por la compañía L.M. Ericsson de Suecia en 1994. Ericsson pretendía unificar, por medio de la tecnología Bluetooth, el mundo de los móviles. En 1998, Ericsson, IBM, Intel, Nokia y Toshiba fundaron la Bluetooth SIG, donde desarrollaron las especificaciones de conexión inalámbrica de corto alcance basados en la tecnología Bluetooth de Ericsson. (Thompson, Kline, & Kumar, 2008)

El inicio del Bluetooth fue para la sustitución de cables para la conexión de teléfonos y sus accesorios, entonces se pensó en una red inalámbrica de corto alcance que permitiera la conexión de estos dispositivos sin la necesidad de un intermediario para la conexión como un "router". (Thompson, Kline, & Kumar, 2008)

Entre las especificaciones que se acordaron por la SIG, se encuentran las Especificaciones del núcleo Bluetooth, Especificaciones de transporte Bluetooth, Especificaciones de Protocolos Bluetooth y Especificaciones de perfiles Bluetooth. Las especificaciones de núcleo definen la arquitectura del Bluetooth como los controladores y especificaciones del anfitrión o "host". Las especificaciones de transporte definen cómo pueden utilizarse los diferentes transportes para la comunicación entre el anfitrión y el controlador, por ejemplo el controlador puede ser UART, USB o Three-Wiere. Las especificaciones de protocolo definen los protocolos de alto nivel sobre los cuales se ejecuta la tecnología Bluetooth, como por ejemplo RFCOMM y OBEX. Las especificaciones de perfil son especificaciones individuales que describen el

perfil de cada Bluetooth, los perfiles son la base de los protocolos Bluetooth y describen cómo las aplicaciones utilizarán la pila del Bluetooth. (Thompson, Kline, & Kumar, 2008)

El protocolo de comunicación por radio frecuencia (por sus siglas en inglés, RFCOMM), es un conjunto de protocolos de transporte, que proporciona conexiones simultaneas para los dispositivos Bluetooth, emulando así el puerto serial RS-232. (Thompson, Kline, & Kumar, 2008)

Existe una interface entre el anfitrión Bluetooth y el controlador Bluetooth conocido por sus siglas en inglés como HCI. El anfitrión Bluetooth se encuentra en el nivel alto de la pila y usualmente se encuentra implementado en software, en general se encuentra integrado con el sistema del dispositivo. Por otra parte el controlador es usualmente un dispositivo electrónico de radio modulación, el controlador se comunica con el anfitrión mediante algunos de los mecanismos estándar de comunicación, por ejemplo UART. (Thompson, Kline, & Kumar, 2008)

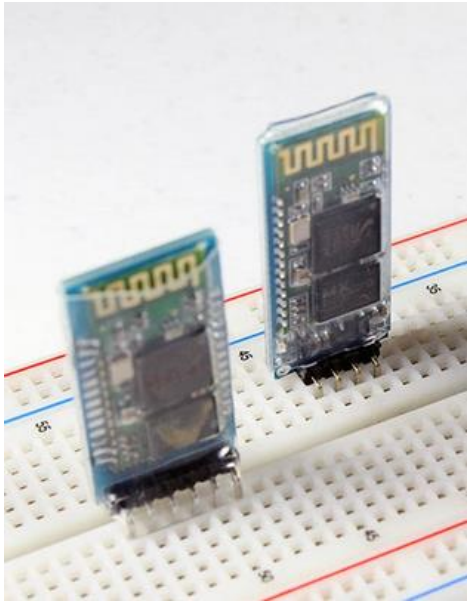
Los dispositivos Bluetooth pueden operar con uno o más perfiles. Los cuatro perfiles básicos son: Perfil de Acceso General (por sus siglas en inglés, GAP), Perfil de Puerto Serial (por sus siglas en inglés, SPP), Perfil de Descubrimiento de Servicios (por sus siglas en inglés, SDAP) y el Perfil de Intercambio de Objetos General (por sus siglas en inglés, GOEP).

El perfil SPP define los requerimientos para el dispositivo Bluetooth necesarios para inicializar la emulación de conexión serial por cable, utilizando el protocolo RFCOMM entre los dispositivos a comunicar. (Thompson, Kline, & Kumar, 2008)

1. Módulo Bluetooth HC-05. El módulo Bluetooth HC-05, es un dispositivo Bluetooth que opera bajo el perfil de puerto serial (SPP), diseñado para configuración de conexión inalámbrica simple. El puerto serial Bluetooth de este dispositivo puede operar a 3Mbps de modulación con frecuencia de transmisión-recepción de 2.4 GHz base. Utiliza un chip CSR Bluecore04-External con tecnología CMOS y un adaptador de frecuencia (por sus siglas en inglés, AFH). En la Figura 15 se puede observar un ejemplo de los módulos Bluetooth HC-05 y HC-06. (ITead Studio, 2014)

El HC-05, posee 34 pines, en la presentación con placa se tienen a disposición 6 pines, 2 para alimentación, 1 para UART_TX, 1 para UART_RX, 1 para control de configuración y finalmente 1 pin para indicar el estado del HC-05. La alimentación del módulo es 3.3V, con voltaje de operación lógica de 1.8V a 3.6 V. (ITead Studio, 2014)

Figura 15. Módulos HC-05 en la izquierda y módulo HC-06 en la derecha, ambos soldados a una placa para facilitar la configuración y conexión de los mismos. (Ruben, Jesus, 2014)



Capaz de transmisión RF de 4dBm, por lo que su alcance es de 10 a 5 metros. Posee una interfaz UART para la programación de su razón de baudios (conocido en inglés como Baud Rate), un baudio es el conjunto de bits que es transmitido. (ITead Studio, 2014)

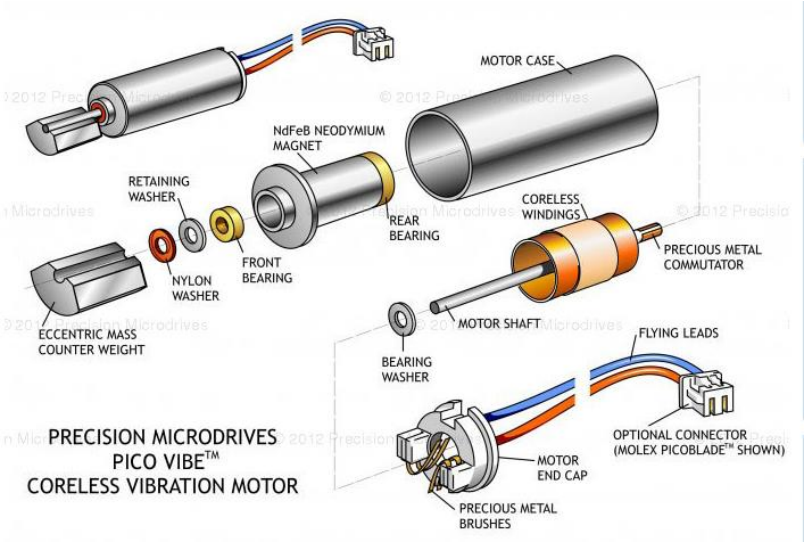
El HC-05, posee parámetros configurables, como: "nombre", "rol", "contraseña" y "baud rate". Estos parámetros pueden configurarse por medio de comandos AT. Los comandos AT son un subconjunto de los comandos Hayes, los cuales son órdenes que permiten controlar y configurar módems. (García & Morales, 2012)

El módulo HC-05 puede encontrar soldado a una placa que facilita su manejo, en esta placa se realizan las conexiones a los pines de voltaje de alimentación, pines de configuración y pines de comunicación serial. El HC-05 puede operar en dos modos: Modo de Configuración y Modo de Comunicación. En el modo de configuración se debe colocar en nivel alto el pin 34 del módulo denominado "Key" en la placa y transmitir a un "baud rate" de 38400 comandos AT, para comunicarnos con el módulo y configurarlos es necesario tener acceso a él por medio una interfaz serial, el módulo tendrá un led parpadeando cada 2 segundos aproximadamente indicando que se encuentra en modo de configuración. El modo de comunicación transmite por RF lo enviado al pin RX y transmite por TX lo recibido en RF. (ITead Studio, 2014)

K. MOTOR DE VIBRACIÓN (masa excéntrica en movimiento rotacional)

El motor de vibración, es un motor que posee una masa excéntrica en su eje, conocido en inglés como ERM Vibration Motor. En un motor de corriente directa una masa, no simétrica, es agregada al eje de forma excéntrica, estos motores de corriente directa son conocidos en inglés como "page motor". En la Figura 16 se puede observar la estructura de un motor de vibración de corriente directa. (Precision Microdrives, 2014)

Figura 16. Vista en explosión de motor de vibración de corriente directa (Precision Microdrives, 2014)

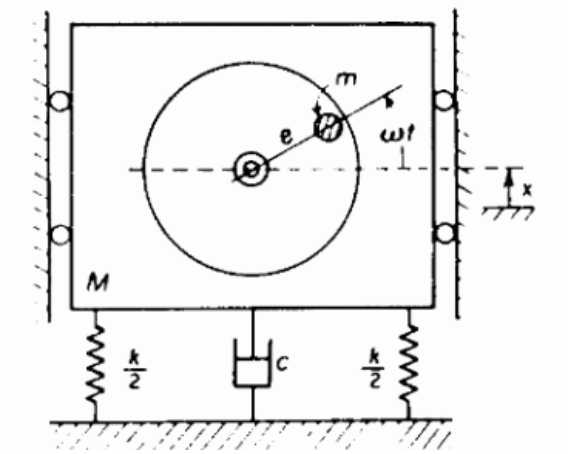


Al momento de girar, se crea una fuerza centrípeta asimétrica debido a la masa no alienada con el eje, como resultado se crea una fuerza centrífuga que genera el desplazamiento del motor. Al encontrarse a altas revoluciones el motor es constante desplaza por estas fuerzas asimétricas. Los desplazamientos constantes son percibidos como vibración. (Precision Microdrives, 2014)

El sistema eje-masa excéntrica puede modelarse tomando las siguientes consideraciones: La entrada del sistema no es el voltaje de alimentación del motor sino el movimiento de rotación de la masa que se encuentra desalineada del dentro del eje. El movimiento de la masa se puede modelar como onda sinusoidal. Se aproxima a tener un solo grado de libertad, lo que indica que el movimiento se genera en una sola dirección, esto es aceptable para vibradores pequeños ya que el desplazamiento del motor en otras direcciones es suficientemente pequeño para considerarse despreciables.

El modelo de vibración con un grado de libertad se puede representar con una masa conectada a un resorte con pistón. En el centro de la masa se encuentra el eje de rotación con una masa posicionada a distancia de desalineación como se muestra en la Figura 17. (Precision Microdrives, 2014)

Figura 17. Representación de modelo de vibración con un grado de libertad, indicando puntos de referencia y variables del modelo (Precision Microdrives, 2014)



Como se mencionó anteriormente la entrada del sistema es el movimiento de la masa "m", la cual se modela como una onda sinusoidal. En el diagrama de la Figura 17, se muestran dos resortes de constante k/2 que puede sustituirse por un solo resorte de constante k, por ley de Hooke´s se obtiene que la fuerza del resorte es:

Ecuación 1: $F = k * x$

Para el pistón la fuerza es proporcional a la velocidad de la masa "m", considerando "c" como constante de amortiguamiento se obtiene que la fuerza del pistón es:

Ecuación 2: $F = c * \frac{dx}{dt}$

El movimiento de la masa del sistema de motor de vibración ERM, restando la masa excéntrica, sigue la segunda ley de Newton:

Ecuación 3: $F = (M - m) * \frac{d^2x}{dt^2}$

La fuerza centrípeta que experimenta la masa "m" está relacionada con la excentricidad y la velocidad rotacional a la que se mueve, de lo anterior se obtiene:

Ecuación 4: $F_C = m * e * \omega^2$

Donde "e" es la distancia a la que se encuentra la masa "m" desde el centro del eje y "ω" es la velocidad rotacional a la que se mueve la masa "m".

La sumatoria de la fuerza del resorte, fuerza del pistón y la fuerza que genera el movimiento del sistema son iguales a la entrada del sistema.

Ecuación 5: $(M-m)*\frac{d^2x}{dt^2}+c*\frac{dx}{dt}+k*x=F_C*\sin(\omega*t)$

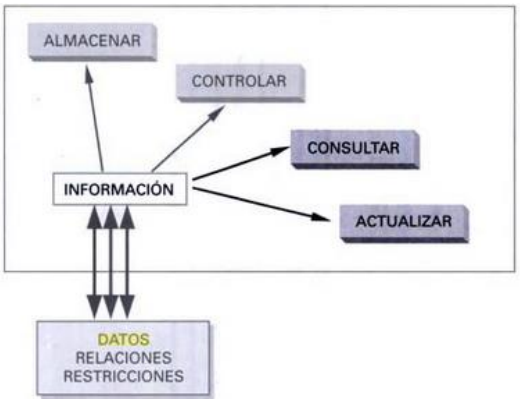
Debido a que la velocidad de rotación de la masa excéntrica está relacionada con el voltaje de alimentación del motor de corriente directa. Es posible en encontrar la función del sistema teniendo como entrada la corriente del motor. (Precision Microdrives, 2014)

L. BASE DE DATOS

Las bases de datos se definen como una forma computarizada para almacenar colecciones de registros e información digital. Éstas se crearon debido a la creciente necesidad de gestionar grandes cantidades de información. Esta forma de guardar datos posee ciertas características útiles gracias a su carácter digital, como por ejemplo son fáciles de acceder, presentan adaptabilidad ante varios problemas, pueden almacenar diferentes tipos de información, proveen seguridad restringiendo accesos, entre otras cosas (Date, 2001).

Una base de datos debe de ser una herramienta capaz de ayudar en la solución de problemas concretos de gestión de información. Para proveer esta funcionalidad se necesita que la base de datos cumpla con la función mostrada en la Figura 18, que indica que se debe de ser capaz de almacenar, controlar, consultar y actualizar la información almacenada, que a su vez se le han aplicado una serie de relaciones y restricciones para facilitar las tareas descritas anteriormente. Esto se puede realizar a través de uno o varios programas informáticos (Date, 2001) (Mora, 2003).

Figura 18. Finalidad de una base de datos (Mora, 2003)



Entre los tipos de bases de datos existentes el más aceptado es el de bases de datos relacionales, propuesto por Edgar F. Codd del IBM San José ResearchLaboratory en el año 1969.Este tipo de base de datos consiste en agrupar la información por tablas con nombres únicos, tal como se muestra en la Figura 19, las filas muestran relación entre diferentes datos, también son llamadas registros, y las columnas son los distintos datos que se desean asociar con el registro, en este modelo cada tabla debe de tener una columna

designada para un valor único por cada registro, a esta columna se le denomina como llave primaria (Mora, 2003).

Figura 19. Ejemplo de tabla de base de datos relacional (Mora, 2003)

Personal

NOMBRE	PROFESION	LOCALIDAD
Pedro	profesor	Santander
Luis	estudiante	Santander
María	estudiante	Las Palmas
Ana	estudiante	Madrid

Los datos asociados con cada registro pueden ser de diferentes tipos dependiendo el software utilizado para la construcción y gestión de la base de datos; algunos de los tipos más comunes son: cadena de caracteres, numéricos, fecha, hora y binarios. Es posible cambiar ciertos parámetros de estos tipos de datos, como el largo de una cadena de caracteres o el formato de la fecha o la hora (Mora, 2003).

M. MySQL

Para gestionar una base de datos se utiliza un sistema de manejo de bases de datos relacionales o RDBMS por sus siglas en inglés, que son sistemas encargados de darle el formato relacional a las bases de datos y también están basados en el trabajo de Edgar Codd. Al comienzo los RDBMS no eran tan utilizados por su alto costo, debido a que al momento de adquirir una licencia software también era necesario contar con hardware de alto rendimiento para asegurar la funcionalidad del sistema (DuBois, 2008).

Actualmente existen varias opciones de bajo costo para gestionar bases de datos, una de las más utilizadas es MySQL, lanzado al público en 1996 para el sistema operativo Linux, también existen distribuciones para Windows y Mac OS. MySQL no es una implementación directa de un RDBMS si no que solamente está basada en este modelo, por lo que cuenta con otras funcionalidades y elimina ciertas restricciones encontradas en los RDBMS. Por ejemplo en MySQL es posible tener tablas sin llaves primarias por lo que se puede dar el caso de tener registros completamente idénticos (Fehily, 2002)

MySQL es un proyecto Open Source que se encuentra bajo la licencia *GNU General PublicLicense(GPL),* lo que significa que es posible obtenerlo sin costo para muchas aplicaciones, mientras éstas cumplan con los términos de uso de GPL. Estos términos indican que el código debe ser gratuito y permanecer gratuito sin importar que cambios se le apliquen y que la licencia debe ser utilizada para restringir la propiedad intelectual sin limitar la disponibilidad del *software*. Las distintas distribuciones de MySQL incluyen una variedad de herramientas entre las que se encuentran un servidor de SQL, que es la

parte que provee acceso, da formato y permite realizar operaciones a las bases de datos, también se incluyenprogramas de utilidades que con las que el usuario puede gestionar las bases de datos, hacer copias de seguridad, revisar tablas, etc; también de incluirse una librería para escribir programas propios en el lenguaje de programación C, la librería también está en ese lenguaje y puede ser utilizada en otros lenguajes de programación como Perl, PHP o Ruby (Fehily, 2002).

La sintaxis SQL es la utilizada en MySQL, ésta tiene la característica de que sus instrucciones son sencillas de entender debido a su similitud con el inglés. En la Figura 20 puede observarse un ejemplo de una instrucción. Las palabras en mayúsculas son palabras reservadas, la primera palabra de la instrucción define qué tipo de operación es la que se desea realizar, *SELECT*se utiliza para obtener datos de la tabla y también existen otras operaciones como *INSERTINTO* que agrega nuevo registros, *UPDATE* que edita registros existentes y *DELETE*que elimina registros de la base de datos. En el ejemplo las palabras "au_fname" y "au_lname" son los nombres de las columnas de las cuales se desea obtener los valores. Éstas deben separarse por coma y pueden ser sustituidas por un asterisco en caso de necesitar la información de todas las columnas de la tabla. La palabra reservada *FROM* indica a que tabla se le desea aplicar la instrucción, en este caso es a "authors". *WHERE* se utiliza para establecer condiciones, en el ejemplo se usa para indicar que se desea aplicar la instrucción solo a registros donde el campo "state" sea igual a "NY", por último *ORDER BY* indica que se desea que el resultado este ordenado según el valor de cierta columna, en este caso es "au_lname", esto funciona con varios tipos de datos como numéricos, fechas, textos, horas, etc. Conociendo lo anterior se puede saber que la instrucción del ejemplo pide los valores "au_fname" y "au_lname" de los registros de la tabla "authors" cuyo valor "state" sea igual a "NY" y que el resultado se ordene según "au_lname". A las instrucciones de SQL se les conoce comúnmente como consultas o *queries* (DuBois, 2008).

Figura 20. Ejemplo instrucción SQL (DuBois, 2008)

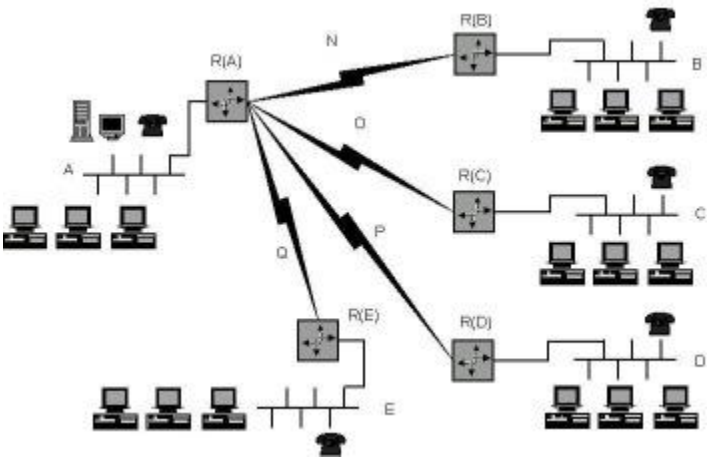
```
SELECT au_fname, au_lname
   FROM authors
   WHERE state = 'NY'
   ORDER BY au_lname;
```

N. SERVIDORES LOCALES Y REMOTOS

Las computadoras tienen varias maneras de comunicarse, actualmente la manera más común es el internet, que es considerado como una colección de redes de computadoras en cooperación. Estas redes contienen diferentes tipos de dispositivos en ellas por lo que es necesario que todos utilicen el mismo protocolo para para comunicarse entre aparatos. El protocolo utilizado es el TCP/IP el cual es el uso en conjunto entre dos protocolos, el TCP y el IP, cada uno con su respectiva función para enviar la información (Barceló, 2008) (Yeager, 1996).

El protocolo IP define que cada dispositivo que se conecte a una red se le otorga una dirección única, ésta contiene información como su identificador dentro de la red y la identificación de la red misma, en otras palabras contiene la ruta que debe de tomar la información para llegar al punto de distribución o la ruta que debe de tomar desde el punto de distribución al dispositivo. La Figura 21 muestra el ejemplo de una red donde las computadoras están conectadas a diferentes terminales, cada terminal le otorga una dirección local a cada computadora única en esta red, sin embargo dos computadoras pueden tener la misma dirección local si están en redes distintas. La dirección IP del dispositivo también incluye la dirección de la terminal así como de la terminal a la que ésta está conectada. Otra función del protocolo IP es incluir en el paquete de datos a enviar la información sobre de dónde viene el paquete y a dónde se dirige. Utilizando las direcciones IP se puede localizar otros dispositivos, para localizar una computadora en la misma red solamente es necesaria la dirección local, y con esta información se puede contactar a otros dispositivos. El protocolo TCP apoya comprobando la integridad del paquete, tanto en que la información sea la correcta así como que haya llegado en el orden correcto (Yeager, 1996).

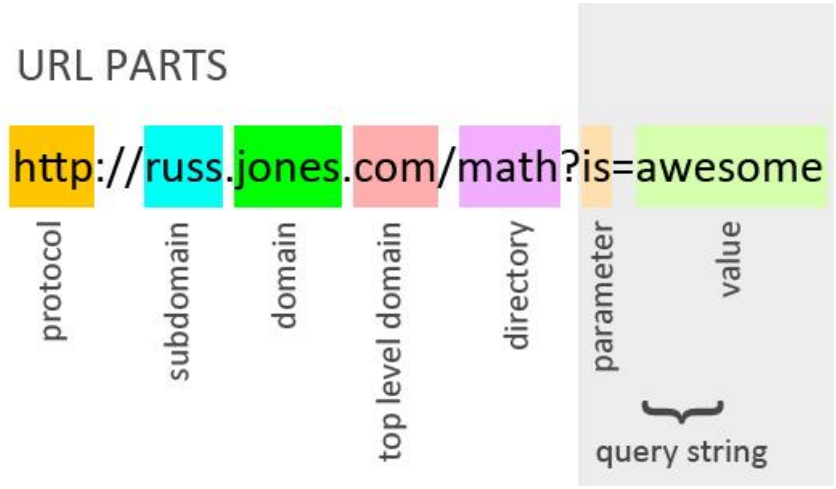
Figura 21. Ejemplo de redes (Rouse, 2008)



Los dispositivos pueden tomar distintos roles al establecer una conexión, entre estos se encuentran los roles de cliente y servidor. La relación entre estos es que el servidor es un dispositivo que espera y realiza peticiones de un dispositivo cliente. Cuando ambos se encuentran en la misma red se dice que el servidor es local, cuando se encuentran en redes diferentes y se deben de comunicar a través de internet se dice que el servidor es remoto u *online*. Al momento de ingresar a una página web el explorador de la computadora actúa como el cliente y el dispositivo que devuelve la información de la página deseada es el servidor (Rouse, 2008).

El programa más utilizado para darle la función de servidor a un dispositivo es Apache, lanzado en 1995 por la compañía *Apache Software Foundation* para ser utilizado de forma gratuita. El software utiliza el protocolo *Hypertext Transfer Protocol (*HTTP*)*, el cual se usa en conjunto con el TCP/IP. Mientras que la función principal del protocolo TCP/IP es transportar la información a través de redes, el protocolo HTTP permite encontrar los archivos requeridos utilizando una dirección conocida como localizador de recursos uniforme (URL), ésta es la que comúnmente se conoce como dirección web y su formato se muestra en la Figura 22, donde se observa que incluye el servidor, el nombre del archivo y la ubicación (Laurie & Laurie, 2003) (Gourley, Totty, Sayer, Aggarwal, & Reddy, 2002).

Figura 22. Ejemplo *URLs* (Adsuara, 2013)



Apache permite el acceso remoto a archivos, los cuales comúnmente son *HyperTextMarkupLanguage(*HTML*)* que es el estándar utilizado para crear páginas web, sin embargo también puede devolver imágenes, audio o simplemente el resultado de una operación. Para poder ingresar a los archivos, éstos deben de encontrarse en la carpeta *htdocs*en donde se encuentre instalado Apache (Laurie & Laurie, 2003).

O. PHP

HypertextPreprocessor(PHP) es un lenguaje de código abierto utilizado comúnmente para el desarrollo de aplicaciones web ya que es posible incrustarlo en código HTML. En la Figura 23 se puede observar un ejemplo de cómo integrar estos dos tipos de código en un mismo archivo. El código HTML se escribe de la manera usual sin embargo todo lo que se encuentre dentro de las etiquetas “<?php” y “?>” será interpretado en modo PHP donde es posible tener variables, ciclos, condiciones lógicas y otras características que se acostumbran tener en un lenguaje de programación (phpMyAdminDevelpmentTeam).

Figura 23. Ejemplo de integración de HTML y PHP (phpMyAdminDevelpmentTeam)

```
<!DOCTYPE HTML PUBLIC "-
//W3C//DTD HTML 4.01 Transitional//EN"
    "http://www.w3.org/TR/html4/loose.dtd">
<html>
    <head>
        <title>Ejemplo</title>
    </head>
    <body>

        <?php
            echo "¡Hola, soy un script de PHP!";
        ?>

    </body>
</html>
```

Una de las características del código en PHP es que éste es ejecutado en el servidor, mientras otras opciones como Javascript se ejecutan en el cliente a pesar de estar almacenadas en el servidor. Al momento de ejecutar código PHP el cliente recibe únicamente el resultado de las operaciones ejecutadas, siendo capaz de realizar operaciones complejas en lugar del cliente, ahorrándole capacidad de procesamiento ya que es posible que el cliente realice la petición de manera asíncrona y realizar otras tareas mientras se recibe el resultado de la petición (phpMyAdminDevelpmentTeam).

PHP cuenta con un amplio soporte para conectarse a bases de datos, y esto es utilizado para crear páginas web con la capacidad de alterar o agregar el contenido de las tablas. Entre las posibles extensiones a utilizar para gestionar bases de datos se encuentra MySQL por lo que es posible realizar consultas de este tipo, que en conjunto con las otras capacidades de PHP permite gestionar perfectamente una base de datos desde otro equipo (phpMyAdminDevelpmentTeam).

P. SERVIDORES XAMP

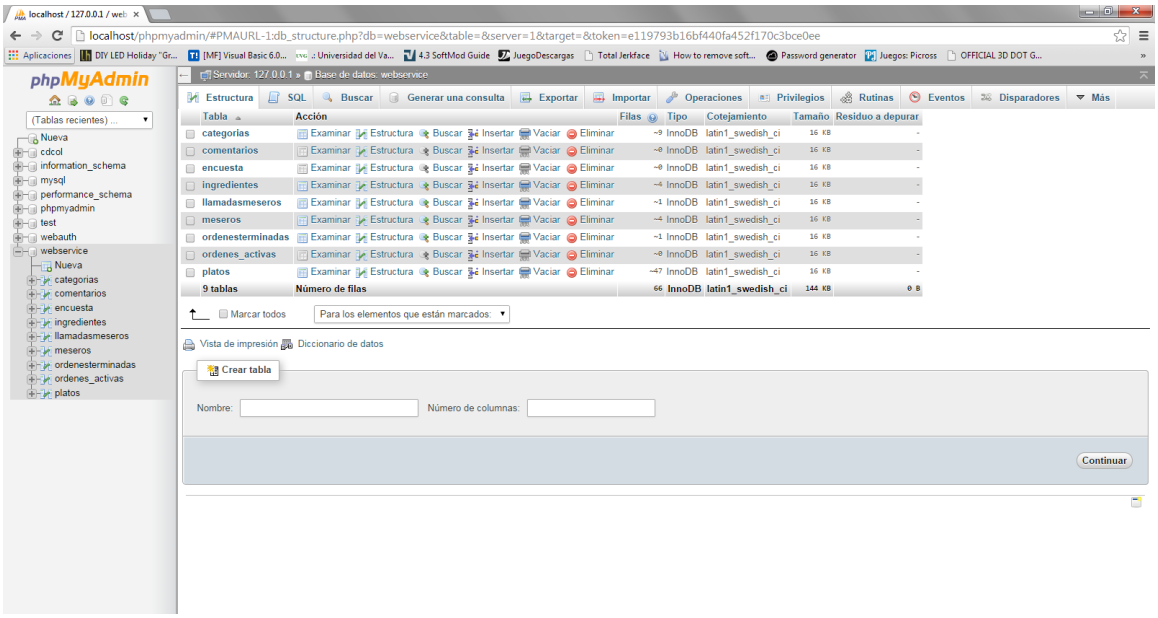
El término servidor XAMP se refiere a un dispositivo que funciona como un servidor y que lo hace utilizando *Apache, MySQL,PERL* o *Python* o *PHP*, por ello su nombre. La primera “X” debe de sustituirse por la primera letra del sistema operativo del servidor, es decir sí se encuentra montado en un sistema con Linux el servidor será un LAMP y sí se encuentra en un sistema con Windows se llamara un servidor WAMP. Gracias a sus componentes los servidores XAMP cuentan con varios puntos positivos como por ejemplo: es gratuito ya que todas sus partes son libres de usar sin costo, puede instalarse en varios sistemas

independiente de su sistema operativo, es posible acceder de forma segura a la información gestionando niveles de control y son sencillos de instalar y configurar (Angoar, 2008).

Estos servidores poseen con una facilidad para modificar bases de datos debido a la relación que existe entre Apache, MySQL y PHP. MySQL es utilizado para dar formato a la base de datos y poder consultas en las tablas utilizando esta sintaxis, mientras PHP tiene la función de modificar los datos y las tablas de la base de datos por medio de las extensiones para realizar consultas en SQL y Apache brinda acceso externo a los archivos PHP (Angoar, 2008).

Varios instaladores incluyen phpMyAdmin que es un software, también gratuito, escrito en PHP, utilizado para gestionar base de datos a través de una interfaz gráfica donde el usuario no necesita saber del lenguaje SQL para crear y modificar tablas, también es posible utilizar SQL directamente si se desea. Es posible modificar los accesos a las bases de datos como por ejemplo agregar o editar usuarios y contraseñas. El acceso al programa es vía un explorador de internet, en la Figura 24 se observa una captura de pantalla utilizando phpMyAdmin a través del explorador *Google Chrome* (phpMyAdminDevelpmentTeam).

Figura 24. PhpMyAdmin desde *Google Chrome*



Q. RASPBERRY PI

La Raspberry Pi es un producto lanzado por la fundación Raspberry Pi en febrero del 2012 y consiste en una tarjeta electrónica de tamaño similar a una tarjeta de crédito con la funcionalidad de una computadora de escritorio. La tarjeta incluye puertos de entradas y salidas para que sea posible conectar un teclado, un ratón y un monitor, para poder utilizarla de la misma manera de la que uno utilizaría una computadora. El enfoque de la fundación Raspberry Pi hacia este producto es que se desea que las computadoras sean accesibles para todas las personas, esto se ve reflejado en el precio ya que la versión más modesta tiene un valor de $25 dólares, mientras que la versión más robusta tiene un costo de $35.16. (Foundation, Introduction)

Figura 25. Raspberry Pi modelo B

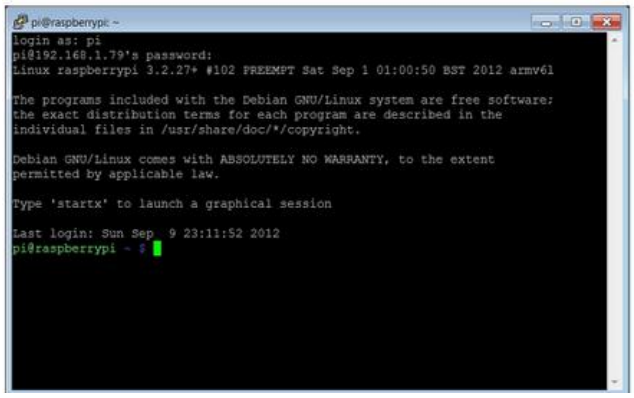
Existen tres modelos de la Raspberry Pi, el modelo A, B y B plus. Las tres versiones cuentan con salida HDMI, puerto para audífonos estéreo, puerto para cámara, puerto micro-USB para alimentar a la tarjeta y un procesador Broadcom BCM2835 100MHz ARM11. Las diferencias más grandes entre modelos consisten en que ambos modelos B cuentan con un puerto Ethernet para conectarse a internet y cuentan con 512 Megabytes de memoria RAM mientras que el modelo A no cuenta con puerto Ethernet, tiene solamente 256 Megabytes de RAM y cuenta con un menor número de puertos USB. Las diferencias entre el modelo B y B plus es que el primero cuenta con 2 puertos USB mientras que el modelo plus tiene 4, el modelo B plus no cuenta con la salida de video análoga que tienen los otros modelos y utiliza una memoria microSD en lugar de una memoria SD de tamaño regular. Actualmente el modelo B plus ha sustituido al modelo B en el mercado ya que el costo es el mismo. (Lawler, 2012) (Foundation, Introduction)

El sistema operativo debe ser almacenado en una tarjeta SD en el caso de los modelos A y B y una microSD en el caso del modelo B plus. Esto se debe a que la Raspberry no cuenta con almacenamiento interno. El sistema operativo soportado es Linux un sistema *open source,* desarrollado por LinusTorvalds

de la Universidad de Helsinki Finlandia. Linux cuenta con varias versiones y la más utilizada en el caso de la Raspberry Pi se denomina Raspbian que es una versión optimizada del sistema operativo Debian optimiza para las características de la Raspberry (Sliever, Figgins, Love, & Robbins, 2009).

Las dos maneras principales de utilizar la Raspberry PI es a través de la interfaz gráfica o la línea de comandos, ambas se pueden utilizar conectando el aparato a un monitor, un teclado y un ratón o utilizando otra computadora conectada en la misma red como entrada y salida de la Raspberry utilizando acceso remoto. Para esta segunda opción se necesita contar con algún software que realice la conexión, es posible utilizar cualquier software que implemente el protocolo *Secure Shell(SSH)*, uno comúnmente utilizado es PuTTY, y solamente es necesario conocer la dirección IP y la credenciales de conexión de la Raspberry para realizar la conexión a la terminal (Golden, 2013).

Figura 26. Terminal de Raspberry Pi desde PuTTY (Golden, 2013)



Desde la línea de comandos es posible actualizar el sistema operativo y descargar nuevas aplicaciones y librerías, como PHP, Apache,MySqly PHPMyAdminpor lo que es posible montar un servidor LAMP en Raspbian. Los archivos del servidor se pueden crear en la Raspberry Pi utilizando el comando "nano" para abrir un editor de texto o pueden transferirse por diferentes medios. Uno de los más prácticos es aprovechar el protocolo de transferencia de archivos (FTP), para esto también es necesario utilizar un software como *Filezilla*para realizar la conexión.El cual permite enviar archivos a través de la red solo con conocer la dirección de la RaspberryPi y sus credenciales de conexión (Golden, 2013).

V. METODOLOGÍA GENERAL

El proyecto comenzó definiendo el área de interés que se tenía para trabajar, en este caso fueron los restaurantes y la forma en la que se llevaban a cabo las ordenes. Con esto en mente se procedió a realizar entrevistas al personal de ciertos restaurantes para obtener información de los métodos utilizados para llevar a cabo la obtención de un pedido y la ruta que lleva éste hasta ser entregado al cliente y pagado. Entre los restaurantes visitados había algunos que ya utilizaban una opción apoyada con tecnología para facilitar sus procesos por lo que se obtuvo información de los problemas que cuentan algunas opciones actuales en el mercado.

Con la información obtenida en la etapa de entrevistas se procedió a definir más concretamente las capacidades del sistema, así como a delimitarlo, también se definieron todos los módulos de manera más concreta. Con el proyecto más definido se procedió a contactar a los gerentes y dueños del restaurante Antic para solicitar su apoyo en la realización del proyecto. Se organizó una reunión donde se explicó a detalle en qué consistía el proyecto y se solicitó permiso para basar el proyecto en las necesidades de su restaurante y poder utilizar su imagen y sus productos en las aplicaciones.

Conociendo las necesidades específicas del restaurante Antic se detallaron más las tareas de cada módulo. Se comenzó a trabajar en cada módulo y se realizaron pruebas periódicamente de las partes del sistema que ya eran funcionales. Al tener el sistema funcionando se realizó una última visita al restaurante Antic para recibir retroalimentación del sistema y de sus capacidades.

VI. *DESARROLLO DE APLICACIÓN EN PLATAFORMA ANDROID PARA EL CONTROL DE COMANDAS EN COCINA Y DESARROLLO DE APLICACIÓN WEB PARA CONTROL GERENCIAL EN EL RESTAURANTE ANTIC*

A. METODOLOGÍA

Este proyecto se inició creando la página web, para lo cual se obtuvo un host gratuito en el que se realizaron las pruebas dado que era necesario manejar una base de datos. Se procedió a realizar borradores de la distribución de la página Web y se inició con la programación de cada una de estas secciones. En la página de inicio, se colocó la función "login", mientras que para las otras secciones, se dejó un espacio especificando que funciones se programarían. Al principio, fue una página simple sin implementar hojas de estilo, por lo que se le dio formato a la página con las opciones estándar de HTML. Se recibieron tutoriales gratuitos para conocer los diferentes lenguajes de programación para páginas web y conocer algunas librerías abiertas al público para implementar mejoras sobre la primera versión.

Al tener estructurada la página web, se procedió a realizar la aplicación; de igual forma se realizaron borradores a mano de la interfaz gráfica, para conocer qué era lo que se debía implementar. Antes de iniciar, se observaron video-tutoriales de internet, para poder crear una aplicación utilizando el SDK de eclipse y aprender sobre el ciclo de vida de una aplicación. Se identificó que se utilizarían dos actividades para realizar esta aplicación, por lo que se creó la interfaz gráfica de ambas actividades para poderlas implementar en cada clase y crear cada uno de los objetos (botones y cuadros de texto). Se definió una función para cada uno de los objetos, que se ejecutaría al ser presionados. Se implementaron las clases que se comunican con la base de datos, pero invocando los métodos una única vez, sin realizar actualizaciones. También se investigó sobre multi-tarea y "Threads", para poder realizar una actualización de las órdenes que el mesero ingresa, revisando constantemente en las dos actividades si hay nuevas órdenes.

Se realizaron pruebas con la integración de la aplicación para meseros y la base de datos local. Se cambiaron los "layouts" de cada actividad para que la aplicación pueda ser utilizada en "tablets" con diferentes dimensiones. Por último se implementó el envío de las órdenes listas a la base de datos, para estar disponible al mesero. La aplicación del bar tiene el mismo comportamiento que la de la cocina, la única diferencia es una variable que indica el lugar al que llegó la información o de dónde se está enviando.

B. RESULTADOS

1. Página WEB

Figura 27. Página de inicio.

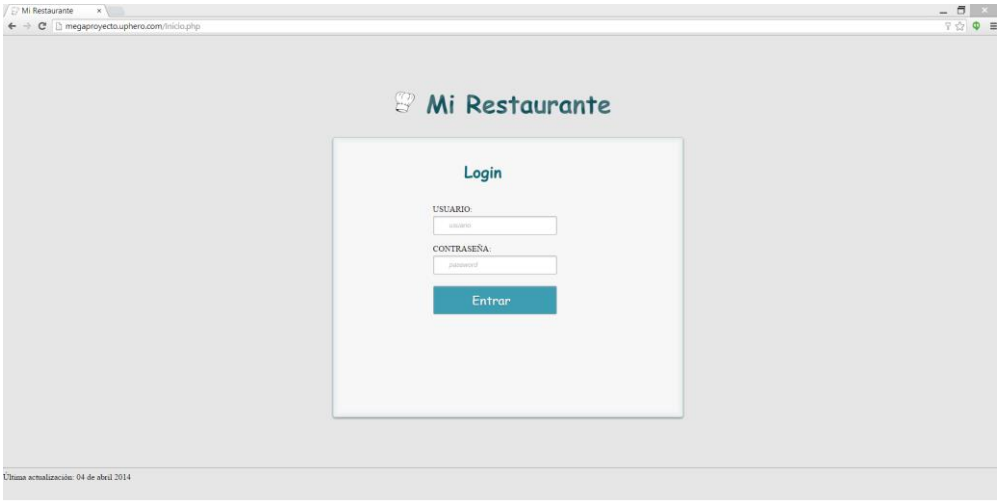


Figura 28. Formato para iniciar sesión con verificación de datos incorrectos

Figura 29. Formulario para iniciar con mensaje de sesión terminada.

Figura 30. Página de inicio con instrucciones de uso de la página web.

Figura 31. Página de inicio para estadísticas.

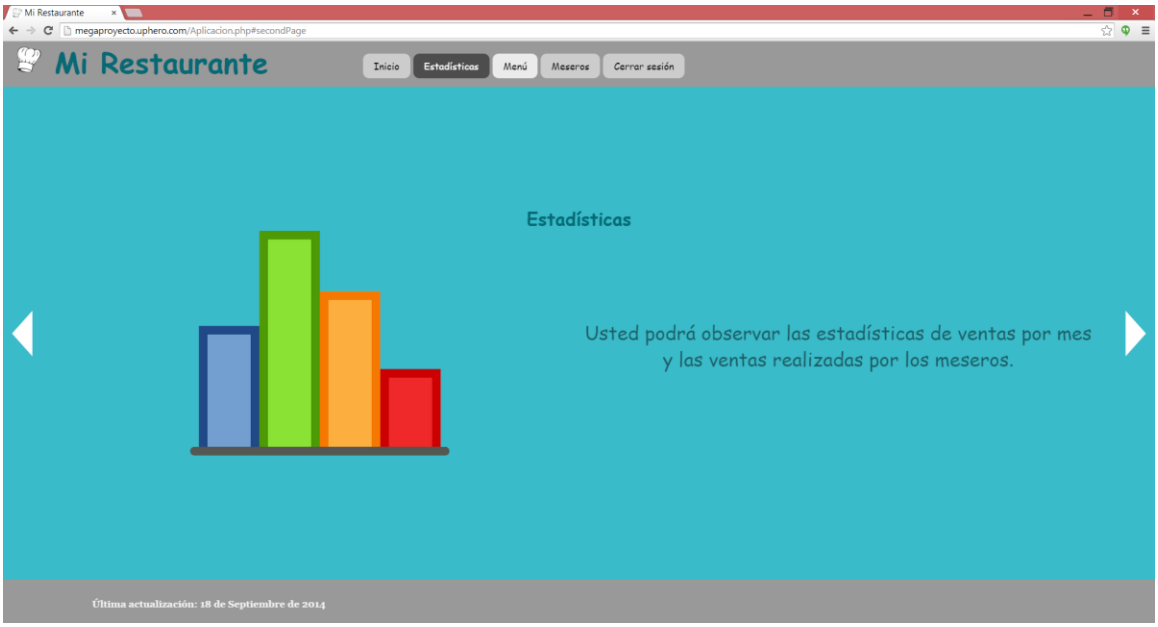


Figura 32. Página para seleccionar y ver gráficas.

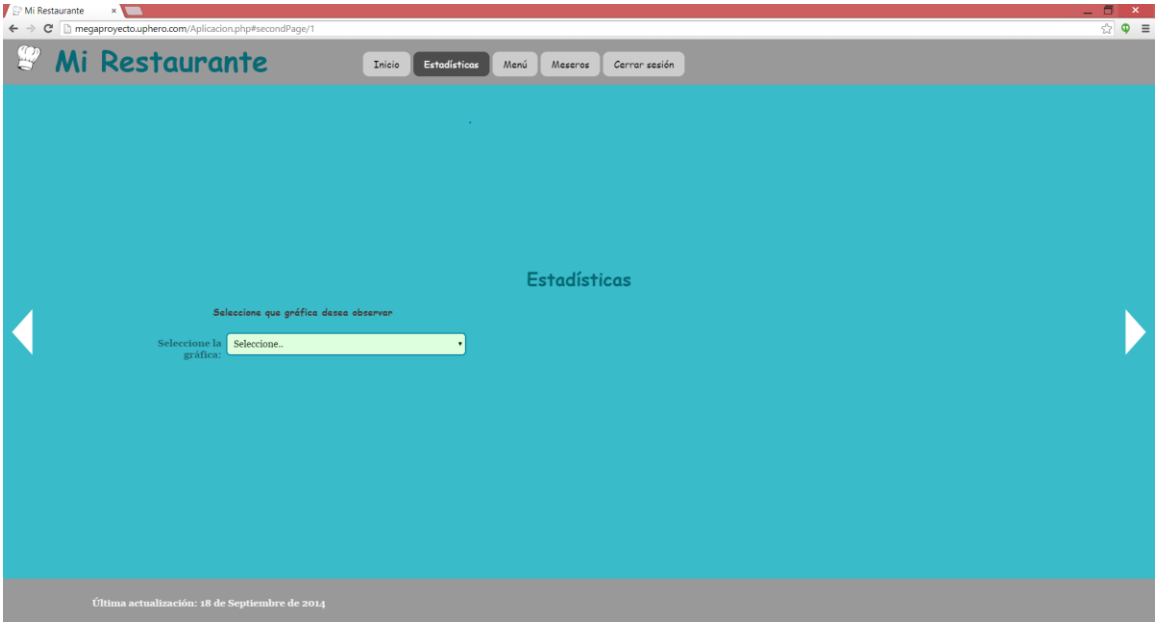


Figura 33. Selección y visualización de la gráfica.

Figura 34. Otras opciones de estadísticas.

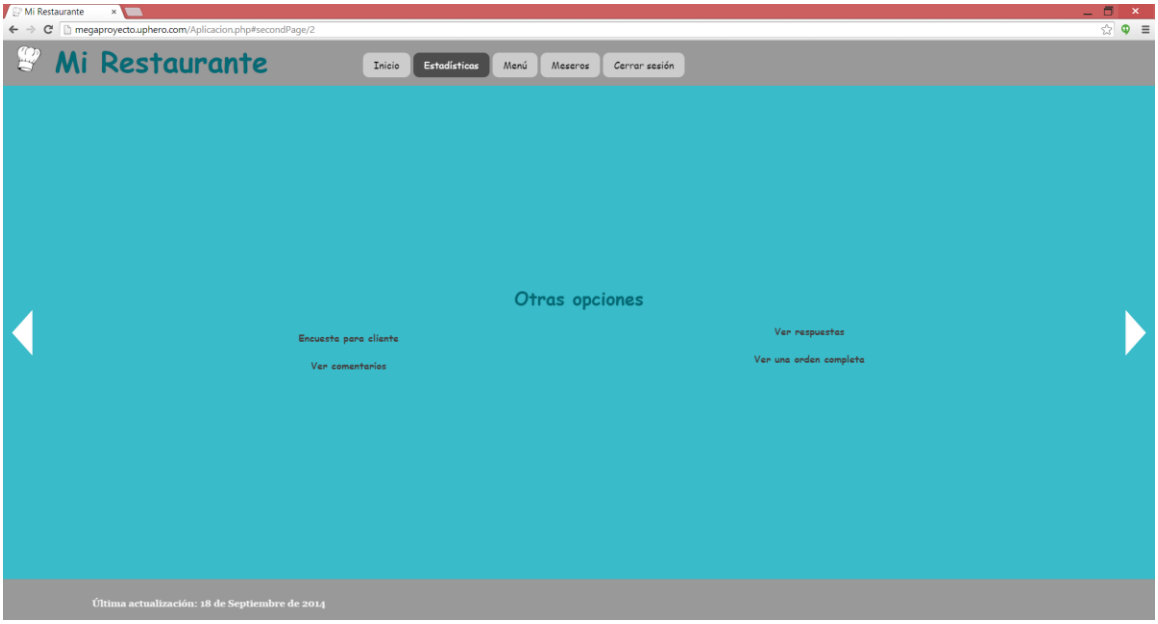


Figura 35. Formulario para modificar encuesta, agregar nuevas preguntas, inhabilitar y habilitar preguntas.

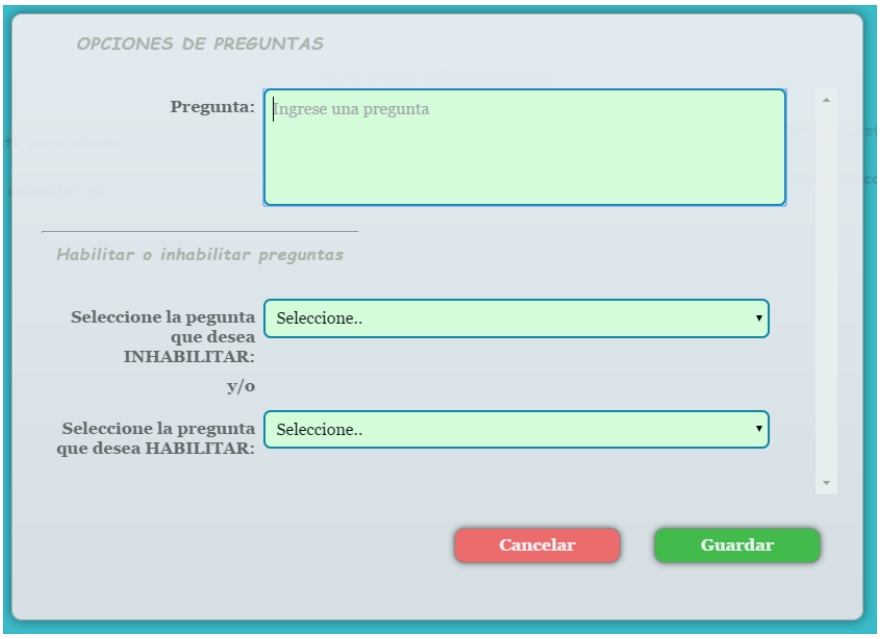


Figura 36. Formulario para observar el número de estrellas dadas por los clientes a cada pregunta.

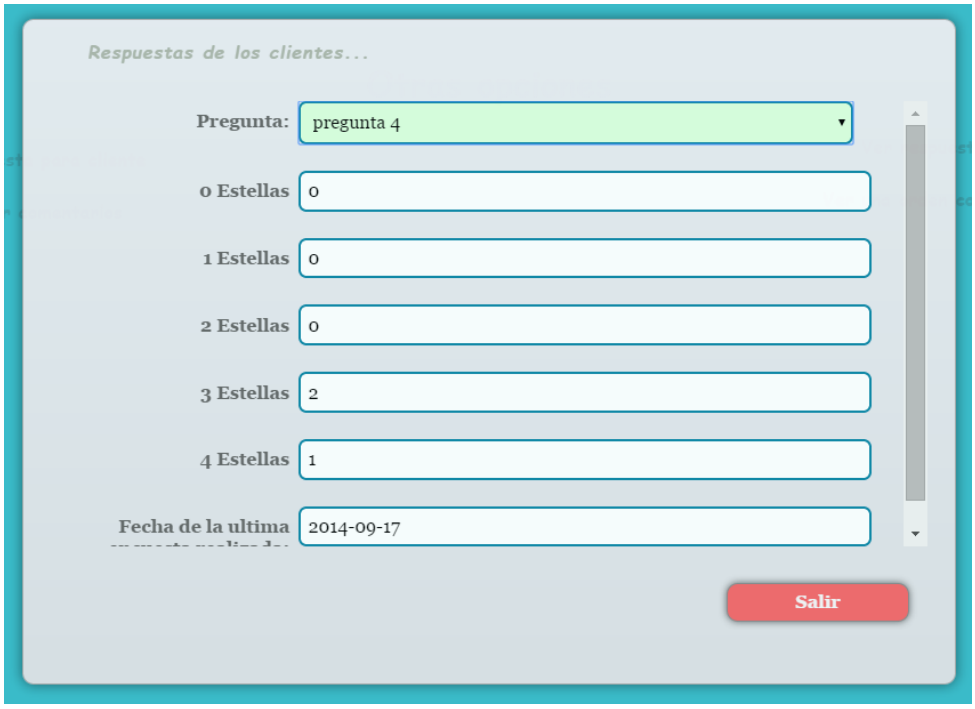


Figura 37. Formulario para observar los comentarios de los clientes.

Figura 38. Formulario para observar una orden completa.

Figura 39. Página principal para las opciones de menú.

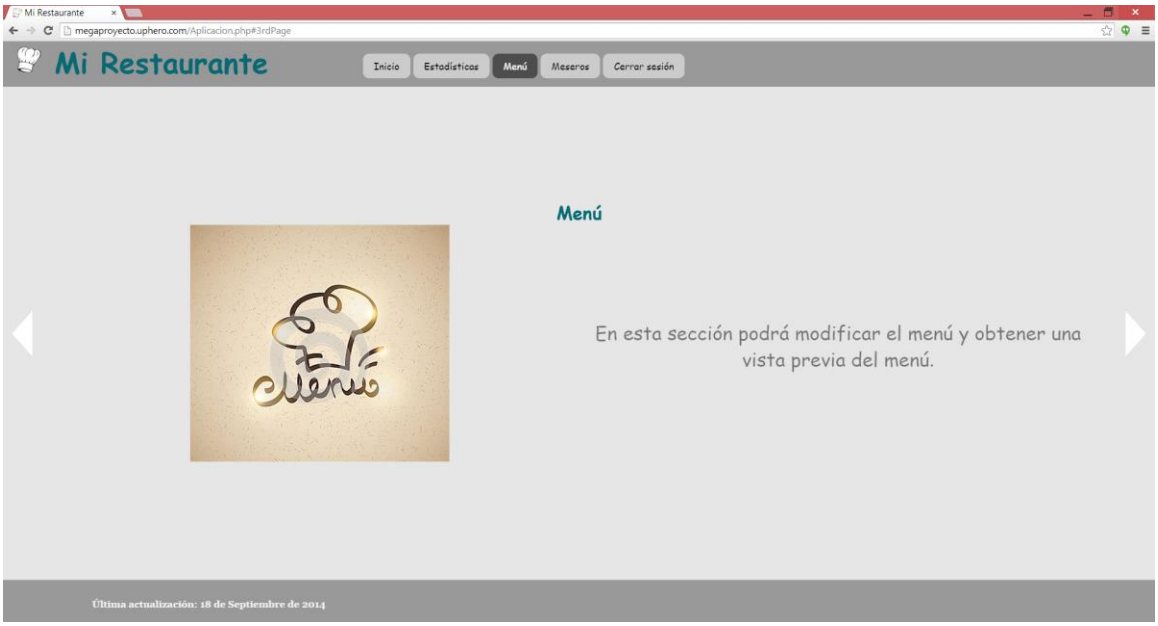


Figura 40. Página con las diferentes opciones de modificaciones al menú.

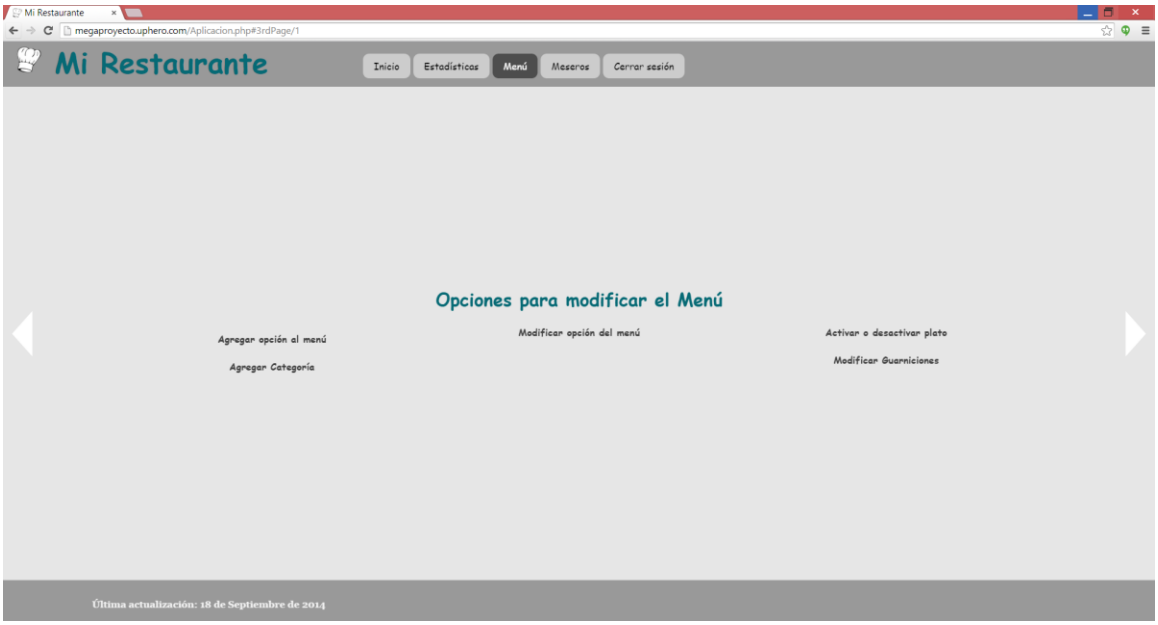


Figura 41. Formulario para incluir un nuevo plato a la base de datos

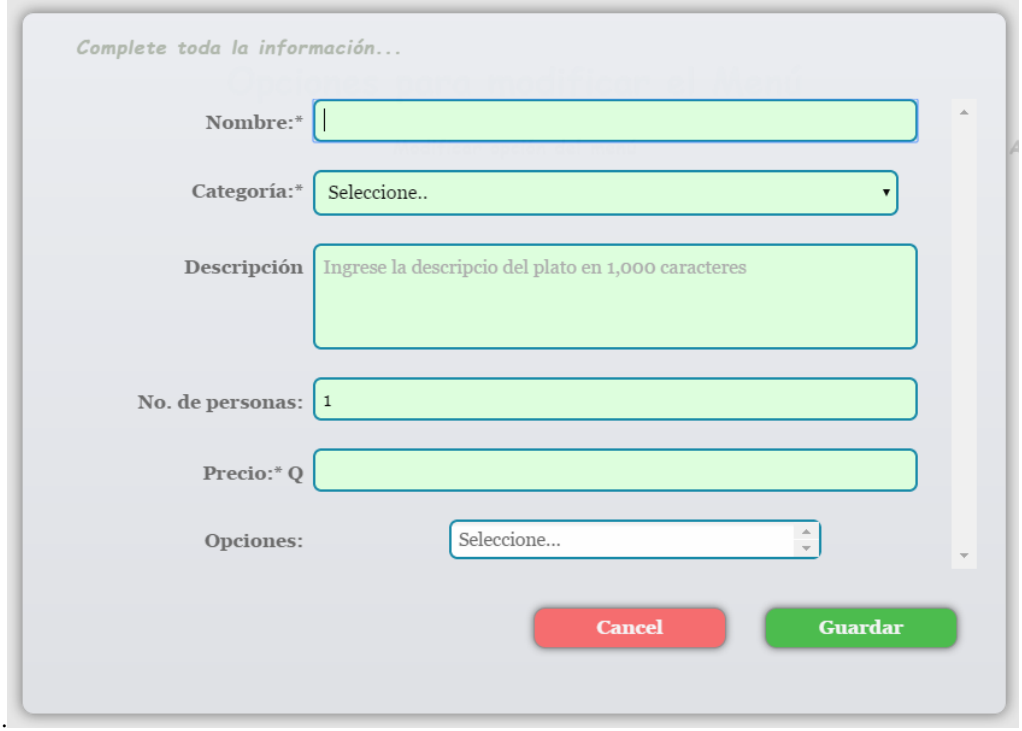


Figura 42. Formulario para realizar modificaciones a un plato específico

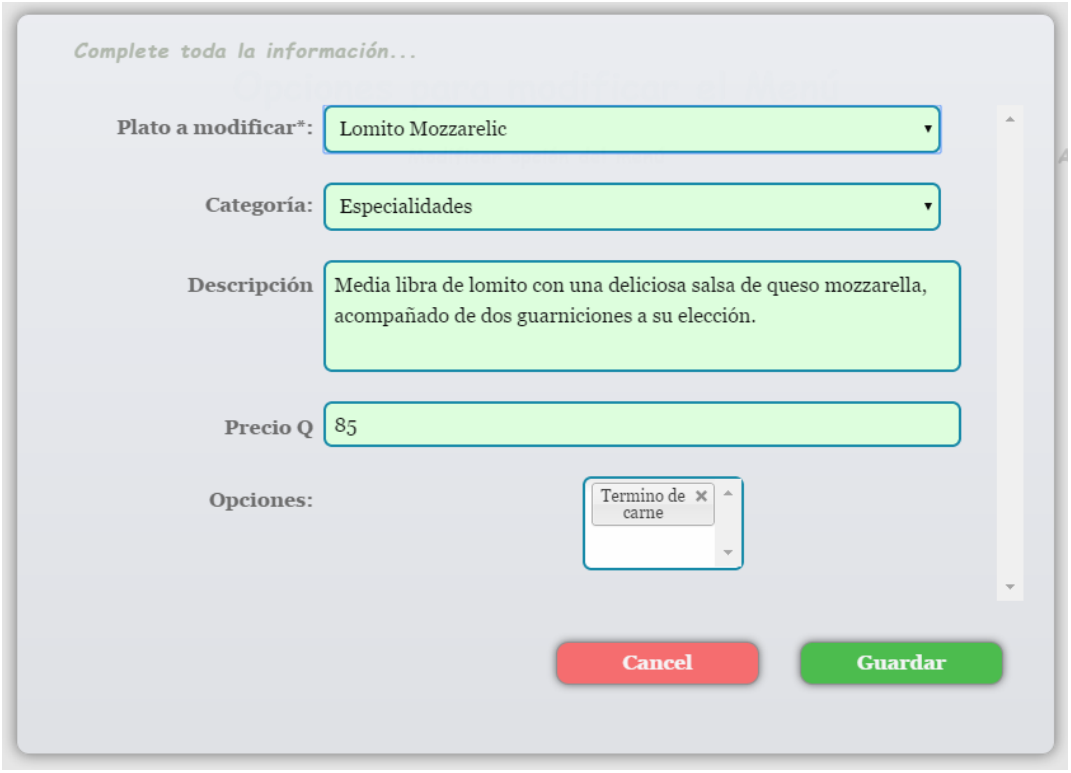


Figura 43. Formulario para desactivar o activar un plato del menú.

Figura 44. Formulario para agregar una nueva categoría, indicando si debe ir a bar o a cocina.

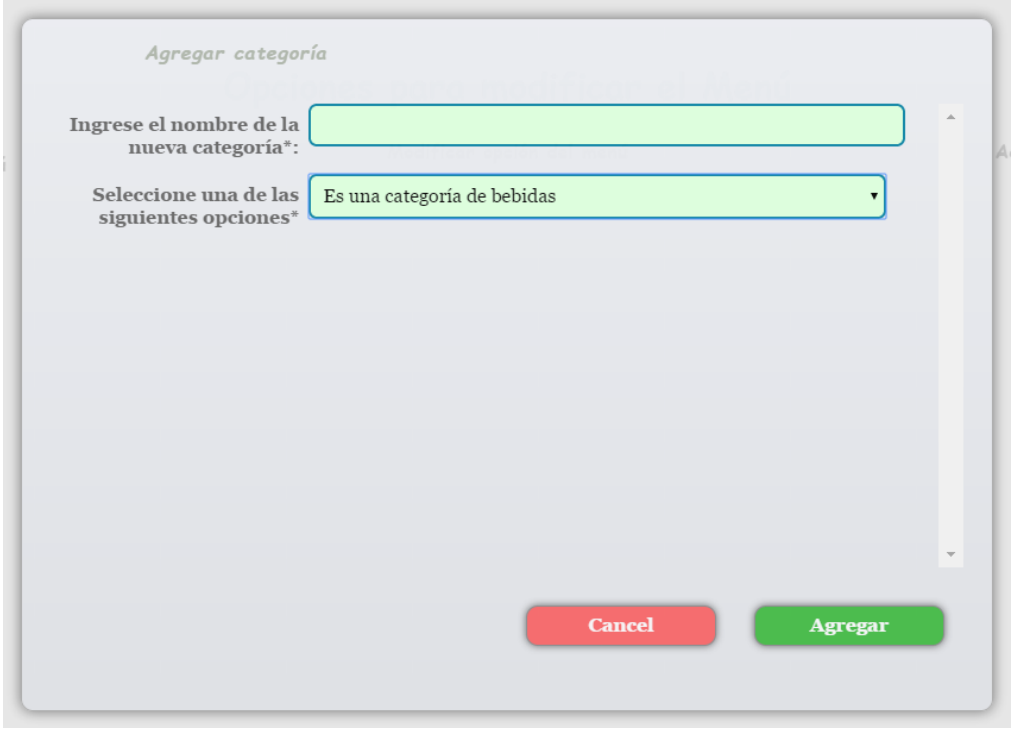


Figura 45. Formulario para agregar, inhabilitar y habilitar guarniciones.

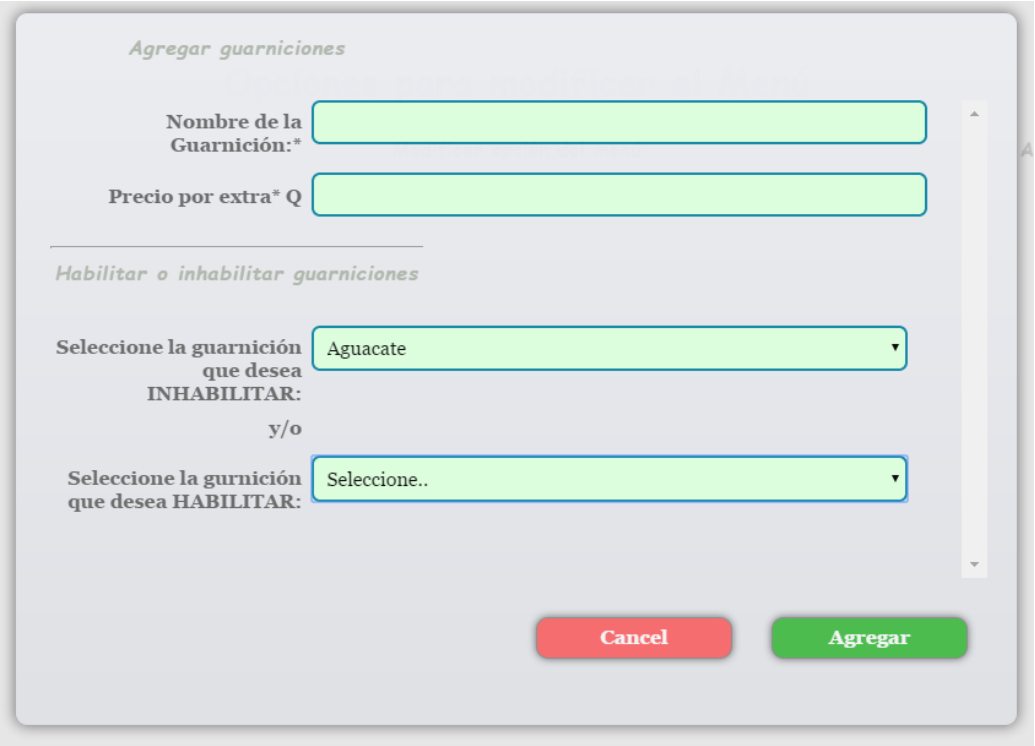


Figura 46. Página principal de opciones de meseros.

Figura 47. Página con las diferentes opciones de meseros.

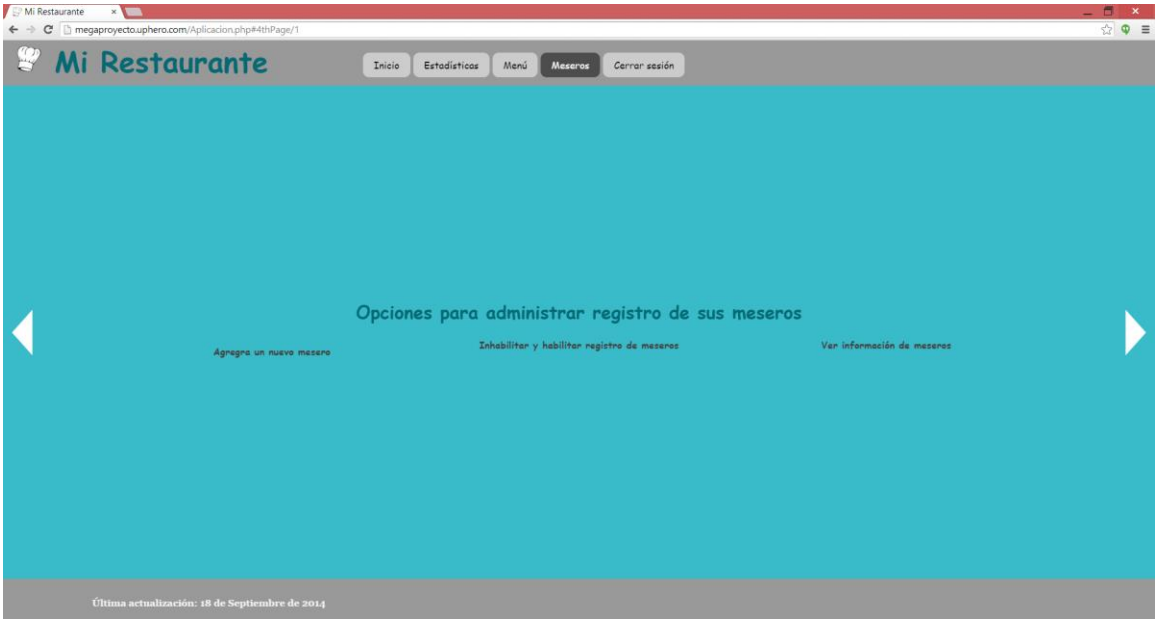


Figura 48. Formulario para ingresar un nuevo mesero a la base de datos.

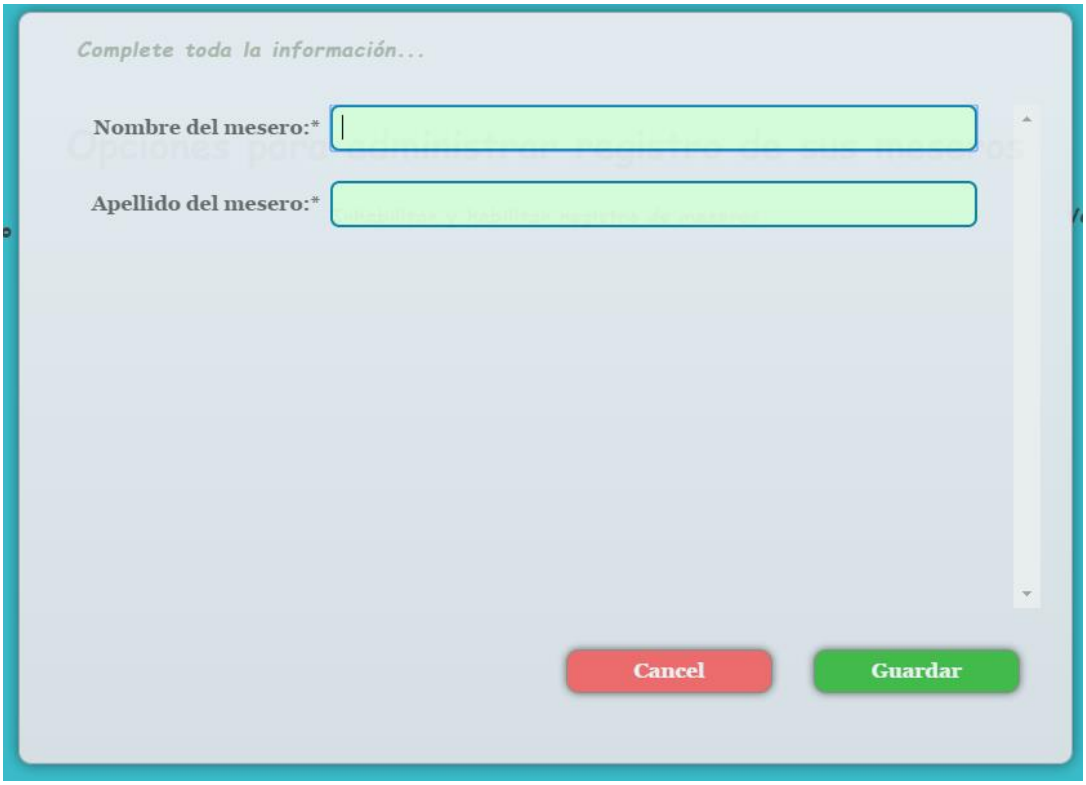


Figura 49. Formulario para habilitar o inhabilitar meseros.

Figura 50. Formulario para observar la información de un mesero.

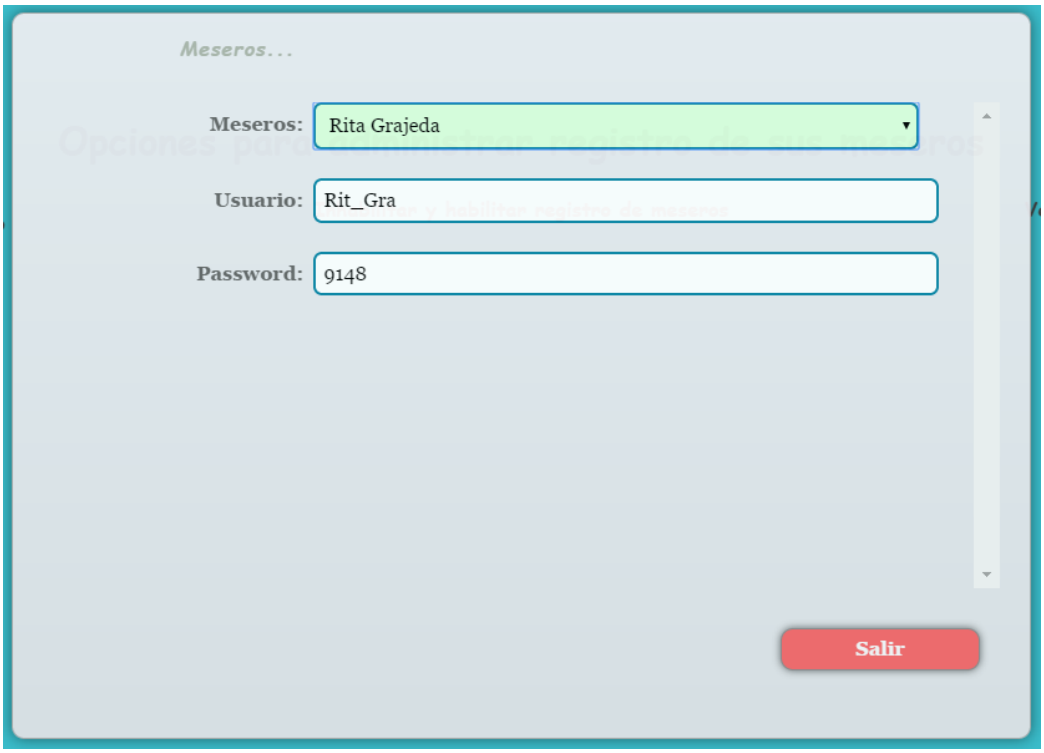
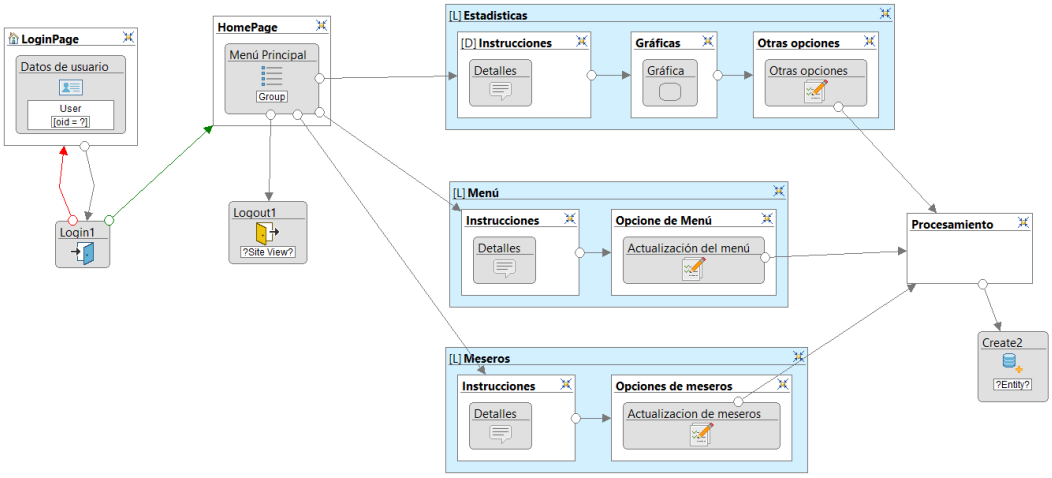


Figura 51. Diagrama de la página web



2. Aplicación Android para cocina y bar

Figura 52. Actividad principal de la aplicación Android.

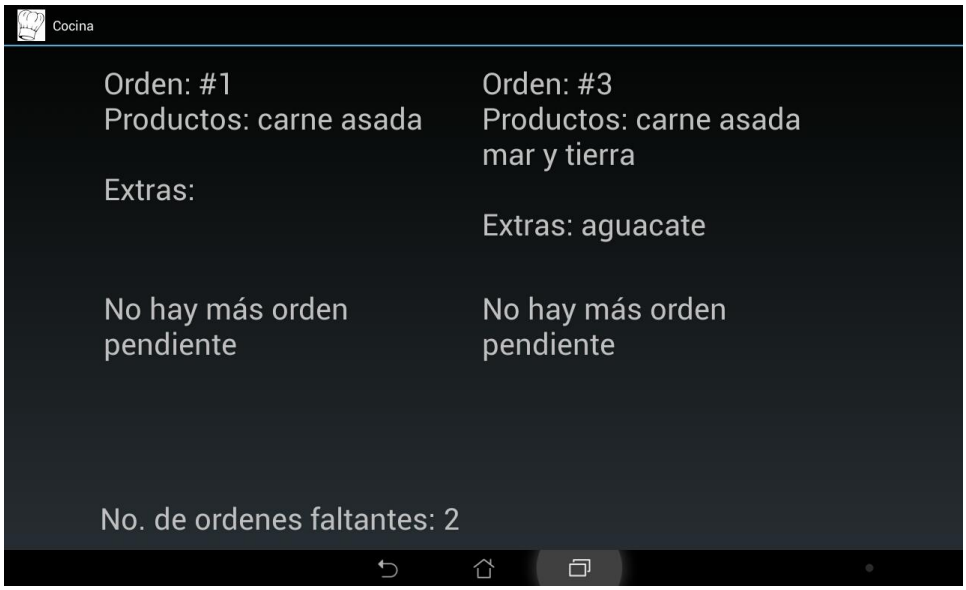
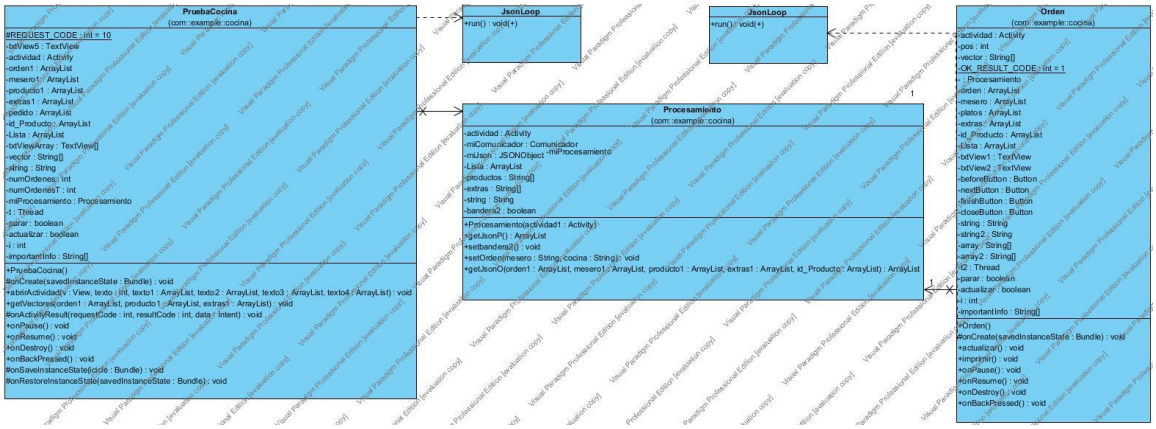


Figura 53. Segunda actividad de la aplicación Android.

Figura 54. Diagrama de clases aplicación Android



C. ANÁLISIS DE RESULTADOS

1. Página Web. Para mantener la seguridad de la información que se maneja, fue necesario implementar el ingreso a la aplicación por medio de un usuario y contraseña registrados. En la Figura 27 a la Figura 29 se observa el sistema para iniciar sesión, notificaciones cuando el usuario o clave sean incorrectas y para indicar se ha terminado un sesión. El usuario y contraseña fueron validados con la información en la base de datos, como toda la información que se desplegará en la página web será extraída mediante mySQL de la base de datos, al igual que la nueva información (nuevas preguntas, platos o meseros) que desea ingresar al sistema. En la Figura 30 se creó una página de inicio para desplegar las instrucciones de cómo navegar en la página web.

De la a la Figura 38 se puede observar la sección de “Estadísticas”. Específicamente en la Figura 31 muestra el acceso a la sección de estadísticas que da una breve explicación del contenido. En la Figura 32 se encuentra el formulario y el espacio para generar diferentes estadísticas, que se pueden apreciar en la Figura 33. En la Figura 34 se encuentran otras opciones para observar el desarrollo del restaurante, al seleccionar cada una de estas se desplegará un formulario. En la Figura 35 se puede observar que se pueden incluir nuevas preguntas a los clientes, habilitar o inhabilitar una pregunta según le convenga al restaurante. En la Figura 36 se puede observar que se mostrar el número de personas que ha elegido una cantidad de estrellas por cada pregunta y la última fecha en la que la respondieron. En la Figura 37 se pueden observar los diferentes comentarios que han dejado los clientes según el número de orden. La última opción definida para las estadísticas se puede observar en la Figura 38 donde se despliega el detalle completo de la orden de la que se desea obtener la información.

De la Figura 39 a la Figura 45 se puede observar la sección de “Menú”. En la Figura 39 una introducción a esta sección, en la Figura 40 se encuentran todas las opciones de esta sección, agregar platos, modificar platos, habilitar o inhabilitar platos, crear nuevas categorías para el menú, modificar guarniciones. En la Figura 41 se observa el formulario para agregar un nuevo plato al menú, con los campos necesarios, sin ser todos los campos obligatorios. En la Figura 42 se puede realizar modificaciones a los platos, al seleccionar uno, se despliega toda la información guardada en la base de datos. En la Figura 43 se pueden activar o desactivar platos, ya sea si era un plato de la temporada, o cualquier otra causa. En la Figura 44 se encuentra el formulario para agregar nuevas categorías como lo son “Postres”, “Platos fuertes” entre otros, es necesario indicar si es una categoría de bebidas o no, para mandar cualquier orden de esa sección a bar o a cocina. Y en la Figura 45 se tiene las opciones de las guarniciones, agregar una nueva, inhabilitar o habilitar una ya existente.

Las opciones de inhabilitar o habilitar un plato o una guarnición pueden ayudar a que los clientes no ordenen alguna opción que no esté disponible, y el mesero pueda notificarle al momento de ordenar, que esa opción no está disponible.

De la Figura 46 a la Figura 50 se puede observar la sección de “Meseros”. En la Figura 46 está la página principal de la sección de “Meseros”. En la Figura 47 se encuentran todas las opciones de esta sección. En la Figura 48 está el formulario para ingresar nuevos meseros a la base de datos, este formulario generara el usuario y la contraseña con la que el mesero puede ingresar a la aplicación de meseros. En la Figura 49 se pueden inhabilitar o habilitar meseros de la base de datos. Y la última opción de la sección de meseros es observar toda la información del mesero, nombre, apellido, usuario y contraseña que se puede observar en la Figura 50.

2. Aplicación Android. En la Figura 52 se puede observar la interfaz gráfica de la primera actividad, en donde se despliega un resumen de cuatro de las órdenes pendientes. Para observar la lista detallada de platos, se puede seleccionar una orden en específico que lleva a la actividad 2 en la Figura 53. En esta última se tiene la opción de regresar a ver el resumen de órdenes, cambiar entre órdenes y terminar una orden para poder notificar que el mesero debe de llegar por la orden. En la Figura 54, es posible visualizar la relación que tienen las clases entre sí, la clase "PruebaCocina" es la actividad 1, la clase "Ordenes" es la actividad 2, cada una de estas tiene un objeto "Procesamiento" el cual obtiene los datos de la base de datos. Cada clase tiene un "JsonLoop" donde se crea el "Thread" para revisar constantemente la base de datos.

D. CONCLUSIONES

- Se logró implementar tres diferentes opciones para modificar a través de la página web el menú del restaurante, se puede incluir un nuevo plato al menú, modificar un plato y activar o desactivar un plato.

- En la sección de estadísticas de la página Web, se desplegaron gráficas del total de ventas por mesero al año, al mismo tiempo se pueden observar la calificación dada por los clientes en las diferentes preguntas personalizada por el administrador del restaurante, como un mecanismo de control de calidad.

- Para asegurar la interacción con el cliente, en la sección de estadísticas se muestran los comentarios que estos dejan para retroalimentar los servicios prestados por el restaurante.

- Por medio de la implementación de "threads" se realizó un proceso activo siempre que la aplicación estuviera corriendo, de esta forma se verifica constantemente si en la base de datos se encuentran nuevas órdenes pendientes para cocina o bar.

- En la interfaz gráfica de la aplicación para cocina y bar, se implementó un botón para que el cocinero indicara cuando una orden está lista.

E. RECOMENDACIONES

- Implementar una mayor cantidad de estadísticas, para que el administrador pueda tener un mayor control de su restaurante, como la recurrencia de venta de un plato o la cantidad de ordenes tomas por un mesero.

- Implementar en la página web una opción de vista previa del menú, es decir un gráfico de la forma que tendrá la aplicación del mesero y el cliente. De esta forma el administrador puede realizar los cambios necesarios sin necesidad de descargar la aplicación en una Tablet o teléfono inteligente.

- Implementar más campos para los meseros, como la fecha de contratación, fecha de cumpleaños, para tener toda la información necesaria en la base de datos.

- Implementar el manejo de inventario, de esta forma poder indicar si el plato o bebida está disponible o no a los clientes y para que el administrador pueda considerar estos números al momento de abastecer la bodega.

- Incluir notificaciones con sonidos al recibir una nueva orden, de modo que no sea necesario estar pendiente de la aplicación y de esta forma no mantener desbloqueada la Tablet todo el tiempo, para consumir menos energía.

VII. DESARROLLO DE UNA APLICACIÓN EN LA PLATAFORMA ANDROID PARA GENERACIÓN DE COMANDAS Y DESARROLLO DE UNA APLICACIÓN PARA LOS CLIENTES DEL RESTAURANTE ANTIC

A. METODOLOGÍA

1. Investigación. Para el desarrollo del proyecto el primer paso fue investigar acerca de las aplicaciones similares que se estuvieran implementando en otros restaurantes en distintas partes del mundo. Luego de realizar esto se estableció una idea de los alcances y tareas que debía ejecutar el proyecto. Posteriormente se investigó los distintos entornos de desarrollo que existen para programar aplicaciones Android, entre los más populares se encontró Eclipse y Netbeans; sin embargo existe mucho más soporte para Eclipse así que se decidió utilizar dicho IDE. Se investigaron también las consideraciones que se deben tener en cuenta al realizar un programa para *tablets* y para teléfonos inteligentes, siendo de mucho interés el posicionamiento de los componentes gráficos en la pantalla con el objetivo de que siempre se vieran correctamente posicionados respecto a las dimensiones de la pantalla del dispositivo. Se investigó la interacción de las aplicaciones con un servidor, ya sea en línea o local.

El proceso de investigación no se limitó solamente al inicio del proyecto sino que se continuó en la fase de desarrollo de aplicaciones.

2. Diseño. Luego de las investigaciones, el segundo paso que se llevó a cabo para la realización del proyecto fue definir los diagramas de flujo, tanto para la aplicación de meseros como para la de clientes. En este proceso se especificó qué información se debía pedir a los usuarios, qué botones debían presionar, cómo se iba a mostrar la información, qué restricciones se iban a tomar según los usuarios. Y a medida que se fue avanzando se fue especificando más a fondo cada tarea.

Posteriormente se organizó por clases lo que los programas debían realizar y se realizó los diagramas UML de ambas aplicaciones; esto a partir de los diagramas de flujo. Esta parte del diseño se fue restructurando durante el desarrollo de las aplicaciones ya que si se definía una nueva clase se volvió a organizar el diagrama UML.

3. Desarrollo de aplicaciones. Esta etapa se inició investigando cómo se realizaba un *Hello World*, un programa donde solamente se muestra un mensaje en pantalla; luego se integró el uso de botones para cambiar el mensaje mostrado. Así sucesivamente se fueron implementando diferentes componentes

gráficos para llevar a cabo las distintas tareas. Luego de ya conocer los componentes básicos se empezó por desplegar las categorías del menú y luego del despliegue de categorías se implementó el despliegue de platos por categoría; luego se implementaron las modificaciones a realizar en cada plato, y el despliegue en una lista los platos agregados a la orden.

Siguiendo con el desarrollo el siguiente paso fue enviar la información de la orden a la base de datos. Esto se realizó uniendo con el módulo de comunicación y utilizando la biblioteca generada por otro integrante del equipo, dicha biblioteca se encargaba en enviar y recibir un objeto JSON que era el responsable de comunicarse con la base de datos así que mi unión con su módulo solamente se basaba en convertir la información a enviar a un objeto JSON y utilizar una función que se generó en su biblioteca. Se logró enviar y recibir los primeros datos de la base de datos.

Para finalizar con la aplicación de meseros se implementó más de lo aprendido en el desarrollo anterior para poder obtener el usuario del mesero corroborando su contraseña en la base de datos, también se obtuvo una orden final a pagar y se envió a la base de datos la forma de pago de la misma.

Para la realización de la aplicación de clientes se investigó más detenidamente los *Intents* ya que componen gran parte las tareas a ejecutar en la aplicación previamente dicha. Se implementó cómo realizar llamadas, enviar correos electrónicos, entrar a páginas web, y además a realizar una ruta desde donde se encuentra el usuario por GPS hace el restaurante Antic. El obtener datos del menú ya no significó investigación para esta aplicación, sin embargo en esta app sí se implementaron diseños realizados en Photoshop para generar una interfaz gráfica más vistosa para los clientes.

4. Pruebas de unión de modulos. Como último paso se unificaron las aplicaciones a los demás módulos, y se realizaron las pruebas de funcionamiento. Se verificó con el módulo de comunicación inicialmente si se recibían correctamente todos los datos del menú. Posteriormente se corroboró que los datos recibidos fueran los mostrados en pantalla. Luego de poder recibir la información el siguiente paso a desarrollar fue enviar los datos a la base de datos: primero se debía enviar las órdenes, si eran para bar o cocina, el nombre de la cuenta y qué platos contenía. Al lograr enviar esta información se envió la información de pago.

La primera prueba que se realizó en la unión con el módulo de comunicación fue verificar que el objeto JSON se recibiera correctamente y que el formato que se estableció previamente fuera respetado por ambos módulos, es decir que los parámetros que se buscan en el objeto JSON tuvieran el mismo nombre para poder ser identificados. Como segunda prueba que se realizó fue la verificación que todos los datos requeridos por un programa fueran devueltos por la base de datos, en este punto se identificó un problema con los contadores que cada módulo los trabajaba independientemente entonces se perdieron algunos datos,

así que para corregirlo se determinó que el conteo de platos fuera liderado solamente por el módulo de comunicación.

Posteriormente se realizó las pruebas de envió de datos para determinar qué platos con sus respectivas modificaciones fueron ordenados en cada cuenta y si se debía ir a bar o a cocina; así que primero se verificó que el JSON enviado cumpliera con los identificadores de parámetros prestablecidos así como se realizó al recibir la información; y luego se verificó que toda la información almacenada en la base de datos tuviera relación con todos los cambios y pedidos realizados a la orden en la aplicación de los meseros. El mayor problema que se presentó en dicha unión fue identificar y enviar correctamente las modificaciones en cada plato así que se decidió por que la aplicación de meseros enviara todas las modificaciones como una cadena de texto y no como identificadores como previamente se había considerado.

Luego de realizar ambas pruebas de enviar y recibir información con el módulo de comunicación se siguió todo el proceso realizando dicho procedimiento para los distintos requerimientos de ambas aplicaciones. En la aplicación de mesero se verificó recibir la orden que se debe pagar y enviar la información de pago; además de la verificación de las contraseñas de los meseros. Para la aplicación de clientes se verificó recibir la información del menú tal y como se realizó en la aplicación de meseros, además obtener la información de las preguntas que los del restaurante Antic desean hacerles a los clientes del mismo, junto con esto se envió la información a la base de datos donde de lo respondido por los clientes.

B. RESULTADOS

1. APLICACIÓN DEL MESERO

Figura 55. Diagrama de flujo principal de app del mesero

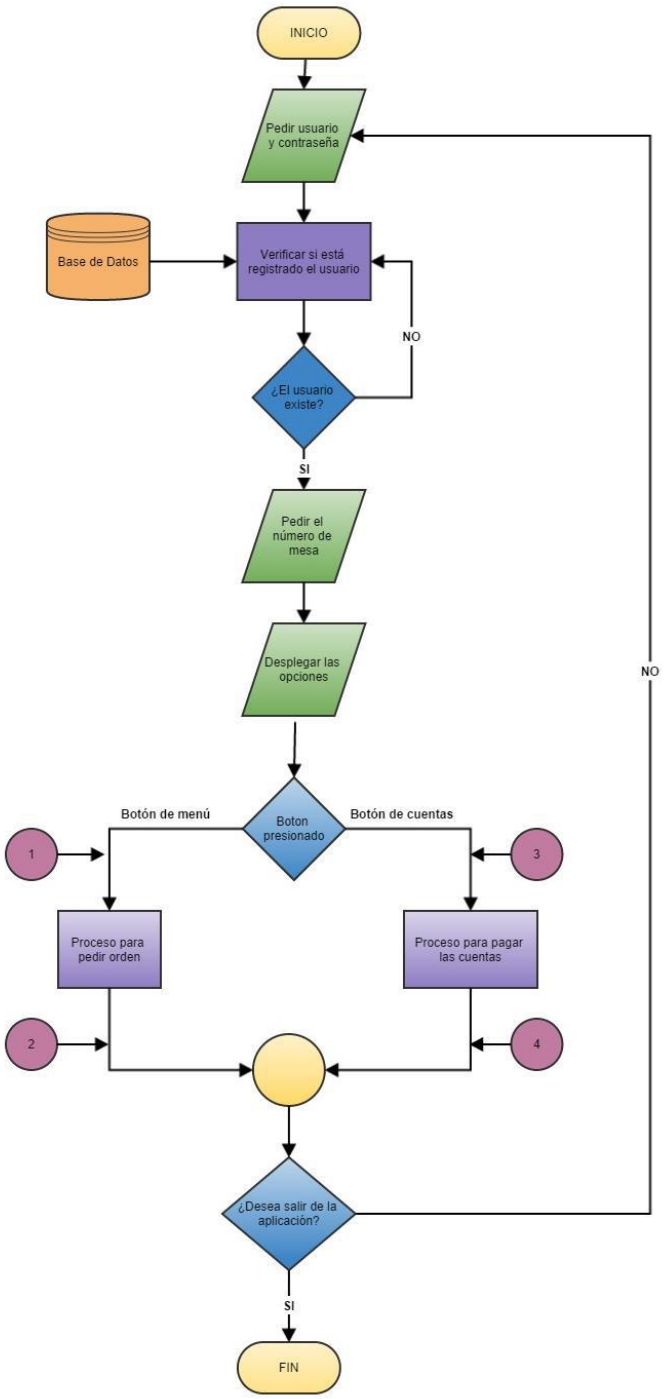


Figura 56. Diagrama de flujo del proceso de pedir una orden

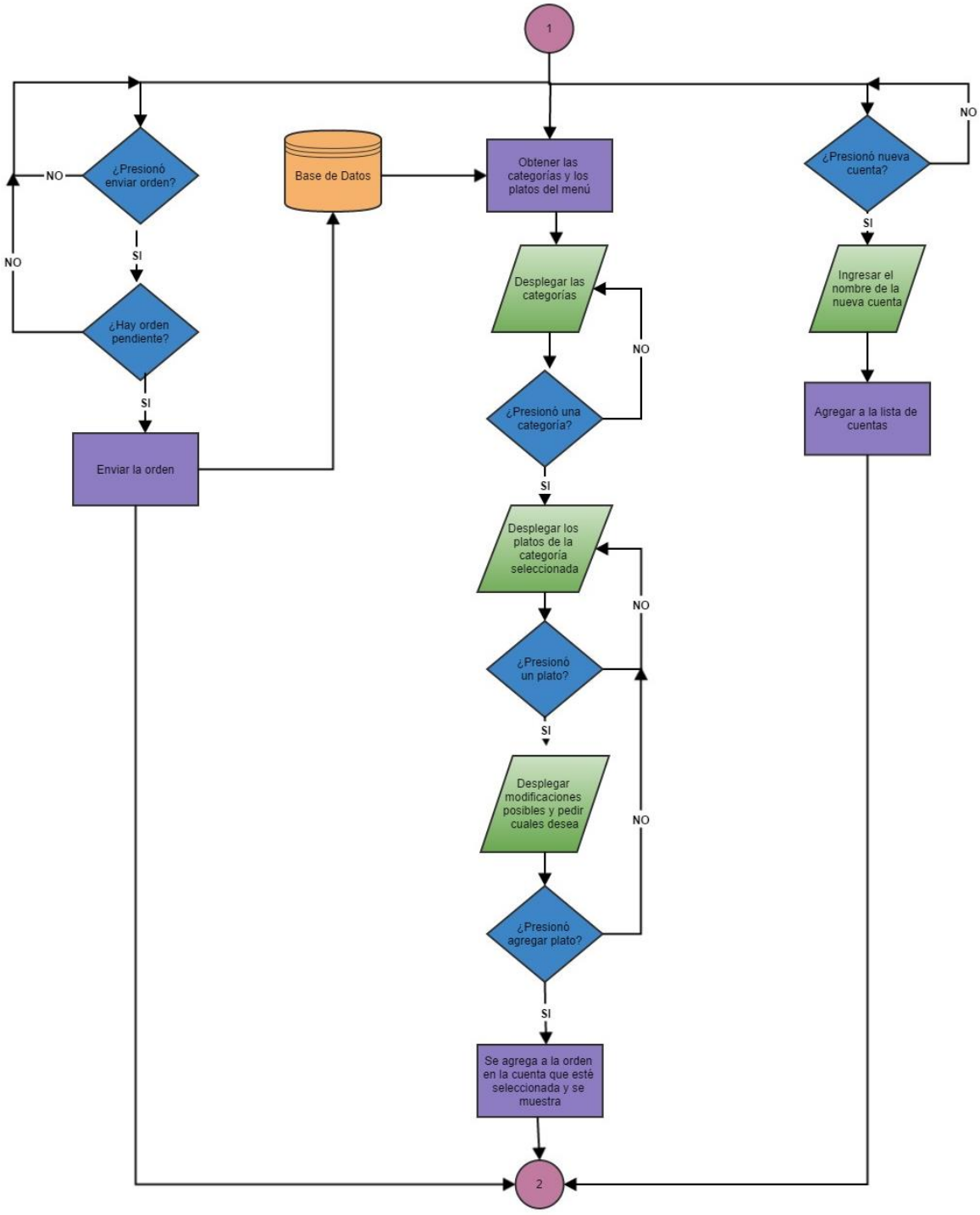


Figura 57. Diagrama de flujo del proceso de pagar cuentas

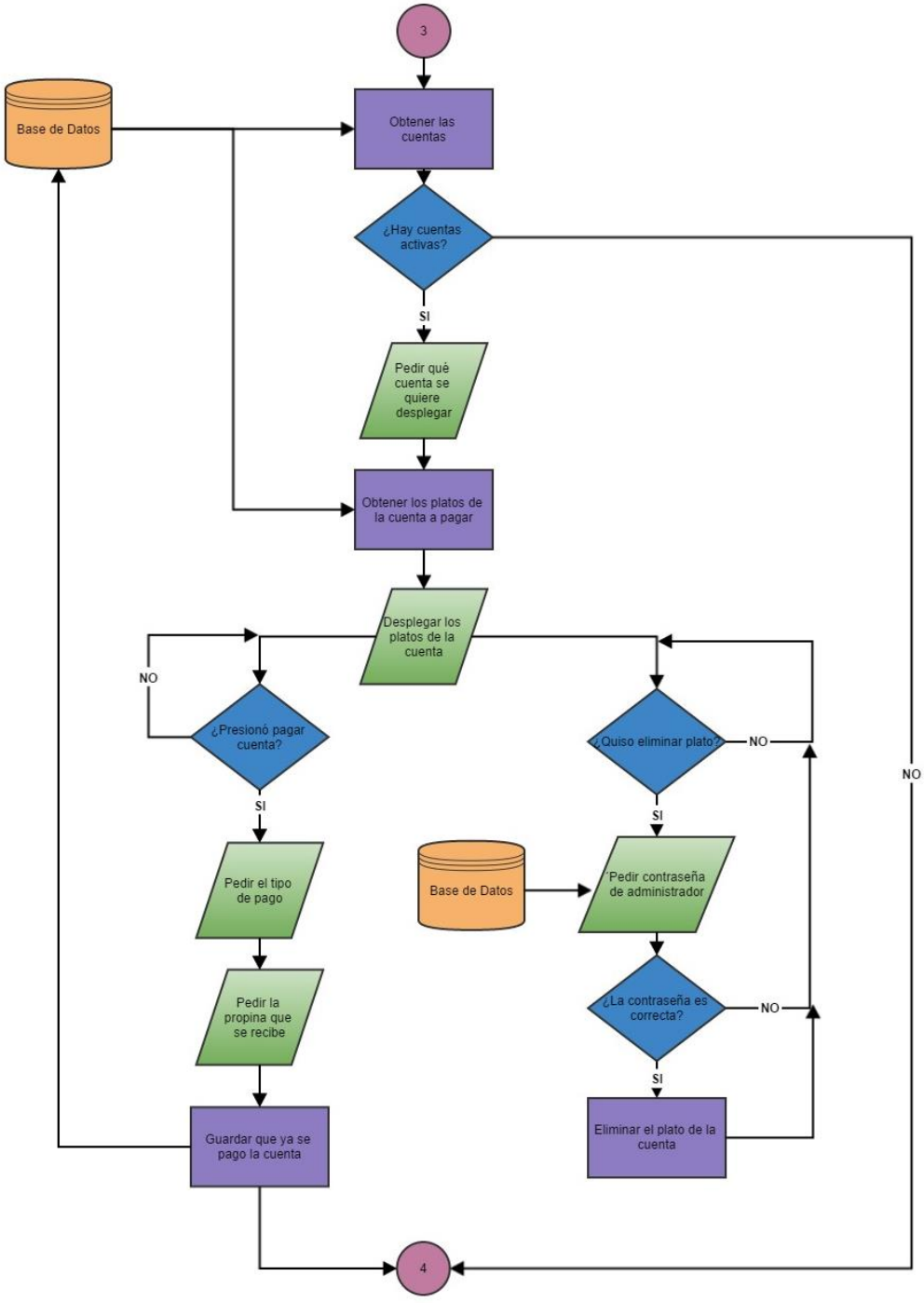


Figura 58. Diagrama UML de aplicación de meseros

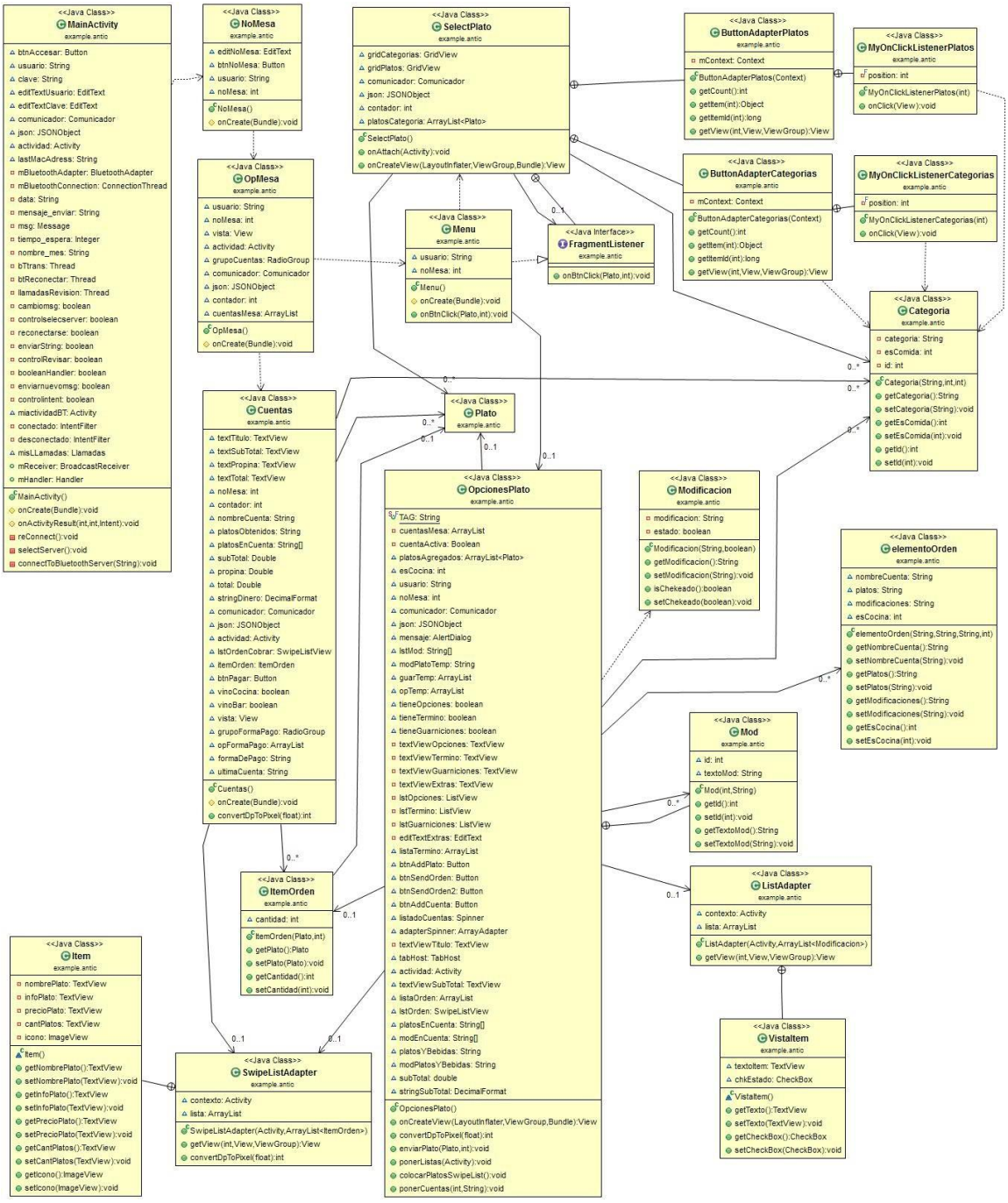


Figura 59. Pedir usuario y contraseña

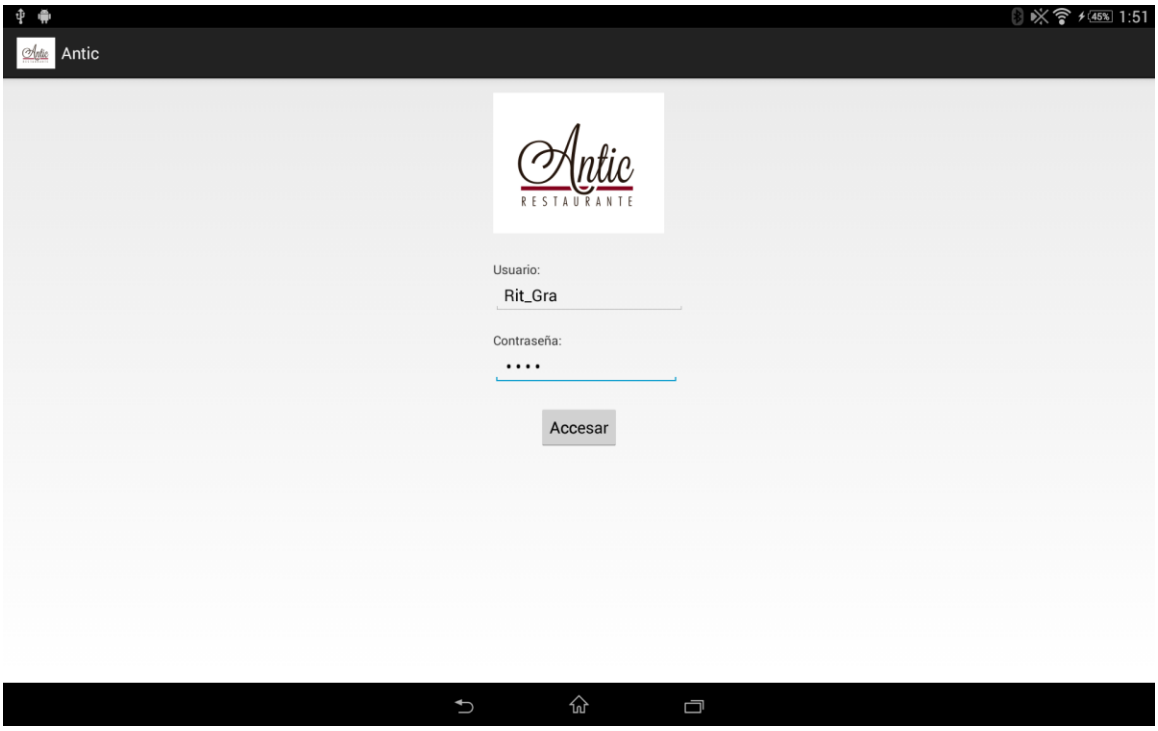


Figura 60. Muestra de error al ingresar mal el usuario

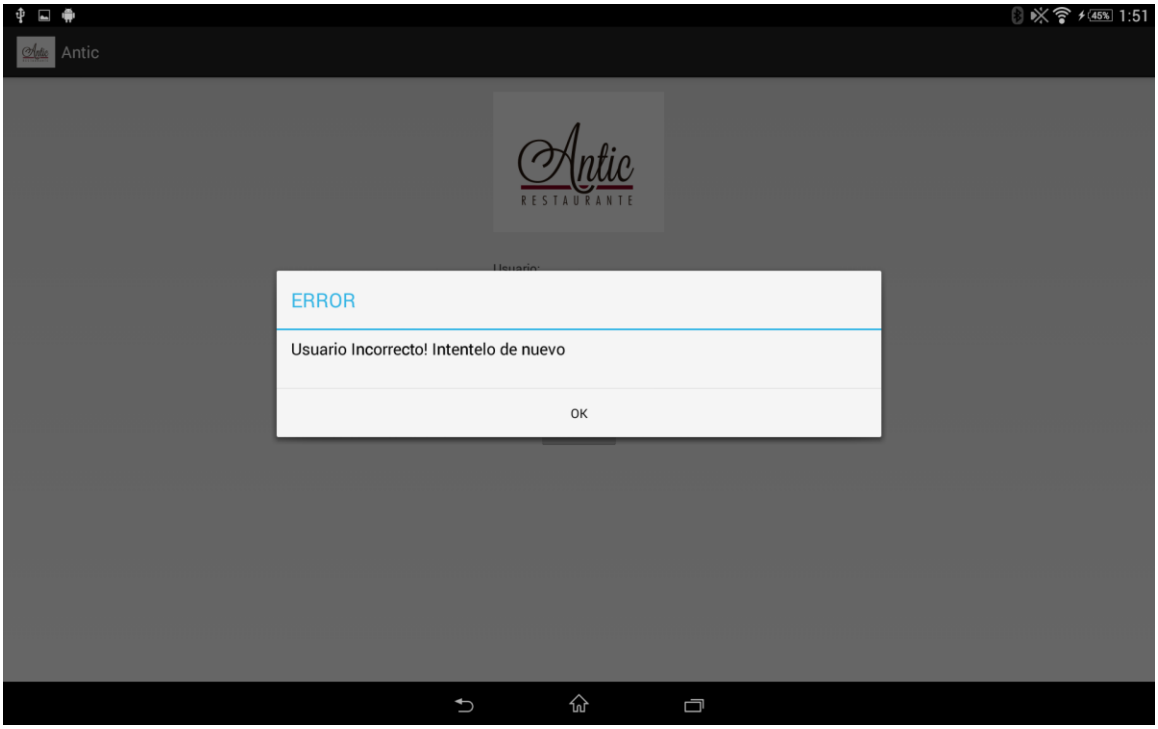


Figura 61. Ingreso de la mesa en la que atiende el mesero

Figura 62. Opciones a realizar por mesa

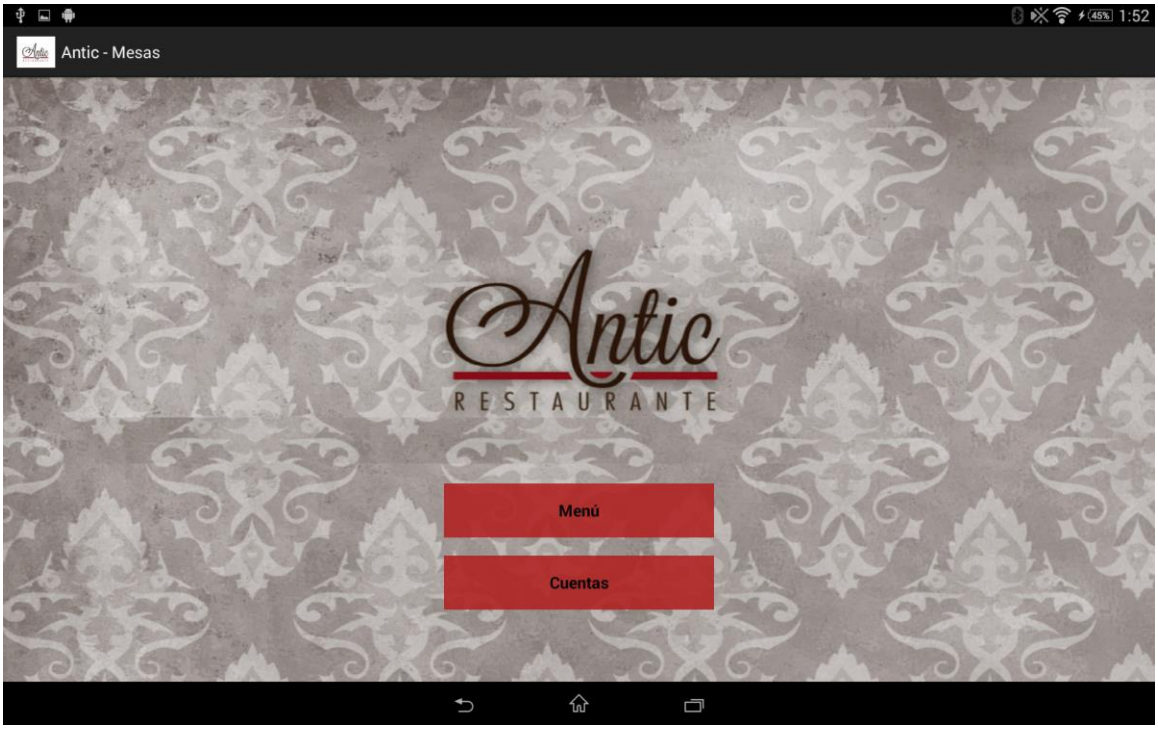


Figura 63. Despliegue de categorías del menú y cuentas de la mesa

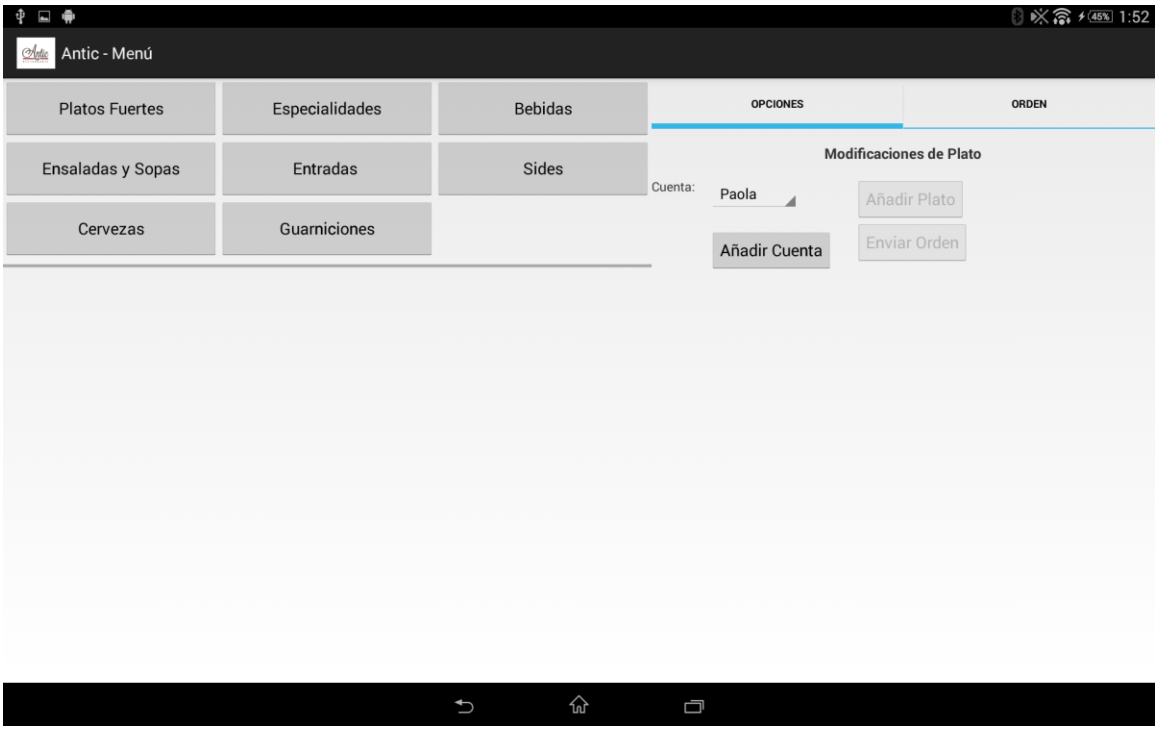


Figura 64. Lista que muestra las cuentas de la mesa

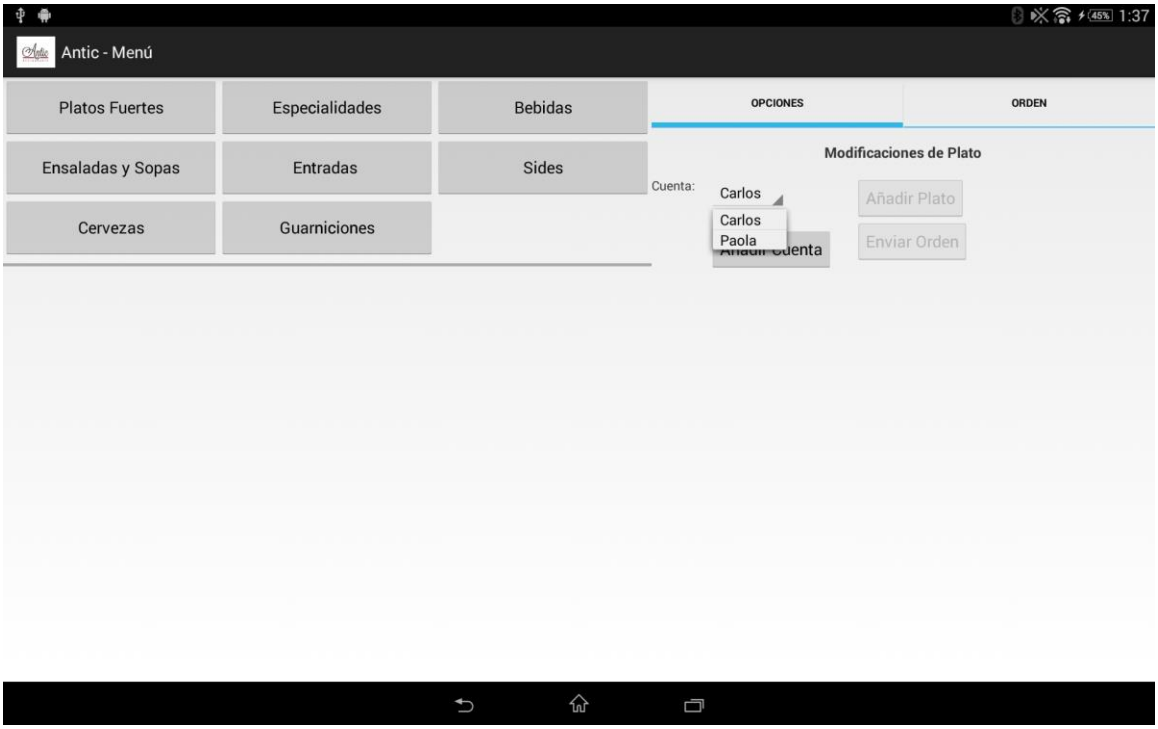


Figura 65. Despliegue de platos por categoría y muestra de las modificaciones por plato seleccionado

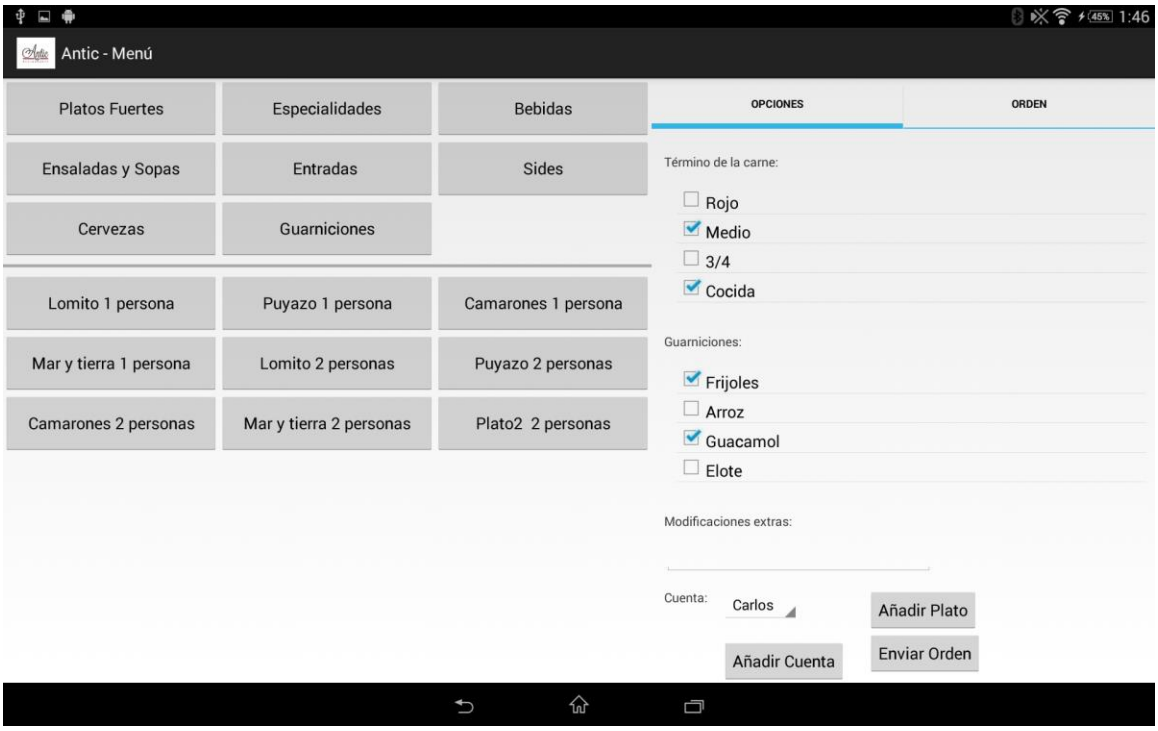


Figura 66. Mensaje de error al seleccionar inadecuadamente el término de la carne

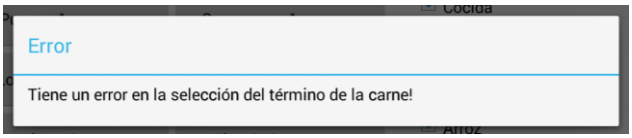


Figura 67. Mensaje de error al seleccionar inadecuadamente las guarniciones del plato

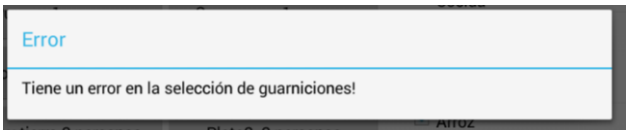


Figura 68. Despliegue de los platos agregados a la orden

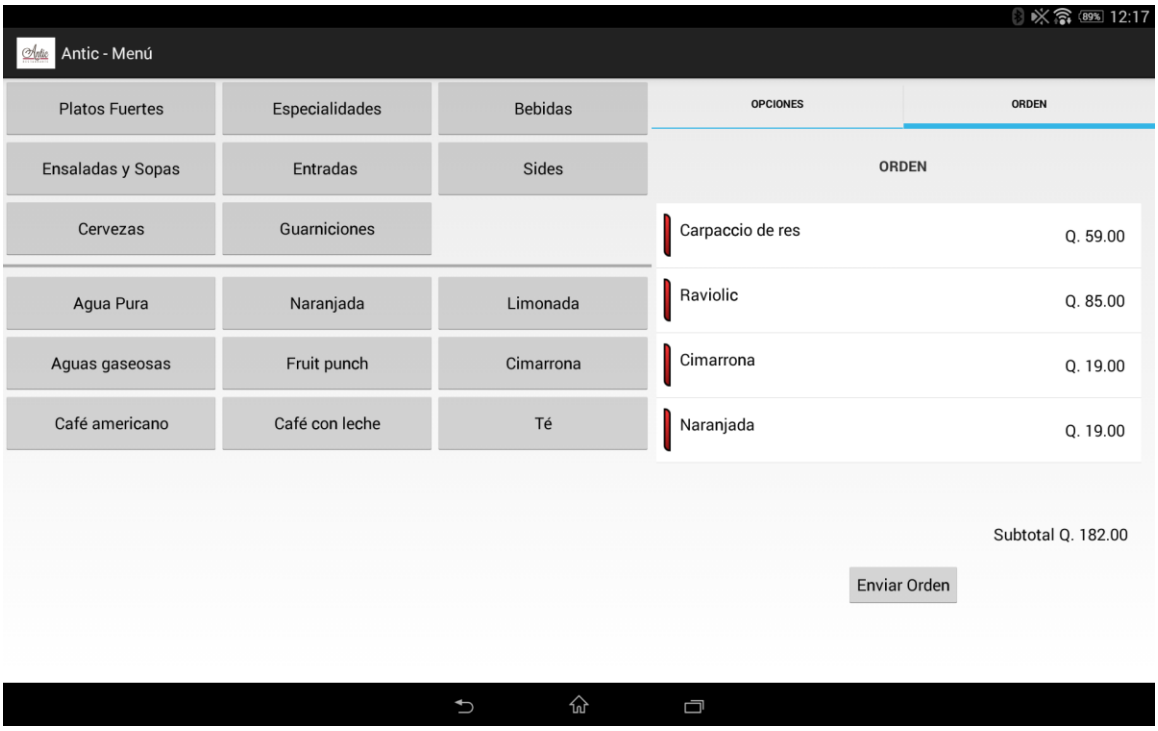


Figura 69. Despliegue de las modificaciones del plato agregado a la orden

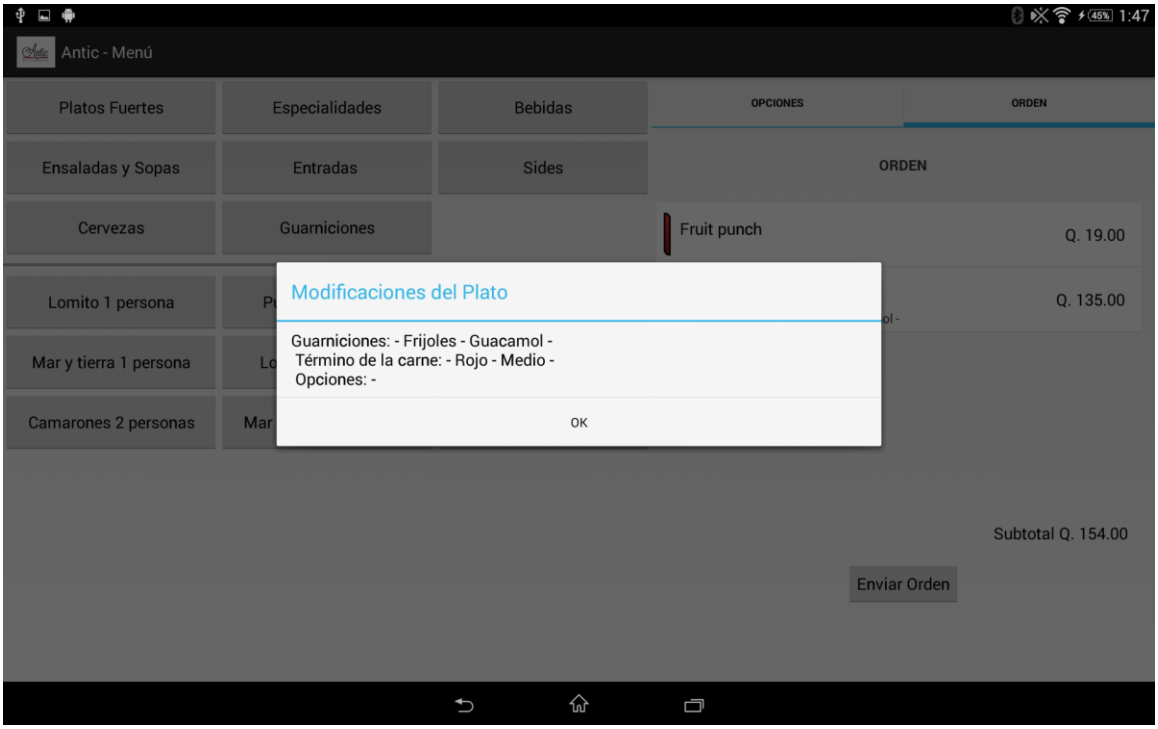


Figura 70. Verificación de eliminación de un plato de la orden

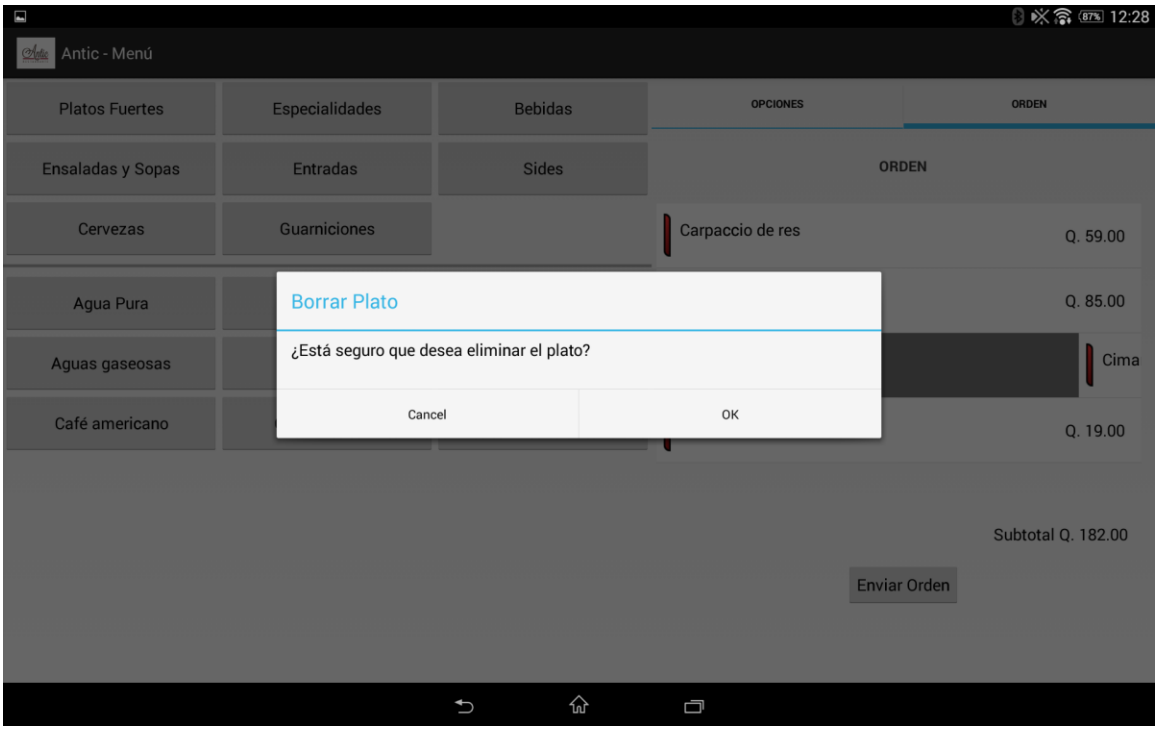


Figura 71. Mensaje generado cuando no hay cuentas pendientes que cobrar en una mesa

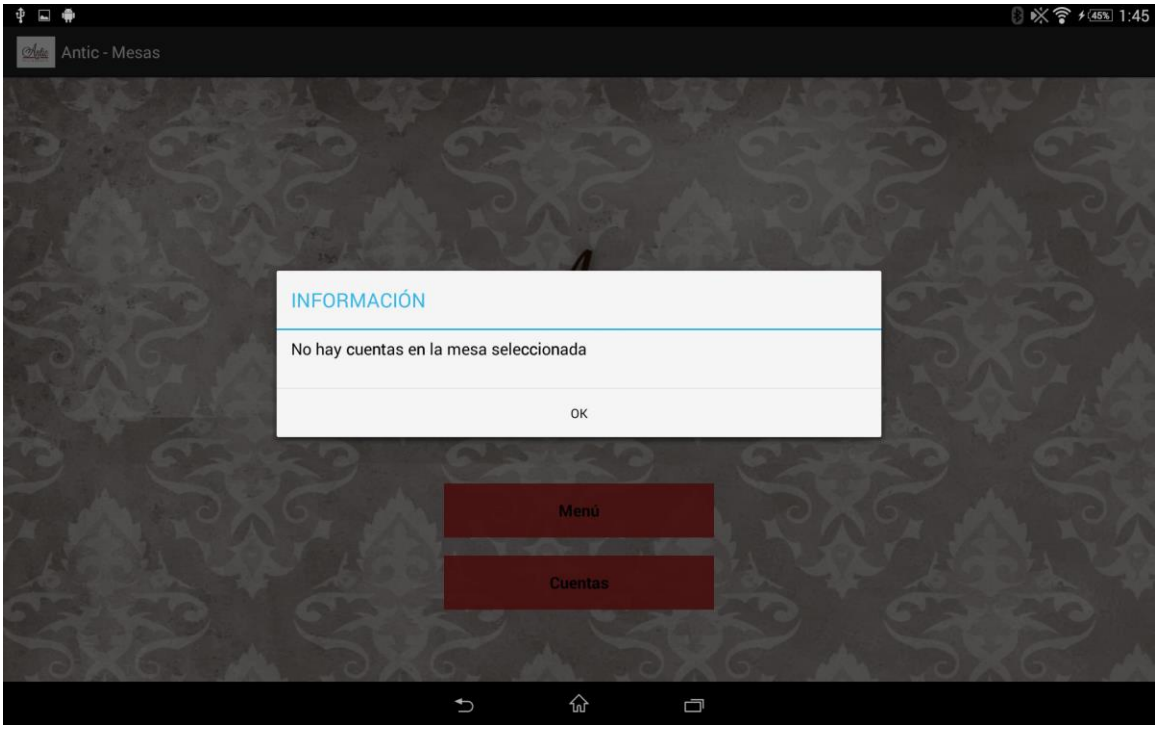


Figura 72. Despliegue de cuentas pendientes por cobrar en una mesa

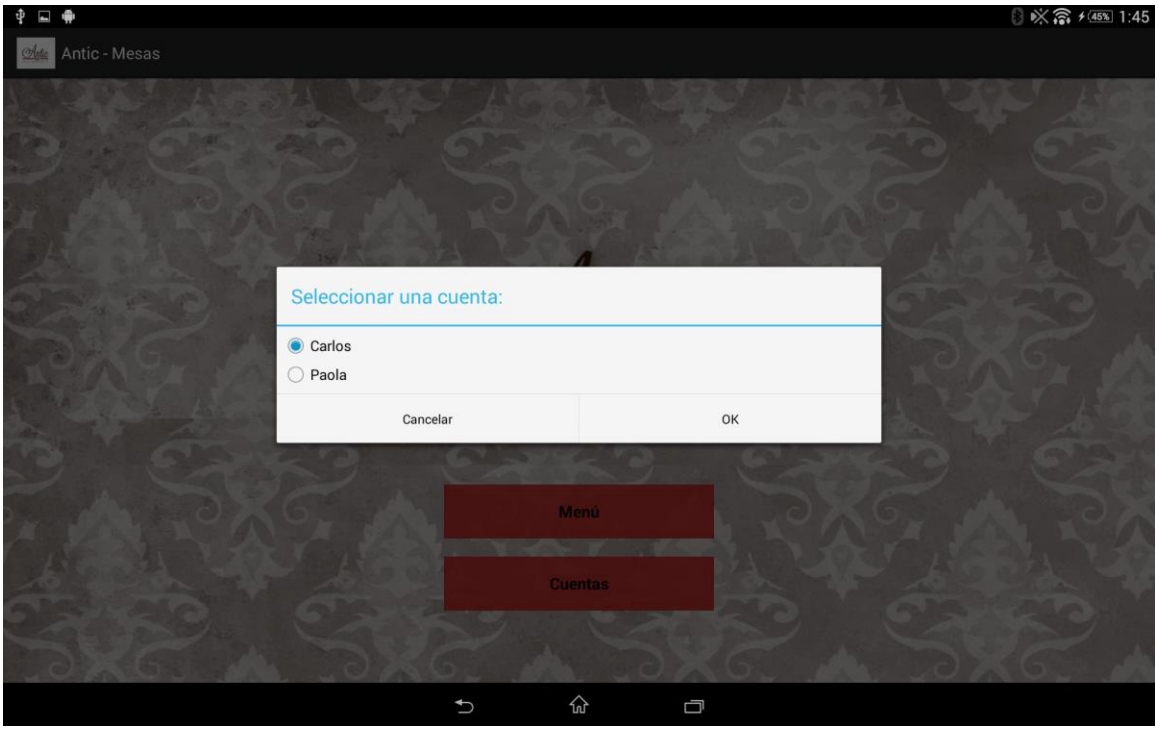


Figura 73. Despliegue de la cuenta por cobrar

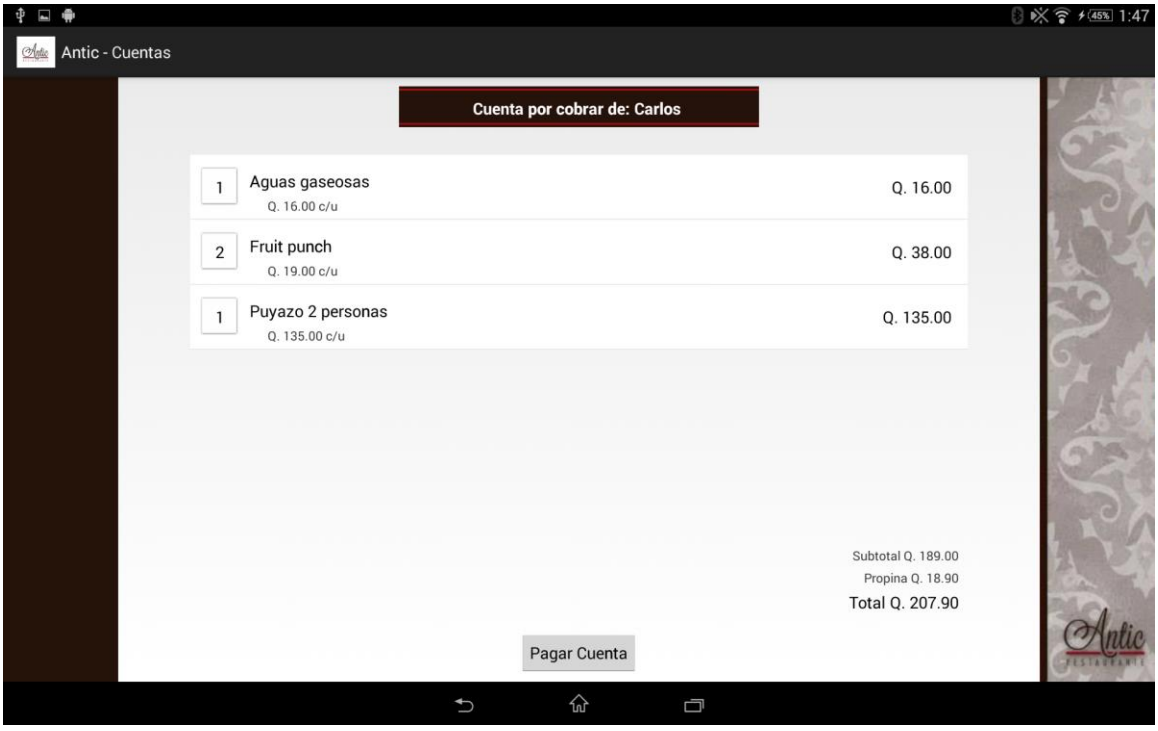


Figura 74. Realización de deslizamiento para eliminar un plato

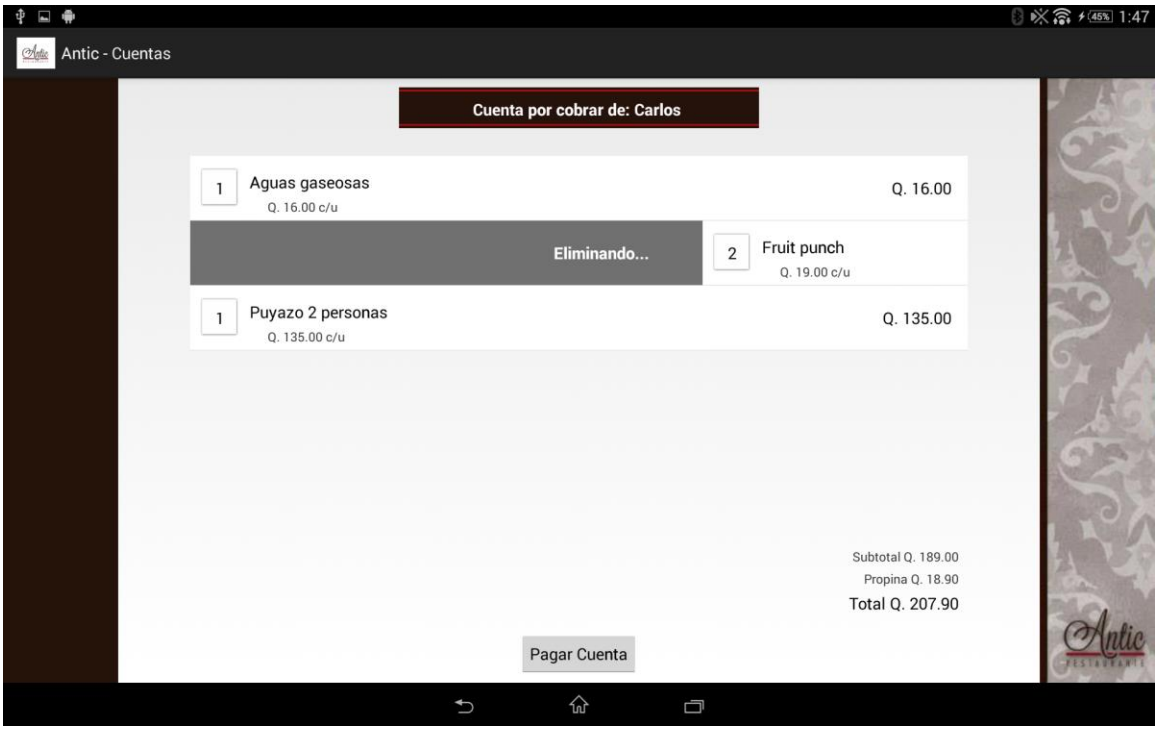


Figura 75. Pregunta de verificación de eliminación de un plato de la cuenta por cobrar

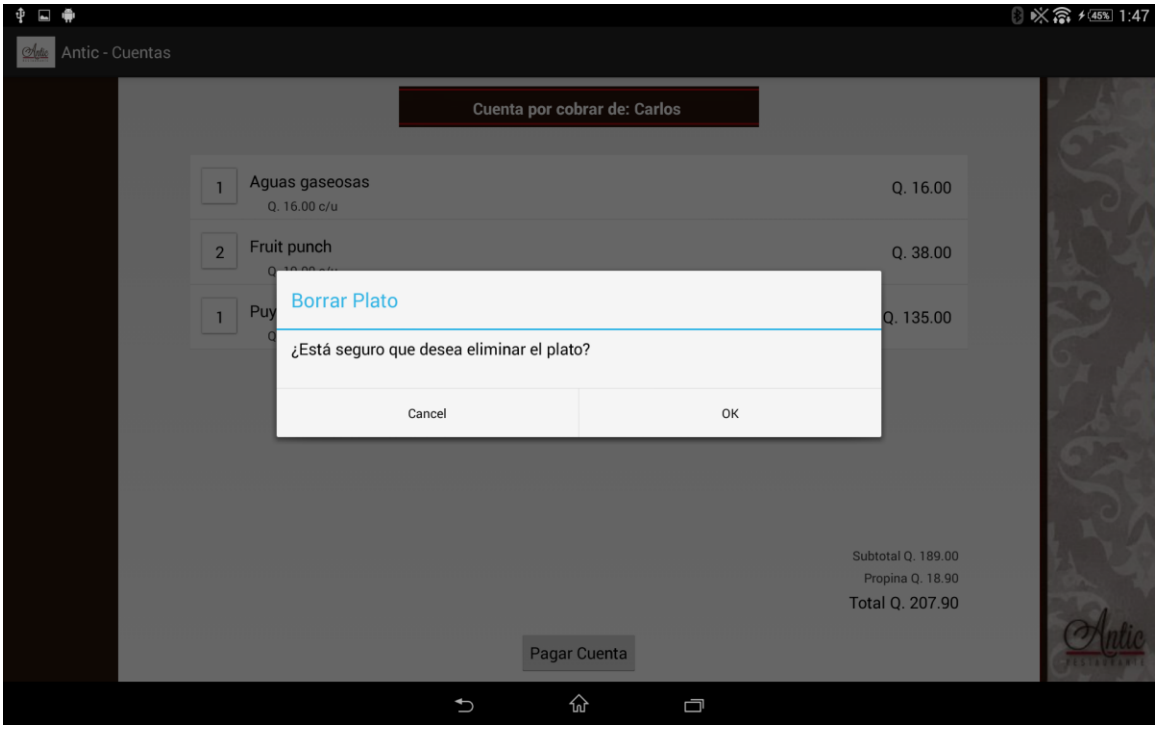


Figura 76. Ingreso de contraseña de administrados para eliminar un plato de la cuenta por cobrar

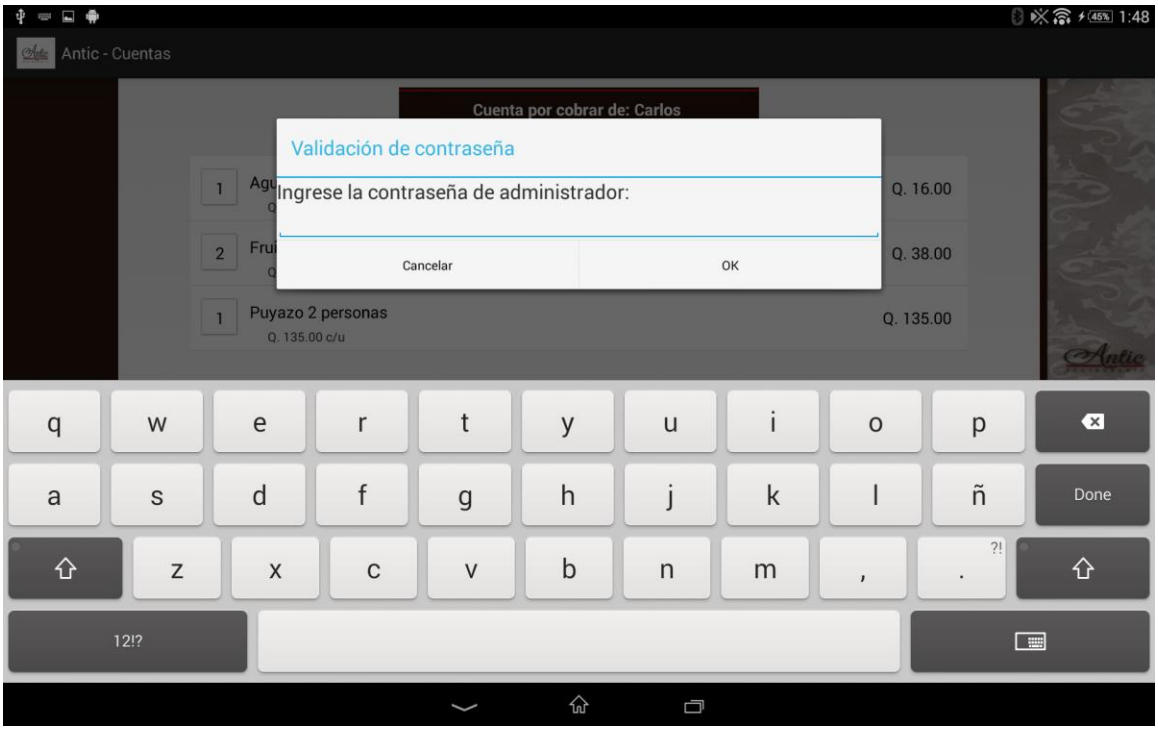


Figura 77. Lista de opciones de pago

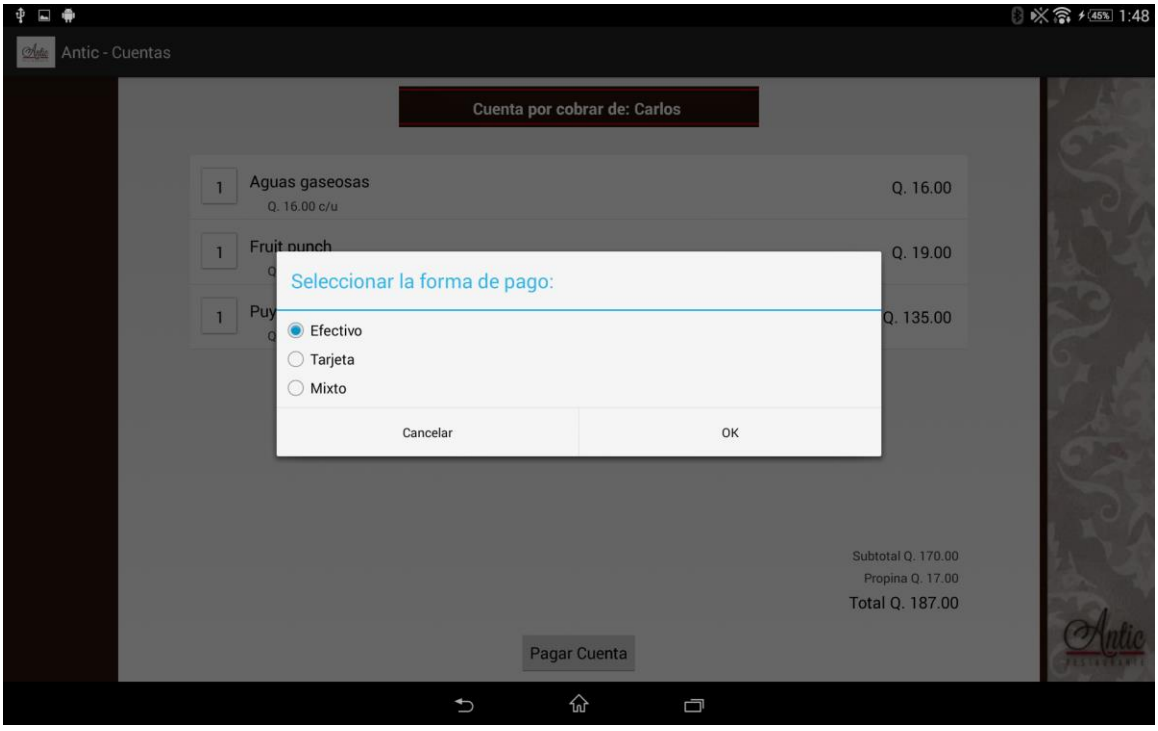


Figura 78. Verificación si se recibió el 10% de propina

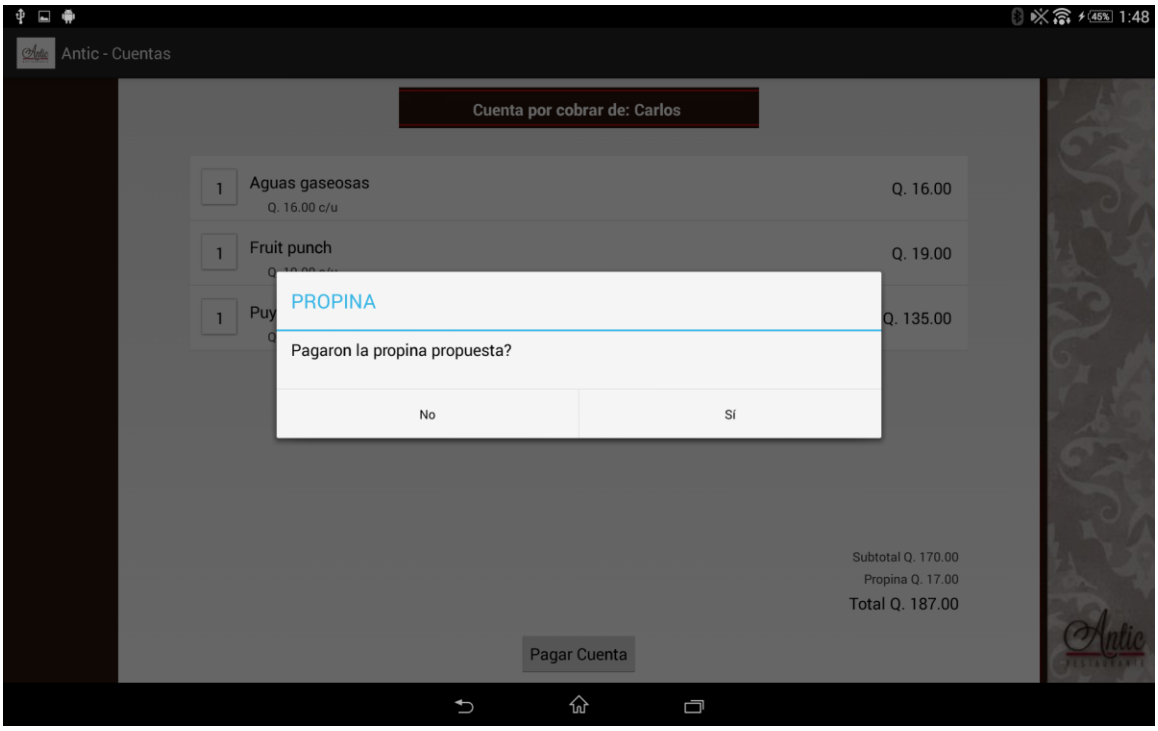
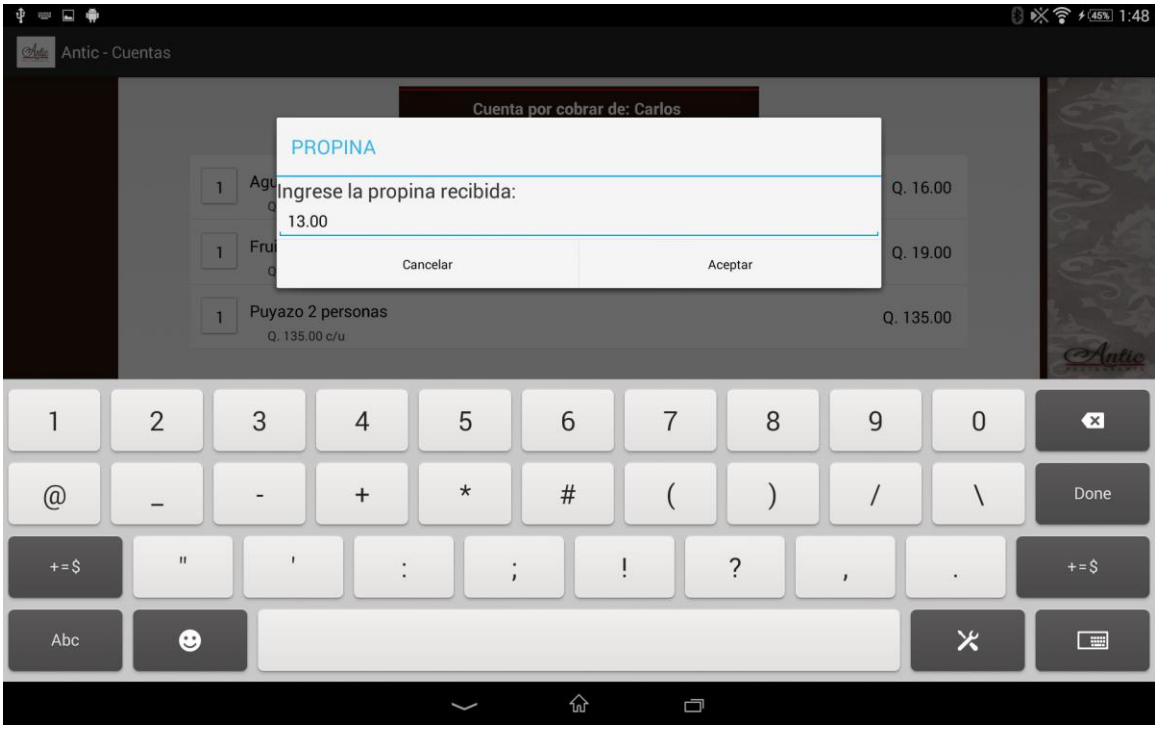


Figura 79. Ingreso de la propina recibida en caso no se recibe lo esperado



2. APLICACIÓN DE LOS CLIENTES

Figura 80. Diagrama de flujo de la aplicación para los clientes

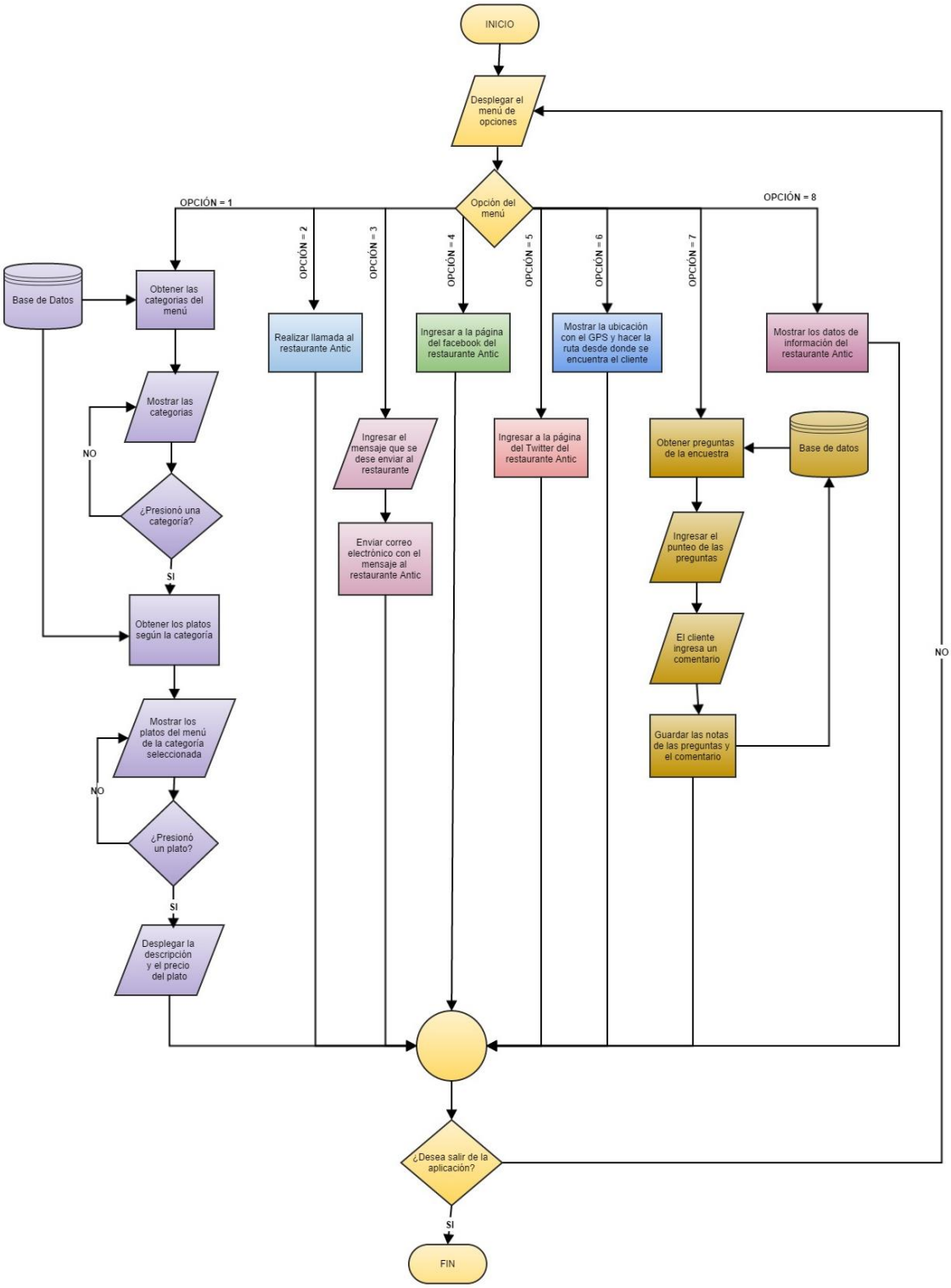


Figura 81. Diagrama UML de la aplicación para los clientes

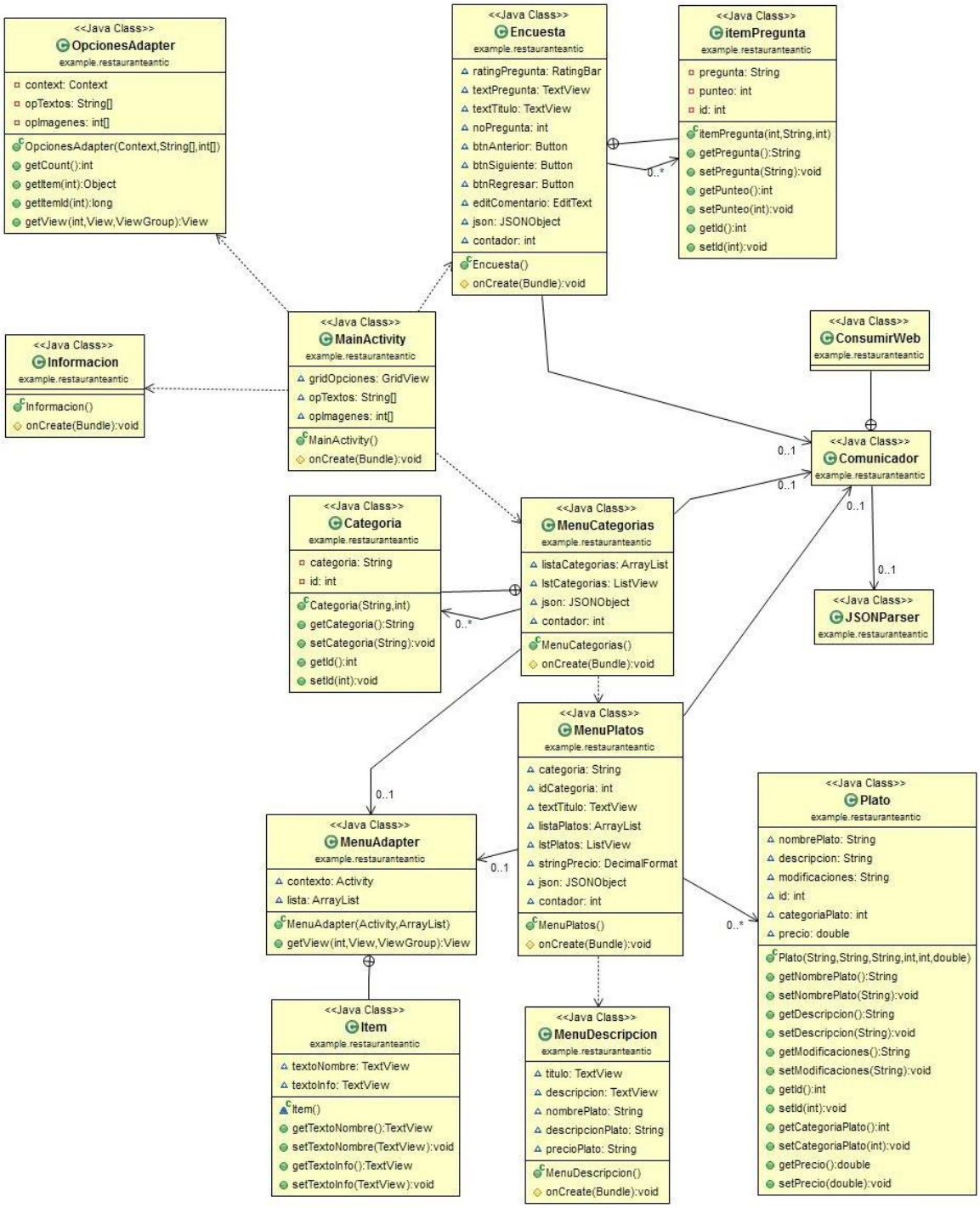


Figura 82. Menú principal de la aplicación para clientes

Figura 83. Despliegue de las categorías del menú en aplicación para clientes

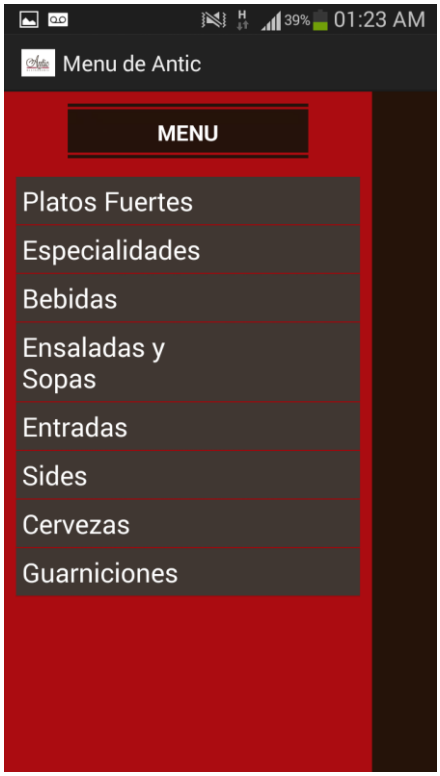


Figura 84. Despliegue de los platos según la categoría seleccionada en aplicación para los clientes

Figura 85. Despliegue de descripción del platos seleccionado en aplicación de clientes

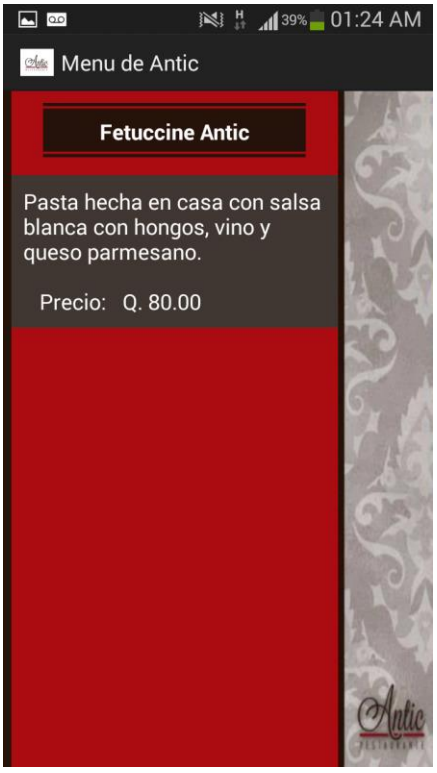


Figura 86. Llamada realizada por la aplicación de clientes

Figura 87. Ingreso de texto para enviar correo electrónico al restaurante Antic desde la aplicación de clientes

Figura 88. Despliegue de ruta hacia el restaurante desde la posición del cliente en GoogleMaps

Figura 89. Despliegue de ruta hacia el restaurante desde la posición del cliente en Waze

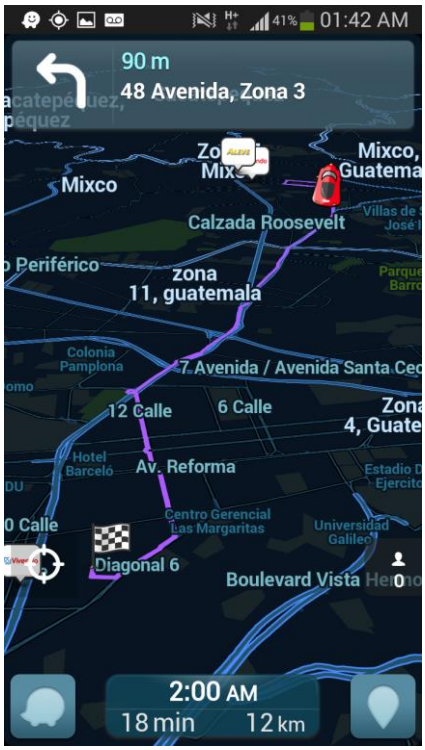


Figura 90. Despliegue de las preguntas de la encuesta para los clientes

Figura 91. Interfaz para que los clientes envíen un comentario al restaurante Antic

Figura 92. Despliegue de la información de contacto del restaurante Antic

C. ANÁLISIS DE RESULTADOS

Se requirió que la versión del SDK fuera la 22.6.4 ya que es la versión que contiene las herramientas necesarias para el API 18, y se requirió de la utilización de dicho API ya que era el requisito mínimo para el uso de la librería descargada *SwipeListView.* La librería mencionada fue utilizada ya que es un elemento que brinda una fácil utilización del programa para los usuarios, otro beneficio es que ahorra espacio en la pantalla ya que brinda opciones sin necesidad de colocar botones sino que interactuando directamente con la lista; la utilización de *swipeListView* facilita la utilización de la app debido a que para un entorno Android se ha vuelto común dicha interacción y los usuarios ya están familiarizados con deslizar los elementos de una lista esperando un evento ante dicha acción; las aplicaciones más utilizadas como lo son Facebook, Twitter, Instagram lo utilizan, así mismo se utiliza en las aplicaciones de funcionalidad básica del teléfono como la aplicación de llamadas, mensajes, etc.

Para la depuración de las aplicaciones se requirió de un dispositivo móvil real con sistema operativo Android en configuración de "modo de desarrollador" debido a que un emulador no tiene soporte para el GPS ni para realizar llamadas, así que la aplicación de clientes solamente podía ser depurada en un dispositivo que brindara soporte GPS y la capacidad de realizar llamadas.

1. APLICACIÓN PARA LOS MESEROS. La aplicación de los meseros tiene como objetivo principal el control de comandas del restaurante Antic. Para enviar una comanda se requiere de saber los datos de qué mesero atendió, qué mesa es en la que se está pidiendo la orden, los platos que se desean, y si ya se debe pagar la cuenta. Los datos del mesero y de la mesa se requieren ya que el restaurante necesita llevar control sobre las ventas de cada mesero y si hay alguna anomalía en alguna mesa saber con quién referirse. Para desarrollar el programa inicialmente se realizó un diagrama de flujo con la finalidad de entender correctamente las fases del proceso de una comanda. En la Figura 55 se puede observar el diagrama de flujo en donde se piden los datos principales de la orden, los cuales son mesero y mesa que se atiente. Con el fin de definir correctamente los procesos, en la Figura 56 se muestran el proceso que sigue una orden desde su realización hasta su envío a cocina o bar, y en la Figura 57 se muestra el proceso de pago de una orden considerando que si la orden tiene un plato incorrecto el plato pueda ser eliminado para no ser cobrado pero solamente con la contraseña de los gerentes. En la Figura 58 se muestra el diagrama UML de la app de meseros en donde se puede observar fácilmente la interacción de todas las clases entre sí. Este diagrama puede ser analizado para entender el flujo lógico que se siguió para desarrollar la aplicación.

Ya que se debe obtener quién ingresó la orden para tener registro de qué mesero atendió cada mesa, al inicio de la aplicación pide que se ingrese el usuario y una contraseña, esto también evita que otras personas ajenas a los meseros realicen pedidos a la cocina (ver Figura 59). En caso el usuario no existiera o la contraseña fuera incorrecta, es decir, que no esté registrada en la base de datos, se muestra un mensaje de error como se observa en laFigura 60. Luego de validar la existencia del mesero en la base de datos se procede a pedir el número de mesa que se atenderá (ver Figura 61). Este dato debe ser trasladado a la siguiente *Activity*, y esto se realiza mediante la colocación de extras en los *Intents*. En la Figura 62 se despliegan las dos opciones que se pueden seleccionar luego de haber ingresado la mesa que se atenderá, los botones en esta sección son más grandes para facilitar el uso del programa y que los meseros no presenten complicaciones de tener que presionar un espacio reducido.

Al presionar la opción de menú se despliega una *Activity* que contiene dos *Fragments*[19] (ver Figura 63); el primer *Fragment* es el responsable de seleccionar un plato, y el segundo de mostrar las modificaciones que se le realizarán y el listado de los platos pedidos en la cuenta para su verificación antes de ser enviada a cocina o a bar. Se realizaron ambas funciones con *Fragments* debido a posibles expansiones en la aplicación de clientes. Al utilizar dichos componentes se brinda la posibilidad de reutilizar el código si, por ejemplo, el restaurante deseara proveer a los clientes la opción de realizar pedidos por medio de la aplicación. Para implementarlo solamente se analiza el tamaño de la pantalla, y si no se pueden desplegar los dos *Fragments* al mismo tiempo, se muestra uno primero y luego el otro; no es necesario

[19] Es un componente similar a la *Activity* con su propio ciclo de vida, pero una *Activity* puede contener varios *Fragments*.

reprogramarlos ya que están implementados como interfaces de usuarios que se integran a una actividad según su conveniencia de espacio. Gracias a la utilización de los *Fragments* se podrán utilizar varias interfaces gráficas en una o varias actividades ya que su ciclo de vida es independiente al de la *Activity*.

Con el fin de facilitar las cuentas a los meseros o a los mismos clientes con la separación del pago, la aplicación es capaz de generar cuantas cuentas se deseen solamente presionando el botón de Añadir Cuenta e ingresando el nombre de la nueva cuenta, verificando que el nombre ingresado no haya sido utilizado. En la Figura 64 se puede observar que ya se han agregado dos cuentas, y que por medio de una lista desplegable se puede seleccionar en cuál se agregarán los platos.

Al seleccionar una categoría se despliegan todos los platos referentes a ella, y además al presionar un plato en específico se muestran todas las posibles modificaciones que se pueden realizar al plato, tales como el término de la carne, las guarniciones que se desean, sin ciertos ingredientes, etc. En la Figura 65 se muestra como ejemplo una orden de un plato fuerte para dos personas. Con el fin de facilitar el uso de la app se despliega las opciones que se tienen para las modificaciones de los platos a manera de lista, y el mesero puede escoger las que desee. Sin embargo el menú presenta restricciones como por ejemplo si se ordenó un plato fuerte para una persona solamente se puede seleccionar un término de la carne, a diferencia si se ordenó un plato fuerte para dos personas, se puede seleccionar dos términos de la carne. Existe la misma restricción para las guarniciones de los platos fuertes; así que al agregar el plato la aplicación despliega un mensaje de error informando al mesero si ingresó incorrectamente el término o las guarniciones del plato (ver Figura 66 y Figura 67).

Al finalizar los pedidos los meseros deben verificar la orden antes de enviarla a cocina para brindar un mejor servicio a los clientes según los requerimientos que presentó el restaurante Antic. Para facilitar dicha tarea se presenta en la parte superior del *Fragment* un menú en TABS y se debe presiona en la opción derecha de la aplicación, con lo que se despliega una lista de todos los platos pedidos con sus precios y el monto total del pedido (ver Figura 68). Si se desea ver las modificaciones de cada plato solamente se debe presionar sobre el plato, y se mostrarán en un mensaje (ver Figura 69). Si al verificar la orden los clientes desean realizar algún cambio es posible eliminar el plato que ya no se desea deslizando el dedo de izquierda a derecha sobre el plato, y se mostrará un mensaje de verificación (ver Figura 70) y al presionar ok se eliminará de la lista. El paso de verificación se colocó como un método para evitar la eliminación de platos por error ya que el mesero podría presionar la pantalla de una forma indeseada y si solamente se elimina podría generar molestias al cliente volviendo a preguntar la orden. Luego de haber verificado la orden puede presionarse el botón “enviar orden” para que sea enviada. Ya que el restaurante cuenta con un área de bar que se encarga de todas las bebidas, la aplicación clasifica cada plato de la orden y envía una suborden a cocina de la comida y otra suborden a bar de las bebidas.

Si en la pantalla que se observa en la Figura 62 se presiona cuentas y no hay ninguna pendiente de cobro se muestra un mensaje de alerta como se observa en la Figura 71; en cambio sí hay cuentas pendientes se despliega una lista para que el mesero seleccione la cuenta que se cancelará en el momento (ver Figura 72). Luego de seleccionar la cuenta se muestra en pantalla el nombre de la cuenta y el listado de platos pedidos, del lado izquierdo se muestra la cantidad de platos iguales y del lado derecho el total del costo, en la parte inferior de la pantalla se muestra el subtotal, la propina propuesta y el total (ver Figura 73). Si ocurrió un error en el listado de los platos de la orden y se desea eliminar un plato se desliza sobre el mismo como se muestra en la Figura 74, y se verifica si se desea eliminar como se observa en la Figura 75; sin embargo como mecanismo de seguridad se pide la contraseña de los administradores como se mira en la Figura 76 y si la contraseña es correcta entonces se elimina el plato. El objetivo de que solamente con la contraseña de los administradores se pueda eliminar y no con las contraseñas de los meseros es debido a que así fue pedido por los dueños del restaurante Antic. Al momento de pagar primero se pide la forma de pago (ver Figura 77), luego para agilizar el proceso se pregunta si se recibió la propina propuesta (ver Figura 78), es decir, el 10% de lo consumido; esto se realizó porque se estableció en el restaurante que al dar ellos la cuenta preliminar siempre colocan la propina propuesta aunque no sea obligatorio pagarla. En caso de que se reciba una propina diferente se despliega un texto donde se ingresa dicha cantidad (ver Figura 79), y finalmente se envía la información a la base de datos con la cancelación de la cuenta y las características de pago.

2. APLICACIÓN PARA LOS CLIENTES. Al igual que en la aplicación de meseros, en la aplicación de clientes se realizó un diagrama de flujo para facilitar el entendimiento de lo que debía realizar la app. En la Figura 80 se puede analizar dicho proceso, y en la Figura 81 se puede analizar el diagrama UML que concreta el código realizado y mejora su entendimiento.

En esta aplicación, a diferencia a la anterior, el objetivo era que fuera llamativa debido a que debe captar la atención de los clientes; por lo tanto para la pantalla principal se seleccionó una imagen institucional del restaurante, la cual es utilizada en su menú como se observa en la Figura 82. En dicha *activity* se muestran las ocho opciones disponibles al clientes las cuales son observar el menú, llamar al restaurante, mandar un correo electrónico, entrar a su página de Facebook o a la de Twitter, mostrar la ubicación del restaurante desplegando la ruta a seguir para que el cliente llegue desde donde se encuentre, pueden los clientes seleccionar hacer una encuesta para que el restaurante tenga realimentación y por último se presenta la opción de desplegar toda la información del restaurante.

Si el cliente desea ver el menú se despliega una pantalla con las categorías del mismo (ver Figura 83), y al presionar una se despliega otra pantalla donde se muestran los platos de la categoría seleccionada (ver Figura 84). En dicha pantalla se despliega el precio de los platos, pero para brindar un mejor servicio a los clientes si se presiona un plato se muestra una descripción del mismo (ver Figura 85). En estas tres

pantallas se utilizó el mismo formato que utiliza el restaurante en sus menús impresos, es decir, los mismos colores y formatos de interfaz gráfica.

Al presionar “llamar al restaurante” se dispara una llamada como se observa en la Figura 86; para realizar esta acción se utilizó un *Intent* con la acción de *CALL*, y además se debió colocar en el *AndroidManifest* el permiso para acceder a esta aplicación de llamadas en el dispositivo móvil, esto fue necesario ya que se dispara la llamada automáticamente al presionar la opción; si en dado caso no se pidiera el permiso, el programa solamente podría utilizar la acción *DIAL* y el número del restaurante automáticamente aparece en la pantalla pero no llama de manera automática. A dicho *Intent* se le proveyeron como datos el URI del teléfono del restaurante Antic.

En el caso de presionar “email” se realiza otro *Intent* con la acción *SEND*; para completar la acción es necesario establecer el tipo de mensaje que se desea enviar, y además se necesita especificar los correos electrónicos a los que debe enviar el mensaje electrónico, así que cambiando el tipo y poniendo como extra la dirección se puede observar en la Figura 87 que se envía un correo al restaurante sin necesidad de teclear su correo electrónico.

Para ingresar a la opción de Facebook y Twitter se realizó un *Intent* con la acción *VIEW*, y para completar la tarea se colocó como URI la dirección url de ambas páginas. Para utilizar ambos logos se han seguido las indicaciones, y se han utilizados las imágenes que Facebook y Twitter proveen para conectar apps con sus redes sociales.

Al presionar “ubicación” también se genera un *Intent* con la acción *VIEW*; el objetivo es generarle al usuario la ruta para llegar al restaurante desde donde se encuentre así que para lograrlo se tomó la longitud y latitud del restaurante Antic, y por medio del URI de GoogleMaps, se estableció el punto geográfico y la ruta, siempre y cuando el GPS del dispositivo móvil se encuentre activado, en caso que el GPS del dispositivo esté desactivado la aplicación solamente muestra dónde se encuentra el restaurante pero no genera una ruta para llegar a él. En esta función se permite al cliente mostrar la ruta en GoogleMaps (ver Figura 88) o en Waze (ver Figura 89).

Para obtener datos de realimentación se realizó una encuesta a los clientes del restaurante. Las preguntas fueron determinadas por el gerente del restaurante Antic y pueden ser modificadas al ser ingresadas en una página web; así que la aplicación de clientes sólo despliega la pregunta y permite al usuario darle una calificación de 0%, 25%, 50% 75% y 100% como se muestra en la Figura 90, y posteriormente permite el ingreso de un comentario para complementar la encuesta como se muestra en la Figura 91. Estos datos son almacenados en la base de datos guardando cuántas personas han respondido cada puntuación, por ejemplo

si un cliente responde en la primera pregunta 50% la respectiva casilla aumenta su cantidad en uno y así se puede generar las estadísticas requeridas por el restaurante.

Si los clientes desean solamente contar con la información del restaurante y saber los horarios de atención al cliente, entonces pueden presionar la opción "información" y se despliegan los datos de contacto del restaurante Antic como se muestra en la Figura 92.

D. CONCLUSIONES

- La aplicación de meseros permitió desplegar el menú y las ofertas del restaurante Antic, teniendo acceso a realizar modificaciones en los plato según lo deseado por los clientes y además desplegó la lista final de la orden antes de ser enviada vía WiFi a la cocina o a bar respectivamente.

- Por medio de la implementación de la clase Plato se almacenó todos los datos pertinentes de cada plato de la base de datos y se ahorró a los meseros el trabajo de dividir la orden en bar y cocina ya que la aplicación enviaba los respectivos platos a cada área.

- La aplicación de los meseros permitió la organización de las órdenes por cuentas evitando así que los meseros o los clientes realicen los cálculos de la división de platos.

- La aplicación de clientes permitió el acceso a los mismos a visualizar el menú y las ofertas del restaurante Antic comunicándose con una base de datos en línea.

- Por medio de la implementación del uso de GPS se ha facilitado a los clientes el encontrar el restaurante ya que despliega la ruta para llegar a él desde cualquier punto donde se encuentre el cliente.

- Se implementó la recepción de realimentación por parte de los clientes ya que en la aplicación de clientes se puede realizar una encuesta evaluando los servicios del restaurante y además tienen la opción de colocar un comentario; dichos resultados y comentarios son almacenados en la base de datos en línea para disposición del restaurante.

- La utilización de *Fragments* en la aplicación de meseros permite expandir la aplicación de clientes cuando se desee pedir por medio de la app de clientes y no se debe reprogramar el código.

- En la aplicación de meseros para eliminar un plato de una orden que se debe pagar se implementó como método de seguridad que deben ingresar la contraseña de los gerentes para que solamente los administradores del restaurante sean capaces de eliminar platos de las cuentas.

E. RECOMENDACIONES

- Se recomienda implementar en la aplicación de clientes la opción de ordenar y pasar a traer la comida, incluyendo la implementación de modificaciones a los platos seleccionados.

- Se recomienda llevar a cabo pruebas de usabilidad, ya que éstas no se llevaron a cabo.

- Para brindar un mejor servicio a los clientes se pueden implementar las notificaciones de nuevos platos o modificaciones en el menú por medio de alertas en los dispositivos móviles.

- Como mecanismo de seguridad, se debería implementar que cada cierto tiempo se cierre la sesión del mesero para evitar que alguien pueda agarrar la *tablet* y pedir un plato como un mesero.

- Implementar notificaciones a los gerentes de anomalías en las ordenes, por ejemplo que en una mesa solamente se pidiera una bebida y cinco platos; así se evita que los meseros regalen mercadería a sus conocidos.

VIII. DISPOSITIVO DE NOTIFICACIONES INALÁMBRICO TIPO BRAZALETE PARA MESEROS DEL RESTAURANTE ANTIC

A. METODOLOGÍA

1. Recolección de información de componentes. Se consideró en base al problema planteado, crear un dispositivo versátil capaz de mostrar mensajes y generar alertas no audibles para la creación de un prototipo.

Se investigó sobre microcontroladores, especialmente los disponibles en el mercado local. Entre los microcontroladores a investigar se decidió obtener información sobre dimensiones físicas. Como primera opción se enfocó en las capacidades del microcontrolador ATmega328P en placa de Arduino Nano. Esta investigación incluye información de hojas de datos.

Se investigó sobre dispositivos de comunicación inalámbrica de corto alcance para dispositivos móviles. Como primera opción se tomó el sistema de comunicación Bluetooth, se obtuvo información sobre dimensiones físicas de los módulos disponibles en el mercado local.

Se investigó acerca de dispositivos capaces de mostrar caracteres de letras en el mercado local, también se obtuvo información sobre dimensiones físicas de los módulos. Como primera opción se eligió trabajar con una matriz de LED´s de 8x8.

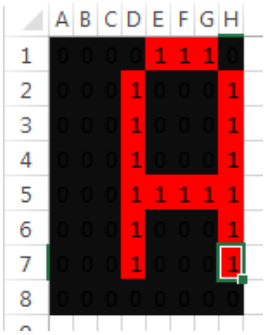
Se investigó sobre dispositivos que permitan notificar o alertar sobre eventos sin generar sonido, se obtuvo información sobre dimensiones físicas de módulos disponibles en el mercado local. La primera opción que se eligió fue un motor de vibración.

2. Prueba de dispositivos. Se inició con pruebas de funcionamiento del motor vibrador, estas pruebas son de control de encendido y apagado. De las hojas de datos se obtuvieron los valores de voltaje de operación y corriente, con lo cual se diseñó un prototipo de controlador de encendido y apagado. El controlador del motor se construyó con un transistor 2N3904.

Se realizaron pruebas de funcionamiento de la matriz de LED´s, de la hoja de datos se obtuvo la información de pines de conexión. Se controló el encendido y apagado de uno de los LED luego se probó encender y apagar en una fila y columna completa de la matriz.

En una hoja de cálculo de Excel se creó una matriz de 8x8, y se generó el carácter “A”. La letra se diseñó para utilizar 7 filas y 5 columnas en su despliegue como se muestra en la Figura 93. Se realizó la conexión en la matriz para crear el mismo efecto con los LED.

Figura 93. Mapa de 8 filas y 8 columnas que muestra el carácter A.



Con transistores 2N3904 y resistencias de 1,000 y 330 ohmios se conectó la matriz al Arduino Nano y se construyeron controladores para los ánodos de la matriz. Los emisores de los transistores se conectaron a los ánodos de la matriz, los colectores a la salida regulada de 5V; las bases se conectaron con resistencias de 1,000 ohmios a los pines 4 al 11 del Arduino Nano; con resistencias de 330 ohmios se conectaron los cátodos de la matriz a los pines 13 al 19.

Para reducir la cantidad de pines utilizados en el control de la matriz se decidió implementar el corredor de bits 74HC595 para la conexión de los catados. Para la conexión de los ánodos se continúa con el cableado a transistores y la base de los transistores al Arduino Nano.

Se generaron los arreglos, para 7 filas y 5 columnas, de la representación de los caracteres que se desean mostrar en la matriz de LED´s. Se utilizó Excel para facilitar la visualización de cada carácter.

Se modificó el código del microcontrolador para poder mostrar un mensaje de cualquier cantidad de caracteres, tomando el mensaje en una variable tipo texto. Se buscó la lectura individual de los caracteres que forman el texto y se realizó la interpretación de los caracteres a los arreglos de cada letra para mostrarlo en la matriz de LED´s. Se buscó mostrar el mensaje de forma clara a una velocidad que permita la lectura del mismo.

Para el prototipo del módulo Bluetooth se utilizó el integrado HC-05. Este se configuró por medio de códigos AT a un “baud rate” de 9600 y con una contraseña y nombre con los cuales se pudiera identificar. La configuración de este módulo se puede realizar con el mismo Arduino Nano.

Se programó el microcontrolador para que este se comunicara vía pines RX/TX con el módulo Bluetooth y así poder configurar este último.

Una vez configurado el módulo Bluetooth se realizaron pruebas de transmisión y recepción de mensajes con otros dispositivos como celulares o computadoras. Se utilizó una "hyperterminal" en la computadora para observar y enviar datos por medio de la conexión del puerto serial del Bluetooth y también una aplicación para celulares capaz de desplegar los datos recibidos.

Se modificó el código del microcontrolador de tal manera que el mensaje recibido en el pin RX sea mostrado en la matriz de LED´s. Adicionalmente se configuró que el motor vibrador sea activado y desactivado el tiempo necesario para generar la alerta de vibración mientras se muestra el mensaje en la matriz.

3. Diseño y manufactura de circuitos impresos. A partir de las pruebas realizadas se diseñó el circuito esquemático de la interconexión de los componentes. Basado en este diseño se desarrolló el diseño del circuito impreso tomando en cuente que se utilizaran transistores, capacitores y resistencias de superficie, así como las dimensiones de la matriz y la placa de Arduino Nano. Es importante mencionar que la alimentación de 5V debe hacerse a través del puerto mini-USB para poder habilitar el regulador de voltaje de 3.3V de la plataforma Arduino Nano. Los 3.3V son indispensables para alimentar el módulo HC-05.

4. Desarrollo de aplicación para dispositivo móvil. Se investigaron lenguajes de programación y librerías de software para el desarrollo de aplicaciones en la plataforma Android. Por sus X&Y propiedades se escogió tal y cual.

Se inició con el desarrollo de una aplicación que permita la conexión Bluetooth además de ser capaz de enviar y recibir mensajes. La revisión de conexión de dispositivo y el envío de mensajes se ejecutan en *threads* separados y ajenos al principal. La aplicación se comunica con la tabla de la base de datos que contiene la información de las comandas que ya han sido terminadas por cocina o bar.

La aplicación de comunicación con el brazalete se ejecuta en segundo plano junto con la aplicación de creación de comandas del mesero. La aplicación envía un mensaje vía Bluetooth al brazalete con la cadena "Cocina", "Bar" o "C/B" para que las comandas que se encuentran listas en cocina o bar, sea enviado por medio de Bluetooth un mensaje de "Cocina", "Bar" o "C/B", dependiendo si la comanda es por parte de cocina, bar o ambos respectivamente.

Para conocer si una orden está lista es necesario leer información de una base de datos. El acceso a esta se realiza por medio de una librería y debe ser ejecutado a intervalos cortos y constantes. En la base de datos existe una tabla con información de la comandas terminadas; dentro de esta se busca por nombre de mesero que comandas han sido terminadas. Si se encuentra una terminada se verifica de qué lugar provino la comanda. Dependiendo del lugar al cual proviene se envía por medio de Bluetooth al dispositivo el mensaje correspondiente.

La comunicación con la base de datos se realizó con la implementación de las clases en Java del módulo: *Desarrollo de una interfaz de programación de aplicaciones y estructuración de base de datos para el restaurante Antic*, desarrollado por Gustavo Andrés Sánchez Cardona. El código se encuentra en la sección de anexos.

B. RESULTADOS

Figura 94. Muestra de letra "e" en espacio de 8 filas y 8 columnas

Figura 95. Visualización de letra "e" en desplazamiento obtenida en la matriz 8x8 de LED´s RG de 3.8 cm X 3.8 cm.

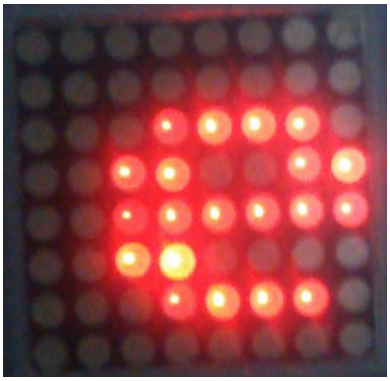


Figura 96. Diagrama esquemático del controlador encendido/apagado del motor vibrador

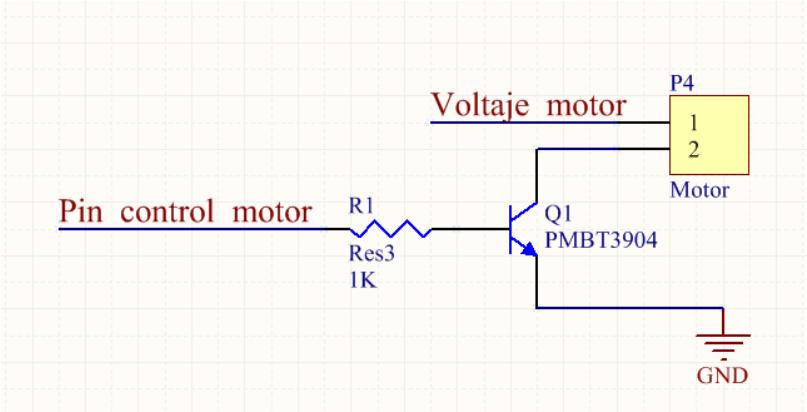


Figura 97. Diagrama esquemático de las conexiones de los ánodos de la matriz de LED´s, los identificadores D5-D12 indican los pines del Arduino Nano

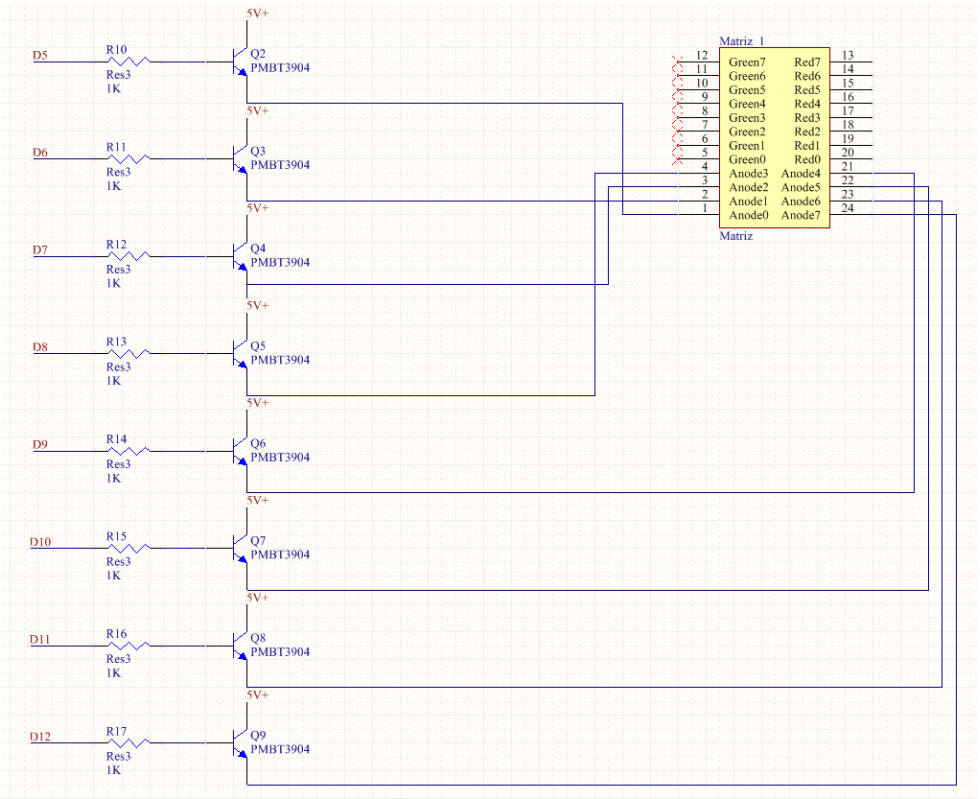


Figura 98. Diagrama de conexión de los cátodos para color rojo de la matriz de LED´s con el 74HC595, los identificadores A0-A2 indican los pines del Arduino Nano

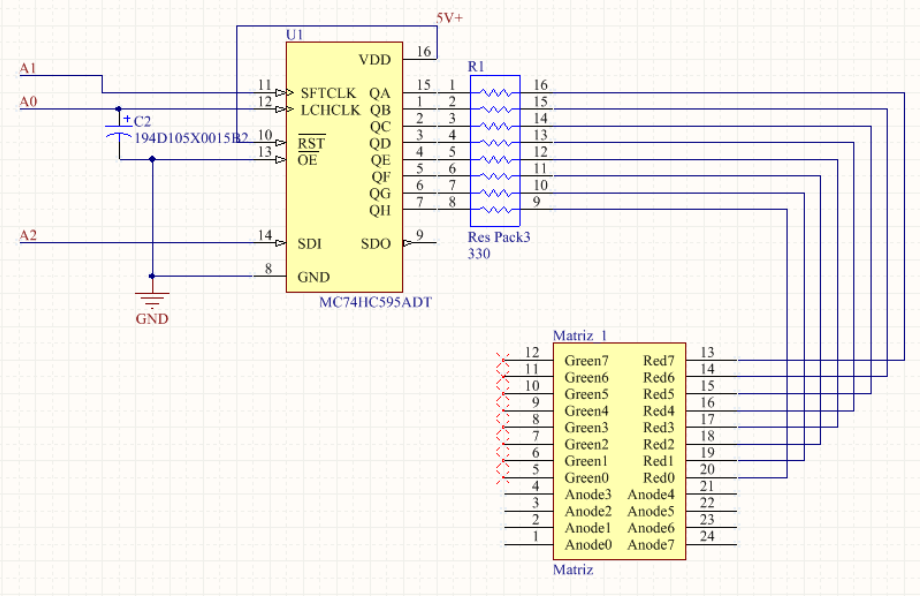
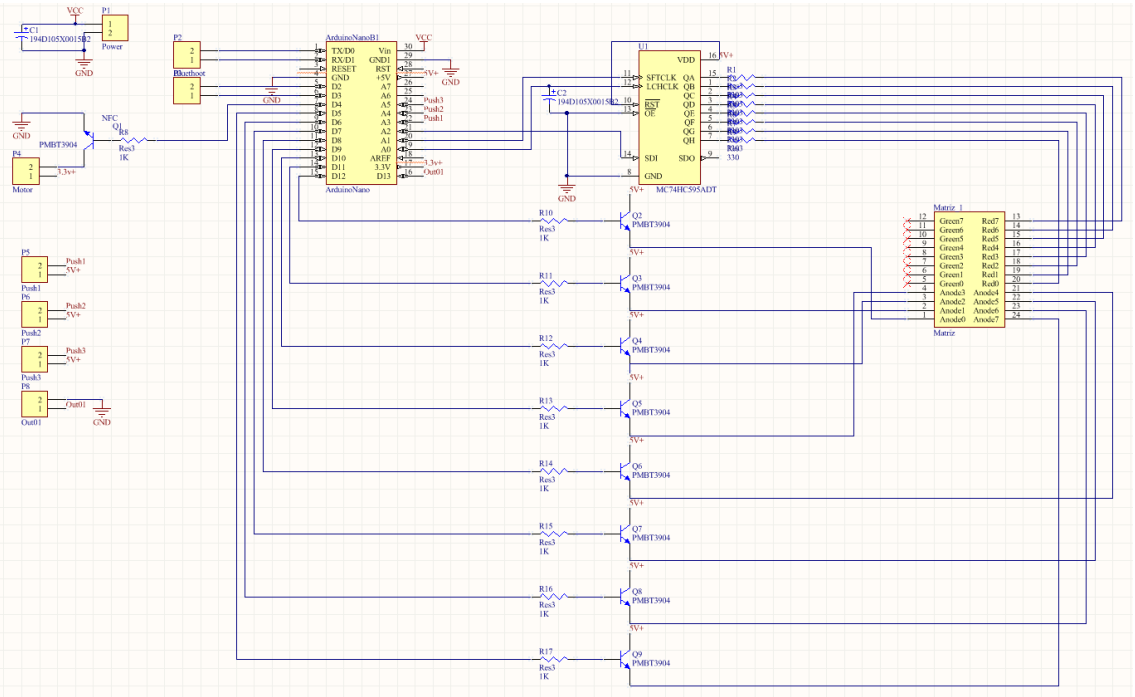


Figura 99. Diagrama esquemático de conexión de componentes



Cuadro 4. Conexiones de los pines de Arduino Nano con otros componentes

Identificador de Arduino Nano	Componente al cual conecta
TX	Pin 2 Conector 1 RX Bluetooth
RX	Pin 1 Conector 1 TX Bluetooth
D2	Pin 2 Conector 2 libre para actualización
D3	Pin 1 Conector 2 libre para actualización
D4	Resistencia de base de transistor controlador de motor vibrador (Q1)
D5	Resistencia de base de transistor controlador de ánodo (Q2)
D6	Resistencia de base de transistor controlador de ánodo (Q3)
D7	Resistencia de base de transistor controlador de ánodo (Q4)
D8	Resistencia de base de transistor controlador de ánodo (Q5)
D9	Resistencia de base de transistor controlador de ánodo (Q6)
D10	Resistencia de base de transistor controlador de ánodo (Q7)
D11	Resistencia de base de transistor controlador de ánodo (Q8)
D12	Resistencia de base de transistor controlador de ánodo (Q9)
D13	Pin 1 Conector 3 (libre para actualización)
A0	Pin 12 de 74HC595
A1	Pin 11 de 74HC595
A2	Pin 14 de 74HC595
A3	Pin 1 Conector 4 (libre para actualización)
A4	Pin 1 Conector 5 (libre para actualización)
A5	
A6	Pin 1 Conector 6 (libre para actualización)
3V3	Pin 1 Conector 7, Vcc de Bluetooth
+5V	Con colectores de Q2 a Q9 y Pin 16 de 74HC595
Vin	Pin 1 Conector 7 para alimentación externa del Arduino Nano.

Figura 100. Vista inferior del circuito impreso

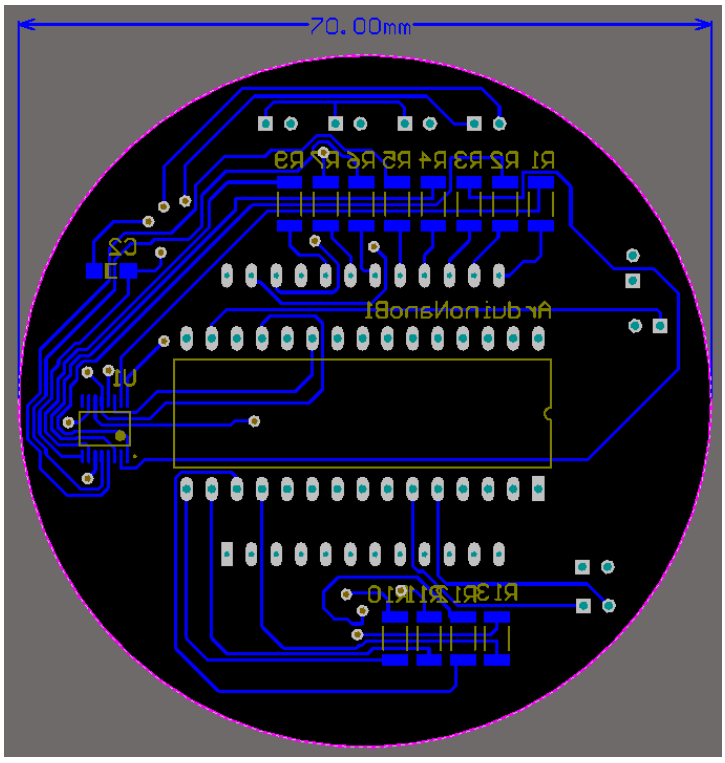


Figura 101. Vista inferior de circuito manufacturado y soldado

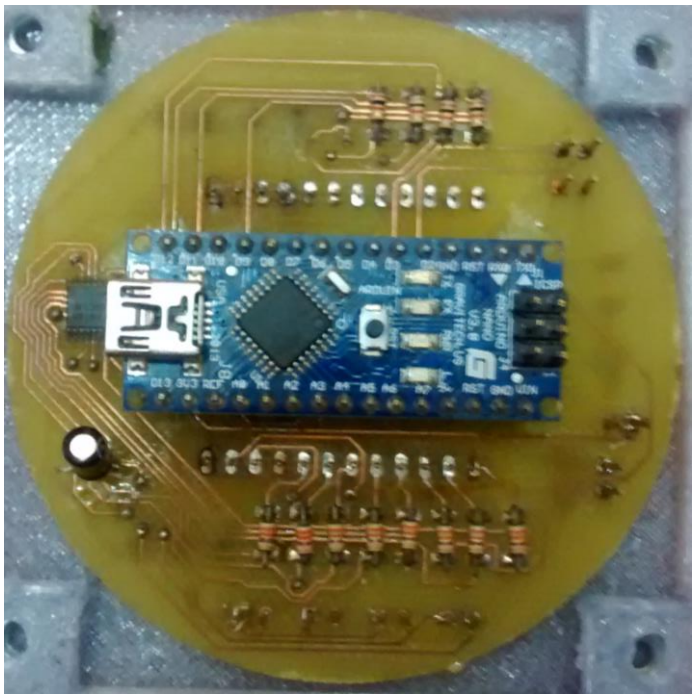


Figura 102. Vista superior del circuito impreso

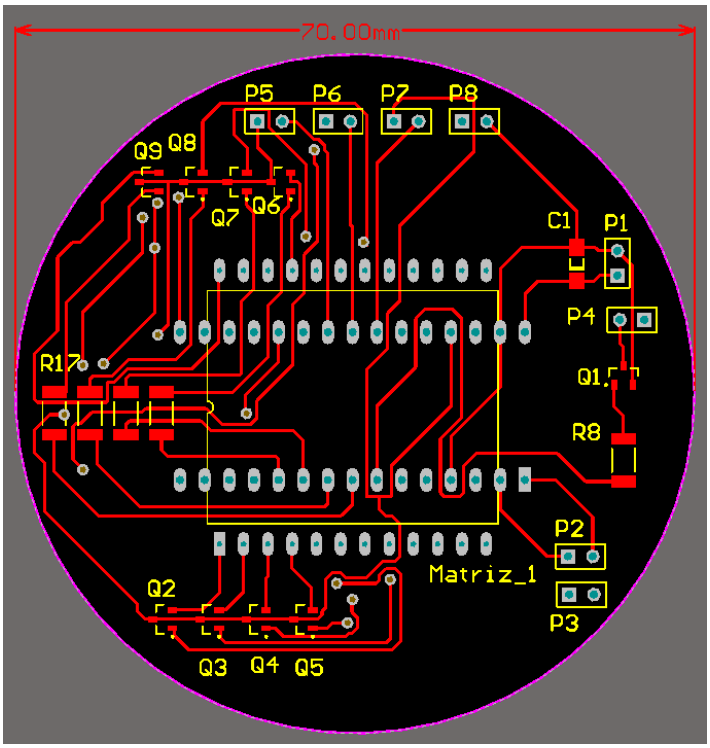


Figura 103. Vista superior de circuito manufacturado y soldado

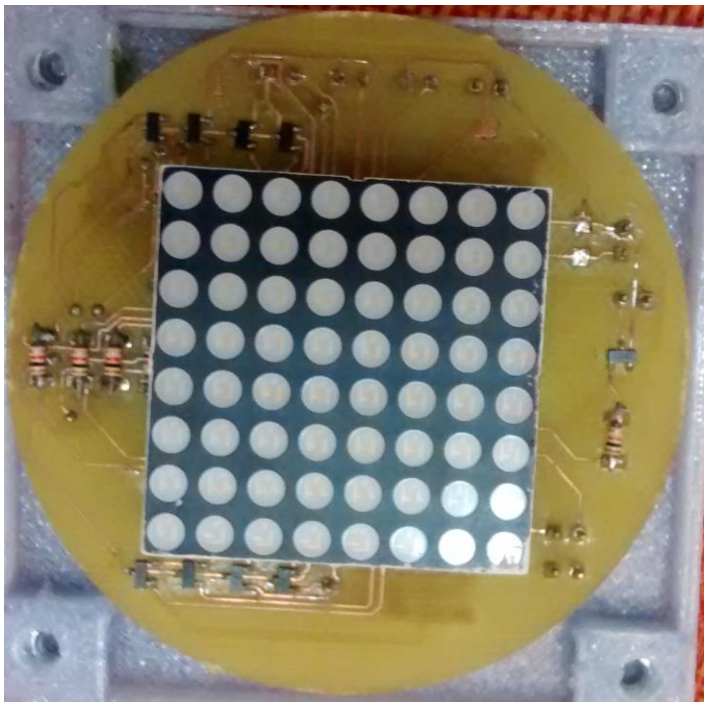


Figura 104. Brazalete encapsulado

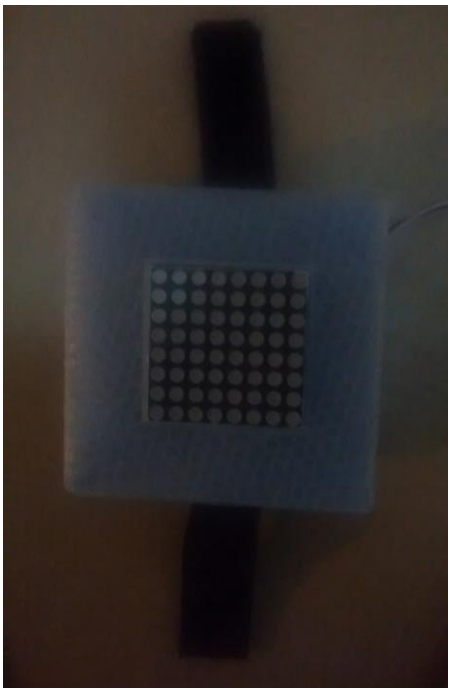


Figura 105. Vista de brazalete montado en brazo

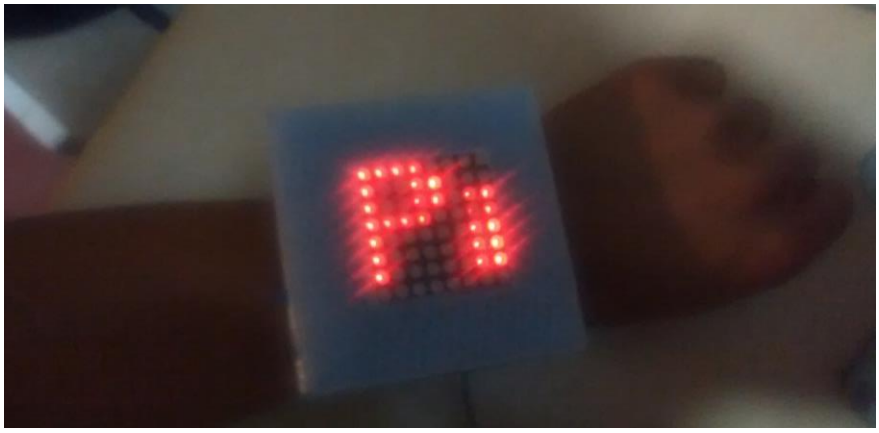


Figura 106. Inserción de módulo HC-05 en pulsera para brazalete

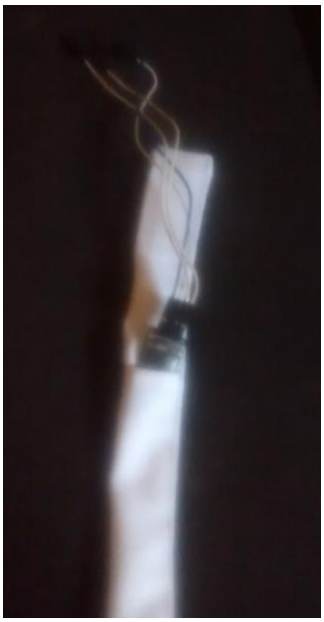


Figura 107. Pulsera con módulo Bluetooth en su interior

Figura 108. Diagrama de flujo código para Arduino Nano 1

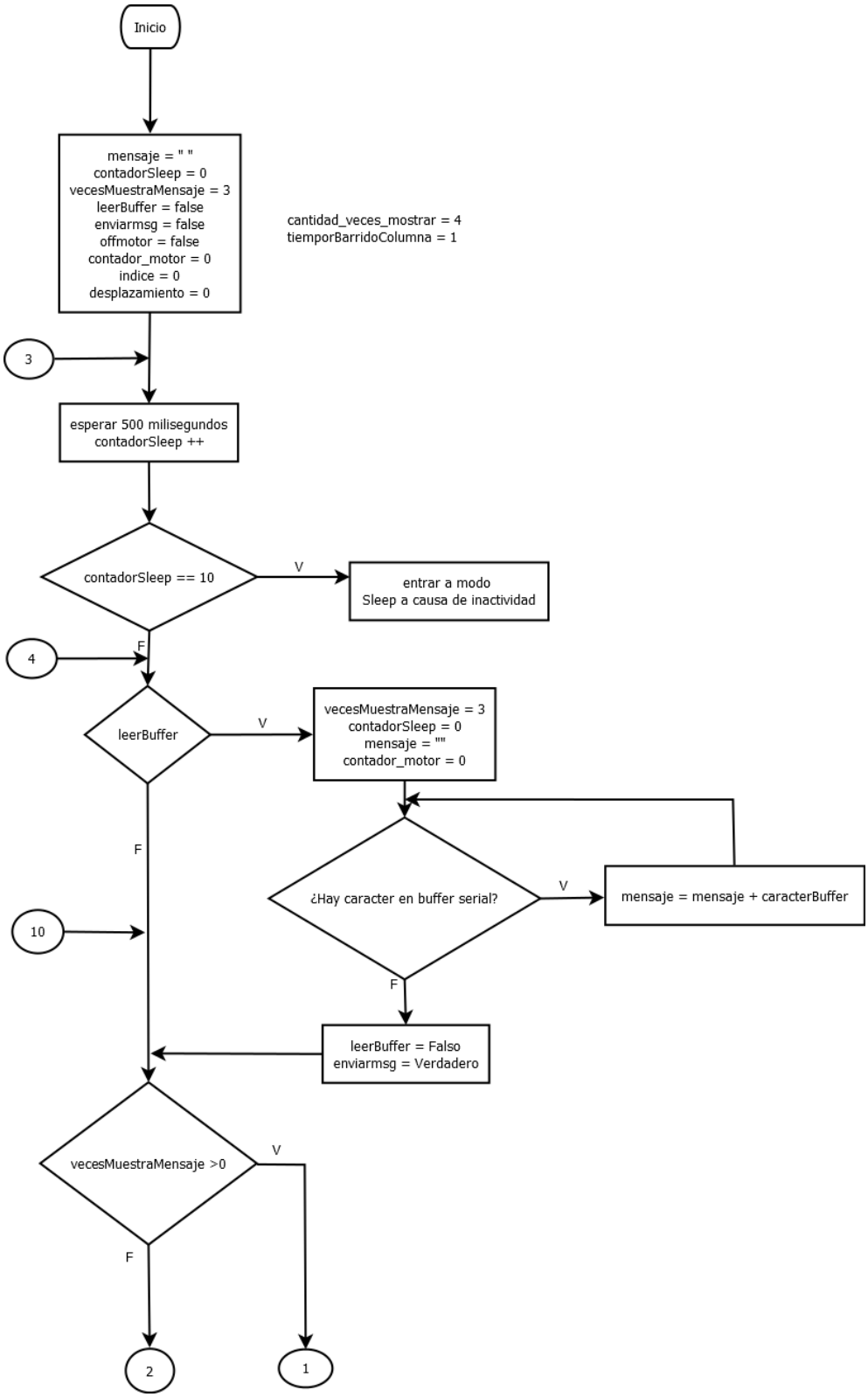


Figura 109. Diagrama de flujo código para Arduino Nano 2

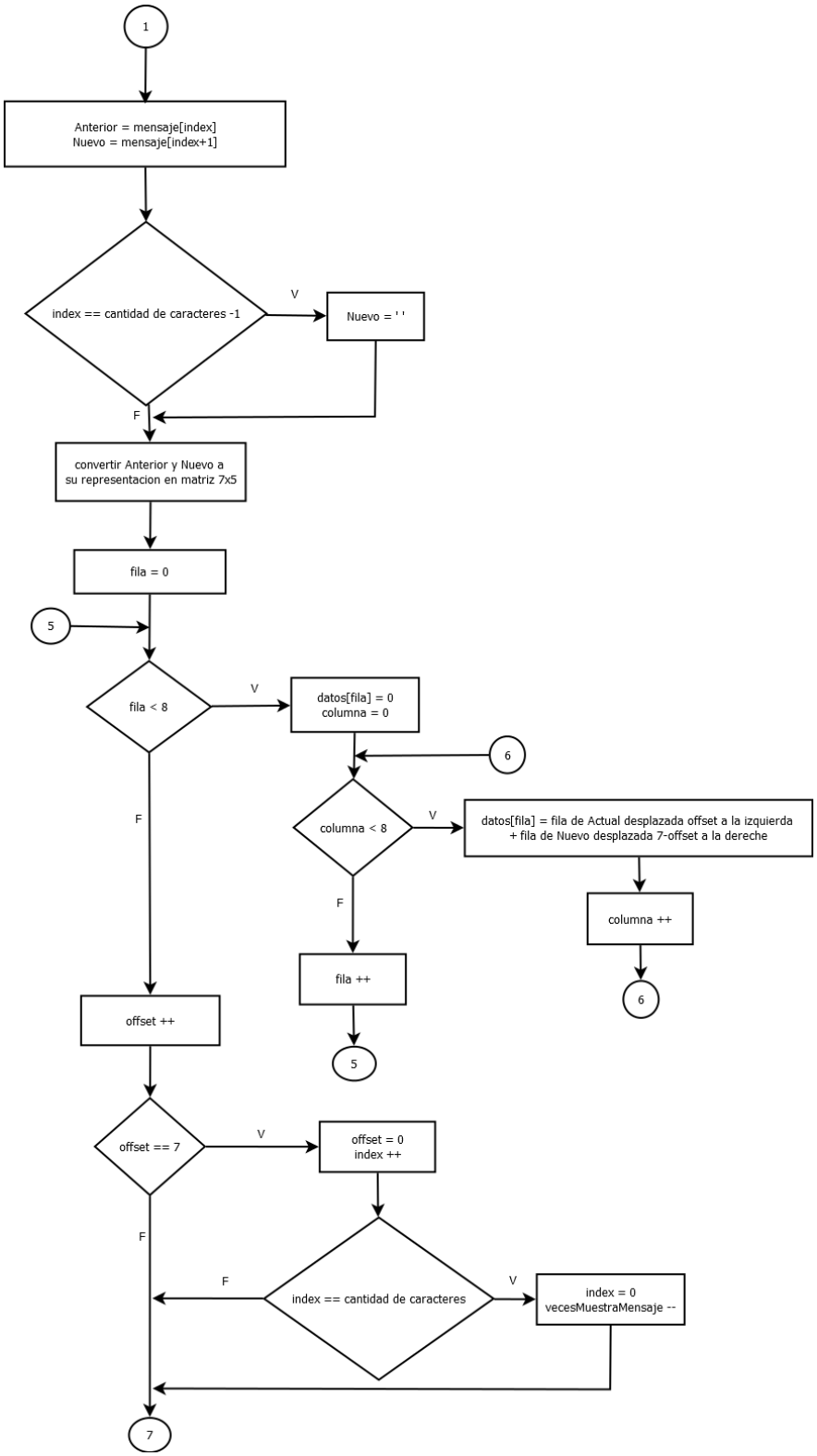


Figura 110. Diagrama de flujo código para Arduino Nano 3

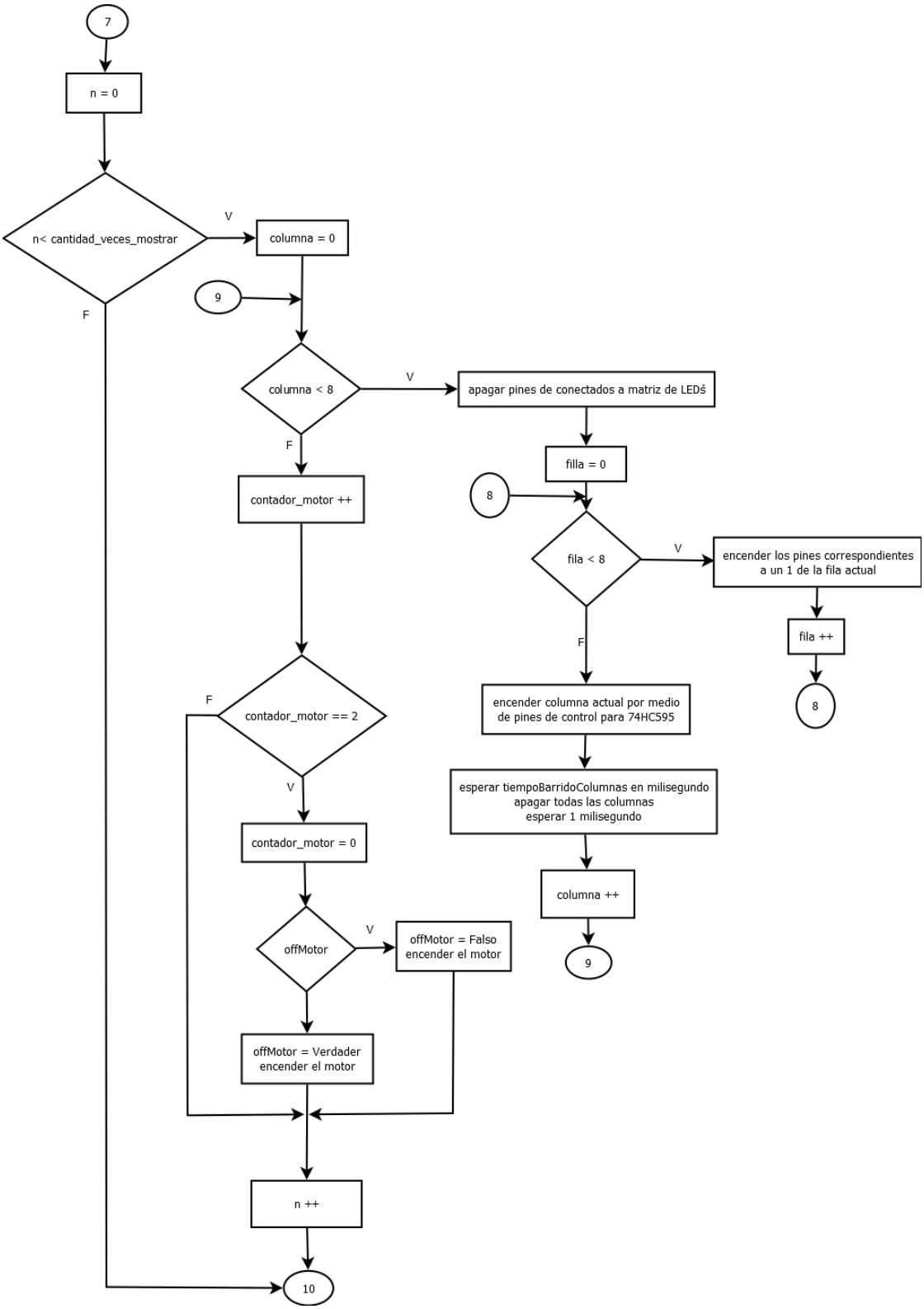


Figura 111. Diagrama de flujo código para Arduino Nano 4

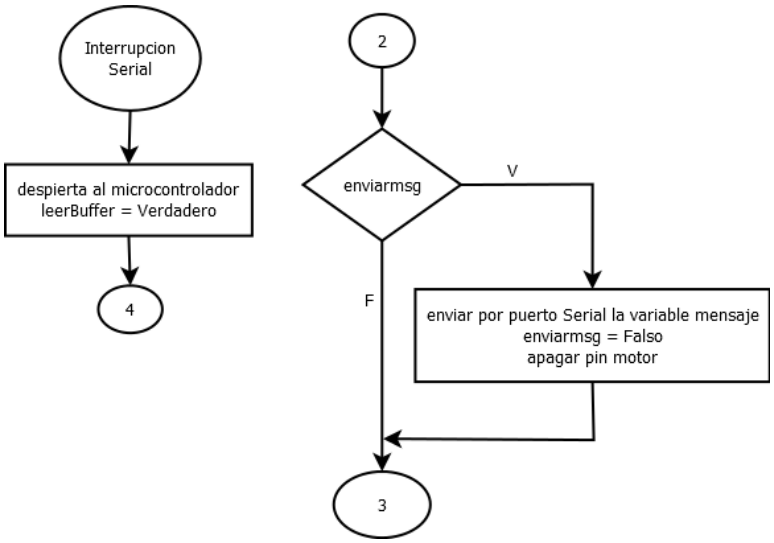
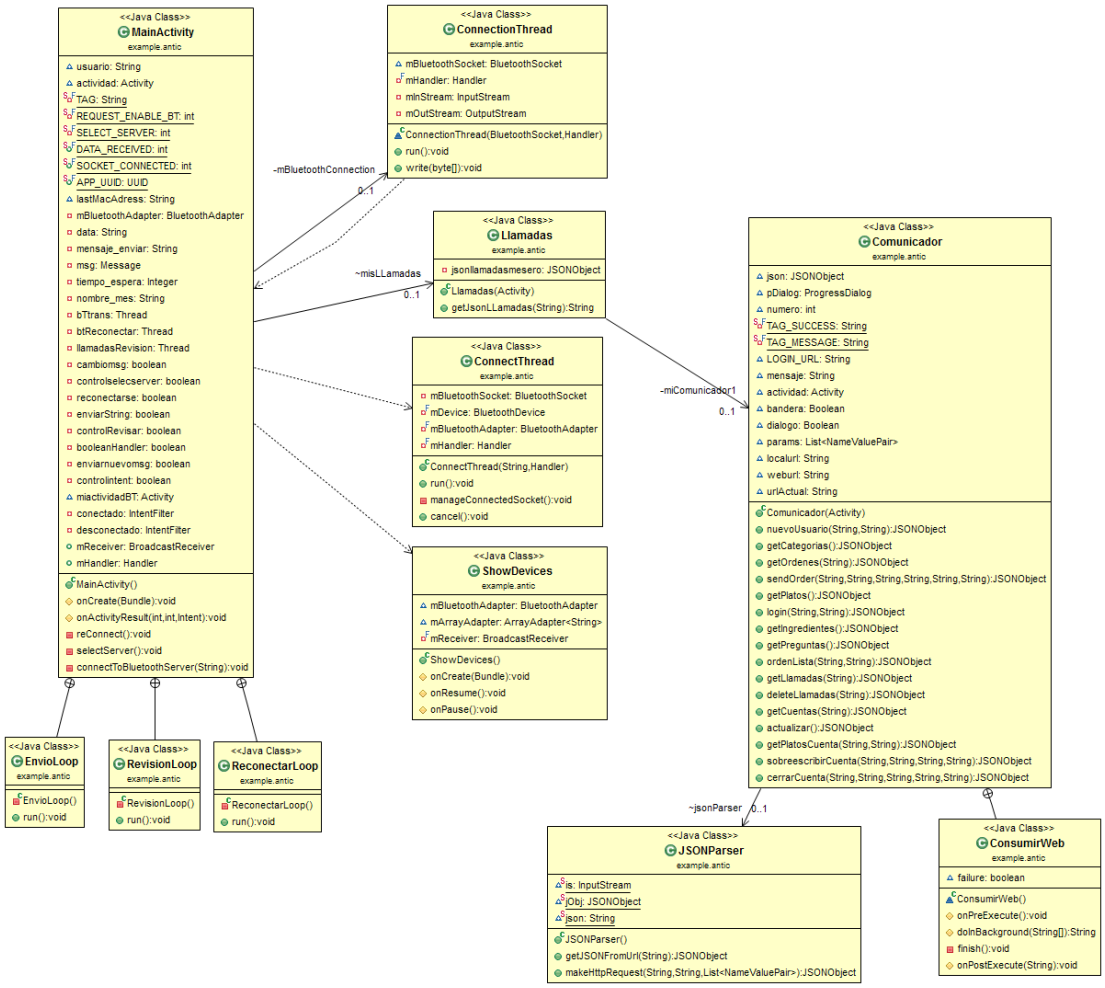


Figura 112. Diagrama de clases aplicación Bluetooth para Android



Cuadro 5. Mensaje enviado vía Bluetooth por parte de la aplicación en función de la información obtenida de la tabla de comandas finalizadas

Información obtenida al encontrar comanda terminada	Mensaje enviado vía Bluetooth
Cocina1 = 0	"Bar"
Cocina 1 = 1	"Cocina"
Cocina 2 ≠ Null	"C/B"

C. ANÁLISIS DE RESULTADOS

La Figura 94 muestra el despliegue deseado en la matriz de LED's. La Figura 95 muestra el despliegue en la matriz física. Como se puede apreciar el resultado es el esperado. Es importante mencionar que es posible desplegar tanto un solo carácter como una cadena de caracteres siempre y cuando esta no exceda los 64 bytes del bus serial del microcontrolador; estos caracteres se despliegan uno después de otro en un movimiento de corrimiento de derecha a izquierda.

La matriz utilizada, aunque fue la más pequeña disponible en el mercado local, resulto ser grande para la aplicación de brazalete. Las dimensiones de la matriz de LED´s fueron, en mayor parte, la causa de las dimensiones finales del brazalete.

Utilizando el circuito de la Figura 96 en combinación con el microcontrolador se desarrolló una secuencia de encendido y apagado de un motor vibrador. El propósito de esta secuencia es notificar al mesero que deber ver su brazalete para conocer de qué lugar proviene la comanda. Aunque es un sistema simple, es eficiente en lugares concurridos y con un ambiente bullicioso. Además, esta configuración permite alimentar al motor con una fuente regulada distinta a la alimentación del resto del brazalete. Con estructuras de altas dimensiones, comparadas con las dimensiones del motor vibrador, las vibraciones son casi imperceptibles, es decir, si el encapsulado del dispositivo es demasiado grande en comparación al tamaño del motor vibrador la percepción de la vibración se dificulta.

Manejar la conexión a voltaje de la matriz como se muestra en la Figura 97 permite la opción de alimentar los LED´s con un voltaje diferente al que otorgar un pin de salida del microcontrolador. Los transistores mostrados en la Figura 97 controlan las filas de la matriz. El integrado 74HC595 se encarga de la conexión a tierra de los cátodos de la matriz como se observa en la Figura 98. Los cátodos controlan las columnas de la matriz. La interconexión de todos los componentes se puede observar en la Figura 99.

El uso de un corredor de bits, IC 74HC595, simplifico el diseño del circuito impreso disminuyendo las conexiones de la matriz hacia el Arduino Nano. Se redujo espacio utilizando las versiones de encapsulado para soldadura de superficie (SMD) de resistencias, transistores, capacitores y circuitos integrados.

Para facilitar la depuración de errores se realizó una lista de referencia de las conexiones de los pines del microcontrolador con los componentes al cual conecta cada uno. En el Cuadro 4 se observa dicha lista.

El diseño del circuito impreso basado en el diagrama esquemático de la Figura 99 se puede observar en la Figura 100 y Figura 101. La manufactura resultante se observa en la Figura 101 y la Figura 102. Se utilizó una placa con dos caras de cobre, dejando en la parte superior a la matriz de LED´s y en la parte inferior a la placa de Arduino Nano. Esta disposición de componentes permitió reducir las dimensiones del diseño final. Sin embargo, debido al espacio requerido por la matriz, las dimensiones finales fueron grandes para la aplicación de brazalete. Otro componente que contribuyo al tamaño final del circuito impreso fue la placa de Arduino Nano: el espacio requerido lineal era similar al de la matriz de LED´s.

El encapsulado para la placa fue realizado en una impresora 3D. Este encapsulado permite el ajuste para la muñeca por medio de una cinta de velcro. El montaje del dispositivo con pulsera se observa en la Figura 104 y la Figura 105.

El módulo Bluetooth HC-05 no causo ningún aumento en las dimensiones del diseño del circuito impreso ya que este se conectaba por medio de buses cableados. Además fue posible esconder el modulo dentro de la pulsera que sostiene al dispositivo de notificaciones como se observa en la Figura 106 y la Figura 107.

El diagrama de flujo de la programación del microcontrolador se puede observar en la Figura 108, Figura 109, Figura 110 y Figura 111. El algoritmo desarrollado para el Arduino Nano es capaz de recibir una cadena de datos igual o menor a 64 bytes, tomar cada byte y relacionarlo con un arreglo de 7x5 bits para la representación de ese valor como carácter alfanumérico. Además, el microcontrolador entra en modo “sleep” luego de aproximadamente 5 segundos sin recibir datos en el puerto serial y “despierta” por medio de la interrupción de puerto serial para recibir el nuevo mensaje. Adicionalmente realiza el control del encendido y apagado de cada LED de la matriz de para mostrar el arreglo de bits asignado a cada byte recibido. La versión SMD del microcontrolador ATmega328P fue capaz de realizar todas las tareas requeridas para el funcionamiento del brazalete.

El algoritmo para el dispositivo con sistema operativo Android versión 4.1 fue desarrollado en JAVA. La interacción de las clases utilizadas se observa en la Figura 112. La aplicación es capaz de: realizar la conexión con el módulo Bluetooth HC-05; leer la tabla de la base de datos relacionada con las comandas terminadas; enviar el mensaje correspondiente según la información obtenida de la lectura de la tabla de comandas terminadas como se observa en el Cuadro 5. Si por algún motivo la conexión Bluetooth entre el brazalete y el dispositivo Android es interrumpida luego de haber logrado la conexión inicial; la aplicación es capaz de detectar esta desconexión; interrumpir el envío de mensajes almacenando el último mensaje no

enviado; restablecer la conexión con el ultimo dispositivo con él que tuvo conexión y reanudar el envío de mensajes iniciando con el mensaje no enviado.

Aunque las dimensiones finales del dispositivo de notificaciones inalámbrico para meseros lo hace incómodo para usarse como brazalete, este cumple con los requerimientos de alerta y despliegue de información necesaria para que el mesero sea capaz de dirigirse al lugar donde se encuentra su comanda terminada. Como segunda opción el dispositivo puede utilizarse en el cinturón del pantalón en lugar de la muñeca del mesero.

D. CONCLUSIONES

- La implementación de una matriz de 8x8 LED´s, permitió desplegar mensajes para informar al mesero a qué lugar dirigirse.

- Se logró llamar la atención del mesero en ambiente con niveles de ruido alto, utilizando el sistema de vibración diseñado.

- La implementación del módulo Bluetooth HC-05 permitió exitosamente la transmisión de información entre el dispositivo Android y el dispositivo de notificaciones inalámbrico para meseros.

- Se logró la comunicación entre dispositivo de notificaciones y la aplicación Android por medio del algoritmo desarrollado en lenguaje JAVA.

E. RECOMENDACIONES

- Se recomienda sustituir la matriz 8x8 de LED´s, de 3.8 cm x 3.8 cm, por un diseño propio de matriz 8x8 de LED´s con encapsulado para soldadura de superficie. La matriz utilizada en este proyecto posee LED´s RG, por lo que contiene pines no utilizados para esta versión del proyecto. Al realizar una versión propia de matriz 8x8 con LED´s de un único color y que además se encuentran en encapsulado de superficie código 0603, se estaría disminuyendo el espacio requerido de la matriz. Esto a su vez disminuye las dimensiones del circuito impreso.

- En lugar de utilizar la placa del Arduino Nano, se puede utilizar el microcontrolador ATmega328P con encapsulado de superficie directamente en el diseño del circuito impreso. Adicionalmente se deben hacer las modificaciones para agregar los componentes necesarios para el funcionamiento del microcontrolador y que este pueda ser reprogramable. Esta sustitución se recomienda con el fin de buscar disminuir las dimensiones del circuito impreso final.

- Se puede crear un espacio en el mismo circuito impreso para sujetar allí el motor vibrador, tomando como ejemplo el diseño de sujeción de los motores vibradores de celular. Tomar en cuenta que deben añadirse al diseño de la placa agujeros para la sujeción mecánica del circuito impreso al encapsulado que lo contendrá. Esto transmitirá de mejor manera la vibración mecánica por todo el encapsulado, mejorando la percepción de las vibraciones para el usuario.

IX. DESARROLLO DE UNA INTERFAZ DE PROGRAMACIÓN DE APLICACIONES Y ESTRUCTURACIÓN DE BASE DE DATOS

A. METODOLOGÍA

La metodología utilizada para llevar a cabo este proyecto consistió en una etapa inicial de investigación, cuando aún no se había definido con qué restaurante se iba a realizar el proyecto en mente se hicieron entrevistas a otros restaurantes. Esto se hizo para conocer el proceso que lleva una orden en cada lugar, las diferentes formas de manejar la información importante de la misma. También se preguntó sobre posibles problemas o situaciones que atrasan el proceso para entregar una orden. Con esta información se empezaron a definir los diferentes módulos, así como las funciones de cada uno. A partir de esto se comenzaron a realizar pruebas comunicando una aplicación para el sistema operativo Android con una base de datos en *MySQL*.

Con el proyecto completo más definido se habló con el gerente del restaurante Antic para solicitar permiso para basar el proyecto en su restaurante y utilizar su menú, su imagen y su información de contacto en el proyecto. Se aprovechó la oportunidad para preguntar sobre el proceso de órdenes para conocer las necesidades específicas de Antic. También se preguntó sobre la información que deseaba almacenar sobre cada orden o mesero, qué expectativas tenían para un proyecto de éste tipo y qué tipo de retroalimentación les gustaría recibir para poder modelar la base de datos.

Para el proceso de diseño de la base la base de datos se hizo utilizando la información que obtenida previamente, sin embargo esto solo se utilizó para saber qué información era indispensable almacenar. El número de tablas y el resto de las características se decidieron dependiendo de las operaciones que se necesitaban para cada aplicación. Con esto se revisó la relación entre los distintos módulos del proyecto la cuál puede apreciarse en la Figura 118.

Al definir la base de datos se comenzó a agregar funciones a la librería en Android, las funciones necesarias se definieron después de la visita Antic y se fueron programando según las prioridades de cada módulo. Los demás integrantes del grupo fueron libres de sugerir funciones que ellos consideraron necesarias para la realización de su módulo.

B. RESULTADOS

Figura 113. Diagrama de base de datos local

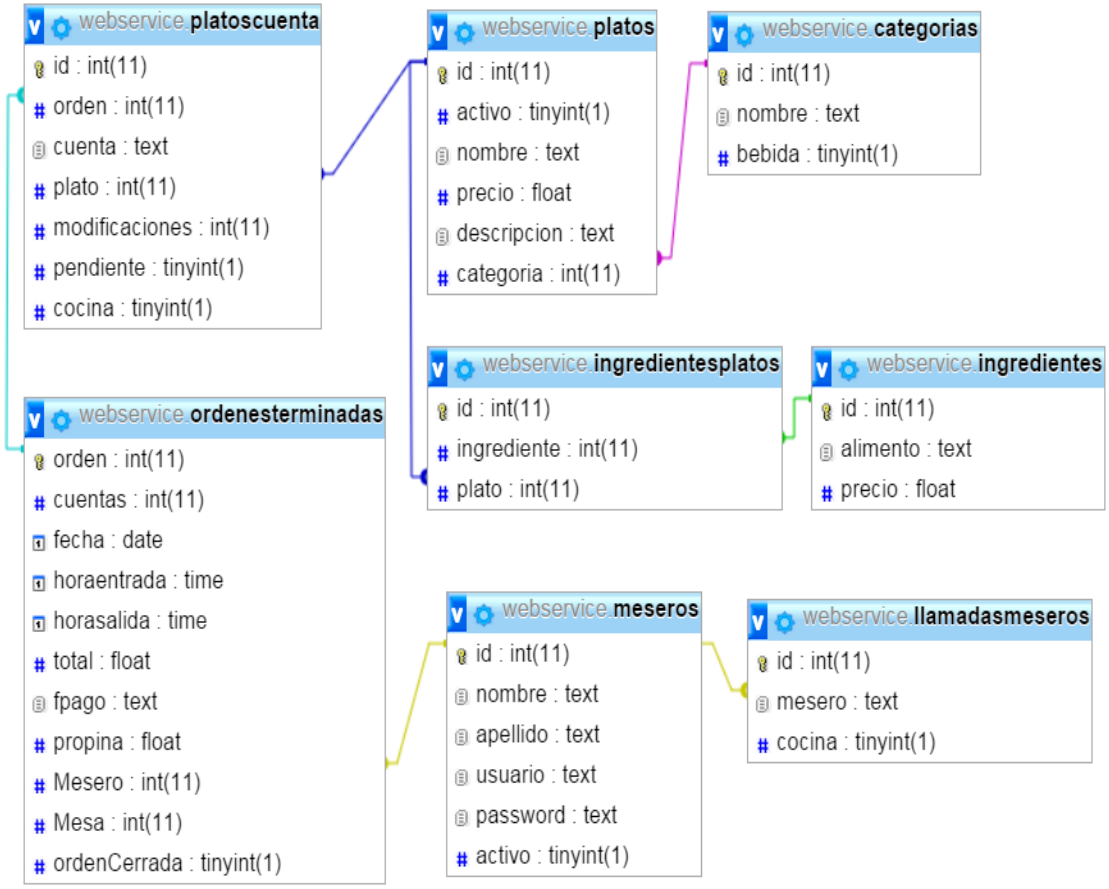


Figura 114. Diagrama de base de datos online

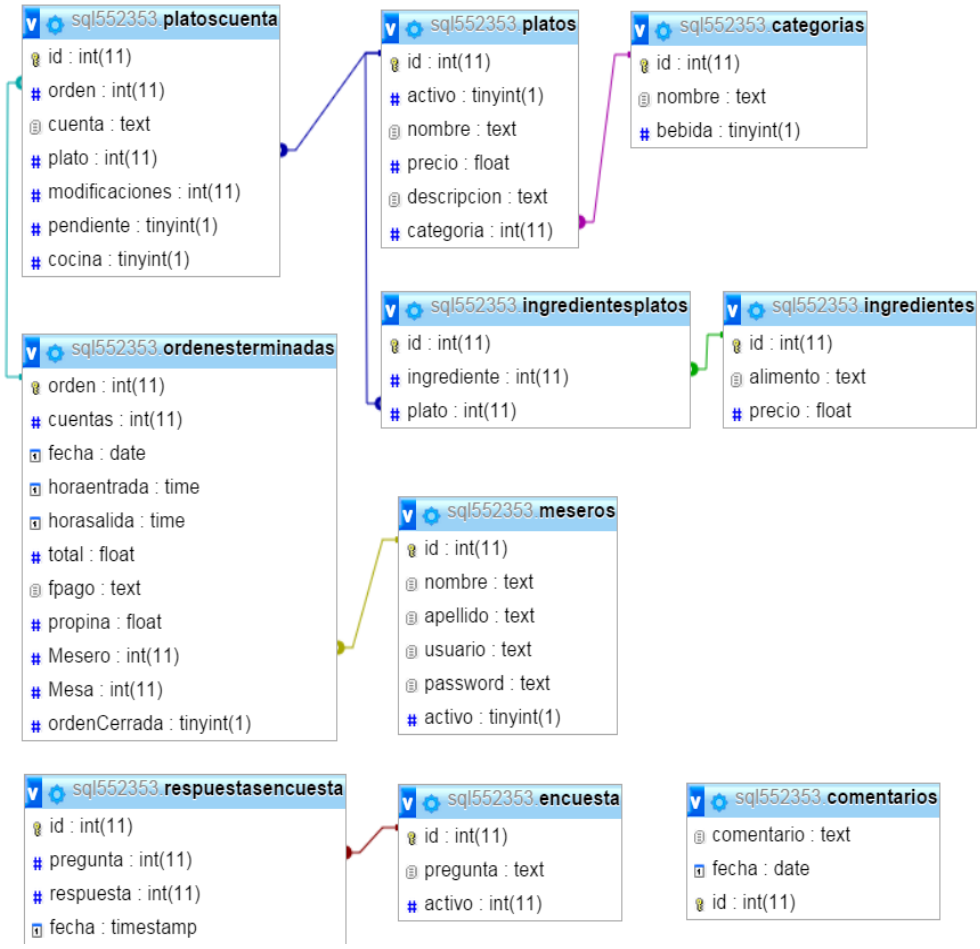
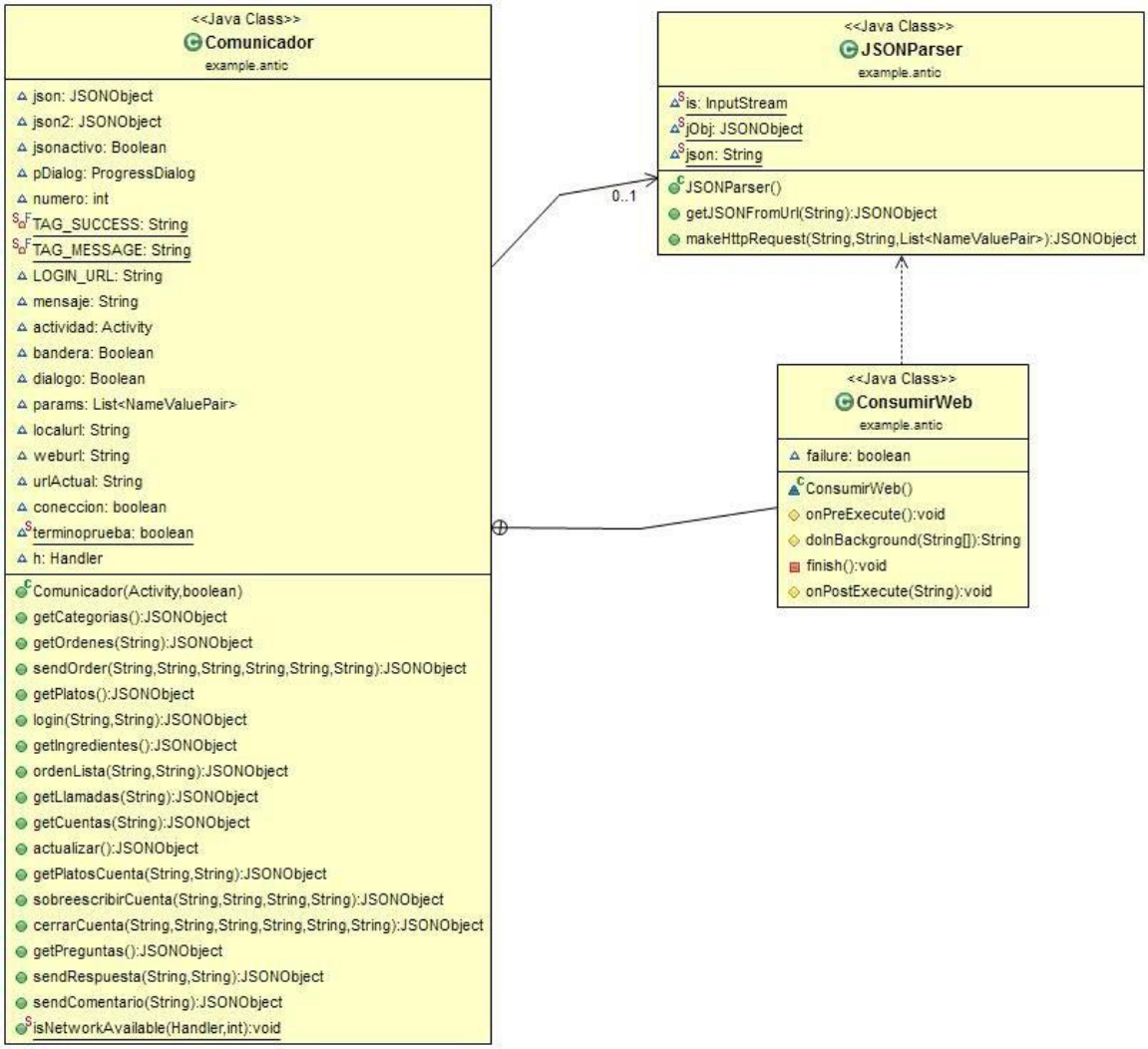


Figura 115. Diagrama UML de Comunicador



Cuadro 6. Resumen de funciones de clase Comunicador para Android

Función	Descripción	PHP Relacionado	Tipo de conexión
getCategorias	Se obtienen todas las categorías del menú.	getCategorias.php	Depende de la aplicación
getOrdenes	Se obtienen las órdenes activas ya sea de bar o cocina.	getOrdenes.php	Local
sendOrder	Se envían los datos de una orden ya sea nueva o que ya esté activa.	sendOrder.php	Local

Continuación Cuadro 6

Función	Descripción	PHP Relacionado	Tipo de conexión
getPlatos	Se obtienen todos los platos del menú.	getPlatos.php	Depende de la aplicación
login	Se confirma que los datos para conectar a un usuario sean correctos.	login.php	Local
getIngredientes	Se obtiene una lista con todos los ingredientes que hay en el menú.	getIngredientes.php	Depende de la aplicación
ordenLista	Se indica que la orden ya está lista y se llama al mesero.	ordenLista.php	Local
getLlamadas	Se obtienen todas las llamadas para un mesero en específico.	getLlamadas.php	Local
getCuentas	Se obtienen todas las cuentas activas asociada a una mesa.	getCuentas.php	Local
actualizar	Se actualizan los datos a los que se tienen acceso con la versión más reciente.	buscarOrdenesActivas.php getMaximos.php getDatosOnline.php sobreescribirDatosLocal.php getDatosLocal.php sobreescribirDatosOnline.php	Local
getPlatosCuenta	Se obtienen todos los platos solicitado por una cuenta en específico.	getPlatosCuenta.php	Local
sobreescribirCuenta	Se borran los platos de una cuenta y se sustituyen por un nuevo listado.	sobreescribirCuenta.php	Local
cerrarCuenta	Se cierra una cuenta.	cerrarCuenta.php	Local
getPreguntas	Se obtienen todas las preguntas para la encuesta.	getPreguntas.php	Online

Continuación Cuadro 6

Función	Descripción	PHP Relacionado	Tipo de conexión
sendRespuesta	Se envía la respuesta a una pregunta individual.	sendRespuesta.php	Online
sendComentario	Se envía el comentario de retroalimentación.	sendComentario.php	Online

Cuadro 7. Entradas, salidas y procesos de funciones de clase Comunicador parte 1

Función	Entradas	Salidas	Proceso
getCategorias	Ninguna	Devuelve todas las categorías en la base de datos.	Se solicita a la tabla categorías todos sus campos.
getOrdenes	Cadena de texto que indica sí están solicitando ordenes de cocina o bar, donde “1” es cocina y “0” es bar.	Retorna los números de orden, los productos pendientes, sus modificaciones y el nombre del mesero responsable; de todas las órdenes de cocina o del bar.	Se solicita de la tabla ordenes_activas los valores de los campos orden, productos, ingredientes y Mesero de todos los registros donde el campo cocina sea igual al parámetro enviado por el usuario. Se borran los campos de productos e ingredientes que fueron leídos para no realizar lecturas repetidas.
sendOrder	Nombre del comensal, lista de productos, lista de las modificaciones de los productos, número de mesa e indicador si la orden es de cocina o bar.	Un indicador denominado como “success” el cual es 1 sí se logró la operación y 0 si hubo un error.	Sí no existe una orden activa con esta mesa y este nombre se crea una nueva tanto en ordenes_activas como en ordenesterminadas y se almacenan los datos de fecha y hora de inicio, también se asigna el número de orden. Sí ya existe una orden activa en esta mesa pero con otro nombre solamente se crea la nueva cuenta en órdenes_activas y sí ya existe una orden para esa mesa y ese nombre se concatenan los nuevos productos a la orden ya existente.

Continuación Cuadro 7

Función	Entradas	Salidas	Proceso
getPlatos	Ninguna	La lista de platos completa incluyendo la categoría a la que pertenecen y sus precios.	Se solicita de la tabla platos todo el contenido de los platos activos.

Cuadro 8. Entradas, salidas y procesos de funciones de clase Comunicador parte 2

Función	Entradas	Salidas	Proceso
login	El usuario y contraseña del mesero que desea ingresar a la aplicación.	Un indicador denominado como “success” el cual es 1 sí se logró la operación y 0 si hubo un error.	Se revisa en la tabla “meseros” que exista el usuario enviado que coincida con la contraseña enviada.
getIngredientes	Ninguna	Lista completa de ingredientes.	Se obtiene de la tabla ingredientes el nombre e identificador de los ingredientes.
OrdenLista	Nombre del mesero e indicador si la función se está llamando desde cocina o bar.	Indicador “success” el cual es 1 sí se logró la operación y 0 si hubo un error.	Se ingresa un nuevo registro en la tabla llamadasmeseros que indica el nombre del mesero y desde donde fue enviada la llamada.
GetLlamadas	Nombre del mesero.	Devuelve todas las llamadas pendientes incluyendo sí son de bar o cocina.	Se solicitan todos los registros de la tabla llamadasmeseros cuyo campo nombre sea el mismo al nombre ingresado.
GetCuentas	Número de mesa.	El nombre asociado a todas las cuentas abiertas actualmente en esta mesa.	Se obtienen de la tabla ordenes_activas los nombres de todos los registros cuyo valor mesa sea igual al ingresado.

Continuación Cuadro 8

Función	Entradas	Salidas	Proceso
Actualizar	Ninguna.	Indicador "success" el cual es 1 sí se logró la operación y 0 si hubo un error.	Se obtienen los valores id más grandes en las tablas categorías, ingredientes, platos, meseros y ordenes activas en la base de datos online y local, se comparan y se agregan a las tablas locales los registros de las tablas online que posean un id mayor al de la tabla. Se realiza el mismo procedimiento con ordenesterminadas pero realizando la copia desde el local hacía el online.

Cuadro 9. Entradas, salidas y procesos de funciones de clase Comunicador parte 3

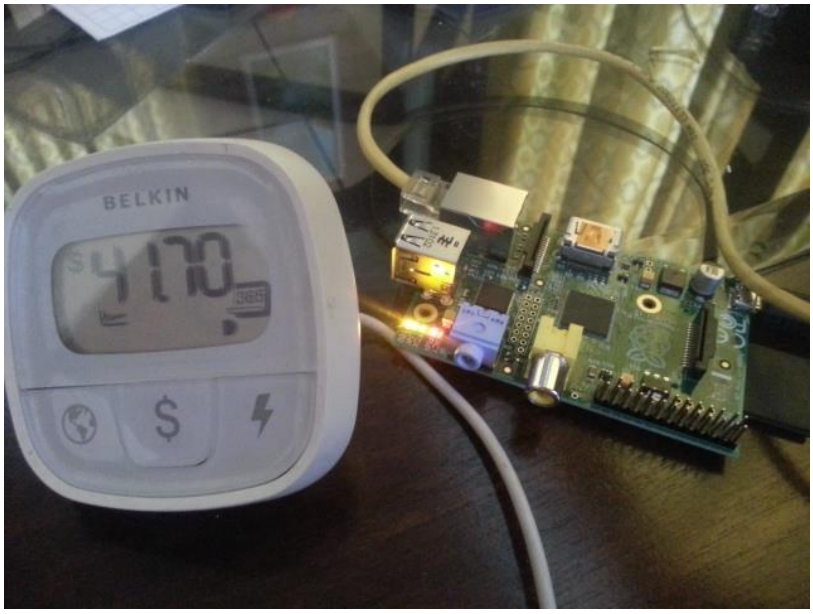
Función	Entradas	Salidas	Proceso
getPlatosCuenta	El número de mesa y el nombre de la cuenta.	El id de los platos ordenados por esa cuenta hasta el momento.	Se solicita de la tabla ordenes_activas el valor del campo productostotales, de los registros que cumpla con tener el nombre y la mesa indicada.
sobreescribirCuenta	El número de mesa, el nombre de la cuenta y la nueva lista de platos.	Indicador "success" el cual es 1 sí se logró la operación y 0 si hubo un error.	Se sobrescribe la columna platostotales del registro que cumpla con tener el mismo nombre y número de mesa, en la tabla ordenes_activas.

Continuación Cuadro 9

Función	Entradas	Salidas	Proceso
CerrarCuenta	El nombre de la cuenta, el número de mesa, la forma de pago, el total, la propina y un indicador que indica si al cerrar esta cuenta también se debe de cerrar la orden.	Indicador "success" el cual es 1 sí se logró la operación y 0 si hubo un error.	Se obtienen los valores de productos e ingredientes de la tabla ordenes_activas y se concatenan a sus respectivos campos en la tabla ordenesterminadas, se suma el total al total almacenado hasta el momento, se guarda la forma de pago, se suma una unidad al número de cuentas al orden y sí es la cuenta pasará a estar cerrada se coloca la hora actual en el campo horasalida y se marca el campo indicando que la orden está terminada. Por último se borran los registros de esa cuenta y orden de la tabla ordenes_activas.
GetPreguntas	Ninguna	Se devuelven todas las preguntas activas, incluyendo sus ids.	Se solicitan todos los id y pregunta de los registros en la tabla encuesta que cuenten con un "1" en su valor de activo.
sendRespuesta	Id de la pregunta, y la calificación dada por el usuario.	Indicador "success" el cual es 1 sí se logró la operación y 0 si hubo un error.	Se suma una unidad al campo que coincida con la calificación ingresada por el usuario, y se modifica el valor de la casilla fecha por la fecha actual.
sendComentario	Comentario del usuario.	Indicador "success" el cual es 1 sí se logró la operación y 0 si hubo un error.	Se agrega un nuevo registro con el comentario ingresado por el usuario y la fecha actual.

Figura 116. Consumo energético Raspberry Pi

Figura 117. Gasto anual de energía por Raspberry Pi



Cuadro 10. Costos del módulo

Componente	Costo inicial	Costo anual
Raspberry Pi modelo B	Q281.28	Q41.70
Router Almond Plus como extensor de señal	Q800.00	Q113.00
Servidor en 000webhost.com	Q0.00	Q0.00
Total	Q1,081.28	Q154.70

C. ANÁLISIS DE RESULTADOS

Las Figura 113 y Figura 114 muestran los diagramas para las bases de datos local y *online* respectivamente, éstos muestran que ambas bases de datos son distintas y que ninguna es una réplica exacta de la otra. Esto se hizo debido a que cada base se comunica con diferentes módulos. Por ejemplo la aplicación de clientes solamente se comunica con la base de datos *online* para enviar las respuestas de las encuestas, por lo que las tablas relacionadas con este proceso serian inútiles en la base de datos local, el mismo caso aplica para la tabla "llamadasmeseros" que solamente se utiliza dentro del restaurante por lo que no está en la base de datos *online*. Teniendo en cuenta esto se dejó de perseguir replicar ambas bases de datos.

Ambas bases de datos fueron normalizadas principalmente para evitar información duplicada ya que como se utiliza un servidor gratuito para la base de datos online se cuenta con un espacio limitado. Por esto se decidió reducir el tamaño de la base de datos para poder continuar utilizando una opción gratuita para el *alojamiento* de la información *online*.

La conexión a las bases de datos se hizo vía un *webservice* con archivos PHP, debido a que no todos los módulos contarían con sistema operativo Android por lo que se prefirió esta opción que permitía reutilizar los mismos archivos con varios módulos. El módulo que no utilizaría Android fue eliminado ya avanzado el proyecto por lo que no se cambió la implementación de lo que ya se tenía.

Se utilizó MySQL en acuerdo con la persona encargada de realizar el módulo de la página *web*, debido a que ambos proyectos interactuaron con la base de datos online. Para esta decisión se tomó en cuenta la disponibilidad de la información ya que MySQL es la opción más popular actualmente por lo que se contó con bastante documentación para su manejo e implementación.

La base de datos *online* debía permitir almacenar y agregar nuevos productos al menú, almacenar las respuestas de las encuestas en la aplicación para clientes, agregar nuevos meseros y tener un registro de las órdenes. La información que se decidió almacenar de las cuentas se eligió tomando en cuenta comentarios de los gerentes del restaurante Antic. La base de datos local debía permitir obtener el menú, los meseros que están trabajando actualmente, crear y cerrar órdenes y manejar las llamadas a los meseros. Muchas de las necesidades de ambas bases de datos están interconectadas en el sentido en que mientras una debe de poder editar una tabla la otra debe de poder solamente ingresar a ésta. En el caso de la creación de un plato nuevo, esto se hace desde la página *web* por un gerente y al hacer esto se agregan registros las tablas "platos" e "ingredientesplatos", estos cambios se hacen en la base de datos *online* por lo que la base de datos local debe ser actualizada con esta información para poder brindar a la aplicación de meseros el menú más actualizado. Esto se da en varios casos y son estas tablas las que se encuentran en ambas bases de datos, y las que se deben de replicar cada cierto tiempo, esto se hace cada vez que se abre una nueva orden en la aplicación de meseros. Esto se hace de en este punto debido a que se tomó prioridad que se pudieran ordenar los platos nuevos lo más pronto posible desde que son agregados. Por lo que sin importar en qué momento se agregue un plato todas las órdenes iniciadas después de eso ya contarían con la opción de agregarlo, contrario a que se realizara al final de un orden ya que la primera orden después del cambio no percibiría la actualización.

Estas tablas que satisfacen necesidades para ambas bases de datos son las que se encuentran en ambas y son las que son necesarias replicarse para contar con la información más reciente para todos los módulos. Las tablas "categorías", "platos", "ingredientes", "ingredientesplato" y "meseros" solo pueden modificarse vía *online* por lo que las mismas tablas en la base de datos local son una réplica de éstas, mientras que las tablas "ordenesterminadas" y "platoscuenta" solamente son modificadas localmente por lo que la réplica se encuentra en el servidor *online.* Las tablas que se encuentran solo en una base de datos no presentan información relevante para la otra por lo que no son replicadas para no desperdiciar espacio.

La principal razón para realizar réplicas de éstas tablas, especialmente de "categorías", "platos", "ingredientes", "ingredientesplato" y "meseros" es para que el sistema no estuviera limitado por la disponibilidad del internet, ya que los gerentes del restaurante Antic nos informaron que no contaban con señal todo el tiempo. En el caso de no haber internet en el restaurante la aplicación de meseros, de cocina, de bar y el reloj inteligente continúan funcionando de la manera habitual solamente no se podrá realizar las actualizaciones a las bases de datos *online*, por lo que si se agrega un nuevo plato este no aparecerá hasta que se reanude la conexión, así como no se actualizarán las ordenes terminadas para verlas en línea. Las actualizaciones pendientes se realizaran cuando se vuelva a contar con conexión a internet dentro del restaurante.

Para almacenar la información del menú se utilizan 4 tablas, “categorías”, “platos”, “ingredientesplatos” e “ingrdientes”. La tabla “categorías” almacena el nombre de las diferentes categorías de alimentos que ofrece el restaurante, por ejemplo platos fuertes o entradas, especifica si es una categoría de bebidas y se le asigna un identificador único denominado id. La tabla platos guarda del nombre de plato, su precio, una descripción, a qué categoría pertenece utilizando el id de ésta y también se le asigna un id único a cada plato. Es posible que los clientes deseen realizar modificaciones a un plato al momento de ordenarlo, por ejemplo especificar el término de la carne o indicar que se prefiere una bebida sin hielo, se creó la tabla “ingredientes” esta le asigna un id a cada posible modificación así como un precio en caso de ser algo que requiera un aumento. Como un plato puede tener varias opciones intercambiables, en un plato fuerte puede modificarse la guarnición y el término de la carne, y una modificación puede ser válida para varios platos, ordenar una bebida sin hielo debería de ser posible para todas las bebidas que contengan hielo, se realizó una tabla intermedia para obtener esta relación. La tabla “ingredientesplato” es una tabla intermedia encarga de realizar una relación de muchos a muchos entre la tabla “platos” y la tabla “ingredientes” donde cada registro almacena el id de un plato y el id de un ingrediente, esto indica que al plato con ese id se le puede aplicar la modificación con el id que está en el registro. Esto significa que si un plato tiene 5 posibles modificaciones este tendrá 5 registros en la tabla “ingredientesplatos” con 5 diferentes ingredientes. Con estas relaciones al recorrer la tabla platos se puede saber el nombre, la descripción, el precio, la categoría y los ingredientes intercambiables de cada plato obteniendo así toda la información importante del menú.

Las tablas “ordenesterminadas” y “platoscuenta” son las necesarias para abrir una nueva orden, enviar la información a cocina y cerrar la orden. Al momento de abrir una nueva cuenta se crea un registro en “ordenesterminadas” donde se le asigna un número de orden, se almacena la fecha, la hora de entrada, el id del mesero realizando la orden y se especifica que la orden aún no está cerrada con el campo “ordenCerrada”. Esta orden aún no tiene cuentas ni platos asociadas, esta información se almacena en la tabla “platoscuenta” la cual realiza una relación de uno a muchos, relacionando la orden con varios platos. En esta tabla se almacena qué platos fueron ordenados en cada cuenta y ciertas propiedades, también cuenta con un valor que indica si el plato ya fue enviado a cocina o bar ara evitar que sea enviado más de una vez. Los campos de “cuentas”, “horasalida”, “total”, “fpago” y “propina” de “ordenesterminadas” se van actualizando conforme se van cerrando cuentas, esto se hace cuando el cliente realiza su pago, por lo que estos datos se van actualizando conforme pasa el tiempo. Cuando se paga la última cuenta se indica que la orden ya ha sido cerrada en su respectivo campo.

El diagrama UML en la Figura 115 resume las clases de la librería Comunicador, la cual es un proyecto en Java que se puede utilizar en otros proyectos al importar la librería en Eclipse. Comunicador permite a aplicaciones de Android interactuar con las bases de datos *online* y local del restaurante, y se diseñó teniendo en cuenta las necesidades de los demás integrantes del grupo. Los usuarios de la librería deben

crear un objeto Comunicador donde en su constructor deben especificar si se utilizará para un módulo local u *online*, esto se hizo de esta manera debido a que la aplicación de clientes y la de meseros comparten varias necesidades, por lo que se reutilizaron varios métodos. Las información para realizar la conexión con la bases de datos ya se encuentra almacenada dentro de la librería por lo que al indicar si se utilizará *online* o no, solo se le indica a la librería qué datos de conexión utilizar.

Los métodos de la librería devuelven la información en formato JSON, esto se acordó con los demás integrantes del grupo ya que al realizarlo así se le permitió a cada usuario de la librería manejar la información como mejor se adecuara a sus necesidades. Por ejemplo la persona que diseñó la aplicación de clientes decidió crear objetos más complejos con la información mientras que para el reloj inteligente se utiliza la información directamente desde el JSON.

El Cuadro 6 muestra con qué base de datos interactúa cada función de Comunicador. Las funciones que se utilizan para obtener el menú completo se utilizan tanto en la aplicación de clientes como en la de meseros y en la de cocina y bar, para este caso es que se utiliza el parámetro del constructor que indica que información de conexión utilizar. Los demás métodos solamente tienen utilidad dentro o fuera del restaurante, todo lo relacionado con la creación de órdenes y las llamadas a meseros se conecta únicamente con la base de datos local, mientras que lo relacionado con la encuesta y comentarios es solamente *online* debido a que solo la aplicación de clientes cuenta con la sección de retroalimentación y ésta está pensada para ser utilizada fuera del restaurante.

En los Cuadro 7, Cuadro 8 y Cuadro 9 se explican los procesos que se llevan a cabo durante cada función, éstos se hacen en su mayoría del lado del servidor en código PHP, ya que varías funciones afectan a más de una tabla. Esto facilito que todos los módulos pudieran seguir siendo trabajados en paralelo porque al momento de una situación inesperada, o de realizar alguna optimización de algún proceso era posible realizarla del lado del servidor, sin que los demás integrantes del grupo tuvieran que agregar o editar algo en sus respectivos proyectos.

La Figura 116 muestra el consumo de energía de la Raspberry Pi en Watts y la Figura 117 muestra su gasto anual en energía con una cuota de Q.1.69/kWh, esta cuota se obtuvo de un recibo por consumo eléctrico del mes de septiembre 2014. Estas mediciones se realizaron utilizando un medidor de potencia marca Belkin durante uno de los días donde se realizó el mayor número de pruebas con el fin de simular la actividad en un restaurante. Este aparato utiliza la información de consumo a lo largo del día para proveer un mejor estimado en su lectura.

El Cuadro 10 muestra un resumen de los costos de los materiales y costos necesarios para agregar la comunicación al sistema completo, en éstos no se incluye el costo de los demás módulos del sistema.

También se tuvo en cuenta que el restaurante Antic ya cuenta con un *Router* sin embargo éste no cubre toda el área del restaurante por lo que se agregó un segundo dispositivo como extensor de rango. Este modelo se utilizó debido a que ya se contaba con el dispositivo y que el tiempo con el que se contó fue bastante limitado como para adquirir uno diferente.

D. CONCLUSIONES

- Se logró entregar el API a los demás integrantes del grupo distribuyéndola como un proyecto de Java para Android.

- El costo inicial de agregar comunicación al sistema para un restaurante en las condiciones de Antic es Q.1081.28 más un gasto anual de electricidad de Q.154.70.

- Se alcanzó el objetivo de montar un servidor local en una Raspberry Pi.

- La librería realizada permitió a los dispositivos obtener acceso y editar a los contenidos de la base de datos abstrayendo la implementación de los procesos para realizar los mismos.

- Se hicieron dos bases de datos distintas debido a que cada una se comunicó con diferentes módulos por lo que atendía distintas necesidades, sin embargo aún comparten la mayoría de las tablas.

- Las tablas que se replican cada cierto tiempo son las necesarias para almacenar el menú y las ordenes ya que son las únicas que proveen funcionalidad a los módulos con los que ambas bases de datos se comunican.

E. RECOMENDACIONES

- Agregar funciones más generales a la librería, como crear tablas o modificar campos, para que el usuario pueda crear sus propias operaciones.

- Agregar funciones de inventario para poder llevar un mejor control de la disponibilidad de los platos.

- Se recomienda llevar a cabo pruebas de estrés en condiciones de restaurante.

X. RESULTADOS GENERALES

Figura 118. Comunicación entre módulos

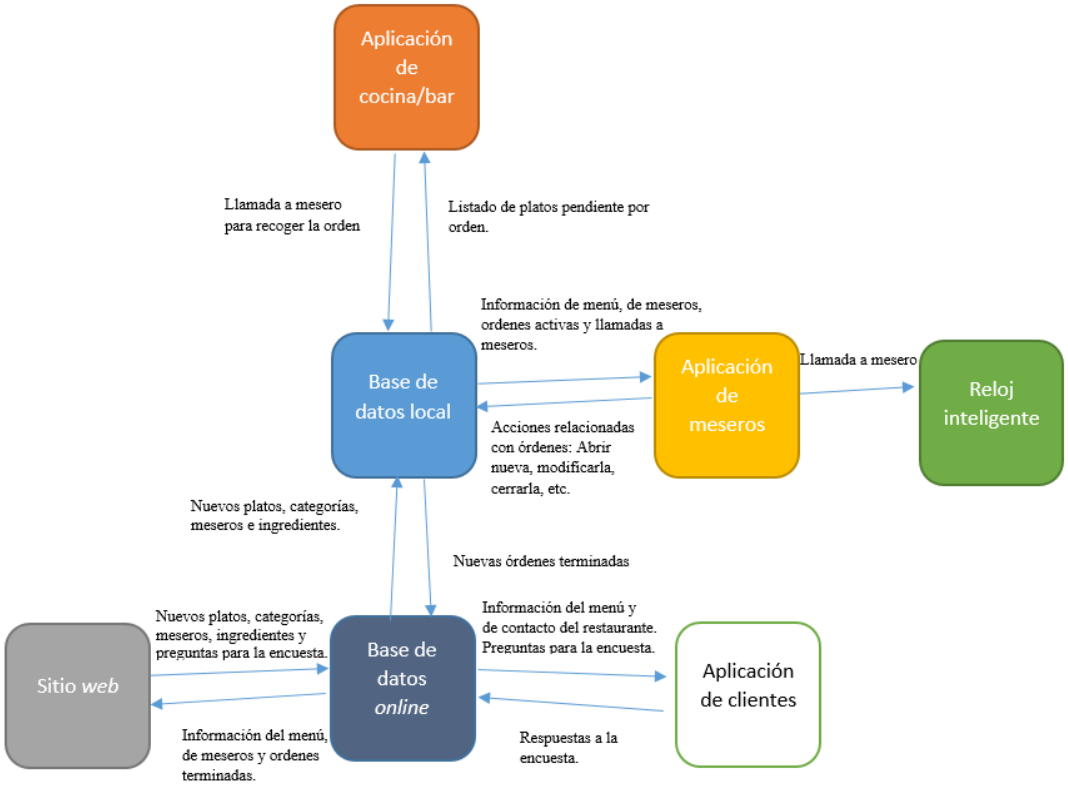


Figura 119. Descripción gráfica de página web para manejo de menú y meseros

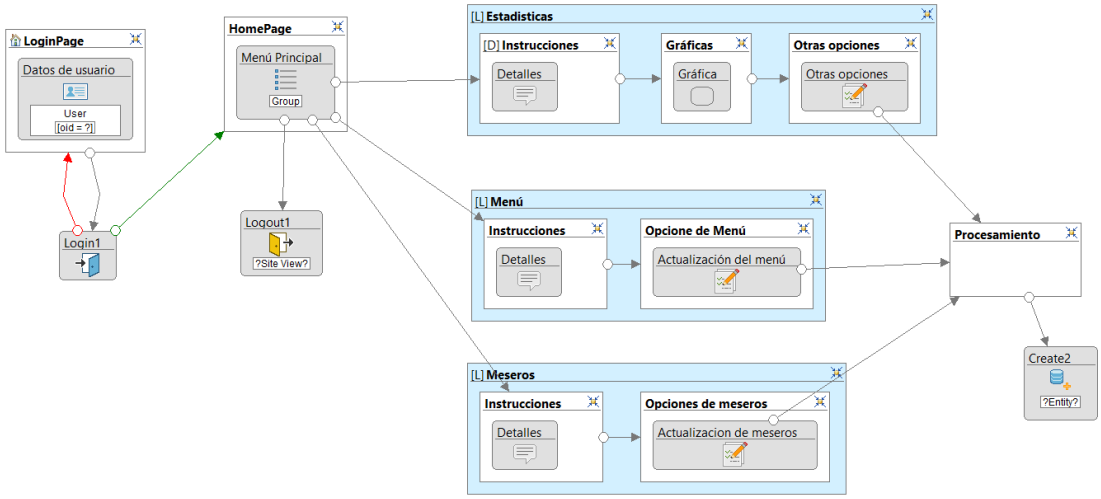


Figura 120. Página web, solicitud de ingreso de usuario

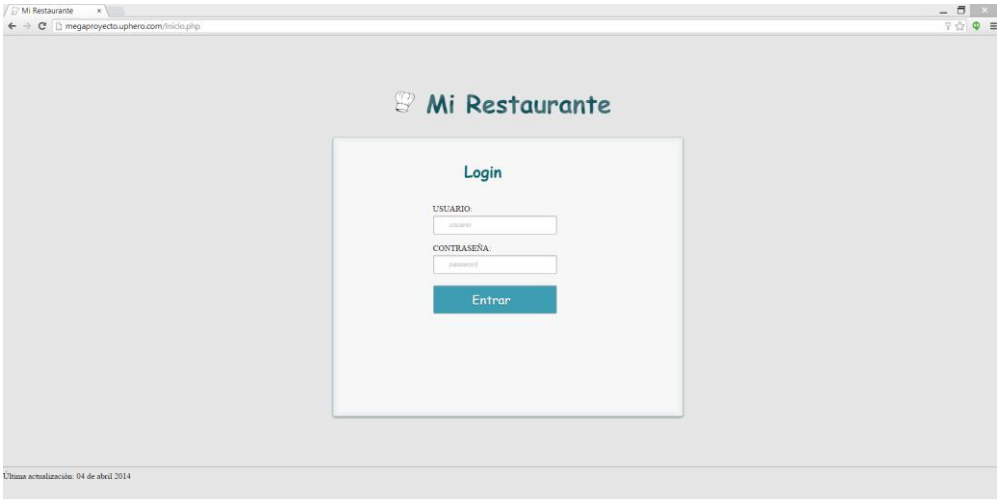


Figura 121. Opciones para control de menú del restaurante.

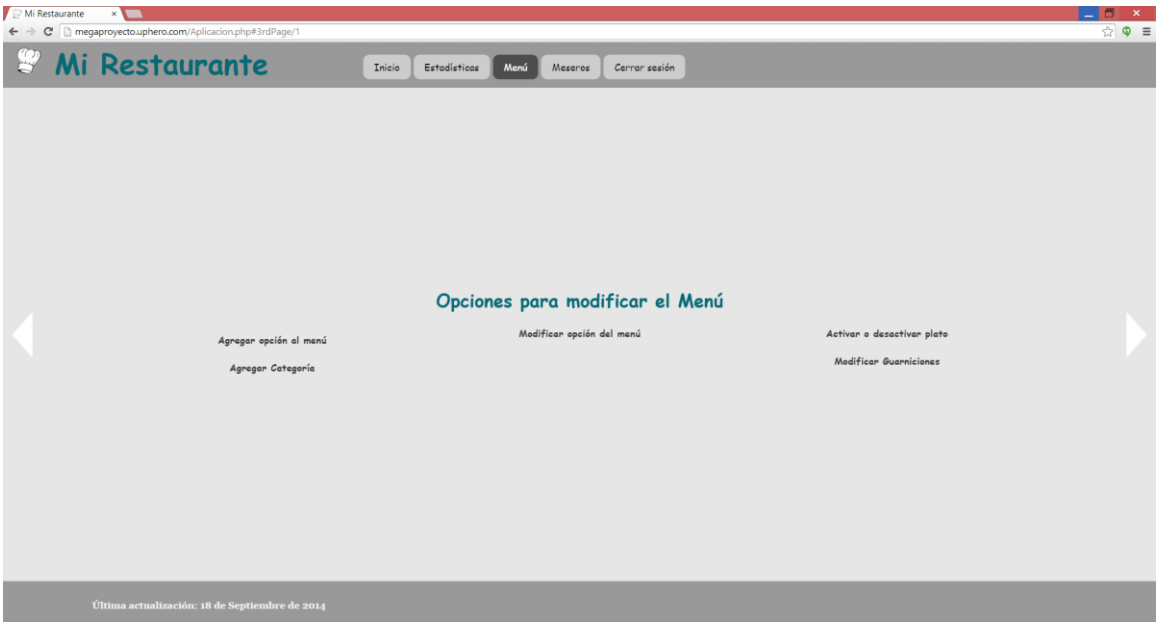


Figura 122. Ventana de ingreso de nuevo plato

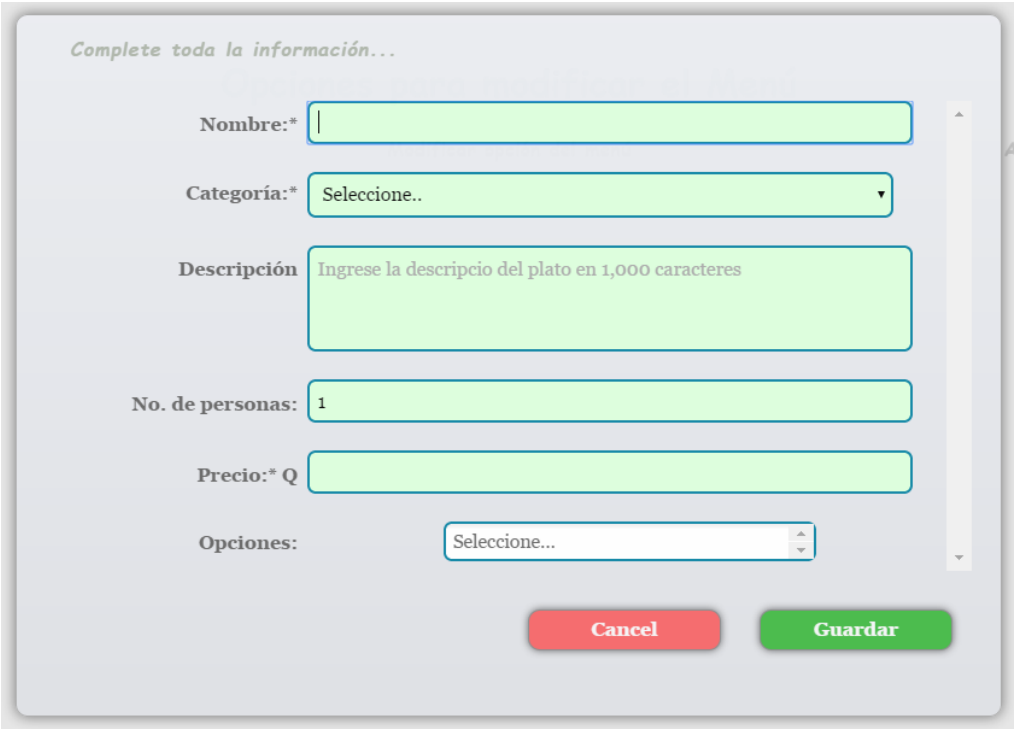


Figura 123. Ventana de modificación para platos existentes

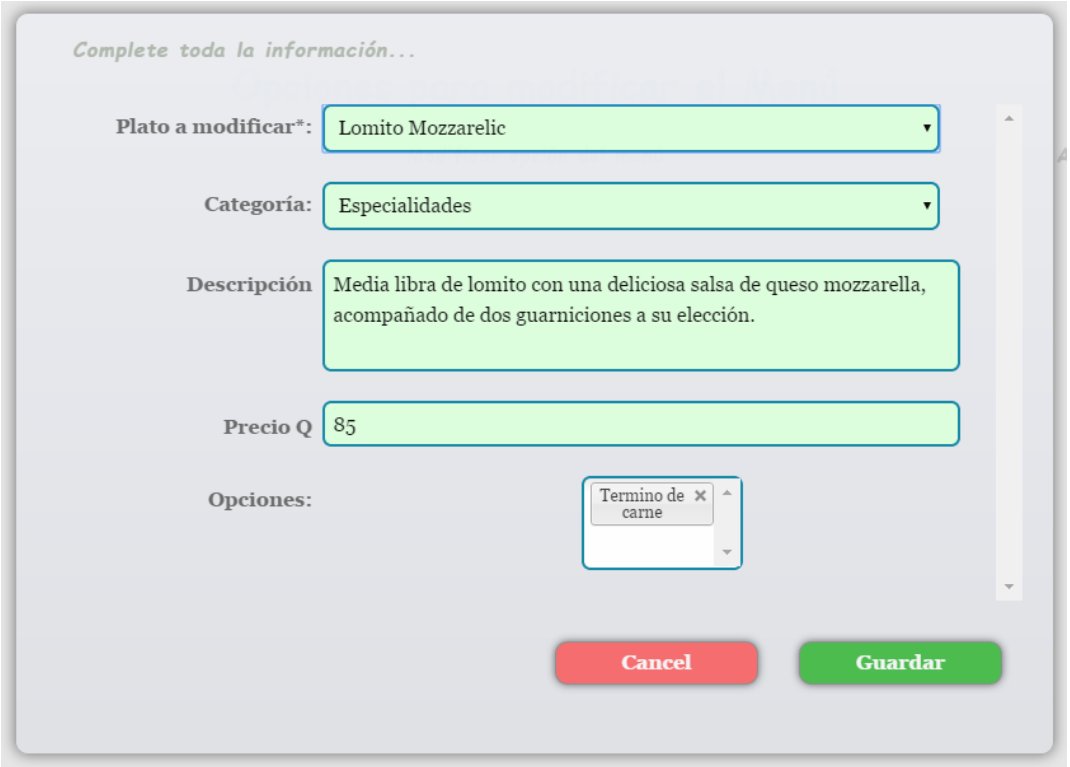


Figura 124. Sección de control de meseros en página web

Figura 125. Ventana de ingreso para nuevo mesero a la base de datos

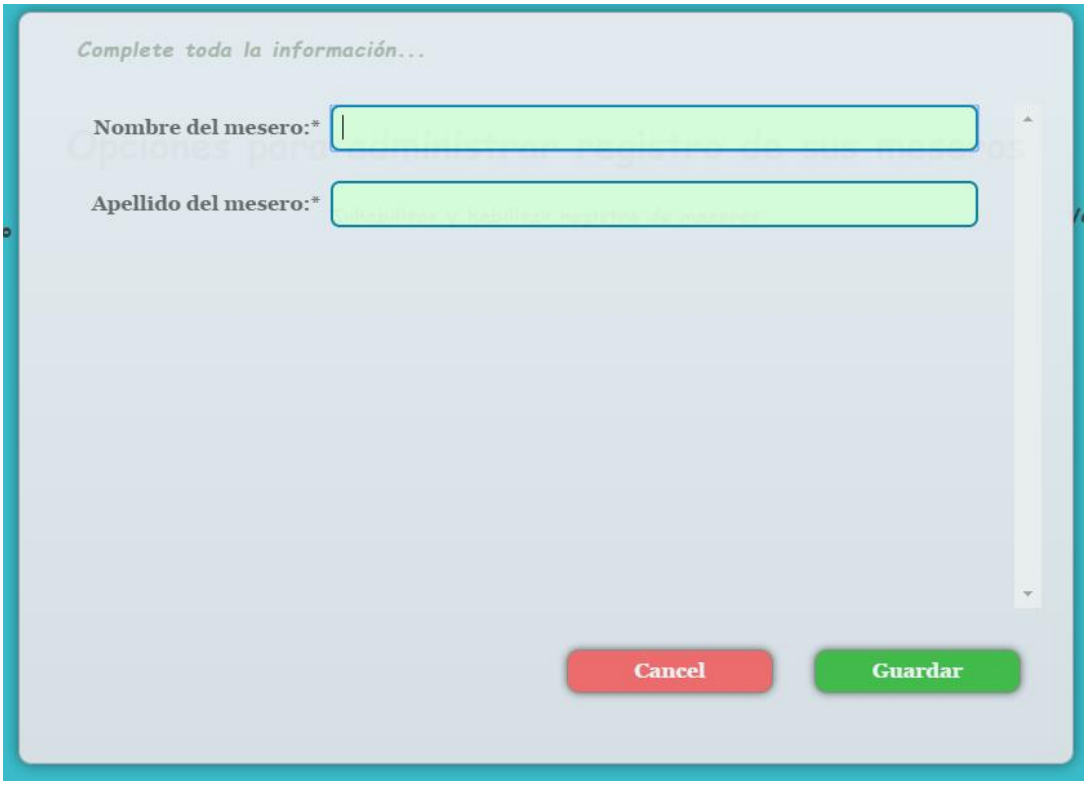


Figura 126. Control de meseros existentes en base de datos

Figura 127. Vista de órdenes recibidas en cocina/bar

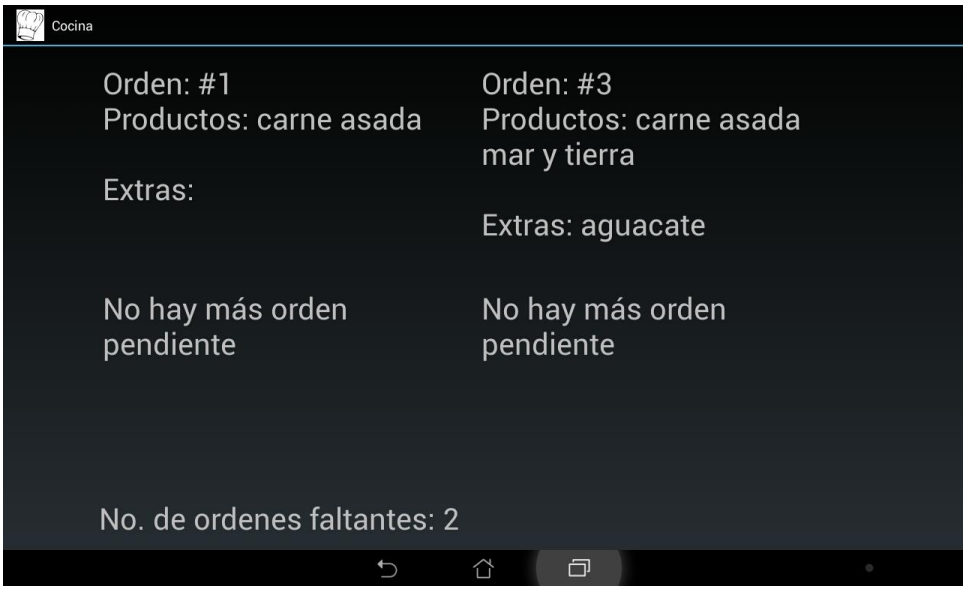


Figura 128. Vista detalla por orden recibida en cocina/bar

Figura 129. Vista de ingreso aplicación para meseros

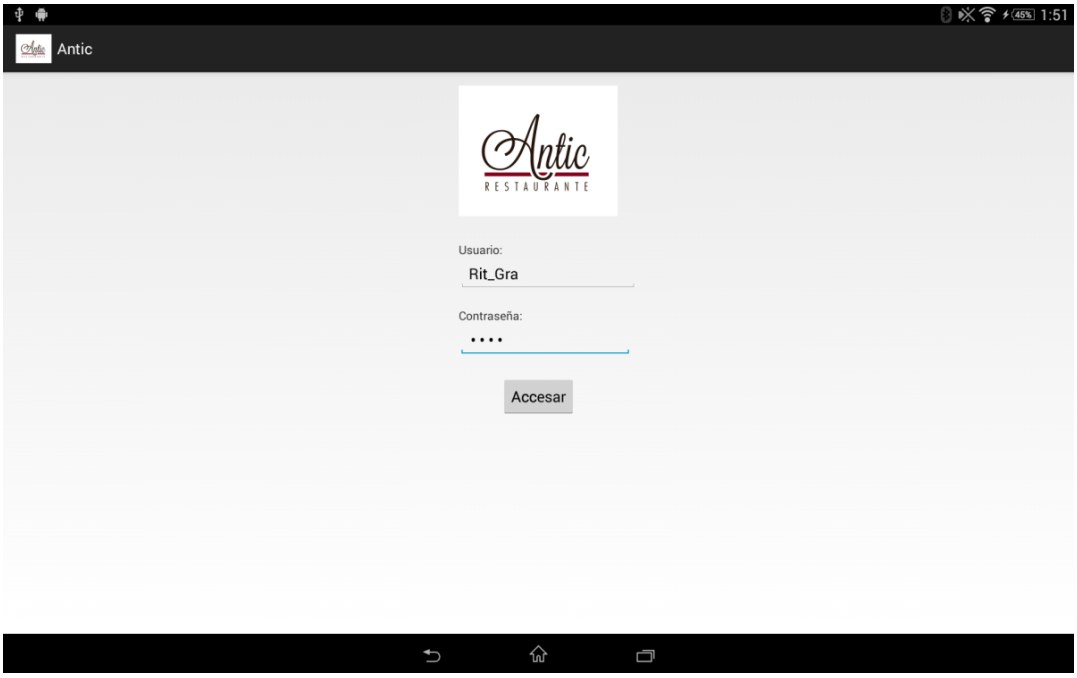


Figura 130. Ingreso de mesa asignada a mesero

Figura 131. Opciones por mesa

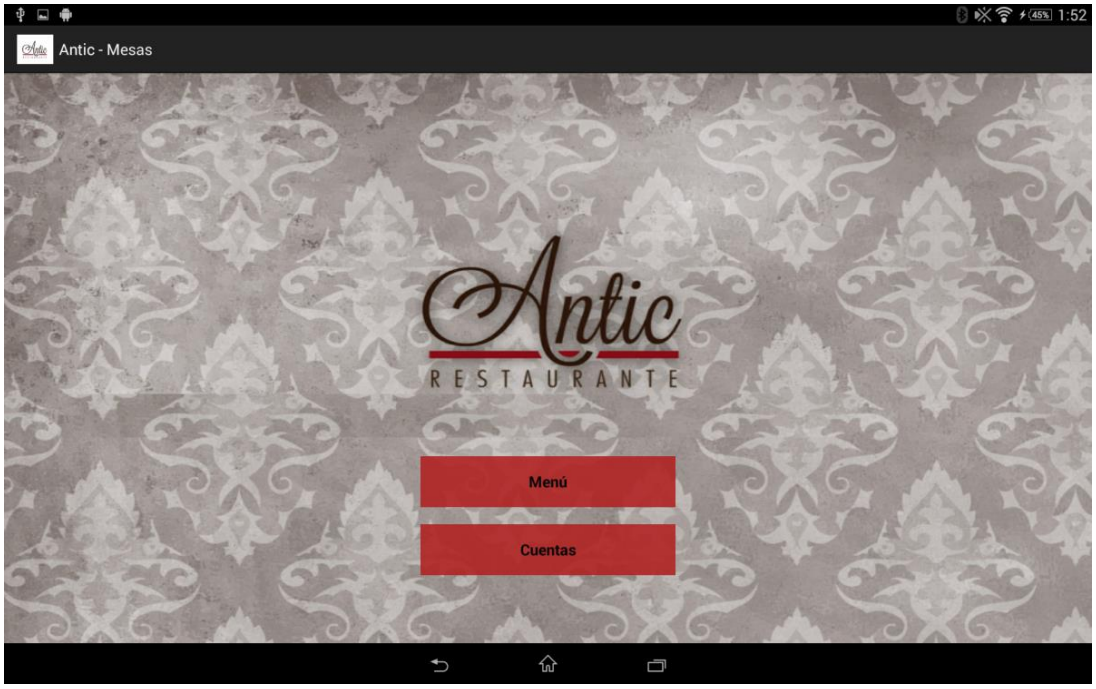


Figura 132. Generar nueva cuenta o manejo de cuenta existente

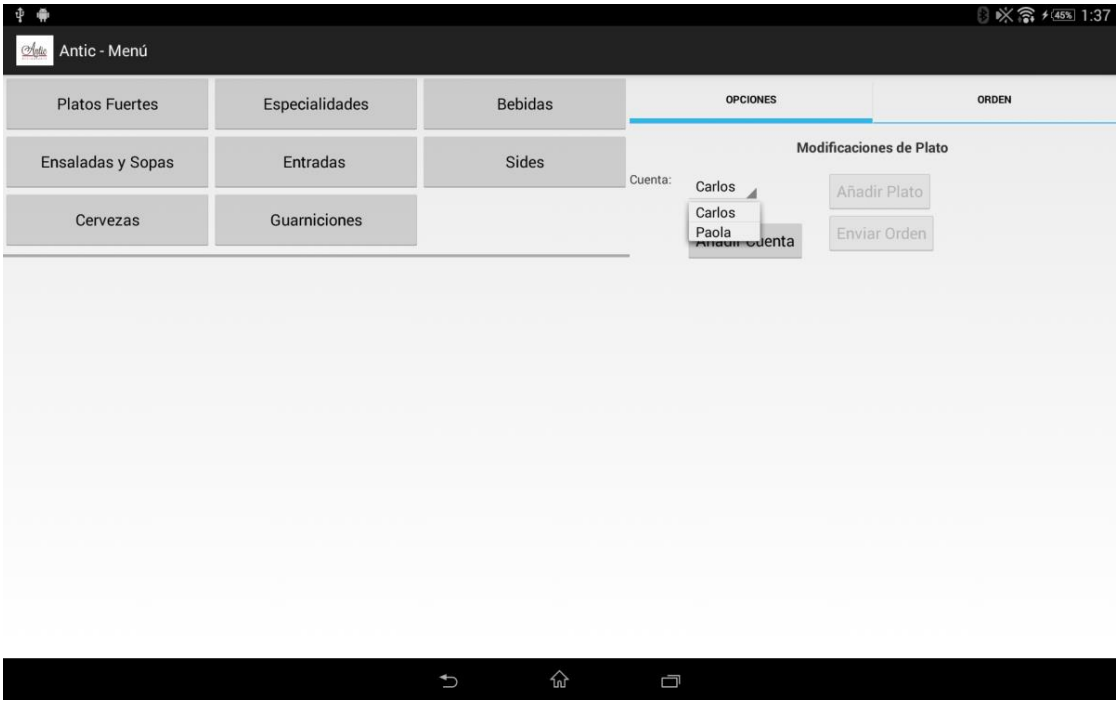


Figura 133. Selección de platos y bebidas para nueva comanda

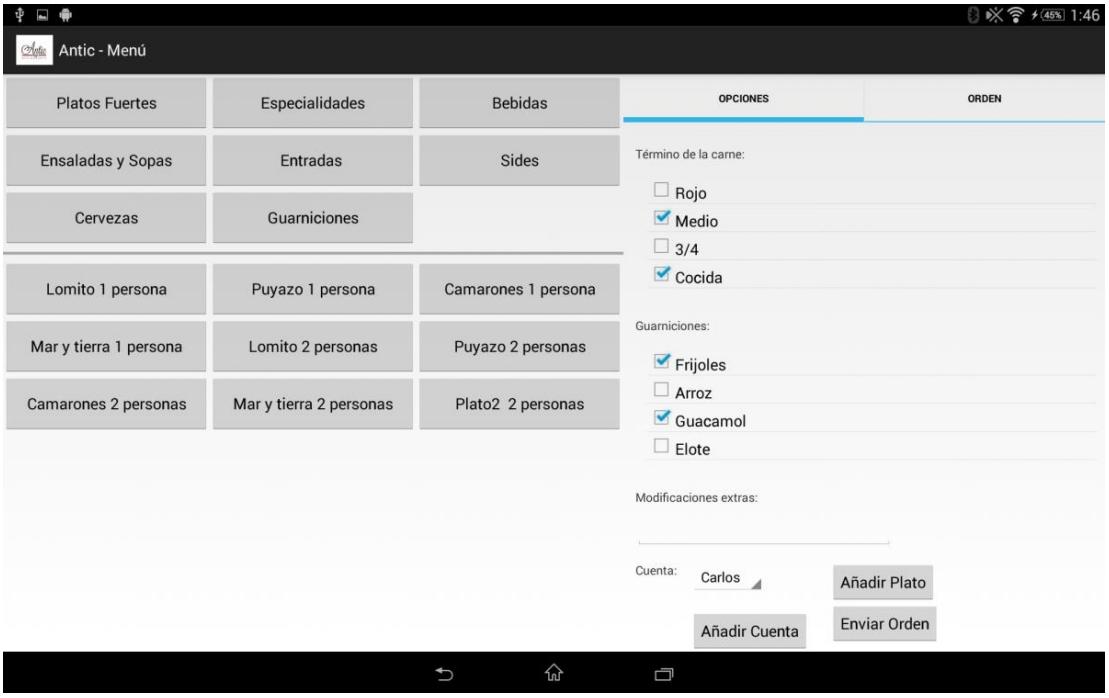


Figura 134. Comanda generada y lista para ser enviada a cocina y bar

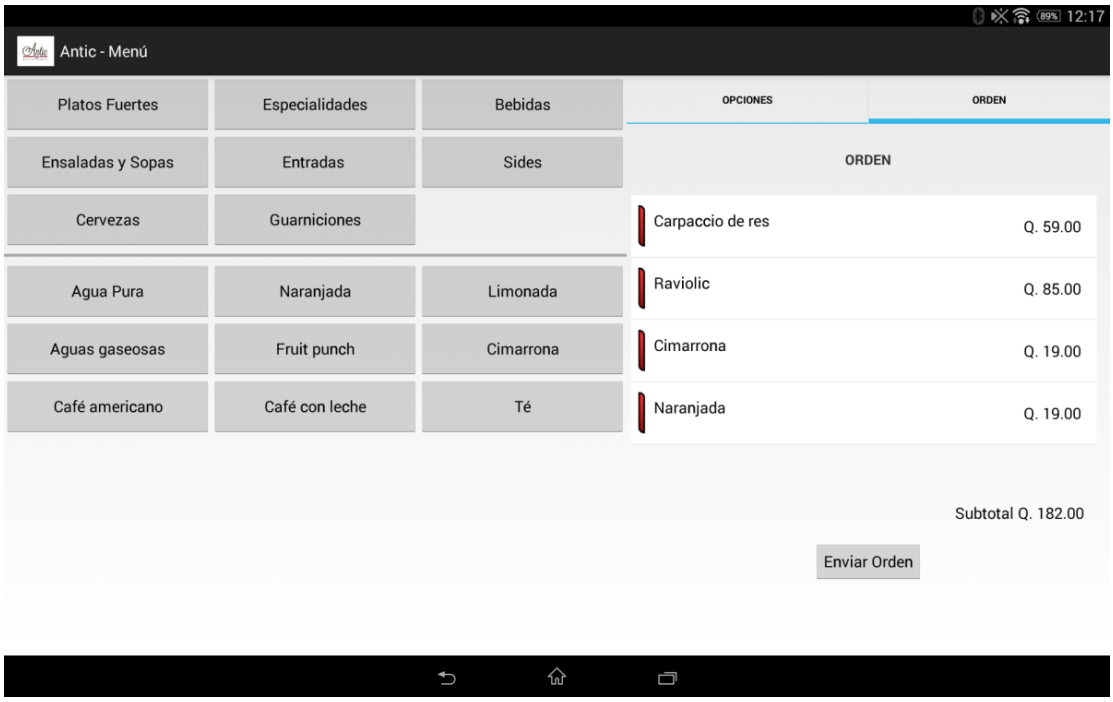


Figura 135. Solicitud de contraseña de administrador para eliminar plato de comandas enviadas a cocina

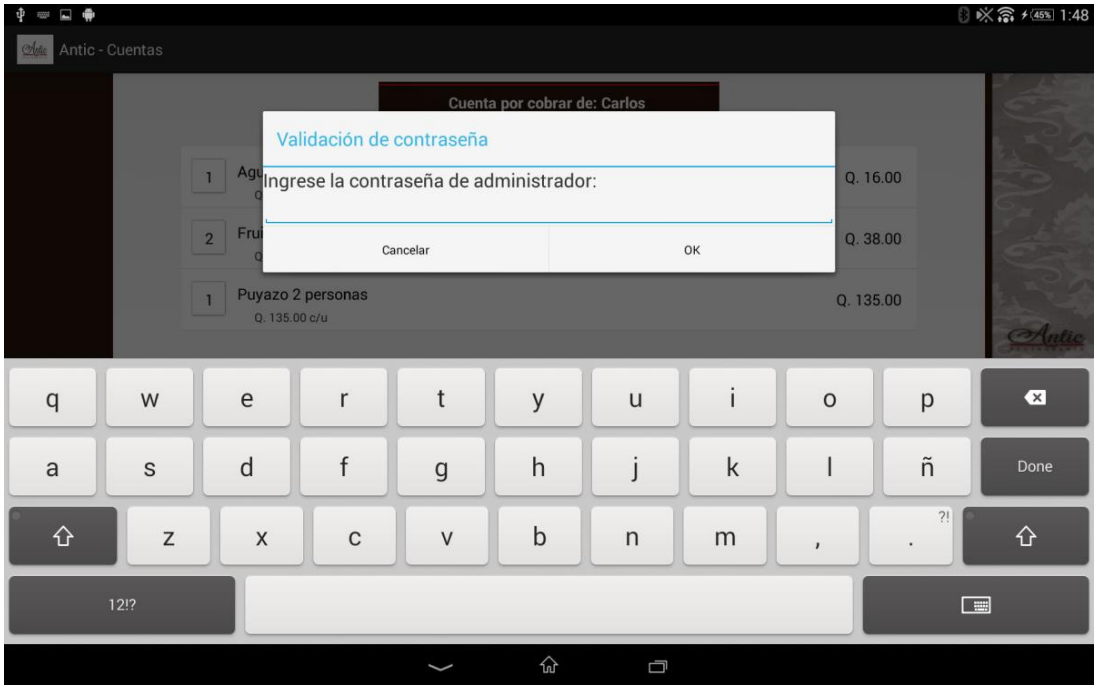


Figura 136. Pantalla principal aplicación para clientes

Figura 137. Muestra de menú por categoría

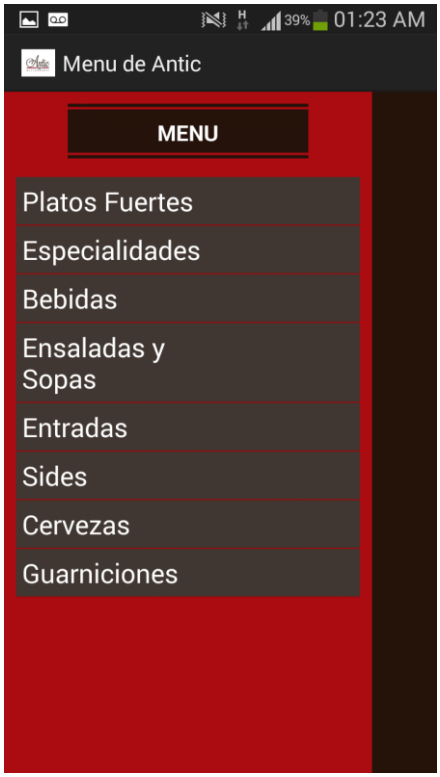


Figura 138. Muestra de descripción y precio por plato

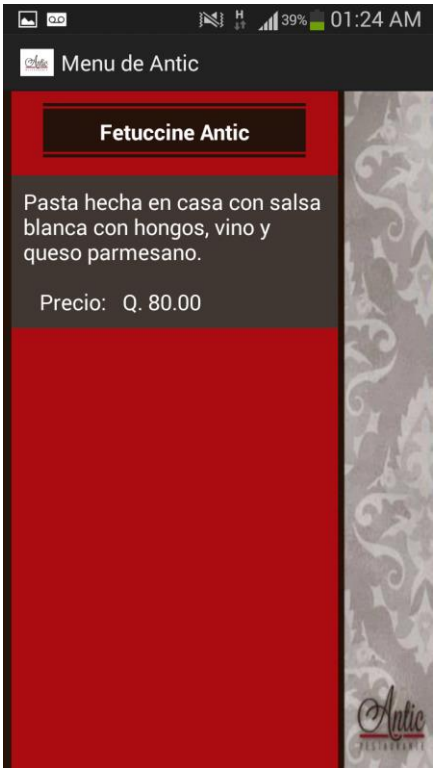


Figura 139. Dispositivo de notificaciones inalámbrico para meseros

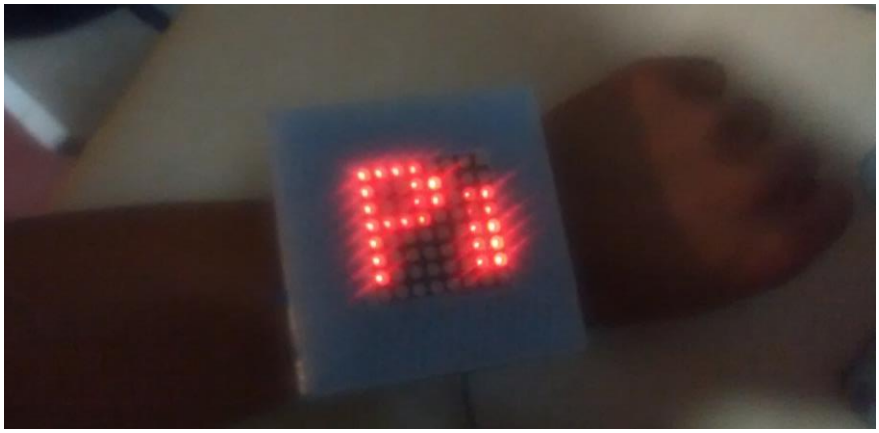
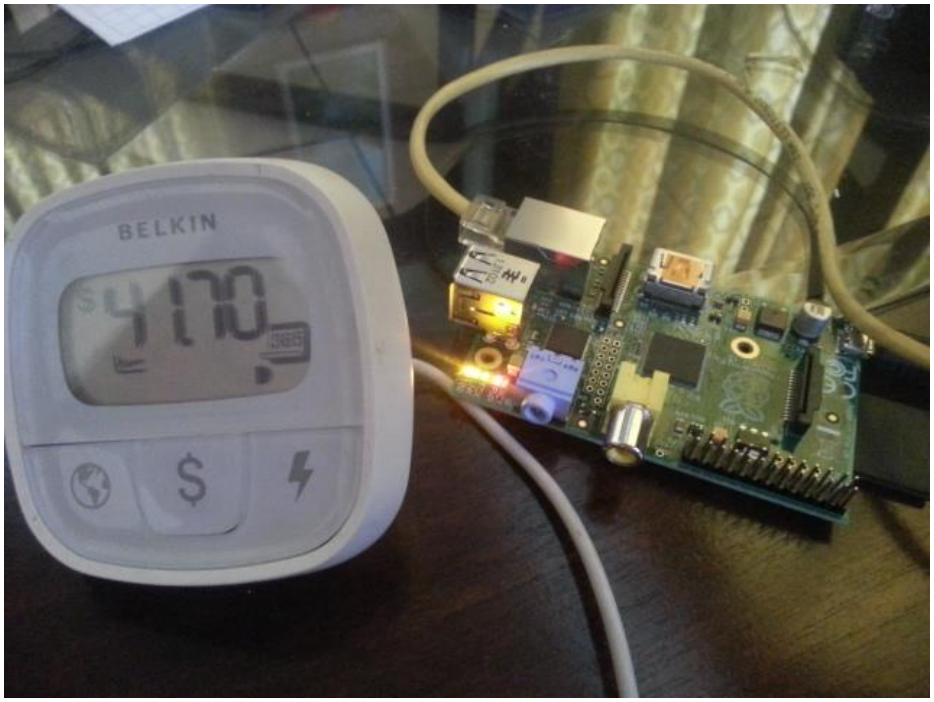


Figura 140.Gasto anual de energía por Raspberry Pi



XI. ANÁLISIS DE RESULTADOS GENERALES

El trabajo presentado tiene como objetivo principal, crear y diseñar un sistema de apoyo gerencial y control de comandas para el restaurante Antic, para lograrlo se realizado la distribución de tareas principales que se querían llegar a alcanzar por módulos, siendo la combinación de estas las que permite tener un control de comandas y ofrecerle al restaurante un apoyo gerencial. Es importante mencionar, que la administración de un restaurante es bastante compleja e incluye procesos como lo es el inventario, que no se han incluido para este trabajo, pero se recomienda incluirlo.

Se logró definir la comunicación entre cada módulo al iniciar con el megaproyecto, como se observa en la Figura 118, de esta forma se definió que información se debía enviar a cada módulo diseñado para su buen funcionamiento, siendo más fácil realizar la unión de los diferentes módulos y buscar errores en los programas. Se buscó que el sistema implementado, Raspberry pi, para interfazar los módulos independientes fuera de bajo costo y consumo energético, esto se puede observar en la Figura 140 que representa el gasto anual. Esta parte fuera realizada por el modulo "Desarrollo de una interfaz de programación de aplicaciones y estructuración de base de datos".

Como se expuso en los capítulos anterior se trabajaron cuatro diferentes módulos, en la cual el modulo "Desarrollo de aplicación en plataforma Android para el control de comandas en cocina y desarrollo de aplicación WEB para control gerencial en el restaurante Antic" tiene como objetivó, como su nombre lo indica, diseñar y desarrollar una página WEB para administración del menú, como podemos observar en la Figura 119 se tiene el esquema de la forma en la que se comunicaran o navegara en la página WEB. De la Figura 120 a la Figura 126 podemos observar las diferentes opciones que se lograron implantar en la página WEB, de las cuales la Figura 122 y Figura 123 hacen referencia a las opciones de modificación de platos y la Figura 124 a la Figura 126 al control del personal de servicio. Como parte del objetivo también se debía diseñar y desarrollar una aplicación en Android para el despliegue de órdenes en cocina, dado que para el Restaurante Antic, el bar y la cocina están ubicados en lugares diferentes, se usó la misma aplicación, con el cambio del valor de una variable para indicar si era la aplicación en cocina o bar. Los resultados obtenidos se pueden observar en la Figura 127 y Figura 128.

El módulo de "Desarrollo de una aplicación en la plataforma Android para generación de comandas y desarrollo de una aplicación para los clientes del restaurante Antic" tiene como objetivo diseñar una aplicación para *tablets* con sistema operativo Android para generación de comandas, esta aplicación se diseñó para que los meseros pudieran realizar diferentes órdenes indicando la mesa a la que pertenecía cada orden y teniendo disponible los diferentes platos del menú que fueron ingresados y modificados desde la

página WEB por el administrador del restaurante, esto se puede observar en de la Figura 129 a Figura 135. Al tener la orden completa de la mese, el mesero con presionar un botón de confirmación, indicaba que esta orden debía ser enviada a cocina o bar, donde la aplicación se encuentra leyendo constantemente la base de datos para encontrar las nuevas órdenes y desplegar toda la información.

Como parte del objetivo el desarrollo de una aplicación para teléfonos inteligentes con sistema operativo Android que les permita a los clientes obtener información sobre los servicios del restaurante Antic, como se puede observar en de la Figura 136 a la Figura 138, donde se incluye acceso a las diferentes redes sociales, teléfono, dirección, correo, como al menú actualizado y disponible. También se deseaba obtener la realimentación de los clientes, para que los administradores del restaurante pudieran realizar cambios en su servicio de ser necesario.

El módulo "Dispositivo de notificaciones inalámbrico tipo brazalete para meseros del restaurante Antic" tiene como objetivo diseñar y construir un dispositivo tipo brazalete que reciba notificaciones de texto por parte de un dispositivo Android vía Bluetooth, para poder indicar al mesero el lugar al cual debe dirigirse, dado que esta notificación únicamente se realiza cuando una orden esta lista para su entrega. Después de que las comandas generadas por el mesero desde la aplicación Android fueron enviadas a la aplicación en cocina, los cocineros tienen la opción para indicar cuando han terminado de cocinar cada uno de los platos de una orden en específico, enviando una notificación al módulo que interfaza los módulos para que la aplicación de la *tablets* que posee el mesero lea constantemente esta notificación y vía Bluetooth indique al mesero que debe ir a bar o a cocina por la orden. Este dispositivo se puede observar en la Figura 139.

XII. CONCLUSIONES

- Se diseñó un sistema de manejo y control de comandas por medio de aplicaciones Android y dispositivo de notificaciones inalámbrico.

- Se desarrolló un sistema de apoyo gerencial para restaurante Antic, por medio de página web y base de datos.

- Se logró diseñar y desarrollar una página web, que ofrece opciones para crear, modificar y deshabilitar platos del menú, junto con una herramienta para evaluar el desempeño del personal de servicio.

- Se desarrolló una aplicación para sistema operativo Android en el que se despliegan las órdenes de cocina y bar.

- Se consiguió diseñar y desarrollar una aplicación en sistema operativo Android que brindó a los clientes una fácil comunicación con el restaurante Antic, la aplicación tiene la capacidad de realizar llamadas al restaurante, enviar correos, visitar su página de Facebook y Twitter.

- Se alcanzó diseñar y desarrollar una aplicación en sistema operativo Android que facilita la gestión de comandas para los meseros del restaurante Antic.

- Se diseñó y construyó un dispositivo que cumple con la función de notificar rápidamente al mesero el lugar al cual dirigirse, aunque por las dimensiones finales no se puede usar como brazalete.

- Se obtuvo la interconexión entre diferentes módulos de un sistema de automatización de restaurantes utilizando redes inalámbricas.

XIII. RECOMENDACIONES

- Se recomienda implementar en la aplicación de clientes la opción de ordenar y pasar a traer la comida, incluyendo la implementación de modificaciones a los platos seleccionados.

- Se recomienda llevar a cabo pruebas de usabilidad y estrés en el restaurante.

- Implementar notificaciones a los gerentes de anomalías en las ordenes, por ejemplo que en una mesa solamente se pidiera una bebida y cinco platos; así se evita que los meseros regalen mercadería a sus conocidos.

- Realizar en la página web una opción de vista previa del menú, así el administrador puede realizar los cambios necesarios sin necesidad de verificar cómo se observan los cambios en la *tablet* o en los *smartphones*.

- Añadir más campos para los meseros, como la fecha de contratación, fecha de cumpleaños, etc. para tener toda la información en la base de datos.

- Desarrollar el manejo de inventario, así se puede indicar si el plato o bebida está disponible para los clientes y para que el administrador pueda considerar estos números al momento de abastecer la bodega.

- Efectuar en la aplicación de bar o cocina las notificaciones con sonidos al recibir una nueva orden, de modo que no sea necesario estar pendiente de la aplicación.

- Se recomienda sustituir la matriz 8x8 de LED’s utilizada en el dispositivo inalámbrico tipo brazalete por un diseño propio de un único color para reducir las dimensiones del circuito impreso y así reducir el tamaño del dispositivo y que sea más cómodo para los meseros.

- Agregar funciones más generales a la librería, como crear tablas o modificar campos con la finalidad de que el usuario pueda crear sus propias operaciones.

XIV. BIBLIOGRAFÍA

(s.f.). Recuperado en Agosto de 2014, de http://www.openhandsetalliance.com/oha_overview.html

(s.f.). Recuperado en Agosto de 2014, de http://www.javabeginner.com/learn-java/java-threads-tutorial

Adsuara, J. (2013). *Resolviendo la ecuación del subdominio*. Recuperado el 17 de Septiembre de 2014, de http://juanadsuara.wordpress.com/2013/09/29/resolviendo-la-ecuacion-del-subdominio-1/

Alcalde, A. (05 de Septiembre de 2014). *El Baúl del programador*. Recuperado el Agosto de 2014, de http://elbauldelprogramador.com/programacion-android-trabajar-con/

Amadeo, R. (15 de Junio de 2014). *ArsTechnica*. Recuperado el Agosto de 2014, de http://arstechnica.com/gadgets/2014/06/building-android-a-40000-word-history-of-googles-mobile-os/

Amaro Soriano, J. E. (2012). *Android Progamación de dispositivos móviles a través de ejemplos*. España: Marcombo.

American Dialect Society. (8 de Enero de 2011). Recuperado el Agosto de 2014, de http://www.americandialect.org/app-voted-2010-word-of-the-year-by-the-american-dialect-society-updated

Anand Kumar, A. (2009). *Switching Theory and Logic Design*. Nueva Delhi: PHI Learning Private Limited.

Andorid. (2014). *Android Developer*. Recuperado el Septiembre de 2014, de http://developer.android.com/guide/topics/manifest/manifest-intro.html

Android. (20 de Septiembre de 2014). *Activities*. Obtenido de Android Developers: https://developer.android.com/guide/components/activities.html

Android. (2014). *Android Developer*. Recuperado el Septiembre de 2014, de http://developer.android.com/guide/components/processes-and-threads.html

Android. (2014). *Android Developer*. Recuperado el Septiembre de 2014, de http://developer.android.com/tools/sdk/eclipse-adt.html

Android. (17 de Septiembre de 2014). *Android Developer*. Recuperado el Septiembre de 2014, de http://developer.android.com/reference/java/lang/Thread.html

Android. (17 de Septiembre de 2014). *Android Developer*. Recuperado el 19 de Septiembre de 2014, de http://developer.android.com/reference/android/os/Bundle.html

Android. (17 de Septiembre de 2014). *Android Devloper*. Recuperado el Septiembre de 2014, de http://developer.android.com/reference/android/app/Activity.html

Android. (Septiembre de 2014). *Android Devloper*. Recuperado el Junio de 2014, de http://developer.android.com/reference/android/os/AsyncTask.html

Android. (20 de Septiembre de 2014). *Application Fundamentals*. Obtenido de Android Developers: https://developer.android.com/guide/components/fundamentals.html

Android. (20 de Septiembre de 2014). *Get Android SDK*. Obtenido de Android Developers: http://developer.android.com/sdk/index.html

Android. (20 de Septiembre de 2014). *Thread*. Obtenido de Android Developers: http://developer.android.com/reference/java/lang/Thread.html

Angoar, M. (2008). Taller de Instalación de Servidores LAMP WAMP. Sevilla.

Arduino . (18 de Septiembre de 2014). *Arduino*. Obtenido de Arduino Nano: http://arduino.cc/en/pmwiki.php?n=Main/ArduinoBoardNano

Arduino. (19 de Septiembre de 2014). *Arduino*. Obtenido de Arduino ShiftOut Tutorial: http://arduino.cc/en/pmwiki.php?n=Tutorial/ShiftOut

Arghire, L. (30 de Octubre de 2012). *Softpedia.* Recuperado el Agosto de 2014, de http://news.softpedia.com/news/Windows-Phone-Store-Has-120-000-Apps-Now-More-to-Come-302992.shtml

Bai, Y. (2006). *The Windows Serial Port Proframming Handbook.* Boca Raton, Florida: Taylor & Francis e-Library.

Barceló, J. M. (2008). Protoolos y aplicaciones Internet. Barcelona: UOC.

Bates, S. (14 de Enero de 2014). Recuperado el Agosto de 2014, de http://manifesto.co.uk/history-mobile-application-development/

Bates, S. (Enero de 2014). *A History of Mobile Application Development*. Recuperado el 17 de Septiembre de 2014, de http://manifesto.co.uk/history-mobile-application-development/

Berjon, R. (Ed.). (29 de Mayo de 2014). *W3C*. Recuperado el 17 de Septiembre de 2014, de http://www.w3.org/TR/html-json-forms/

Carrero. (s.f.). *Introducción a los Layouts para Android*. Recuperado el 5 de Junio de 2014, de http://programacion.net/articulo/introducion_a_los_layouts_para_android_400

Clark, J. F. (s.f.). *University of Kentucky*. Recuperado el Agosto de 2014, de http://www.uky.edu/~jclark/mas490apps/History%20of%20Mobile%20Apps.pdf

Condesa. (2 de Octubre de 2011). *Androideity*. Recuperado el Julio de 2014, de http://androideity.com/2011/10/03/testear-aplicaciones-android-en-tu-telefono/

d. (s.f.).

Date, C. (2001). Introducción a los sistemas de bases de datos.

DocumentationGroup, P. (s.f.). *¿Qué es PHP?* Recuperado el 19 de Septiembre de 2014, de http://php.net/manual/es/intro-whatis.php.

DuBois, P. (2008). MySql. Estados Unidos: Pearson Education.

Fehily, C. (2002). SQL. Estados Unidos.

Foundation, R. P. (s.f.). *General*. Recuperado el 19 de Septiembre de 2014, de http://www.raspberrypi.org/help/faqs/#generalDifference

Foundation, R. P. (s.f.). *Introduction*. Recuperado el 19 de Septiembre de 2014, de http://www.raspberrypi.org/help/faqs/#introWhatIs

Foundation, R. P. (s.f.). *Introduction*. Recuperado el 19 de Septiembre de 2014, de http://www.raspberrypi.org/help/faqs/#introWhatIs

Gallardo, D. (2014). *Apuntes de Arduino Nivel Enteraillo.* Laguna de Tollón. Obtenido de http://es.scribd.com/doc/214900993/Apuntes-Arduino-Nivel-Enteraillo

García, C., Hernández, D., & Berlingen, J. (2010). Desarrollo de aplicaciónes en Android. Salamanca, España: Universidad de Salamanca.

García, J., & Morales, G. (2012). *Instalaciones de radiocomunicaciones.* Madrid: Paraninfo.

Gargenta, M., & Nakamura, M. (2014). Learning Android: Develop Mobile Apps Using Java and Eclipse. O´Reilly, Estados Unidos.

Golden, R. (2013). Raspberry Pi Networking Cookbook. Packt Publishing.

Gorbatchev, A. (2012). *TextExtJs.* Recuperado el Septiembre de 2014, de http://textextjs.com/manual/plugins/ajax.html

Gordon, S. A. (2014). *AndroidPIT*. Recuperado el 19 de Septiembre de 2014, de http://www.androidpit.com/android-for-beginners-what-is-an-apk-file

Gourley, D. e. (2002). HTTP the Definittive Guide. O'Reilly, Estados Unidos.

Gourley, D., Totty, B., Sayer, Aggarwal, & Reddy. (2002). *HTTP The definitive Guide.*

Inc., T. C. (2014). *Technology Products Review*. Recuperado el Agosto de 2014, de http://www.pcmag.com/encyclopedia/term/37856/api

Indvik, L. (11 de junio de 2012). *Mashable.* Recuperado el Septiembre de 2014, de http://mashable.com/2012/06/11/wwdc-2012-app-store-stats/

ITead Studio. (19 de Septiembre de 2014). HC-05. Obtenido de http://biblioteca.geekfactory.mx/Bluetooth_Bee_Pro/datasheet_hc-05.pdf

Janalta Interactive Inc. (20 de Septiembre de 2014). *Application Software*. Obtenido de techopedia: http://www.techopedia.com/definition/4224/application-software

Janalta Interactive Inc. (20 de Septiembre de 2014). *Operating System (OS)*. Obtenido de techopedia: http://www.techopedia.com/definition/3515/operating-system-os

Lanacion. (2011). *¿Qué son y para qué sirven las "apps"?* Recuperado el 17 de Septiembre de 2014, de http://www.lanacion.com.ar/1365035-que-son-y-para-que-sirven-las-apps

Laurie, B., & Laurie, P. (2003). Apache The Definittive Guide. O'Reilly, Estados Unidos.

Lawler, R. (2012). *Raspberry Pi credit-card sized Linux PCs are on sale now, $25 Model A gets a RAM bump*. Recuperado el 19 de Septiembre de 2014, de http://www.engadget.com/2012/02/29/raspberry-pi-credit-card-sized-linux-pcs-are-on-sale-now-25-mo/.[

LV, M. (31 de Enero de 2011). *Androides y Pingüinos.* Recuperado el Julio de 2014, de http://miguelangellv.wordpress.com/2011/01/31/creando-tareas-asincronas-en-android-con-asynctask/

Mednieks, Z. L. (2012). Programming Android . Estados Unidos: O'Reilly.

Mora, E. e. (2003). Iniciación a las bases de datos con Access 2002.

Morris Mano, M. (1982). *Lógica Digital y Diseño de Computadores.* Naucalpan, México: Prentice Hall Hispaniamericana, S.A.

Open Handset Alliance. (December de 2008). Recuperado el Agosto de 2014, de http://www.openhandsetalliance.com/android_faq.html

Ostrander, J. (2012). Android UI Fundamentals Develop and Design . Estados Unidos: Peachpit Press.

Pécheron, S. (2012). Android Guía de desarrollo de aplicaciones para Smartphones y Tabletas.

Pérez, D. (Junio de 2012). *Diseño de una aplicación Android para calcular el potencia de generación*. Recuperado el 18 de Septiembre de 2014, de http://upcommons.upc.edu/pfc/bitstream/2099.1/15650/1/82319.pdf

phpMyAdminDevelpmentTeam. (s.f.). *Bringin MySQL to the web*. Recuperado el 19 de Septiembre de 2014, de http://www.phpmyadmin.net/home_page/index.php

Porter, A. (2014). *Cursera*. Recuperado el Enero de 2014, de https://www.coursera.org/course/android

Pozo, G. (4 de Febrero de 2012). *Aprendiendo de Android y más*. Recuperado el 18 de Septiembre de 2014, de http://www.aprendiendodeandroidymas.com/2012/02/ciclo-de-vida-de-una-aplicacion.html

Precision Microdrives. (19 de Septiembre de 2014). *Precision Microdrives*. Obtenido de Vibration Motors: http://www.precisionmicrodrives.com/vibrating-vibrator-vibration-motors/pager-motors-erm-motors

Ratabouil, S. (2012). Android NDK. Estados Unidos: PacktPub.

Reddy, M. (2011). API Design for C++. Elsevier.

Robledo, D., & Robledo, D. (2012). Programación en Android. España: Aula Mentor.

Roldayan. (2 de Julio de 2013). *Desarrollo de aplicaciones en Android*. Recuperado el 17 de Septiembre de 2014, de http://www.mibqyyo.com/articulos/2013/07/02/desarrollo-en-android-1-preparando-entorno-de-trabajo/#/vanilla/discussion/embed/?vanilla_discussion_id=0

Rouse, M. (2008). *Server*. Recuperado el 21 de Septiembre de 2014, de http://whatis.techtarget.com/definition/server

Ruben, Jesus. (21 de Febrero de 2014). *Bluetooth HC-05 y HC-06 Tutorial de Configuración.* Obtenido de Geek Factory: http://www.geekfactory.mx/radio/bluetooth-hc-05-y-hc-06-tutorial-de-configuracion/

Sacristrán, C., & Fernández, D. (2013). Programación en Android. España: Aula Mentor.

Sanchís, E. (2002). *Sistemas electrónicos digitales.* Valencia: Maite Simon.

Siegler, M. (11 de Junio de 2008). *VentureBeat.* Recuperado el Septiembre de 2014, de http://venturebeat.com/2008/06/11/analyst-theres-a-great-future-in-iphone-apps/

Sliever Ellen, e. a. (2009). Linux in a Nutshell. O'Reilly, Estados Unidos.

Sliever, Figgins, Love, & Robbins. (2009).

Spicemobiles. (09 de Enero de 2014). *Visual.ly*. Recuperado el Agosto de 2014, de http://visual.ly/brief-history-android

Steiner, C. (2005). *The 8051/8052 Microcontroller.* Boca Raton, Florida: Universal Publishers.

Tech Terms. (30 de Julio de 2010). Recuperado el 19 de Septiembre de 2014, de http://www.techterms.com/definition/smartphone

Tech Terms. (15 de Abril de 2010). Recuperado el 19 de Septiembre de 2014, de http://www.techterms.com/definition/sdk

TechTerms. (s.f.). Recuperado el Agosto de 2014, de http://www.techterms.com/definition/api

Thompson, T., Kline, P., & Kumar, C. (2008). *Bluetooth Application Programming with the Java API's.* Elsevier.

Tomás, J. (2013). El gran libro de Android. Barcelona, España: Ediciones Técnicas marcombo.

Tomasi, W. (2003). *Sistemas de Comunicaciones Electrónicas* (4ª ed. ed.). México: Pearson Educación.

treehouse. (s.f.). *treehouse.* Recuperado el Junio de 2014, de http://elbauldelprogramador.com/programacion-android-trabajar-con/

Valdés Pérez, F. E., & Pallas Areny, R. (2007). *Microcontroladores: Fundamentos y Aplicaciones con PIC.* España: MARCOMBO.

Vela, H. (15 de Enero de 2011). *Así es el aborrote.* Recuperado el Agosto de 2014, de http://asieselabarrote.blogspot.com/2011/01/evitar-que-la-pantalla-de-android-se.html

Vico, Á. (4 de Julio de 2011). *Arquitectura de Android*. Recuperado el 18 de Septiembre de 2014, de http://androideity.com/2011/07/04/arquitectura-de-android/

Vilela, I. (2012). *¿Qué es una App?* Recuperado el 17 de Septiembre de 2014, de http://www.startcapps.com/blog/que-es-una-app/

W3C. (18 de Noviembre de 2010). *W3C*. (I. Hickson, & I. Google, Editores) Recuperado el 18 de Septiembre de 2014, de http://www.w3.org/TR/webdatabase/

W3C. (2014). Recuperado el 17 de Septiembre de 2014, de http://www.w3.org/Consortium/

W3C. (2014). (W3C) Recuperado el 17 de Septiembre de 2014, de http://www.w3.org/Help/#funds

W3C. (2014). *W3C*. Recuperado el 17 de Septiembre de 2014, de http://www.w3.org/standards/webdesign/script

W3C. (s.f.). *W3C.* Recuperado el 19 de Septiembre de 2014, de http://www.w3.org/Help/

w3Schools. (s.f.). *w3Shools*. Recuperado el 17 de Septiembre de 2014, de http://www.w3schools.com/php/php_intro.asp

Wikipedians. (s.f.). World Wide Web. PediaPress. Obtenido de http://books.google.es/books?id=1oa-vWbdH3EC&pg=PA15&dq=world+wide+web++history&hl=es&sa=X&ei=9QYaVJO_EbPksATC84LICg&ved=0CCIQ6AEwAA#v=onepage&q=world%20wide%20web%20%20history&f=false

Winder, S. (2008). *Power Supplies for LED Driving.* Burlington: Elsevier.

Womack, B. (29 de Octubre de 2012). *BloombergBusinesswee*. Recuperado el Septiembre de 2014, de http://www.businessweek.com/news/2012-10-29/google-says-700-000-applications-available-for-android-devices

Yeager, N. (1996). Web Server Technology. San Francisco: Morgan Kaufmann Publishers.

XV. ANEXOS

Código fuente de los módulos se encuentra adjunto en CD.

XVI. GLOSARIO

Arduino Nano: Es una placa maraca Arduino, que contiene al microcontrolador ATmega328P en versión SMD , esta placa permite el desarrollo de proyectos de electrónica.

Apache v2, open source license: Es una licencia comercial para código abierto. Permite que los manufactureros y los usuarios de móviles innovar usando la plataforma sin la obligación de ofrecer sus innovaciones a la comunidad de código abierto, dado que pueden mantener su propiedad. (FAQ, 2008)

APK (Application PacKage File): es el formato del archivo para instalar aplicaciones en el sistema operativo Android. (Gordon, 2014)

Bluetooth: Técnologia de comunicación inalambrica de bajo rango.

Bundle: mapeo de objetos “Strings” ha objetos “Parcelable”. (Android, Bundle, 2014)

Comanda: del francés commander, pedir, es el pedido realizado por los clientes de un restaurante.

Fragment: es un componente similar a una Activity; es una porción de una interfaz gráfica que contiene su propio ciclo de vida independiente al ciclo de vida de la actividad que lo contiene.

Hoja de datos: conocida en inglés como “datasheet”, son documentos proporcionados por los fabricantes de los componentes que contienen información sobre el funcionamiento del componente mismo e información adicional como dimensiones y factores de operación.

IDE: es un entorno de desarrollo integrado con el objetivo de programar aplicaciones Android desde un ordenador.

Internet: Sistema global de redes interconectadas que intercambian paquetes de datos utilizando protocolos estandarizados como TCP/IP (Internet Protocol Suite). (W3C, W3C)

LED: Acrónimo inglés para Light Emitting Diode, se traduce como diodo emisor de luz. Es un componente fabricador de dos materiales uno saturado de carga positiva y otro saturado con carga negativa, ambos materiales son unidos. El funcionamiento es similar al de un diodo.

SMD: se refiere a los dispositivos de montaje superficial, por sus siglas en inglés SMS.

Software Development Kit (SDK): es un conjunto de programas para desarrollar aplicaciones de un sistema operativo específico. Entre estos programas se encuentra el IDE (Integrated Development Environment) que es un ambiente para desarrollar o programar; permite depurar, compilar y proporciona un editor visual para modificar la interfaz de usuario o GUI (Graphical User Interface). (SDK, 2010)

Teléfono inteligente o "Smartphone": Dispositivo multifuncional con aplicaciones integradas y acceso a internet, con la capacidad de realizar varias funciones de una computadora, usualmente tiene una pantalla táctil. (Smartphone, 2010)