SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE CD. GUZMÁN

TITULACIÓN INTEGRAL

TESIS

TEMA:

TOC-TOC SERVICIOS A DOMICILIO

QUE PARA OBTENER EL TÍTULO DE:

INGENIERO INFORMÁTICO

PRESENTA:

LUIS AZAEL JIMÉNEZ FAJARDO

ASESOR(A):

M.C. RAQUEL OCHOA ORNELAS

CD. GUZMÁN JALISCO, MÉXICO, SEPTIEMBRE DE 2018

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Cd. Guzmán

Cd. Guzmán, Municipio de Zapotlán el Grande, Jal. 13/09/2018

ASUNTO: Liberación de Proyecto para Titulación integral

M.C. FAVIO REY LÚA MADRIGAL
JEFE DE LA DIVISION DE ESTUDIOS PROFESIONALES
Presente

Por este medio informo que ha sido liberado el siguiente proyecto para la Titulación integral:

Nombre del Egresado	**Luis Azael Jiménez Fajardo**
Carrera:	Ingeniería Informática
No. De Control	**13290769**
Nombre del proyecto:	**"Toc-toc Servicio a domicilio"**
Producto	TESIS

Agradezco de antemano su valioso apoyo en esta importante actividad para la formación profesional de nuestros egresados.

Atentamente,

M.C. Rubén Zepeda García
Jefe del Depto. Sistemas y Computación

SEP TecNM
I. T. DE CD. GUZMÁN
SISTEMAS Y COMPUTACIÓN

M.C. Raquel Ochoa Ornelas	Dr. Daniel Fajardo Delgado	Dra. María Guadalupe Sánchez Cervantes
Nombre y Firma del Asesor	Nombre y Firma del Revisor	Nombre y Firma del Revisor

C.p. Archivo

DLAS/CRM/RZG/ljss*

RESUMEN

El objetivo de una ciudad inteligente es incrementar las inversiones en educación, infraestructura, control de energías, tecnologías y transporte para impulsar una mejor calidad de vida. De igual forma, el desarrollo económico y el cuidado del medio ambiente promueven una participación responsable, además con la participación de un gobierno comprometido que pueda ofrecer servicios seguros y de mejor calidad, logran mejorar tiempos y recursos generando una ciudad inteligente. En este documento se presenta el desarrollo de una plataforma para promover a empresas del sector de servicios a través de la atención de solicitudes de usuarios que realizan desde una aplicación móvil. El proyecto ofrece varios escenarios para monitorear la administración de la plataforma y una aplicación instalada en cada empresa para dar seguimiento a las solicitudes de servicios. Los principales actores implicados en este proyecto de ciudad inteligente, son los industriales y empresarios del sector de servicios, así como los usuarios de la aplicación. Las empresas privadas registradas deberán ser formales, ubicadas en domicilios fidedignos y estar registradas ante el SAT. Los usuarios evalúan el nivel del servicio atendido generando la mejora continua en la calidad de servicios que se ofrecen a través de la plataforma, impulsando a las empresas a ser más competitivas, creando fuentes de trabajo y mejorando el nivel económico de una región.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

Las ciudades inteligentes se basan en las Tecnologías de la Información y Comunicación (TIC) para la prestación de servicios públicos de alta calidad, seguridad, productividad, competitividad, innovación, emprendimiento, participación, formación y capacitación.

Una ciudad o complejo urbano resulta inteligente en la medida que las inversiones que se realicen en capital humano, en aspectos sociales, en infraestructuras de energía, tecnologías de comunicación e infraestructuras de transporte, contemplen una calidad de vida elevada, un desarrollo económico-ambiental sostenible, una gobernanza participativa, con una gestión prudente y reflexiva de los recursos naturales, y un buen aprovechamiento del tiempo de los ciudadanos.

Bouskela et. al. (2016) afirman que una ciudad inteligente integra sus diferentes áreas utilizando redes de comunicación de banda ancha, computación en la nube, dispositivos móviles, programas de análisis y sensores. Por ejemplo, existen sensores para proporcionar información en tiempo real para los niveles de ruido, contaminación ambiental, tráfico y condiciones climáticas. Así también, existen sistemas de iluminación pública que a través de las condiciones del entorno maneja de manera dinámica los niveles de iluminación. En saneamiento, se utilizan sensores para monitorear el volumen de los residuos. En cuanto a seguridad existen sistemas de cámaras para la prevención de delitos implementando la georreferenciación de datos y el análisis de incidencias de crímenes para tomar medidas preventivas.

Quiñones, Ureña y Carruyo (2016) exponen que el desarrollo de aplicaciones de control sistematizado es la base para el desarrollo de una ciudad inteligente, ya que por medio de ellas es posible desarrollar sistemas estratégicos que permitan mantener el crecimiento de las comunidades, la movilidad personal, comodidad, calidad y bajos costos para los ciudadanos. Es por ello importante el desarrollo y aplicación de tecnologías que favorezcan procesos en el ámbito empresarial impulsando y haciendo efectiva la gestión de servicios con el uso de las TIC.

Los principales actores del sector privado implicados en un proyecto de ciudad inteligente, son los industriales y empresarios de sectores clave, tales como energía, agua, transportes y servicios, así como la administración y la consultoría.

Las nuevas tecnologías juegan un papel importante en el desarrollo de proyectos integrales para modernizar las ciudades. En la vida diaria, las personas requieren de manera continua la atención de servicios y no disponen de una herramienta o directorio con información de contactos confiables para localizar empresas que atiendan sus necesidades o solicitudes, mencionando algunos de los siguientes sectores:

- ✓ Automotriz
- ✓ Electricidad
- ✓ Electrónica y línea blanca
- ✓ Entretenimiento
- ✓ Servicios para pagos diversos

- ✓ Fontanería
- ✓ Mantenimiento de equipo de cómputo
- ✓ Otros

Con el desarrollo de este proyecto se creó un sistema de información integral que incluye una aplicación móvil con acceso a un catálogo de empresas y servicios a domicilio que son calificados por los usuarios de manera permanente.

Los usuarios una vez que solicitan los servicios, estos son atendidos por la empresa correspondiente a través de una aplicación Web y monitoreados por un administrador general de la plataforma en una aplicación de escritorio.

2. ANTECEDENTES

Piñar (2017) definen a una ciudad inteligente como la ciudad que mejora la vida a través del desarrollo de tecnologías que optimizan recursos y mejoran la calidad de vida de los habitantes.

Méndez (2017) menciona que desde el año 2005 en Estados Unidos se planteó aplicar la tecnología utilizando parámetros y procesos de la teoría de sistemas en las ciudades para hacerlas más eficientes y promover su desarrollo. En la última década se ha ido configurando el significado, utilidades y conveniencia de la llamada ciudad inteligente, más conocida como Smart City. Las ciudades inteligentes traen consigo una mejora en las condiciones de vida de las personas y pueden llegar a facilitar la sostenibilidad.

En los últimos años se han realizado propuestas para desarrollar las ciudades donde se incorpore a las tecnologías de la información y de la comunicación (TIC), desarrollo sustentable, innovación, calidad de vida y la participación ciudadana (Piñar, 2017).

Koekkoek (2011), indica que 2011 se caracterizó por el desarrollo exponencial de aplicaciones para dispositivos móviles que permitían a los usurios disponer de diversas funcionalidades, servicios y productos.

Moreno (2014) comenta que en México son muy pocas las iniciativas enfocadas al desarrollo de aplicaciones móviles. En el ámbito empresarial mexicano existen prácticas para estimular el desarrollo de aplicaciones, sin embargo, se requiere mayor impulso mediante organización, difusión y patrocinios. México se identifica actualmente como importador de aplicaciones móviles, por lo que es necesario fomentar el desarrollo de aplicaciones en el país. Adoptar tecnologías y aplicaciones móviles (apps) representan una oportunidad de crecimiento para las pequeñas y medianas empresas en México, ya que el 95.4% de las unidades económicas son pymes, según datos del Censo Económico del Instituto Nacional de Estadística y Geografía (Inegi).

Por otra parte, en la implementación y desarrollo de proyectos para ciudades inteligentes se debe tener en cuenta a la técnica, el diseño urbano, la arquitectura y los derechos. Los derechos son las consideraciones en torno a la transparencia y acceso a la información pública y en torno al respeto a la protección de datos (Piñar, 2017).

El acceso a la información pública en un entorno de datos abiertos y el respeto a la protección de datos han de ser especialmente relevantes. Por un lado, para garantizar el acceso a datos y evitar que la protección de los datos quede reducida a la nada, con la aplicación de técnicas de *Big Data* y en el entorno de la *Internet de las cosas (Internet of things).*

Durante la investigación, fueron localizadas algunas tecnologías similares al proyecto TOC-TOC como los son Ay Fix, Taster y Aliada. Sin embargo, ninguna competencia integra los tres escenarios involucrados en TOC-TOC como lo son la aplicación móvil, aplicación Web y aplicación de escritorio conectadas a un servidor de base de datos por Internet, que brinda además diversos informes y estadísticas a las empresas.

3. OBJETIVOS

3.1 Objetivo general

Desarrollar un sistema de información integral que permita atender diversos tipos de servicio a domicilio a través de una aplicación móvil, enviando las solicitudes a un servidor de Internet para ser atendidas en una aplicación Web y administradas por una aplicación de escritorio.

3.2 Objetivos específicos

- ✓ Ofrecer servicios a domicilio a usuarios.
- ✓ Impulsar el uso de aplicaciones móviles en negocios de la región.
- ✓ Controlar y monitorear los servicios.

4. MARCO TEÓRICO

4.1 Internet de las cosas

Las recientes innovaciones de la electrónica, la informática y las tecnologías de la información y la comunicación (TIC) han propiciado, de una parte, un crecimiento exponencial de la capacidad material de procesamiento de los sistemas de tratamiento de información y permitido la miniaturización de microprocesadores empleados como sensores para la captación de datos. Estas invenciones, se construyen y amplifican con las tecnologías a través de los mundos físico y digital. Por otro lado, el proceso continuo de expansión de Internet y gestación de nuevas tecnologías, servicios y plataformas han generado el fenómeno conocido como Internet de las Cosas (Internet of Things, habitualmente denominado por sus siglas en inglés IoT) como la próxima revolución en nuestro mundo interconectado (Barrio, 2018).

Ciudades inteligentes o infraestructuras públicas, pueden aprovechar el potencial de esta tecnología en beneficio de los ciudadanos, las empresas y las administraciones públicas, pero también genera amenazas referentes a la intimidad-privacidad y a la seguridad. El IoT ha sido llamado como un agente de la cuarta revolución industrial, la Industria 4.0, junto con la inteligencia artificial, la robótica, la impresión 3D y 4D, la nanotecnología, la biotecnología o la ciencia de los materiales, por nombrar algunos de sus motores fundamentales. Hoy en día, Internet intercomunica no sólo ordenadores, incluyendo no sólo a los primeros dispositivos como los teléfonos inteligentes (Smartphones) o las tabletas (Tablets), sino también ropas tecnológicas o wearables (relojes, pulseras inteligentes o gafas de realidad aumentada), electrodomésticos (frigoríficos, aspiradoras, etc.), televisores, videoconsolas, automóviles, elementos de edificios (cámaras de seguridad, controles de acceso, sensores de temperatura, etc.), hasta infraestructuras públicas. La incorporación de capacidades inteligentes a todos estos objetos pasivos a través de dispositivos hardware específicos o sensores inalámbricos, pueden recopilar datos para su envío a centros de procesamiento por medio de una estructura de red interconectada, transmitiendo, compilando y analizando datos. Se espera un número de dispositivos estimados a veinte billones para el año 2020, ya que el IoT crece diariamente donde los objetos pasan a ser un origen de datos (Barrio, 2018).

4.2 Programación en entornos móviles

Ramírez, Contreras y Contreras (2014) definen a un dispositivo móvil como un aparato de pequeño tamaño, con capacidades de procesamiento, conexión permanente o intermitente a una red, memoria limitada, diseñado para una función. Existen varios tipos de dispositivos móviles, desde los reproductores de audio portátiles hasta los navegadores GPS, incluyendo los teléfonos móviles, los PDA’s o los Tablet PC.

La programación móvil genera aplicaciones para celulares, Smartphones, PDA's, PocketPC y dispositivos con recursos limitados. Los Sistemas Operativos para móviles disponibles son Symbian, Palm OS y Windows Mobile, Android, iOS y Linux para móviles. Existen varias plataformas móviles; Bada de Samsung, Symbian principalmente de Nokia, Windows Phone de Microsoft, iOS de Apple, BlackBerry OS de RIM, Android de Google, entre otras.

Bada es una plataforma desarrollada por Samsung que incluye herramientas como el SDK (Software Development Kit, Kit de Desarrollo de Software), el IDE (Integrated Development Environment, Entorno de Desarrollo Integrado) y emuladores. Sin embargo, presenta limitaciones a los desarrolladores en el uso de ciertas características como el GPS, acelerómetro, etc. El OS Symbian es una plataforma desarrollada por empresas como Samsung, Ericsson, Nokia, Siemens, etc. Actualmente, es mantenida principalmente por Nokia. Utiliza una IDE QT y un SDK para el lenguaje de programación C++, el cual es el lenguaje nativo de esta plataforma.

Windows Phone es una plataforma desarrollada por Microsoft siendo una versión mejorada del antiguo SO Windows Mobile, para dispositivos como PDA's entre otros. Este sistema operativo móvil ofrece la plataforma de desarrollo .NET, sus lenguajes principales como C# y Visual BASIC en el entorno Visual Studio.

La plataforma iOS es desarrollada por Apple para sus dispositivos móviles iPhone, iPod, iPad y Apple TV. El desarrollo para ésta plataforma se lleva a cabo de manera nativa mediante el IDE Xcode, el cual incluye todas las herramientas necesarias, como el SDK y emuladores, para los dispositivos Apple. Utiliza el lenguaje de programación Objective C. El desarrollo nativo se realiza bajo el Sistema Operativo OSX.

La plataforma BlackBerry es desarrollada por la empresa canadiense RIM (Research In Motion). Es posible desarrollar en las plataformas, haciendo uso de tecnologías Web; Java de manera nativa, o adaptar aplicaciones para Android de tal forma que se puedan ejecutar en el BlackBerry OS.

4.3 Visual Basic y su integración a Toc-Toc

Visual hace referencia al método que se utiliza para crear la interfaz gráfica de usuario (GUI). Sin escribir demasiadas líneas de código para implementar la apariencia y la ubicación de los elementos de la interface, ya que se arrastran y colocan objetos prefabricados en la pantalla (Del Sole, 2017).

La palabra Basic hace referencia al lenguaje Basic (Begginers All-Purpose Symbolic Introduction Code), un lenguaje utilizado más que ningún otro lenguaje en la historia de la informática o computación. Visual Basic ha evolucionado a partir del lenguaje BASIC original y contiene instrucciones, funciones y palabras clave, relacionadas con la interfaz gráfica de Windows. Los principiantes pueden crear aplicaciones útiles con pocas palabras clave, pero la eficacia del lenguaje permite a los profesionales desarrollar grandes proyectos (Fossati, 2017).

Características de Visual Basic:

- Barra de título: muestra el nombre del proyecto y del formulario actual.
- Barra de menús: agrupa menús despegables con todas las operaciones de Visual Basic.
- Barra de herramientas estándar: contienen elementos que se utilizan con mayor frecuencia en un proyecto.
- Ventana de formulario: es el área donde se diseña la interfaz gráfica, es decir, es donde se inserta elementos gráficos, como botones, imágenes, casilla de verificación, cuadros de listas, etc.

- Cuadro de herramientas: presenta todos los controles necesarios para diseñar una aplicación, como cuadros de texto, etiquetas, cuadros de listas, botones de comandos, etc.
- Ventana de proyecto: muestra los elementos involucrados en el proyecto, como formularios, módulos, controles, etc.
- Ventana de posición del formulario: muestra la ubicación que tendrá el formulario en la pantalla, cuando ejecute la aplicación.
- Ventana propiedades: muestra todas las propiedades del control actualmente seleccionado.

4.4 Aplicaciones Web

Las aplicaciones inicialmente se ejecutaban en una única máquina, que eran además la máquina que alojaba los datos que se manipulaban. Posteriormente se hicieron populares las arquitecturas cliente/servidor, en las que la interfaz de usuario de las aplicaciones de gestión se ejecuta en la máquina del cliente, pero los datos que se manipulaban suelen almacenarse en un sistema de gestor de base de datos.

La creación de aplicaciones Web, requiere la existencia de software ejecutándose en el servidor que genere automáticamente los ficheros HTML que se visualizan en el navegador del usuario. Desde el punto de vista de un programador, se puede tener una amplia gama de herramientas a su disposición. Alguna de las opciones entre las que puede elegir el programador puede ser el manejo de ASP.NET incluidas en la plataforma .NET (Berzal, Cortijo y Cubero, 2007)

4.5 ASP.NET

Berzal (2007) comenta que ASP.NET es un modelo de desarrollo Web unificado que incluye los servicios necesarios para crear aplicaciones Web empresariales con el código mínimo. ASP.NET forma parte de .NET Framework y tiene acceso a las clases en .NET Framework. El código de las aplicaciones puede escribirse en cualquier lenguaje compatible con el Common Language Runtime (CLR), entre ellos Microsoft Visual Basic, C#, JScript .NET y J#. Estos lenguajes permiten desarrollar aplicaciones ASP.NET que se benefician del Common Language Runtime, seguridad de tipos, herencia, etc.

ASP.NET incluye:

- Marco de trabajo de página y controles
- Compilador de ASP.NET
- Infraestructura de seguridad
- Funciones de administración de estado
- Configuración de la aplicación
- Supervisión de estado y características de rendimiento
- Capacidad de depuración
- Marco de trabajo de servicios Web XML
- Entorno de host extensible y administración del ciclo de vida de las aplicaciones
- Entorno de diseñador extensible

4.5.1 Controles y elementos Web ASP.NET

Los elementos Web ASP.NET son un conjunto integrado de controles para crear sitios Web que permiten a los usuarios finales modificar el contenido, la apariencia y el comportamiento de las páginas Web directamente desde un explorador. Las modificaciones se pueden aplicar a todos los usuarios del sitio o a usuarios individuales. Cuando los usuarios modifican páginas y controles, es posible guardar la configuración para conservar las preferencias personales de un usuario en futuras sesiones del explorador; esta característica se denomina personalización. Estas funciones de los elementos Web significan que los desarrolladores pueden permitir que los usuarios finales personalicen dinámicamente una aplicación Web, sin intervención del desarrollador o del administrador (ASP, 2018).

Con el conjunto de controles de elementos Web, el desarrollador puede permitir a usuarios finales lo siguiente:

- Personalizar el contenido de la página.
- Personalizar el diseño de página.
- Exportar e importar controles.
- Crear conexiones.
- Administrar y personalizar la configuración de todo el sitio.

4.5.2 MySQL Connector

MySQL (2018) incluye información de MySQL Connector / NET y permite desarrollar aplicaciones .NET que requieren conectividad de datos segura y de alto rendimiento con MySQL. Implementa las interfaces ADO.NET necesarias y las integra en herramientas compatibles con ADO.NET.

Características proporcionadas por MySQL:

- .NET Core y Entity Framework Core para permitir el desarrollo multiplataforma.
- Compatibilidad con paquetes grandes para enviar y recibir filas y valores de hasta 2 gigabytes de tamaño.
- Compresión de protocolo, que permite comprimir el flujo de datos entre el cliente y el servidor.
- Conexiones utilizando sockets TCP / IP, canalizaciones con nombre o memoria compartida en Windows.
- Conexiones usando sockets TCP / IP o sockets Unix en Unix.
- El framework Open Source Mono desarrollado por Novell.
- Marco de la entidad.
- .NET para Windows 8.x
- Tienda de aplicaciones (aplicaciones de Windows RT Store).
- Connector / NET admite versiones completas de Visual Studio 2008, 2010, 2012, 2013, 2015 y 2017, aunque la extensión del soporte puede ser limitada dependiendo de las versiones de Connector / NET y Visual Studio.

Existen varias versiones de conectores MySQL entre ellos se utiliza la versión 6.10 para el sistema de administración general de la plataforma el cual se describe a continuación:

Connector / NET 6.10 incluye soporte de Entity Framework Core y habilita la compresión en la versión .NET Core del controlador para soporte expandido multiplataforma a Linux y macOS cuando se usa .NET Core framework. Es posible encontrar diferentes versiones para proyectos Web en este caso se utiliza la versión 8.0 para el sistema de integración Web, que realiza funciones gracias a ASP.NET, esta versión incluye soporte para versiones 6.10 en adelante. El software de código abierto MySQL se proporciona bajo la Licencia GPL. Connector / NET 8.0 es una continuación de Connector / NET 7.0 e incluye los siguientes paquetes:

- MySql.Data
- MySql.Web
- MySql.Data.EntityFrameworkCore
- MySql.Data.EntityFrameworkCore.Design.

Clases y métodos para conexiones MySQL utilizadas

Clase MySqlConnection

La versión utilizada es la 6.10 para sistema de administración general de plataforma. Se importan las librerías correspondientes con el software y controladores previamente descargados. Para este ejemplo se implementó un apartado o módulo para realizar las conexiones de manera global a nivel sistema.

```
Imports MySql.Data

Imports MySql.Data.MySqlClient
```

Se declaran las variables correspondientes para realizar la conexión a la base de datos ya sea remota o local. En los parámetros que se presentan a continuación se insertaron para conexión remota a un servidor en Internet (los datos de parámetros se eliminaron por motivos de seguridad), de esta manera se obtiene acceso a la base de datos correspondiente.

```
Public cadenaConexion As String

Public Conexion As New MySqlConnection

cadenaConexion                                                                    =
("server='IP';port='PUERTO';uid='USER';pwd='PASSWORD';database='DATABASE';Persist
Security Info='true';protocol='TCP';Min Pool Size='0';Max Pool Size='100';")

Conexion = New MySqlConnection(cadenaConexion)

Conexion.Open()
```

Clase MySqlCommand

Cuando se establece una conexión con la base de datos MySQL, el siguiente paso es llevar a cabo las operaciones de base de datos deseadas. Esto se puede lograr mediante el uso del *MySqlCommand*.

Después de haber sido creado el objeto, es accesible a tres métodos principales de interés:

- *ExecuteReader* para consultar la base de datos. Los resultados generalmente se devuelven en un *MySqlDataReader* creado por *ExecuteReader*.

- *ExecuteNonQuery* para insertar, actualizar y eliminar datos.
- *ExecuteScalar* para devolver un solo valor

Una vez que se ha creado un objeto *MySqlCommand*, se llamará a uno de los métodos anteriores para llevar a cabo una operación de base de datos, como realizar una consulta. Los resultados generalmente se devuelven a un *MySqlDataReader* y luego se procesan.

Clase MySqlDataReader

El *MySqlReader* contiene los resultados generados por el query ejecutado en el objeto de comando. Una vez que los resultados han sido obtenidos en un *MySqlReader*, pueden ser procesados. En este caso, la información se imprime mediante un *while*. Finalmente, el *MySqlReader* se elimina ejecutando el método *Close()*.

En el siguiente ejemplo se muestra una consulta a una tabla MySQL de nombre TABLA, el cual obtendrá el name_client de dicha tabla y será asignado a un elemento DataGridView.

```
ConexionGlobal()
Dim Comando As New MySqlCommand("select TABLA.name_client from TABLA", Conexion)
Dim Lector As MySqlDataReader
Lector = Comando.ExecuteReader
While Lector.Read()
       DG.Rows.Add(Lector(0))
End While
Lector.Close()
```

ConexionGlobal(): hace referencia al módulo que se creó anteriormente con la clase *MySqlConnection*, instanciando a la variable "Conexión" para realizar dicha conexión.

Comando: es la variable de tipo *MySqlCommand* encargada de ejecutar cualquier sintaxis SQL dentro de un conjunto de registros en una base de datos.

Lector: es la variable de tipo *MySqlDataReader* que se encarga de recibir los datos o registros emitidos por el comando, que en conjunto con *ExecuteReader* se obtienen los datos y con el método Read() puedan ser manipulados por el usuario. Para concluir el *DataReader* que se encuentra activo, se ejecuta el método *Close()* que puede devolver un false en la lectura de registros.

Método ExecuteNonQuery

El procedimiento para realizar un *ExecuteNonQuery* se usa para insertar, actualizar y borrar datos, en el siguiente ejemplo se observa su uso para insertar datos en una tabla.

```
Dim R As String
R = "insert into TABLA ([Campos y/o Columnas]) values ('" & dato1 & "', '" & dato2 & "')"
Dim Comando As New MySqlCommand(R, Conexion)
Comando.ExecuteNonQuery()
```

La sintaxis se construye, posteriormente se crea el objeto de *Comando* y se invoca al método `ExecuteNonQuery`. Se puede acceder a la base de datos MySQL y verificar que la actualización se haya realizado correctamente.

Método ExecuteScalar

El método `ExecuteScalar` se usa para devolver un solo valor en una variable simple. El siguiente ejemplo muestra la manera de usar *ExecuteScalar* para adaptarlo a un incremento para un nuevo registro obteniendo en N el número exacto de registros en la base de datos hasta el momento y al sumar uno incrementamos para un nuevo registro.

```
Dim N As Integer
Dim nuevoRegistro As String
Dim Comando As New MySqlCommand("SELECT COUNT(*) FROM TABLA", Conexion)
N = Comando.ExecuteScalar() + 1
nuevoRegistro = N
```

4.5.3 ASP.NET y su integración a Toc-Toc

La utilización de ASP.NET dentro del proyecto Toc-Toc conforma la aplicación Web destinada a la parte administrativa del negocio o empresa que esté vinculada a Toc-Toc, se cuenta con acceso a datos de empleados de un negocio en específico y se filtra la información por medio del identificador o clave del empleado.

El uso de *Master Page* se hace presente en la aplicación Web que simplifica el diseño de encabezados y funciones principales dentro de la aplicación, su acceso por medio de Login con validación permite tener mayor seguridad y en conjunto con los controles de acceso a datos, definiendo un ambiente seguro y confiable para su utilización.

Los controles, clases y métodos de acceso a la base de datos son exactamente los mismos que se encuentran en la aplicación de escritorio realizada en Visual Basic, solo teniendo mayor énfasis en las versiones Web de *MySQL Connector*, la cual otorga mejor comunicación cliente/servidor a la hora de realizar funciones.

Se pueden encontrar diferentes versiones para proyectos Web en este caso se utiliza la versión 8.0 para el sistema de integración Web, que realiza funciones gracias a ASP.NET.

4.6 JavaScript

JavaScript es un lenguaje de programación para diseñar scripts en las páginas HTML y se ejecutan en el navegador. Estos scripts consisten en funciones que son invocadas desde el propio HTML a través de eventos. Los efectos de los eventos se refieren por ejemplo a un botón para cambiar de forma al pasar el ratón por encima, o abrir una ventana nueva al pulsar en un enlace. En un script se pueden definir estructuras condicionales como iterativas, así como funciones. Los scripts que contienen estas funciones suelen estar en el head del documento. Las funciones son invocadas desde el body del documento, cuando ocurra el evento (Navarrete, 2006).

Ejemplo de script para cargar un seleccionador de fechas:

```
<script type="text/javascript">
        {
```

```
            if (history.forward(1))
                location.replace(history.forward(1))
        }
</script>
     <script type="text/javascript">
         {
             if (history.forward(1))
                 location.replace(history.forward(1))
         }
      </script>
    <link href="styles/jquery-ui.css" rel="stylesheet" />
    <script src="Scripts/jquery-1.10.2.min.js"></script>
    <script src="Scripts/jquery-ui.js"></script>
        <script>
            $(function () {
                $('#txtFecha1').datepicker(
                    {
                        dateFormat: 'yy/mm/dd',
                        changeMonth: true,
                        changeYear: true,
                        yearRange: '1950:2100'
                    });
            })
        </script>
    <script>
        $(function () {
            $('#txtFecha2').datepicker(
                {
                    dateFormat: 'yy/mm/dd',
                    changeMonth: true,
                    changeYear: true,
                    yearRange: '1950:2100'
```

```
            });
        })
    </script>
```

4.7 Lenguaje PHP como integración de Web Services tipo REST con JSON

PHP (Personal Home Page), es un lenguaje interpretado libre, usado originalmente para el desarrollo de aplicaciones que actuarán en el lado del servidor, capaces de generar contenido dinámico en la WWW. Figura entre los primeros lenguajes posibles para la inserción en documentos HTML. El código es interpretado en el lado del servidor por el módulo PHP que también genera la página Web para ser visualizado en el lado del cliente. El lenguaje evolucionó, ofreciendo funcionalidades en la línea de comandos y características adicionales (Dimes, 2016).

PHP es un software libre bajo PHP License, es una licencia de código abierto, certificada por la Open Source Initiative. La licencia de PHP es una licencia que no tiene restricciones de "copyleft" asociadas con la GNU General Public License (GPL) que proviene de la primera licencia con copyleft, escrita por Richard Stallman, debido a las restricciones en los términos de uso de PHP (Arias, 2013).

El lenguaje PHP es un lenguaje de programación de dominio específico, es decir, su alcance se extiende a un campo de actuación que es el desarrollo Web. Su propósito principal es de implementar soluciones Web veloces, simples y eficientes. Sus principales características son:

- Velocidad y robustez
- Estructurado y orientado a objetos.
- Portabilidad – independencia de plataforma
- Ejecución en cualquier lugar
- Tipeado dinámico
- Open-Source

4.7.1 Web Services REST

REST define principios arquitectónicos para diseñar servicios Web enfocados en los recursos del sistema, incluyendo accesos al estado de dichos recursos y transferencia por HTTP hacia los clientes escritos en diversos lenguajes. Esta idea inicia en la Universidad de California, durante una conferencia académica "Estilos de Arquitectura y el Diseño de Arquitecturas de Software basadas en Redes", la cual analizaba un conjunto de principios arquitectónicos de software para usar a la Web como una plataforma de *Procesamiento Distribuido* (De Seta, 2008).

La implementación concreta de un servicio Web REST se basa en cuatro principios de diseño fundamentales:

- Utiliza los métodos HTTP de manera explícita
- No mantiene estado
- Expone URIs con forma de directorios
- Transfiere XML, JavaScript Object Notation (JSON)

Las características de los servicios Web REST es el uso explícito de los métodos HTTP. Por ejemplo, HTTP GET como un método productor de datos.

Los servicios Web REST requieren escalar para satisfacer la demanda. Los servidores intermedios que mejoran la escalabilidad hacen necesario que clientes de servicios Web REST envíen sus peticiones completas e independientes.

Una petición completa que es a su vez independiente, permite que el servidor no tenga que recuperar ninguna información de contexto o estado al procesar la petición. Una aplicación o cliente de servicio Web REST entonces debe incluir dentro del encabezado y del cuerpo HTTP de la petición todos los parámetros, contexto y datos que necesita el servidor para generar la respuesta. Por lo anterior, el no mantener el estado mejora el rendimiento de los servicios Web simplificando el diseño e implementando componentes del servidor, pues al no haber estado en el servidor se elimina la tarea de sincronizar los datos de la sesión con una aplicación externa.

4.7.2 States o estados de Web Service

La Figura 1 muestra un servicio con estado, donde una aplicación realiza peticiones para la activity siguiente en un conjunto de resultados múlti-página, presumiendo que el servicio mantiene información sobre la última activity que solicitó el cliente. En un diseño con estado, el servicio incrementa y almacena en algún lugar una variable Activity_siguiente para poder responder a las peticiones siguientes.

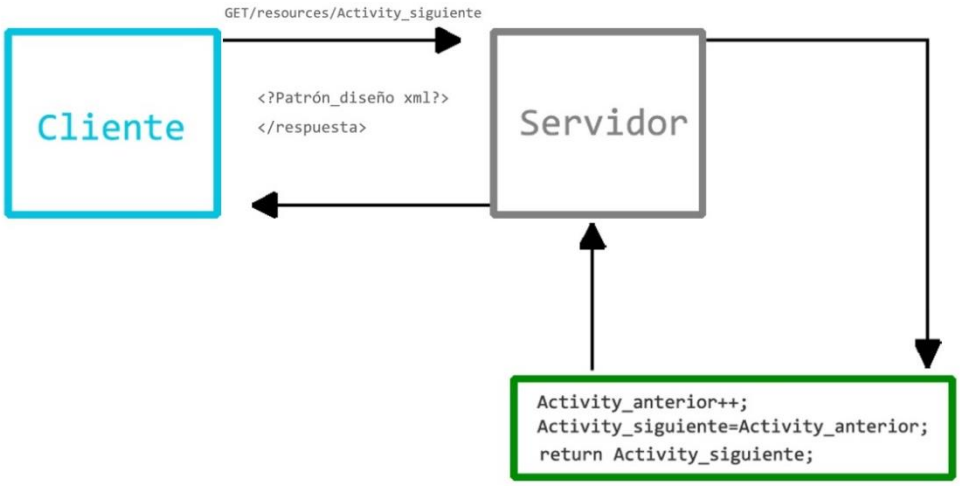

Figura 1. Servicio Web con estado.

Los servicios sin estado son más sencillos de diseñar, escribir y distribuir a través de múltiples servidores. Un servicio sin estado funciona mejor, permitiendo la responsabilidad de mantener el estado al cliente de la aplicación. En un servicio Web REST, el servidor es responsable de generar las respuestas y proveer una interfaz para que el cliente mantenga el estado de la aplicación por su cuenta. Es decir, en el mismo ejemplo de una petición de datos en múltiples páginas, el cliente debe incluir el número de página a recuperar en vez de pedir "la siguiente activity", ver la Figura 2.

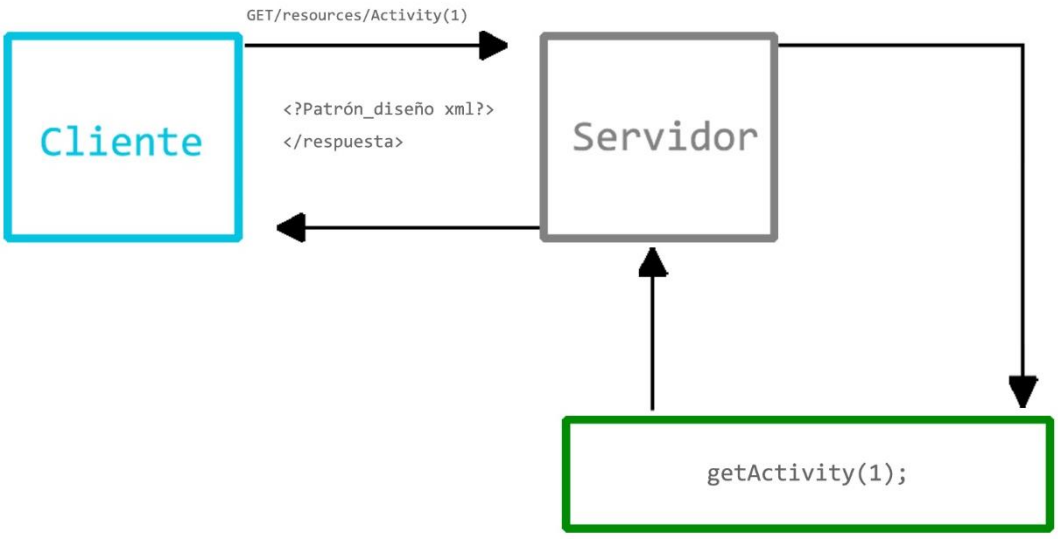

Figura 2. Servicio Web sin estado.

REST transfiere XML, JSON, o ambos

La representación de un recurso refleja el estado actual del mismo y sus atributos al momento en que el cliente de la aplicación realiza la petición. Esto podría ser una representación de un recurso de registro de la base de datos que consiste en asociar etiquetas y columnas en XML que contienen los valores de las filas. Así como de un modelo de datos del sistema. La última restricción que existe al momento de diseñar un servicio Web REST se refiere al formato de los datos que la aplicación y el servicio intercambian en las peticiones/respuestas.

4.7.3 JSON

JSON (JavaScript Object Notation - Notación de Objetos de JavaScript) es un formato sencillo de intercambio de datos. Está fundamentado en un subconjunto del Lenguaje de Programación JavaScript. JSON es un formato de texto que es independiente del lenguaje, utiliza convenciones que son ampliamente conocidos por los programadores de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y otros (JSON, 2018).

JSON se basa en dos estructuras:

- Una colección de pares de nombre/valor, conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo.
- Una lista ordenada de valores, implementada como arreglos, vectores, listas o secuencias.
- El formato de intercambio de datos que es independiente del lenguaje de programación y se basa en su propia estructura.

Interacción PHP + JSON

Para que la comunicación e interacción dinámica de la base de datos sea correcta se necesita la implementación de servicios Web. En el proyecto Toc-Toc, se define un Web service sin estado para no mantener un estado activo del cliente dentro del proceso de intercambio de datos, mejorando el rendimiento de la aplicación móvil y el tráfico de datos en Internet. El contenido de los datos, una vez que se hayan leído y decodificado dentro de la aplicación móvil

se transferirán directamente a formato XML donde un patrón de diseño muy similar al MVC se encarga de realizar los diseños o vistas correspondientes para el usuario final. A continuación, se muestra un código ejemplo para obtener los datos de todas las categorías existentes, realizando una consulta de la Descripción y la cadena de URL en formato String del logotipo para decodificarlo en la aplicación móvil:

```
<?php
define('DB_HOST', IP, SERVER);
define('DB_USER', USUARIO);
define('DB_PASS', 'PASSWORD');
define('DB_NAME', 'BD’);
 $conn = new mysqli(DB_HOST, DB_USER, DB_PASS, DB_NAME);
  if (mysqli_connect_errno()){
    die ('No se pudo conectar a la base de datos' . mysqli_connect_error());}
  $stmt= $conn->prepare("SELECT Descripcion, Logotipo FROM categorias order by Descripcion;");
  $stmt->execute();
  $stmt->bind_result($Descripcion, $Logotipo);
  $product = array();
  while ($stmt->fetch()){
    $temp= array();
    $temp['Descripcion']= utf8_encode($Descripcion);
    $temp['Logotipo']= utf8_encode($Logotipo);
    array_push($product, $temp);}
  echo json_encode($product);
```

La función *json_encode()* realiza el trabajo de codificar los datos o el resultado de la consulta para convertirlos a formato JSON y que puedan ser leídos o interpretados por algún otro lenguaje de programación en este caso en Android para la aplicación Móvil. A continuación, en la Figura 3 se muestra el resultado de la anterior consulta en formato PHP vista desde un navegador de Internet.

El archivo PHP para la realización de esta consulta es llamado *ConsultaDinamicaCategorias.php* que se encuentra alojado en un servidor de Internet, ver Figura 3, y que además puede ser consultado en la siguiente liga:

http://developerscdguzman.com/toctoc/consultaDinamicaCategorias.php

[{"Descripcion":"Automotr\u00edz","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasAutomotr\u00edz.png"},
{"Descripcion":"Belleza","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasBelleza.png"},
{"Descripcion":"Carpinter\u00eda","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasCarpinter\u00eda.png"},
{"Descripcion":"Cerrajer\u00eda","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasCerrajer\u00eda.png"},
{"Descripcion":"Comida","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasComida.png"},
{"Descripcion":"Deportes","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasDeportes.png"},
{"Descripcion":"Electricidad","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasElectricidad.png"},
{"Descripcion":"Electr\u00f3nica y Linea B","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasElectr\u00f3nica y Linea B.png"},
{"Descripcion":"Entretenimiento","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasEntretenimiento.png"},
{"Descripcion":"Fontarer\u00eda","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasFontarer\u00eda.png"},
{"Descripcion":"Guarder\u00eda","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasGuarder\u00eda.png"},
{"Descripcion":"Inform\u00e1tica","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasInform\u00e1tica.png"},
{"Descripcion":"Jardiner\u00eda","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasJardiner\u00eda.png"},
{"Descripcion":"Mascotas","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasMascotas.png"},
{"Descripcion":"Mudanzas","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasMudanzas.png"},
{"Descripcion":"Reposter\u00eda","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasReposter\u00eda.png"},
{"Descripcion":"Salud","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasSalud.png"},
{"Descripcion":"Transporte","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasTransporte.png"},
{"Descripcion":"Vinos y licores","Logotipo":"http:\/\/developerscdguzman.com\/LogosEmpresas\/LogosEmpresasVinos y licores.png"}]

Figura 3. Resultado de consulta de formato JSON.

4.8 Android como sistema operativo para la aplicación móvil

Android es un sistema operativo creado para teléfonos con pantalla táctil de nueva generación o inteligentes, las tablets comunes y las que funcionan con líneas telefónica; entrando en esta gama los relojes inteligentes, televisores y algunos nuevos automóviles. Esta empresa de nombre Android Inc, fue respaldada por Google en el año 2005 (Android, 2018).

4.8.1 Decodificación de datos en formato JSON

En las activities más importantes se implementa este modelo teniendo en cuenta que la activity cuenta con la clase con extensión .class y su archivo XML para su diseño. Se necesita añadir dos clases más las cuales una de ellas tendrá los métodos principales para generar el constructor principal que heredará de la clase principal y además añadir un archivo XML para el patrón de diseño una vez decodificados los datos que entrega el formato JSON. A continuación, se muestra la correcta realización de las clases añadidas a la clase principal (Android, 2018).

La clase ProductCategorias

La clase *ProductCategorias* se define como clase pública a nivel proyecto para declarar las variables de datos que arrojará JSON, generando su constructor y los métodos *get()* de cada variable declarada. A continuación, se muestra el código de la clase:

```
/**
 * Created by luki2 on 30/04/2018.
 */

public class ProductCategorias
{
   private String Descripcion;
   private String Logotipo;
```

```
    public ProductCategorias(String Descripcion, String Logotipo) {  //Constructor
        this.Descripcion = Descripcion;
        this.Logotipo = Logotipo;
    }

    public String getDescripcion()
    {
        return Descripcion;
    }

    public String getLogotipo()
    {
        return Logotipo;
    }
}
```

La clase ProductAdapterCategorias

La clase *ProductAdapterCategorias* define el contexto general de las clases añadidas y hereda directamente de la clase pública estática abstracta *ViewHolder* que a su vez hereda de la librería de soporte v7 donde un *ViewHolder* describe una vista de elemento y metadatos sobre su lugar dentro de *RecyclerView*:

```
public class ProductAdapterCategorias extends
RecyclerView.Adapter<ProductAdapterCategorias.ProductViewHolder>
```

Esta clase define un constructor, métodos y parámetros que serán los contenedores y visualizadores como la lista dinámica para el patrón de diseño dentro del archivo XML que se añadió. Partiendo del constructor se define el contexto general de la clase y las clases heredadas, así como la lista dinámica del tipo *ProductCategorias* la cual obtendrá las variables en forma dinámica cada vez que se le asigne un nuevo elemento o dato de información.

Para la visualización de ese contenedor o de esa lista dinámica se recurre al método *onCreateViewHolder* que su función será mostrar el diseño directamente a un recurso existente dentro del proyecto que será al archivo XML añadido con el patrón de diseño.

Dentro de la clase también se puede encontrar el método o*nBindViewHolder* el cual se encargará de conectar o enlazar los datos obtenidos por los métodos *get()* que se declararon en la clase *ProductCategorias* con el patrón de diseño, así mismo en dicho método se pueden realizar procesos comunes de Android como asignar variables, obtener posiciones de cada elemento añadido, obtener textos de un fragmento o algún otro proceso de un componente en específico, en este caso se obtiene la posición de un elemento *CardView* para saber que elemento fue seleccionado y de esa forma realizar el proceso de *onClickListener*.

Es importante conocer el número de elementos que conforman la lista dinámica para que de esa forma el método *onBindViewHolder* asigne las posiciones correspondientes, y eso se logra realizando el método *getItemCount ()* que devolverá el tamaño de la lista para saber cuántos elementos la conforman.

Finalmente se declara una clase interna miembro que de igual manera heredará de la clase pública estática abstracta *ViewHolder* para inicializar los componentes y/o variables dentro del archivo XML con el patrón de diseño que se le asignará. A continuación, se muestra el código de la clase *ProductAdapterCategorias*:

```
/**
 * Created by luki2 on 30/04/2018.
 */

public class ProductAdapterCategorias extends
RecyclerView.Adapter<ProductAdapterCategorias.ProductViewHolder>
{
    private Context mCtx;
    private List<ProductCategorias> productList;
    public ProductAdapterCategorias(Context mCtx, List<ProductCategorias> productList)
    {
        this.mCtx = mCtx;
        this.productList = productList;
    }
    @Override
    public ProductViewHolder onCreateViewHolder(ViewGroup parent, int viewType)
    {
        View holder=
LayoutInflater.from(parent.getContext()).inflate(R.layout.list_layout_categorias, parent,
false);
        return new ProductViewHolder(holder);
    }

    @Override
    public void onBindViewHolder(final ProductViewHolder holder, final int position)
    {
        final ProductCategorias product= productList.get(position);
        holder.tvCategoria.setText(product.getDescripcion());
        holder.tvLogo.setText(product.getLogotipo());

        holder.CardNeg5.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view)
            {
                if(holder.tvCategoria.getText().toString().equals(product.getDescripcion()))
                {
                    Toast.makeText(mCtx, "Presionaste " + holder.tvCategoria.getText().toString(),
Toast.LENGTH_SHORT).show();
                    Intent intent=new Intent(mCtx, RankingNegocios.class);
```

```
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP |
Intent.FLAG_ACTIVITY_SINGLE_TOP);
                intent.putExtra("categoria", holder.tvCategoria.getText().toString());
                mCtx.startActivity(intent);
            }
        }
    });
  }

  @Override
  public int getItemCount()
  {
    return productList.size();
  }

  class ProductViewHolder extends RecyclerView.ViewHolder
  {
    ImageView imgCategoria, idCategoria;
    TextView tvCategoria, tvLogo;
    CardView CardNeg5;

    public ProductViewHolder(View itemView)
    {
      super(itemView);
      imgCategoria= itemView.findViewById(R.id.imgCategoria);
      tvCategoria=itemView.findViewById(R.id.tvCategoria);
      tvLogo=itemView.findViewById(R.id.tvLogo);
      CardNeg5=itemView.findViewById(R.id.CardNeg5);
    }
  }
}
```

4.8.2 Clase principal y los métodos para la decodificación de datos JSON

La clase principal es el archivo .class que se crea automáticamente cuando se genera una nueva activity, y es la encargada de mandar tanto los datos de diseño principal con un *setContentView* o un *Inflater* como los códigos o algoritmos que contengan la lógica de programación. Esta clase es la que se ejecuta en primera instancia a la hora de correr la aplicación.

Dentro de esta aplicación se declaran las variables y los objetos que se necesitarán para mostrar en pantalla todos los procesos realizados anteriormente, los objetos que se declaran a continuación, se declaran públicos a nivel clase:

```
private static final String PRODUCT_URL ="http://TU_SERVIDOR_URL ";
RecyclerView recyclerView;
```

```
ProductAdapterCategorias adapter;
List<ProductCategorias> productList;
```

adapter es el adaptador de contenido que se mostrará en la pantalla dentro del componente *RecyclerView* y es de tipo *ProductAdapterCategorias*.

productList es la lista dinámica que representa el arreglo principal del contenedor encargado de almacenar los datos que se decodificarán del archivo JSON que se envía.

El método principal que se encarga de decodificar los archivos JSON y así poder manipularlos en la clase principal y se utiliza la clase *StringRequest* que definirá que método HTTP se utilizará, además de la URL estática que se definió. Seguido del método sobreescrito que llamará a la respuesta generada del *StringRequest* llamado *onResponse ()*, este método dará respuesta para realizar la decodificación de los datos JSON, siendo así, se tiene que ejecutar el proceso dentro de un try ya que se manejan datos de archivos.

Se declara un objeto de la clase *JSONArray* para recibir el paquete de información del JSON y almacenarlos en el objeto arreglo de dicho tipo. Posteriormente se declara un ciclo o bucle que se encargará de realizar las iteraciones según la longitud del archivo JSON y lo recibirá el objeto de tipo *JSONObject* que asignará los datos que arrojará JSON para así poder manipularlos y guardarlos en variables. Es posible lo anterior si el nombre que se asigna al *JSONObject* con los métodos *get()* es igual al nombre del parámetro o variable que se programó en el archivo PHP que genera el archivo JSON. De esta manera una vez que se asignan los datos decodificados a las variables inmediatamente se agregan a la lista dinámica *productList* del tipo *ProductCategorias*. A continuación, se muestra el código del método encargado de dichas funciones:

```
private void loadProducts()
{
   StringRequest stringRequest = new StringRequest(Request.Method.GET, PRODUCT_URL,
new Response.Listener<String>()
   {
      @Override
      public void onResponse(String response)
      {
         try{
            JSONArray products = new JSONArray(response);
            for(int i=0; i<products.length(); i++){
               JSONObject productObject= products.getJSONObject(i);
               String Descripcion= productObject.getString("Descripcion");
               String Logotipo=productObject.getString("Logotipo");
               ProductCategorias product = new ProductCategorias(Descripcion, Logotipo);
               productList.add(product); }
            adapter= new ProductAdapterCategorias(IngresoUsuario.this, productList);
            recyclerView.setAdapter(adapter);
         } catch (JSONException e){e.printStackTrace();}
      }
```

```
    }, new Response.ErrorListener()
    {
        @Override
        public void onErrorResponse(VolleyError error)
        {
            Snackbar.make(recyclerView, error.getMessage(),
Snackbar.LENGTH_SHORT).show();
        }
    });
    Volley.newRequestQueue(this).add(stringRequest);
}
```

4.8.3 Librería Volley para el uso de peticiones HTTP

Volley es una librería desarrollada por Google para optimizar el envío de peticiones HTTP desde las aplicaciones Android hacia servidores externos. Este componente actúa como una interfaz de alto nivel, liberando al programador de la administración de hilos y procesos tediosos de *parsing*, permitiendo publicar de manera sencilla resultados en el hilo principal (Revelo, 2015).

Volley es un cliente HTTP creado por Google que facilita la comunicación de red en las aplicaciones Android. *Volley* está enfocado en las peticiones, evitando la creación de código repetitivo para manejar tareas asíncronas por cada petición o incluso para parsear los datos que vienen del flujo externo.

Las características más importantes son:

- Procesamiento concurrente de peticiones.
- Priorización de las peticiones.
- Cancelación de peticiones.
- Gestión automática de trabajos en segundo plano.
- Implementación de caché en disco y memoria.
- Capacidad de personalización de las peticiones.
- Provee información detallada del estado y flujo de trabajo de las peticiones.

4.8.4 Componente RecyclerView

El *RecyclerView* es una versión de *ListView* que trabaja con varios componentes diferentes para mostrar sus datos. El contenedor general para su interfaz de usuario es un *RecyclerView* que agrega a su diseño.

RecyclerView se llena de las vistas proporcionadas por un administrador de diseño que es proporcionado. Se pueden usar los distintos administradores de diseño estándar (como *LinearLayoutManager* o *GridLayoutManager*) o implementar uno propio.

Las vistas en la lista están representadas por los objetos del titular de la vista que son instancias de una clase que define extendiendo *RecyclerView.ViewHolder*.

El titular de la vista es el encargado de mostrar un solo elemento con una vista. El *RecyclerView* crea sólo la cantidad de titulares de vista. A medida que el usuario se desplaza

por la lista, *RecyclerView* elimina las vistas fuera de pantalla y las vuelve a vincular a los datos que se desplazan por la pantalla.

Los objetos del titular de la vista son administrados por un adaptador, que se crea extendiendo *RecyclerView.Adapter*. El adaptador crea titulares de vista según sea necesario. El adaptador vincula los titulares de vista a los datos. Lo hace asignando el titular de la vista a una posición y llamando al *onBindViewHolder()* del adaptador. Ese método usa la posición del titular de la vista para determinar cuál debe ser el contenido, en función de su posición en la lista.

4.8.5 Diseño de clase principal formato XML con RecyclerView

El formato XML para el diseño de la interfaz de la activity principal contiene el componente *RecyclerView* que representa un papel importante para la adaptación del contenido dinámico que se decodificará del JSON al que se realizó la petición. A continuación, se muestra el código en formato XML, el Layout principal tendrá que estar ya sea en *RelativeLayout*, *LinearLayout*, *CoordinatortLayout* o *FrameLayout* en medidas *MatchParent* en este caso para poder realizar la adaptación de forma correcta seguido del componente *RecyclerView* en las medidas que se necesite la adaptación.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout

    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:design="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

        <android.support.v7.widget.RecyclerView
            android:layout_below="@+id/linearLayout"
            android:id="@+id/recyclerView"
            android:layout_width="match_parent"
            android:layout_height="match_parent">
        </android.support.v7.widget.RecyclerView>


</LinearLayout>
```

Cabe mencionar que esta librería de soporte no viene directamente agregada o asignada con Android Studio y por consecuente tendrá que agregarse en el *Grandle Scripts*.

La Figura 4 muestra la pantalla con el diseño y el componente *RecyclerView* como contenedor en la activity principal.

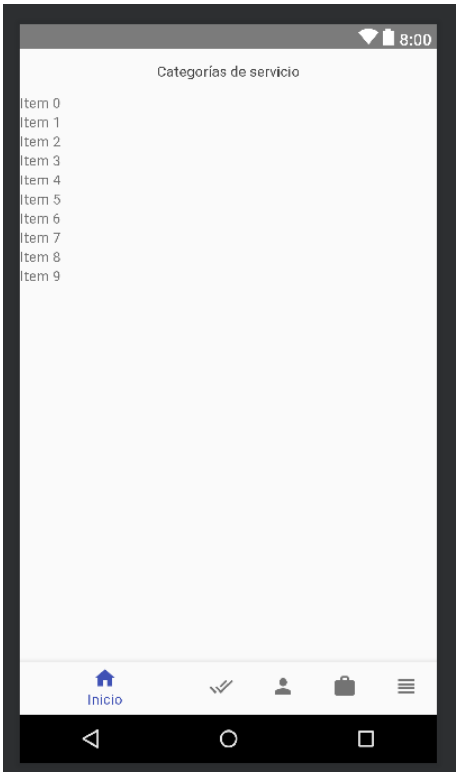

Figura 4. Diseño de la activity principal.

4.8.6 Patrón de diseño XML para lista dinámica de elementos

Al igual que en el archivo XML anterior, el Layout principal tendrá que estar a las medidas que se requieran ya sea en *RelativeLayout*, *LinearLayo*ut, *CoordinatortLayout* o *FrameLayout*. En este caso como este diseño será el patrón dinámico, se tendrá que realizar una sola vez como platilla principal y se requerirá un diseño llamativo, e intuitivo al usuario ya que será el *Layout* que se mostrará en pantalla. Se recurrirá a la librería de *Material Design* para un mejor diseño de los componentes y/o elementos que serán mostrados. Para esto se implementarán *CardViews* y *NestedScrollView* para el movimiento vertical de la pantalla. A continuación, se muestra el código XML del patrón de diseño:

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:design="http://schemas.android.com/apk/res-auto"
    xmlns:card_view="http://schemas.android.com/apk/res-auto"
    android:orientation="vertical"
    android:id="@+id/activity_ingreso_usuario"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <LinearLayout
        android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:orientation="vertical">
<android.support.v4.widget.NestedScrollView
   android:layout_width="match_parent"
   android:layout_height="match_parent">
   <LinearLayout
      android:id="@+id/LayoutContenedor"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:foregroundGravity="right"
      android:orientation="vertical">
      <android.support.v7.widget.CardView
         android:id="@+id/CardNeg5"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         android:clickable="true"
         android:stateListAnimator="@animator/animacion"
         design:cardCornerRadius="5dp"
         design:cardElevation="5dp"
         design:cardUseCompatPadding="true">
         <ImageView
            android:id="@+id/imgCategoria"
            android:layout_width="match_parent"
            android:layout_height="100dp"
            android:scaleType="center"/>
         <TextView
            android:id="@+id/tvLogo"
            android:visibility="invisible"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
         <TextView
            android:id="@+id/tvCategoria"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="left|bottom"
            android:fontFamily="sans-serif-condensed"
            android:padding="16dp"
            android:text="Automotríz"
            android:textAppearance="@style/TextAppearance.AppCompat.Medium"
            android:textColor="@color/colorTransparentWhite"
            android:textSize="16sp" />
      </android.support.v7.widget.CardView>
   </LinearLayout>
```

```
        </android.support.v4.widget.NestedScrollView>
    </LinearLayout>
</android.support.design.widget.CoordinatorLayout>
```

La Figura 5 muestra la pantalla con el diseño y los componentes del patrón de diseño para la lista dinámica.

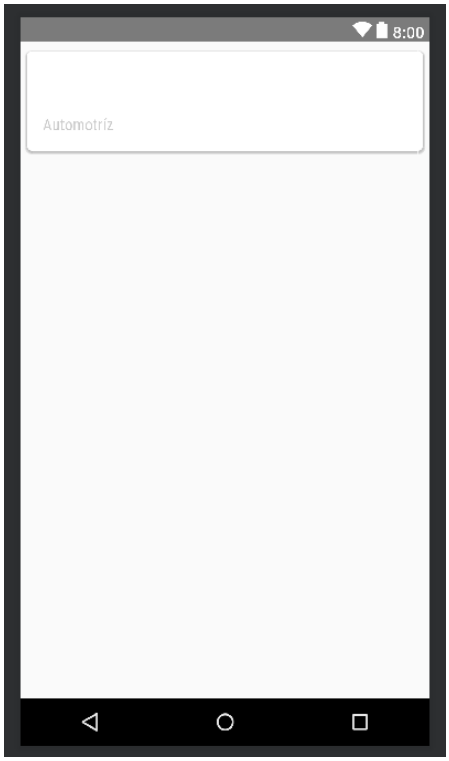

Figura 5. Patrón de diseño.

4.8.7 Pantalla principal con datos decodificados, adaptados y mostrados en tiempo de ejecución

En la Figura 6 se muestra la pantalla principal de los elementos dinámicos que se decodificaron y adaptaron a la vista principal, cabe mencionar que solo se obtuvo la descripción y el logotipo directamente de la base de datos.

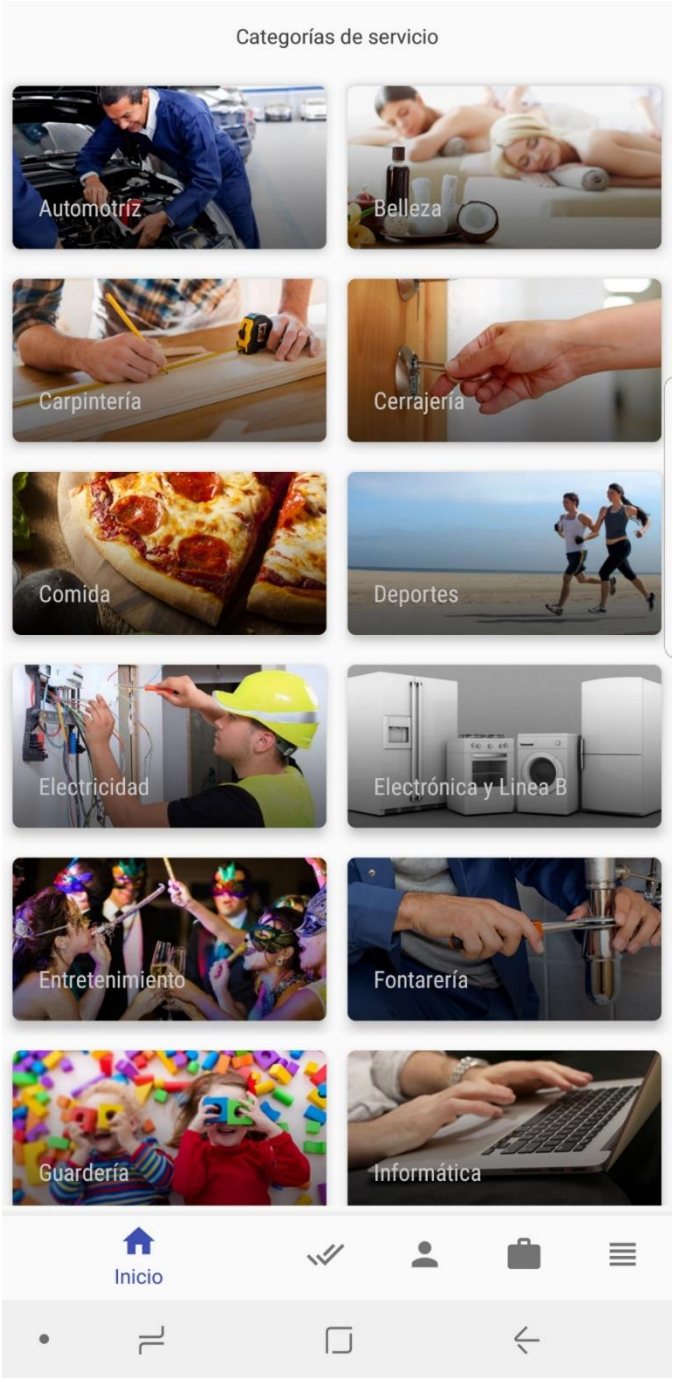

Figura 6. Activity de categorías.

4.8.8 Integración de Google Maps para localización

Google Maps es un servidor de aplicaciones de mapas en la Web que pertenece a Alphabet Inc. Este servicio propicia imágenes de mapas desplazables, así como fotografías por satélite del mundo, e incluso, la ruta entre diferentes ubicaciones o imágenes a pie de calle con *Google Street View*.

Google Maps representa un servicio de mapas, ya sea desde el móvil o en la Web. Combina características y funciones prácticas y atractivas para los usuarios. Existen más de 800.000 sitios utilizan la API de *Google Maps* y hay más de 250 millones de usuarios activos en dispositivos móviles (Google, 2018).

La integración de *Google Maps* en la aplicación móvil es fundamental para agilizar los servicios que se requieran, además de hacer un ambiente de interfaz de usuario superior y más intuitiva, automatizando la localización y destino del cliente solicitante del servicio. Por otra parte, el prestador de servicio agiliza los movimientos a la hora de realizar o atender el servicio con esta ayuda, obteniendo la dirección del cliente de forma automática y trazando las rutas posibles para llegar a dicho destino.

Para la implementación de *Google Maps* en cualquier proyecto, se requiere la utilización de una clave identificadora de proyecto, que se llama *API KEY* la cual se puede obtener directamente en la plataforma *Google Cloud Platform Console* y vinculado a la cuenta normal correspondiente de *Google*, en caso contrario se podrá iniciar con la cuenta *Google Developer* que ofrecerá mejores prestaciones y opciones para la configuración con mejores soluciones. Página de redirección:

https://console.cloud.google.com

4.8.9 Creación de Activity Maps

En la carpeta “Java” de la carpeta “App”, se desplegarán las subcarpetas del proyecto, se selecciona la carpeta principal del proyecto con Click derecho -> Nuevo -> Google -> Google Maps Activity, de esta forma se crea la clase y el XML de la activity, cabe mencionar que la clase hereda de la clase *FragmentActivity* que implementará los métodos sobreescritos:

- onMapReady ()
- onLocationChange ()
- onStatusChange ()
- onProviderEnabled ()
- onProviderDisabled ()

Una vez en la página citada y en vínculo con la cuenta de *Google*, aparecerá la sección “biblioteca” la cual muestra las API´s, SDK´s y librerías de *Google* que están disponibles para vinculación de proyectos. El SDK a utilizar es “Maps SDK for Android”, una vez seleccionada de primera instancia. Cabe mencionar que se requerirá el nombre del paquete del proyecto que se estará realizando, Ejemplo: `package="com.example.ejemplogooglemaps"`. Este nombre de paquete se encuentra en el Manifest de la aplicación.

Una vez localizado el nombre del paquete, se procede a copiarlo y pegarlo en las instrucciones o pasos que marca la página de la activación de la API, el paso siguiente es localizar la clave certificado SHA-1 que se encuentra en el archivo “google_maps_api.xml” en

la carpeta “res” y subcarpeta “values”, esta clave de certificado contiene 40 caracteres de dos dígitos separados por el carácter de dos puntos (:), ejemplo:

```
SHA-1 certificate fingerprint:
XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX:XX
```

En la Figura 7 se muestra la pantalla de los pasos vistos anteriormente.

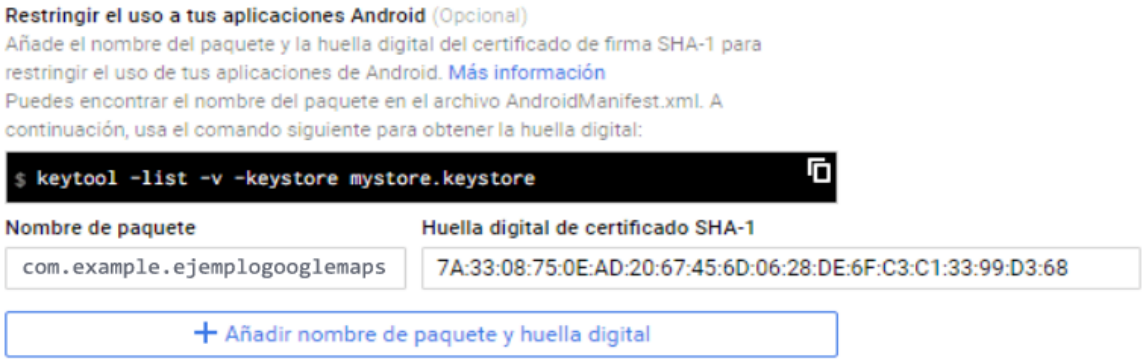

Figura 7. Pasos para obtener la clave de API.

Esta clave será copiada y pegada en los pasos que requiere la página citada anteriormente. Una vez realizados los pasos correctamente, la página retornará una clave de 39 caracteres distinta a la clave SHA-1, esta clave será copiada, y pegada en el archivo “google_maps_api.xml” en el apartado de <String> de resources. Ejemplo:

```
<string name="google_maps_key" templateMergeStrategy="preserve"
translatable="false">
    123456789123456789123456789912345678912
</string>
```

En la Figura 8 se muestra la clave de API generada por la plataforma de google.

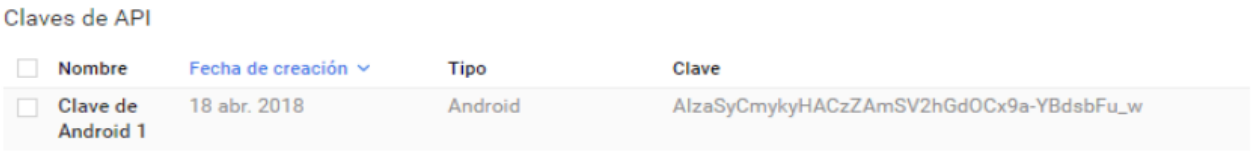

Figura 8. Clave de API generada.

Método onMapReady (GoogleMap googleMap)

Este método recibe un parámetro de la clase principal de la API de *Google Maps* y permite visualizar el mapa directamente con las coordenadas específicas como primera instancia además de generar un objeto del tipo *GoogleMap* ya sea para añadir marcadores, trazar rutas o permitir una animación de enfoque en la cámara cuando se definen nuevas coordenadas, dentro de este método todos los cambios y actualizaciones de ubicación deberán ser declarados para que se puedan mostrar en pantalla.

Método onLocationChange ()

Se usa para recibir notificaciones del *LocationManager* cuando la ubicación ha cambiado. Se invocan estos métodos si el *LocationListener* se ha registrado con el servicio de administrador de ubicación utilizando el *LocationManager.requestLocationUpdates(String, long, float, LocationListener).*

Clase LocationManager

Esta clase proporciona acceso a los servicios de localización del sistema. Estos servicios permiten a las aplicaciones obtener actualizaciones periódicas de la ubicación geográfica del dispositivo o disparar una aplicación especificada cuando el dispositivo ingresa cerca de una ubicación geográfica determinada.

Las instancias de esta clase se deben obtener usando *Context.getSystemService ()*. A continuación, se muestra la implementación de *LocationManager* para llamar al proveedor de servicio GPS, realizando una verificación de permisos para el acceso a datos de localización.

```
if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_FINE_LOCATION) !=
PackageManager.PERMISSION_GRANTED &&
ActivityCompat.checkSelfPermission(this,
Manifest.permission.ACCESS_COARSE_LOCATION) !=
PackageManager.PERMISSION_GRANTED) {
    return;
}
LocationManager locationManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
Location location =
locationManager.getLastKnownLocation(LocationManager.GPS_PROVID
ER);
locationManager.requestLocationUpdates(LocationManager.NETWORK_
PROVIDER,1000,0, locationListener );
```

Clase PlaceAutocompleteFragment

La clase pública *PlaceAutocompleteFragment* hereda de la clase principal *Fragment* que es un fragmento que proporciona autocompletado para lugares. Este *Fragment* es la barra de búsqueda de lugares o direcciones públicas disponibles. Esta clase llama a un método sobreescrito llamado *onPlaceSelected (Place place)* que contiene un parámetro de la clase Place que al igual hereda de la clase pública *Location*. Este método es capaz de obtener la ubicación de forma exacta a la hora de introducir alguna dirección. Esta dirección obtiene la latitud y longitud del lugar seleccionado que contiene el marcador. A continuación, se muestra la manera de realizar correctamente el llamado de la latitud y longitud de la dirección:

```
PlaceAutocompleteFragment edtAddress =
(PlaceAutocompleteFragment)
getFragmentManager().findFragmentById(R.id.place_autocomplete_f
ragment);
```

```
edtAddress.setOnPlaceSelectedListener(new
PlaceSelectionListener() {
    @Override
    public void onPlaceSelected(Place place)
    {
        domicilioPlace = place;
        domiclioFisico= domicilioPlace.getAddress().toString();
        latitud= domicilioPlace.getLatLng().latitude;
        longitud=domicilioPlace.getLatLng().longitude;
    }

    @Override
    public void onError(Status status) {
        Log.e("Error", status.getStatusMessage());
    }
});
```

Las variables utilizadas (latitud/longitud) son declaradas públicas a nivel clase de tipo String y serán utilizadas para almacenarlas en la base de datos.

XML para diseño de Activity Maps

Para este diseño se requieren conocimientos previos y la utilización de *Fragments* para los datos de *Google* que serán visualizados, es importante e indispensable el uso de estos *Fragments* de lo contrario no se podrá implementar la Activity. A continuación, se muestra la manera correcta de cómo implementarlos:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:map="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    xmlns:design="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

        <fragment
            android:layout_marginLeft="25dp"
            android:layout_marginRight="10dp"
            android:layout_marginTop="3dp"
            android:layout_gravity="center"
            android:id="@+id/place_autocomplete_fragment"
            android:name="com.google.android.gms.

           location.places.ui.PlaceAutocompleteFragment"
            android:layout_width="match_parent"
            android:layout_height="match_parent" />
```

```
        <fragment
            android:id="@+id/map"

        android:name="com.google.android.gms.maps.SupportMapFragme
        nt"
            android:layout_width="match_parent"
            android:layout_height="501dp"
            android:layout_alignParentStart="true"
            android:layout_below="@+id/recyclerView" />

        <android.support.design.widget.FloatingActionButton
            android:id="@+id/btFind"
            android:layout_width="50dp"
            android:layout_height="match_parent"
            android:layout_alignEnd="@+id/btManual"
            android:layout_alignParentBottom="true"
            android:layout_marginBottom="107dp"
            android:src="@drawable/search"

        map:backgroundTint="@android:color/background_light" />

        </RelativeLayout>
```

Librerías utilizadas

Cabe mencionar que para utilizar las librerías que se muestran a continuación, la versión mínima de compilación para el SDK deberá ser la versión 26 o posteriores y la versión mínima de SDK deberá ser a partir de la versión 23 o posteriores.

```
dependencies

{
  implementation                    'com.google.android.gms:play-services-places:15.0.1'
  implementation                    'com.android.support.constraint:constraint-layout:1.0.2'
  implementation                    'com.google.android.gms:play-services-maps:15.0.1'
  implementation          fileTree(include:          ['*.jar'],          dir:          'libs')
  androidTestImplementation('com.android.support.test.espresso:espresso-core:2.2.2',

      {
          exclude    group:    'com.android.support',    module:    'support-annotations'
      })
  implementation                    'com.github.karanchuri:PermissionManager:0.1.0'
  implementation                    'com.android.support:Appcompat-v7:26.0.0-alpha1'
  testImplementation                                                  'junit:junit:4.12'
  implementation                    'com.android.support:design:26.0.0-alpha1'
```

```
    implementation 'com.android.support:cardview-v7:26.0.0-alpha1'
    implementation 'com.android.support:recyclerview-v7:26.0.0-alpha1'
    implementation 'com.android.volley:volley:1.1.0-rc1'
    implementation 'com.github.bumptech.glide:glide:4.2.0'
    implementation 'de.hdodenhof:circleimageview:2.1.0'
    implementation 'com.rengwuxian.materialedittext:library:2.1.4'
    implementation 'com.facebook.android:facebook-android-sdk:[4,5]'
    annotationProcessor 'com.github.bumptech.glide:compiler:4.2.0'
}
```

5. METODOLOGÍA

5.1 Análisis

En esta etapa se determinaron todos los requerimientos funcionales y no funcionales de cada tipo de usuario.

5.1.1 Políticas

Se consideraron las siguientes reglas para el diseño de la paltaforma:

- ✓ El Total a cobrar dependerá de un cálculo de tipo de servicios, distancia y tiempo estimado. Por otra parte, la comisión que pagarán los negocios vinculados a esta aplicación se calculará de acuerdo a un porcentaje fijo que depende del total calculado.
- ✓ Los servicios atendidos los pagará directamente el usuario.
- ✓ Se tiene un margen de tiempo para cancelar el estado del servicio por el usuario, si no hay respuesta de parte del empleado del negocio vinculado, el servicio es automáticamente cancelado.
- ✓ Se requiere conexión a Internet para utilizar la aplicación móvil.
- ✓ Se requiere conexión GPS en los dispositivos para localizar la ubicación del servicio.

5.1.2 Lista de entidades externas

- ✓ Usuarios empleados
- ✓ Usuarios clientes
- ✓ Administrador general

5.1.3 Almacenes

- ✓ Servicios
- ✓ UsuariosClientes
- ✓ UsuariosEmpleados
- ✓ NegociosVinculados
- ✓ ServiciosSolicitados
- ✓ Estados
- ✓ Pagos
- ✓ Categorías
- ✓ Administradores

5.1.4 Procesos

- ✓ Registrar servicios solicitados
- ✓ Registrar pagos
- ✓ Comisiones

5.1.5 Diagrama de contexto

En la Figura 9 se muestra el diagrama de contexto del sistema.

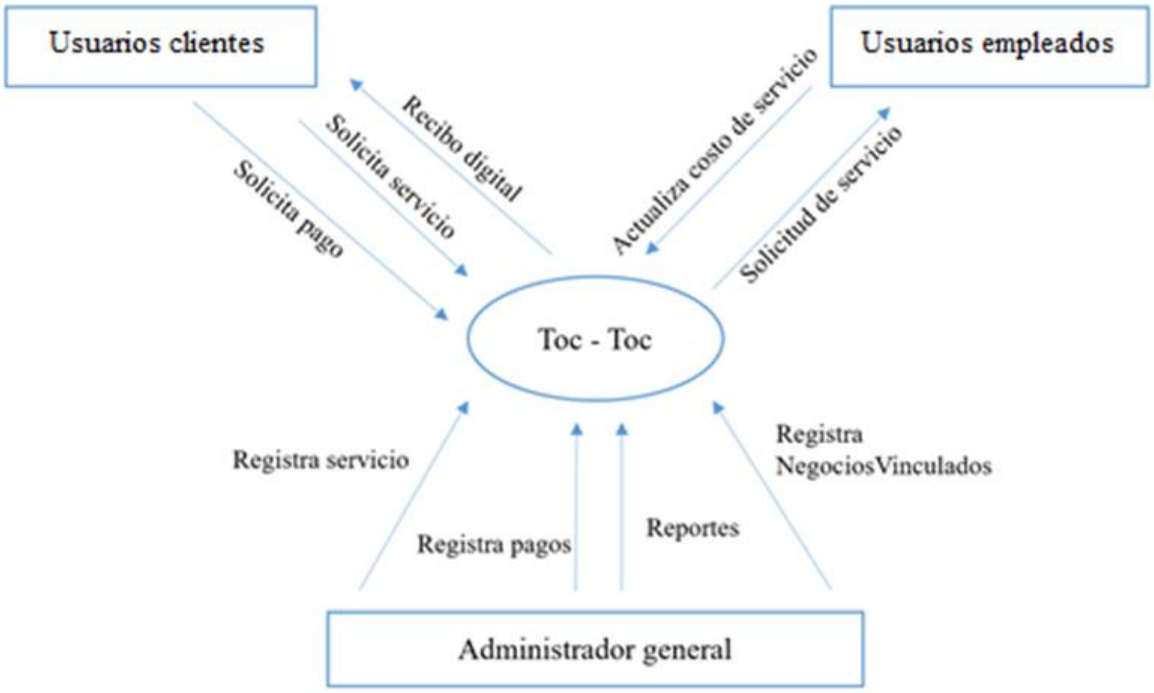

Figura 9. Diagrama de contexto.

5.2 Diseño

5.2.1 Diagrama de Warnier/Orr para administrador

En la Figura 10 se muestran los datos que contiene el diagrama de Warnier/Orr para el rol del administrador aplicación de escritorio.

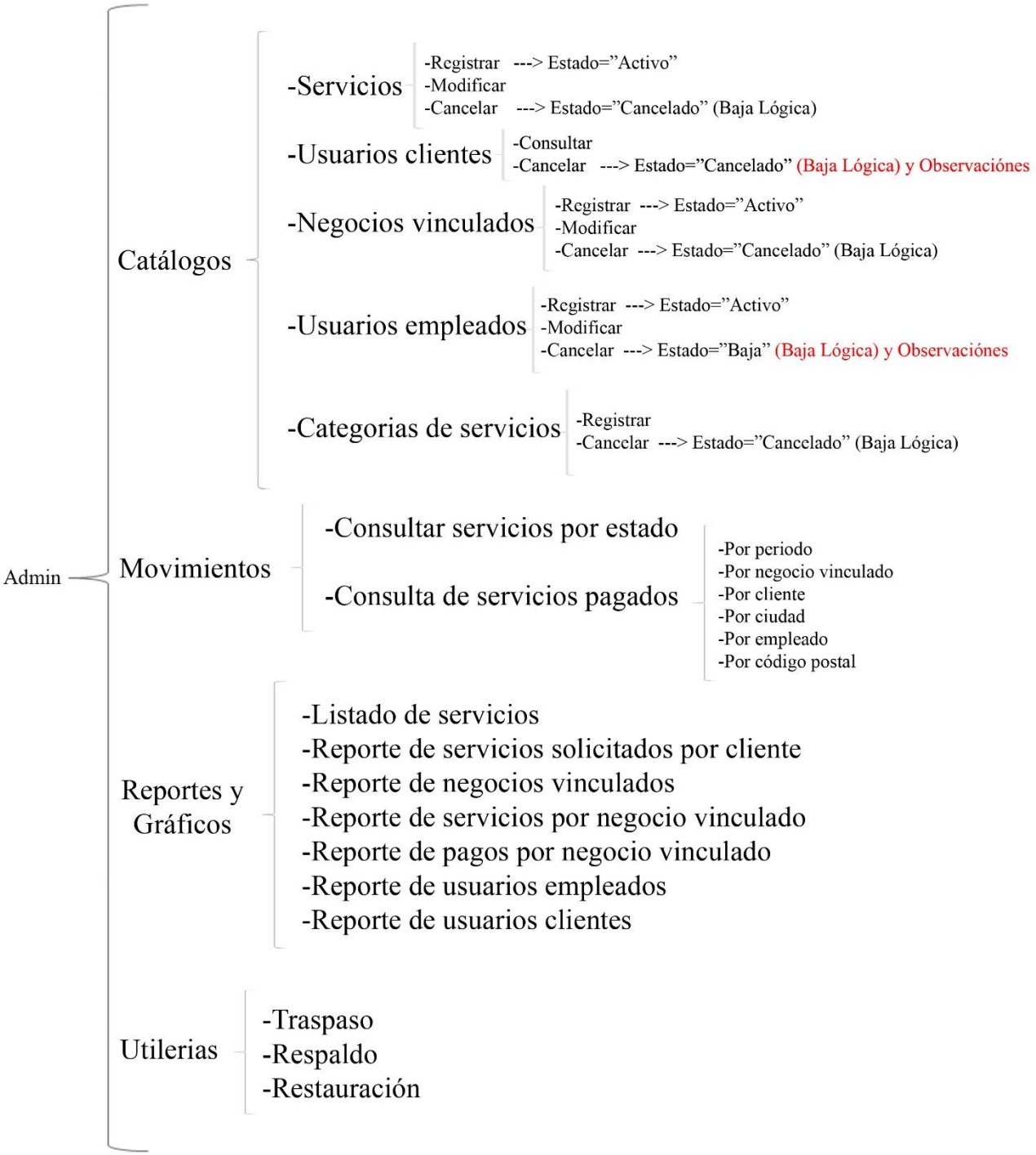

Figura 10. Diagrama de Warnier/Orr para rol de administrador.

5.2.2 Diagrama de Warnier/Orr para usuarios empleados

En la Figura 11 se muestran los datos que contiene el diagrama de Warnier/Orr para el rol de usuario empleados, aplicación Web y móvil.

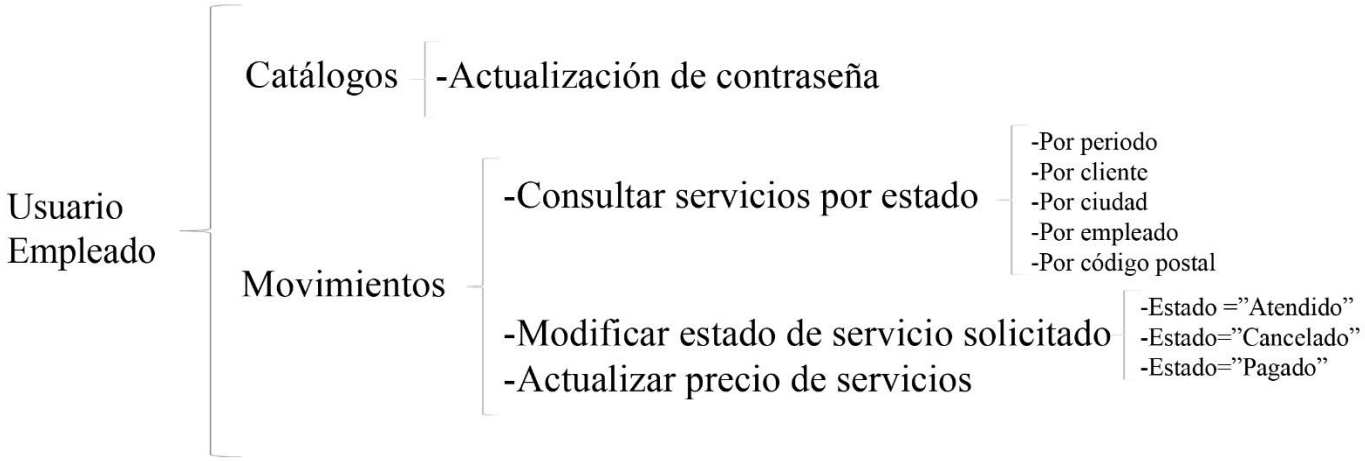

Figura 11. Diagrama de Warnier/Orr para el rol de UsuarioEmpleado.

5.2.3 Diagrama de Warnier/Orr para usuarios clientes

En la Figura 12 se muestran los datos que contiene el diagrama de Warnier/Orr para el rol de usuarios cliente, aplicación móvil.

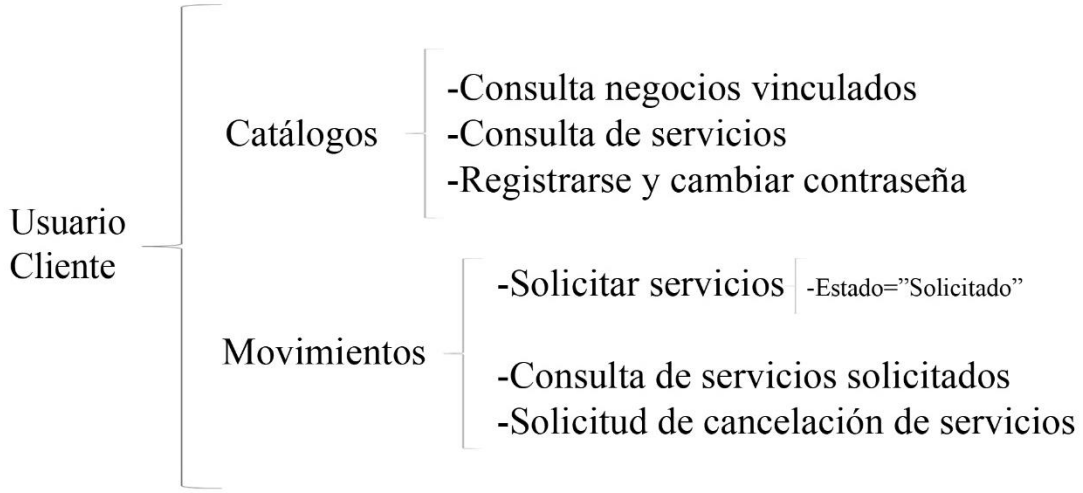

Figura 12. Diagrama de Warnier/Orr para el rol de UsuariosClientes.

5.2.4 Diagrama de caso de uso

En la Figura 13 se muestran los datos que contiene el diagrama de casos de uso general del sistema.

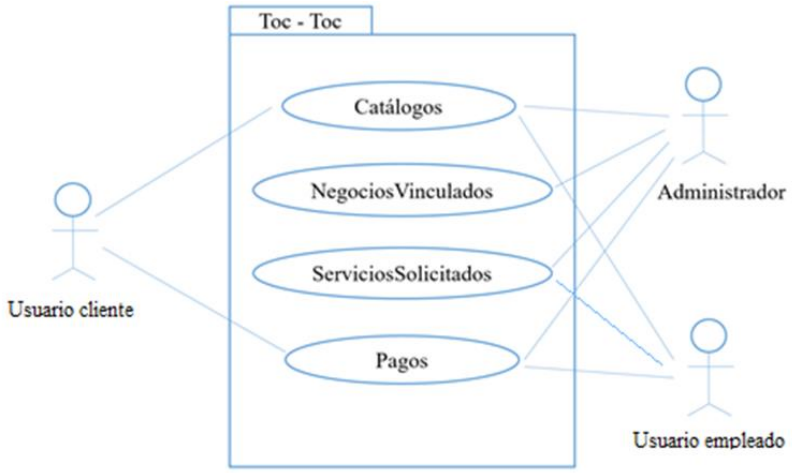

Figura 13. Diagrama de caso de uso general.

5.2.5 Diagrama de clases

En la Figura 14 se muestra el diagrama de clases del sistema.

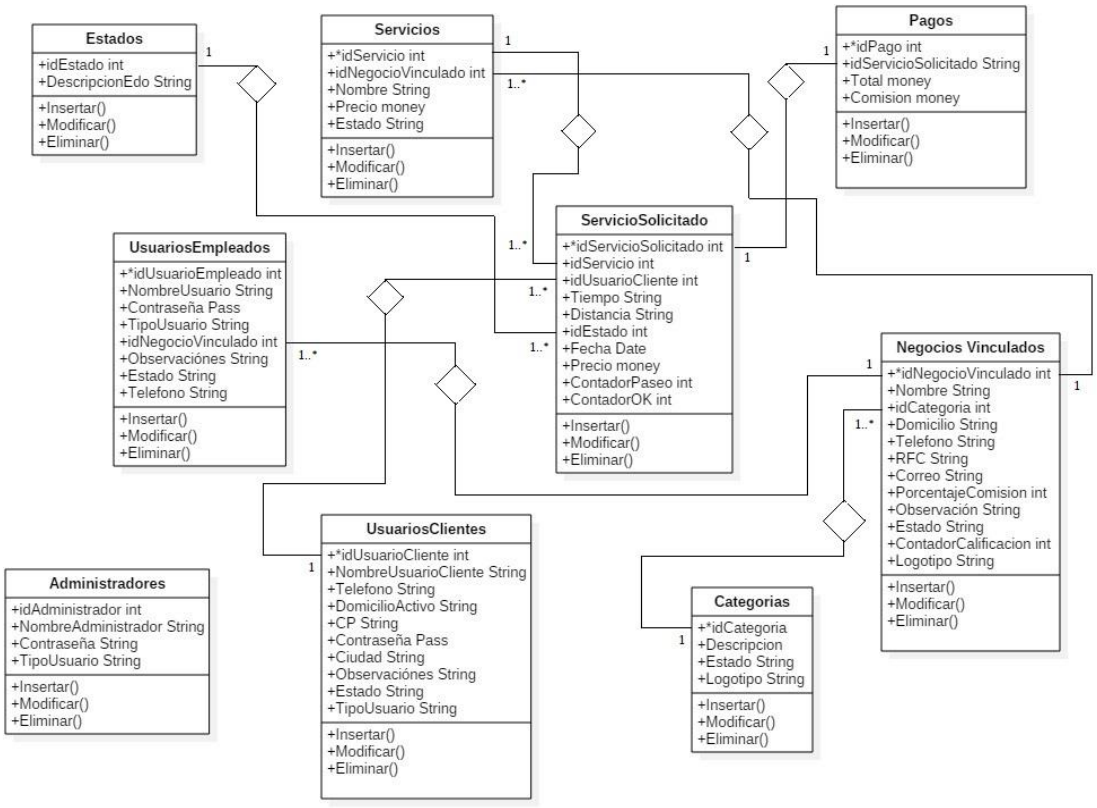

Figura 14. Diagrama de clases.

5.2.6 Diagrama de flujo de datos de servicios solicitados por el usuario cliente

En la Figura 15 se muestra el diagrama de flujo de datos de servicios solicitados por el usuario cliente.

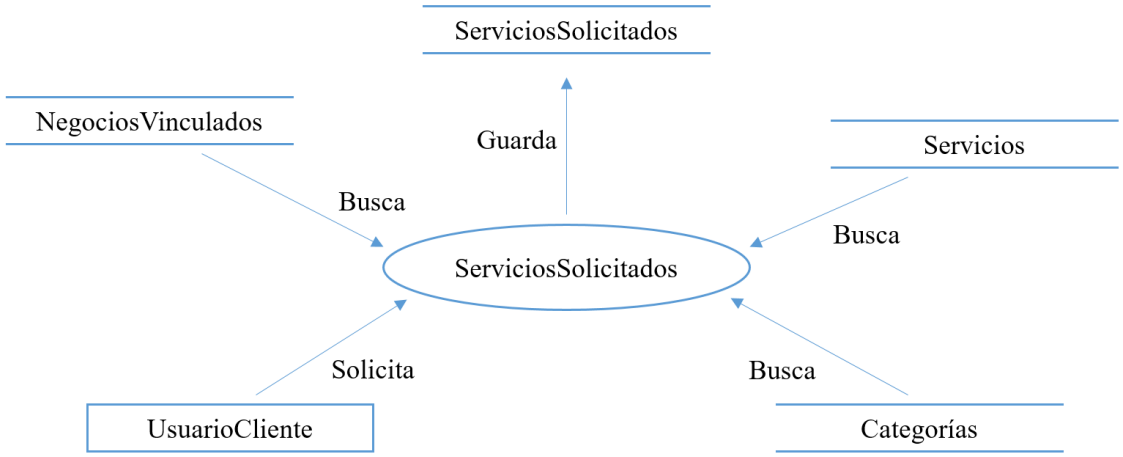

Figura 15. Diagrama de flujo de datos.

5.2.7 Diagrama de flujo de datos para registrar pagos a servicios solicitados

En la Figura 16 se muestran los datos que contiene el diagrama de flujo de datos para registrar pagos a servicios solicitados.

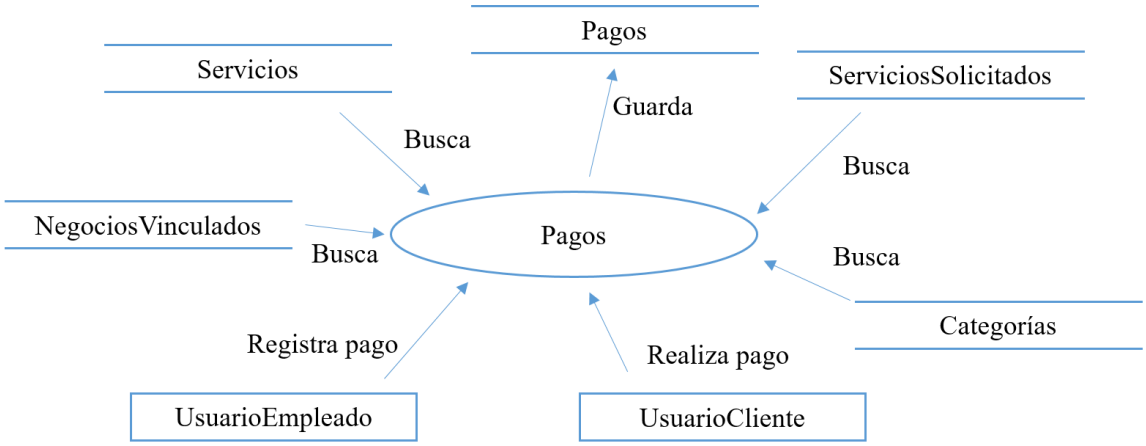

Figura 16. Diagrama de flujo de datos.

5.2.8 Diagrama de flujo de datos para consultar comisiones de servicios solicitados

En la Figura 17 se muestran los datos que contiene el diagrama de flujo de datos para consultar las comisiones de los servicios solicitados.

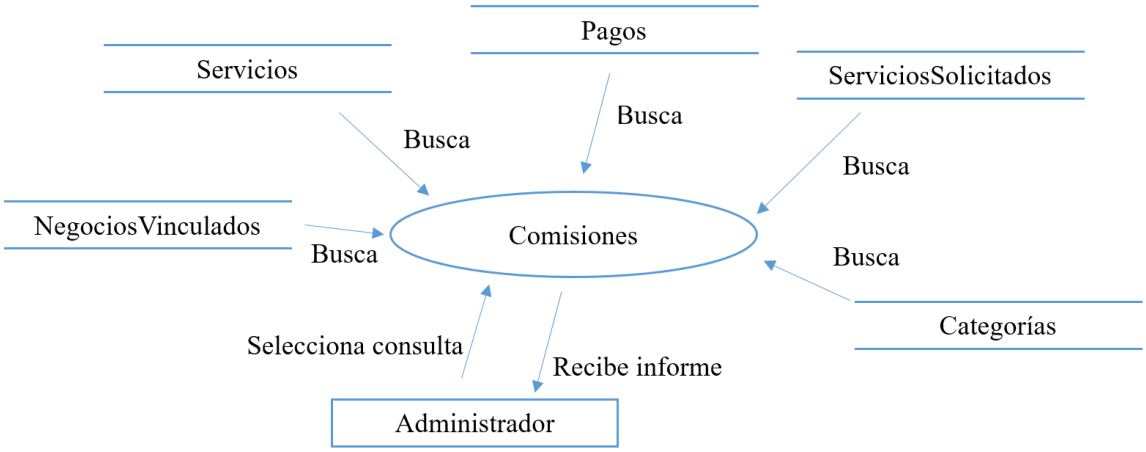

Figura 17. Diagrama de flujo de datos.

5.2.9 Diccionario de datos

Entidad: UsuariosClientes

Objetivo: Almacena los datos de los usuarios clientes que están registrados en el sistema. Ver Tabla 1.

No. de campos: 10

Campo llave: idUsuarioCliente

Tabla 1. UsuariosClientes.

No.	Nombre	Tipo	Longitud	Descripción	Dominio
1	*idUsuarioCliente	Entero		Almacena la clave del cliente.	Dígitos del 0 al 9.
2	NombreUsuarioCliente	Texto	50	Almacena el nombre o usuario del cliente.	Letras de ‘A-Z’.
3	Telefono	Texto	50	Almacena el teléfono del cliente.	Letras de ‘A-Z’.
4	DomicilioActivo	Texto	50	Almacena el domicilio actual con ubicación.	Letras de ‘A-Z’.
5	CP	Texto	20	Almacena el código postal del cliente.	Letras de ‘A-Z’.

6	Contrasena	Texto	50	Almacena la contraseña del cliente.	Letras de ‘A-Z’.
7	Ciudad	Texto	50	Almacena la ciudad de residencia del cliente.	Letras de ‘A-Z’.
8	Observaciones	Texto	150	Almacena las posibles observaciones del cliente.	Letras de ‘A-Z’.
9	Estado	Texto	20	Almacena el estado o status del cliente.	Letras de ‘A-Z’.
10	TipoUsuario	Texto	15	Almacena el tipo de usuario y/o rol dentro del sistema.	Letras de ‘A-Z’.

Entidad: UsuariosEmpleados

Objetivo: Almacena los datos de los usuarios empleados que están registrados en el sistema. Ver Tabla 2.

No. de campos: 8

Campo llave: idUsuarioEmpleado

Tabla 2. UsuariosEmpleados.

No.	Nombre	Tipo	Longitud	Descripción	Dominio
1	*idUsuarioEmpleado	Entero		Almacena la clave del empleado.	Dígitos del 0 al 9.
2	NombreUsuario	Texto	50	Almacena el nombre o usuario del empleado.	Letras de ‘A-Z’.
3	Contrasena	Texto	50	Almacena la contraseña del empleado.	Letras de ‘A-Z’.
4	TipoUsuario	Texto	50	Almacena el tipo de usuario y/o rol dentro del sistema.	Letras de ‘A-Z’.
5	idNegocioVinculado	Entero		Almacena la clave del negocio	Dígitos del 0 al 9.

				vinculado al que pertenece el empleado.	
6	Observaciones	Texto	50	Almacena las posibles observaciones del empleado.	Letras de ‘A-Z’.
7	Estado	Texto	50	Almacena el estado o status del empleado.	Letras de ‘A-Z’.
8	Telefono	Texto	150	Almacena el teléfono del empleado.	Letras de ‘A-Z’.

Entidad: Categorias

Objetivo: Almacena los datos de las categorías disponibles registradas en el sistema. Ver Tabla 3.

No. de campos: 4

Campo llave: idCategoria

Tabla 3. Categorías.

No.	Nombre	Tipo	Longitud	Descripción	Dominio
1	*idCategoria	Entero		Almacena la clave de la categoría.	Dígitos del 0 al 9.
2	Descripcion	Texto	50	Almacena el nombre o descripción de la categoría.	Letras de ‘A-Z’.
3	Estado	Texto	50	Almacena el estado o status de la categoría.	Letras de ‘A-Z’.
4	Logotipo	Texto	50	Almacena la cadena con ruta del logotipo alusivo a la categoría.	Letras de ‘A-Z’ y signos ortográficos.

Entidad: Estados

Objetivo: Almacena los estados en los que se puede encontrar un servicio solicitado. Ver Tabla 4.

No. de campos: 2

Campo llave: idEstado

Tabla 4. Estados.

No.	Nombre	Tipo	Longitud	Descripción	Dominio
1	*idEstado	Entero		Almacena la clave estado.	Dígitos del 0 al 9.
2	DescripcionEdo	Texto	50	Almacena el nombre o descripción del estado.	Letras de 'A-Z'.

Entidad: ServicioSolicitado

Objetivo: Almacena los datos de los servicios solicitados por los usuarios clientes. Ver Tabla 5.

No. de campos: 10

Campo llave: idServicioSolicitado

Tabla 5. ServicioSolicitado.

No.	Nombre	Tipo	Longitud	Descripción	Dominio
1	* idServicioSolicitado	Entero		Almacena la clave servicio solicitado.	Dígitos del 0 al 9.
2	idServicio	Entero		Almacena la clave foránea del servicio.	Dígitos del 0 al 9.
3	idUsuarioCliente	Entero		Almacena la clave foránea del cliente que solicito el servicio.	Dígitos del 0 al 9.
4	Tiempo	Texto	50	Almacena el tiempo estimado en realizar el servicio	Letras de 'A-Z'.
5	Distancia	Texto	20	Almacena la distancia aproximada para realizar el servicio.	Letras de 'A-Z'.

6	idEstado	Entero		Almacena la llave foránea del estado de servicio solicitado.	Dígitos del 0 al 9.
7	Fecha	Texto	50	Almacena la fecha con la hora en la que se solicitó el servicio.	Letras de ‘A-Z’.
8	Precio	Money		Almacena el precio estimado para el servicio.	Dígitos del 0 al 9.
9	ContadorPaseo	Entero		Almacena el contador para entrar a la aplicación.	Dígitos del 0 al 9.
10	ContadorOK	Entero		Almacena el contador para solicitar algún servicio.	Dígitos del 0 al 9.

Entidad: Servicios

Objetivo: Almacena los datos de los servicios que ofrece cada negocio vinculado que están registrados en el sistema. Ver Tabla 6.

No. de campos: 5

Campo llave: idServicio

Tabla 6. Servicios.

No.	Nombre	Tipo	Longitud	Descripción	Dominio
1	*idServicio	Entero		Almacena la clave servicio.	Dígitos del 0 al 9.
2	idNegocioVinculado	Entero		Almacena la llave foránea del negocio vinculado responsable del servicio.	Dígitos del 0 al 9.
3	Nombre	Texto	50	Almacena el nombre o descripción del servicio.	Letras de ‘A-Z’.

4	Precio	Money	50	Almacena el costo estimado para el servicio.	Dígitos del 0 al 9.
5	Estado	Entero		Almacena el estado o status en el que se encuentra el servicio.	Dígitos del 0 al 9.

Entidad: NegociosVinculados

Objetivo: Almacena los datos de los negocios vinculados registrados en el sistema.

No. de campos: 12

Campo llave: idNegocioVinculado

Tabla 7. NegociosVinculados.

No.	Nombre	Tipo	Longitud	Descripción	Dominio
1	* idNegocioVinculado	Entero		Almacena la clave del negocio.	Dígitos del 0 al 9.
2	Nombre	Texto	50	Almacena el nombre del negocio.	Letras de ‘A-Z’.
3	idCategoria	Entero		Almacena la clave foránea de la categoría a la que corresponde el negocio.	Dígitos del 0 al 9.
4	Domicilio	Texto	50	Almacena domicilio del negocio.	Letras de ‘A-Z’.
5	Telefono	Texto	20	Almacena el teléfono del negocio.	Letras de ‘A-Z’.
6	RFC	Entero		Almacena el RFC del negocio.	Dígitos del 0 al 9.
7	Correo	Texto	50	Almacena el correo del negocio.	Letras de ‘A-Z’.
8	Porcentaje	Entero		Almacena el porcentaje de	Dígitos del 0 al 9.

				comisión del negocio.	
9	Observacion	Entero		Almacena las observaciones para el negocio.	Dígitos del 0 al 9.
10	Estado	Entero		Almacena estado o status del negocio.	Dígitos del 0 al 9.
11	ContadorCalificacion	Entero		Almacena la calificación para el negocio	Dígitos del 0 al 9.
12	Logotipo	Texto	50	Almacena en una cadena la ruta del logotipo para el negocio.	Letras de 'A-Z'.

Entidad: Pagos

Objetivo: Almacena los datos de los servicios solicitados pagados que están registrados en el sistema.

No. de campos: 4

Campo llave: idPago

Tabla 8. Pagos.

No.	Nombre	Tipo	Longitud	Descripción	Dominio
1	* idPago	Entero		Almacena la clave del pago.	Dígitos del 0 al 9.
2	idServicioSolicitado	Entero		Almacena la llave foránea del servicio solicitado correspondiente a dicho pago.	Dígitos del 0 al 9.
3	Total	Money		Almacena el total en dinero del servicio solicitado.	Dígitos del 0 al 9.
4	Comision	Money		Almacena la comisión que se calculó en base al porcentaje de cada negocio.	Dígitos del 0 al 9.

5.3 Codificación

5.3.1 Diagrama de módulos de la aplicación de escritorio

En las Figuras 18 y 19 se muestran los formularios y reportes del sistema que integran la aplicación de administrador de la plataforma.

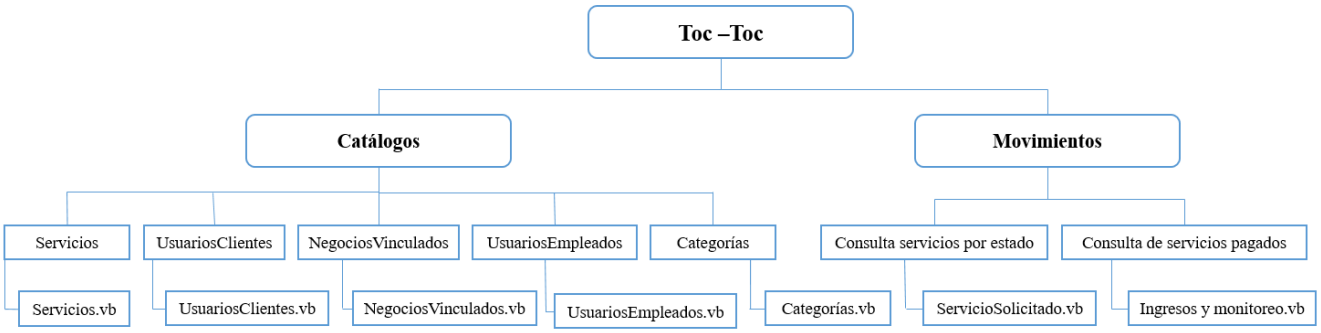

Figura 18. Diagrama de módulos y formularios.

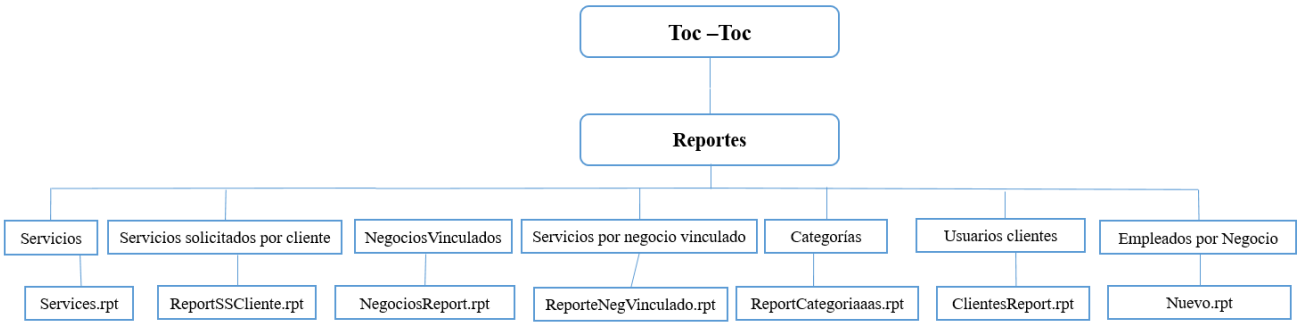

Figura 19. Diagrama de módulos y formularios.

5.3.2 Diagrama de módulos de la aplicación Web

En la Figura 20 se muestran los formularios del sistema que integran la aplicación Web de la plataforma.

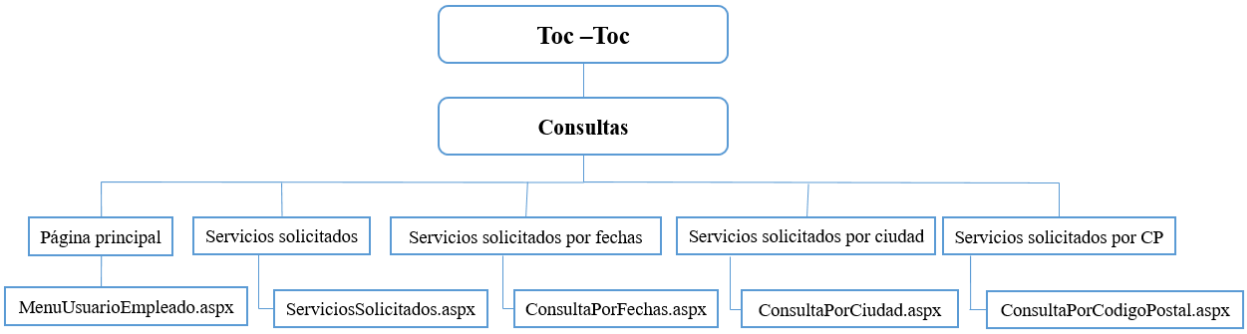

Figura 20. Diagrama de módulos y formularios en la parte Web.

5.3.3 Diagrama de módulos de la aplicación móvil

En la Figura 21 se muestran las clases y diseños XML del sistema que integran la aplicación móvil en la parte del usuario cliente de la plataforma.

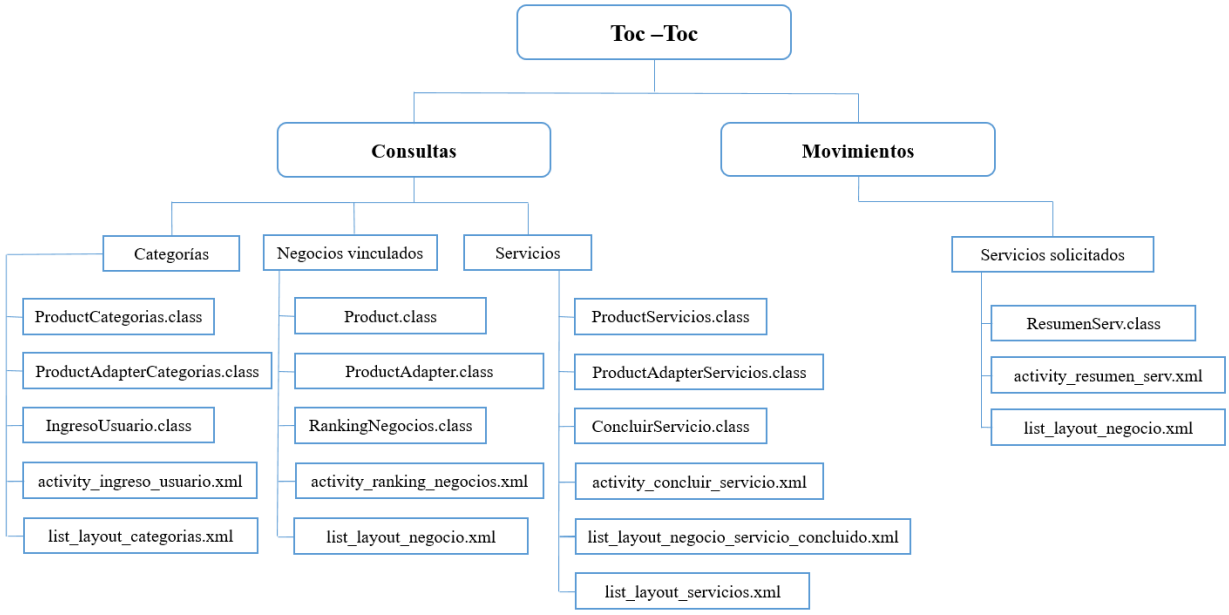

Figura 21. Diagrama de módulos y formularios en la parte móvil de usuario cliente.

5.4 Pruebas

Se realizaron pruebas unitarias a cada proceso, tanto de la aplicación de escritorio, como de la Web y móvil.

En la Tabla 9 se muestran las pruebas realizadas para el movimiento de servicio solicitado por medio de la aplicación móvil de parte del usuario cliente capturando distintos datos desde un dispositivo móvil.

Se definirán parámetros para distancia, tiempo y precio final:

Punto A (Cliente): 1ro de mayo #372.

Punto B (Empleado): Pizza y Come Ciudad Guzmán.

Distancia: 280 metros Aprox.

Tiempo: 17 minutos (Sin contar preparación de pizza, solo entrega).

Tabla 9. Servicio solicitado por la aplicación móvil.

Nombre	**Servicio solicitado**
Descripción	Solicitar un servicio de la categoría de comida desde la aplicación móvil.
Datos de entrada	idServicioSolicitado: 1 idServicio: 8 idUsuarioCliente: 7 Tiempo: 17 minutos Distancia: 280 metros idEstado: 2 Fecha: Tomada del dispositivo Precio: 100 (Sin modificar) ContadorPaseo: 1 ContadorOK: 1
Condiciones	La distancia y el tiempo que se calculará depende de la ubicación del cliente y el negocio vinculado responsable en realizar el servicio. Además el precio se calculará el conforme a los datos calculados en distancia-tiempo.
Secuencia principal	1. El usuario cliente inicia sesión. 2. Selecciona la categoría (Comida). 3. Selecciona el negocio de su preferencia acorde a la categoría. 4. La aplicación móvil obtiene la ubicación en tiempo real y el usuario la confirma, en caso de no ser correcta,

	ingresa desde la barra de ubicaciones de Google. 5. Selecciona el servicio de preferencia. 6. Se manda una notificación de solicitud al usuario empleado responsable del negocio seleccionado. 7. El usuario empleado confirma solicitud y se calcula la distancia-tiempo con servicio Web de Google Maps. 8. El usuario empleado modificará el precio final del servicio. 9. Se manda resumen de pedido y/o solicitud al usuario cliente. (estado = procesando). 10. El empleado realiza el servicio y el cliente paga el monto total. *Nota: El estado del servicio solicitado cambiará a cancelado después de 10 minutos esto en caso de no tener respuesta por parte del usuario empleado.*
Alternativas y errores	1. El usuario cliente no ha iniciado sesión. 2. El usuario cliente no proporciona la ubicación correcta. 3. El usuario empleado no responde la notificación a la solicitud de servicio.
Salida	Servicio: Pizza sin límite de ingredientes Tiempo: 17 minutos Distancia: 280 metros Estado: Procesando Precio: 120 (Precio modificado por el empleado)

En la siguiente tabla se muestran las pruebas realizadas para registrar un nuevo negocio vinculado de la categoría de comida por medio de la aplicación de escritorio en la parte de administrador. Ver tabla 10.

Tabla 10. Registro de un nuevo negocio vinculado.

Nombre	**Negocio vinculado**
Descripción	Registrar un nuevo negocio vinculado para la categoría de comida.
Datos de entrada	idNegocioVinculado: 18 idCategoria: 5 Nombre: Mariscos el Tío Domicilio: Calzada del ejercito # 43 Teléfono: 3424232578 RFC: METSAY9046538B45 Correo: mariscos_eltio@hotmail.com Porcentaje: 12 Observación: Ninguna Estado: Activo Logotipo: "Ruta de archivo" Ciudad: Sayula
Condiciones	1. El logotipo tendrá como contenido la ruta del archivo en el servidor.
Secuencia principal	1. El administrador inicia sesión en la aplicación de escritorio. 2. Selecciona "Negocios vinculados" en el menú de opciones. 3. Presiona "Nuevo" e ingresa los datos de entrada. 4. Selecciona el icono para anexar el logotipo en formato PNG. 5. Presiona "Grabar", y aceptará el mensaje de información emergente.
Alternativas y errores	1. El administrador no ha iniciado sesión. 2. No se dispone del logotipo.
Salida	Negocio vinculado registrado correctamente.

En la Tabla 11 se muestran las pruebas realizadas para modificar el estado y precio de un servicio solicitado por medio de la aplicación de Web en la parte de usuario empleado.

Tabla 11. Modificación de estado y precio de un servicio solicitado.

Nombre	**Servicio Solicitado**
Descripción	Modificar estado y precio de un servicio solicitado.
Datos de entrada	1. Fecha inicial: 2014/04/01. 2. Fecha final: 2018/05/23. 3. Clave de servicio para actualizar: 1 4. Precio a modificar: 120
Condiciones	1. El estado del servicio solicitado debe ser “Atendido” o “Procesando”.
Secuencia principal	1. El usuario empleado inicia sesión en la aplicación Web. 2. Selecciona “Servicios solicitados” en el menú de opciones. 3. Ingresa la fecha inicial y la fecha final como parámetros de consulta. 4. Ingresa la clave a modificar. 5. Presiona “OK”. 6. Modificará el estado a “Pagado”. 7. Modificará el precio basándose en los valores distancia-tiempo de la aplicación móvil, en caso de que lo requiera. 8. Modificará el precio a 120. 9. Aceptará el mensaje emergente.
Alternativas y errores	1. El usuario empleado no ha iniciado sesión. 2. El estado del servicio es pagado o cancelado.
Salida	Estado = Pagado Precio = 120

5.5 Implantación

Se instaló la plataforma en un servidor de Internet real realizando todos los tipos de pruebas necesarios en cada uno de los escenarios (escritorio, Web y móvil).

6. RESULTADOS

6.1 Aplicación de escritorio (rol de administrador general)

6.1.1 Interfaz de Login

En la Figura 22 se visualiza la interfaz de inicio de la aplicación, el cual comienza con un Login solo con privilegios de administrador.

Figura 22. Interfaz de login.

6.1.2 Interfaz principal

En la Figura 23 se visualiza la interfaz principal de la aplicación.

Figura 23. Interfaz principal.

6.1.3 Interfaz con la opción de catálogos

En la Figura 24 se visualiza la interfaz con la opción de catálogos.

Figura 24. Interfaz de la pestaña catálogos.

6.1.4 Interfaz de la opción clientes

En la Figura 25 se visualiza la interfaz de clientes que se encuentra en la opción de catálogos, la cual muestra todos los clientes que fueron registrados en la aplicación móvil.

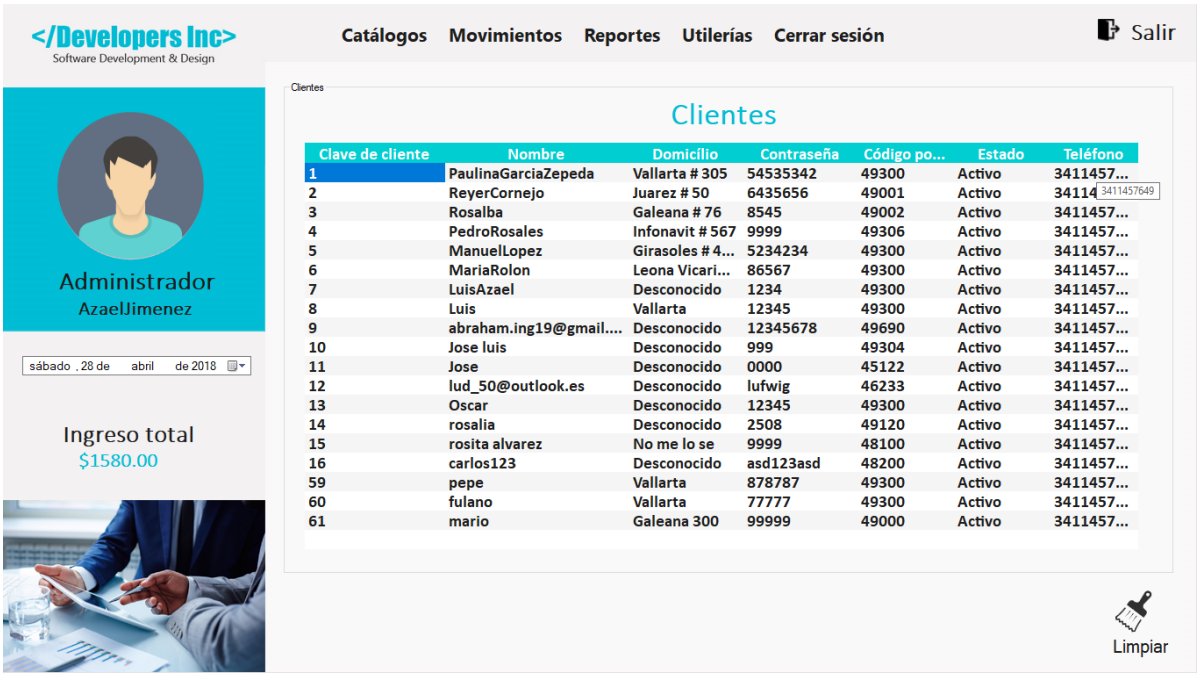

Figura 25. Interfaz de clientes.

6.1.5 Interfaz de la opción categorías

En la Figura 26 se visualiza la interfaz de categorías que se encuentra en la opción de catálogos.

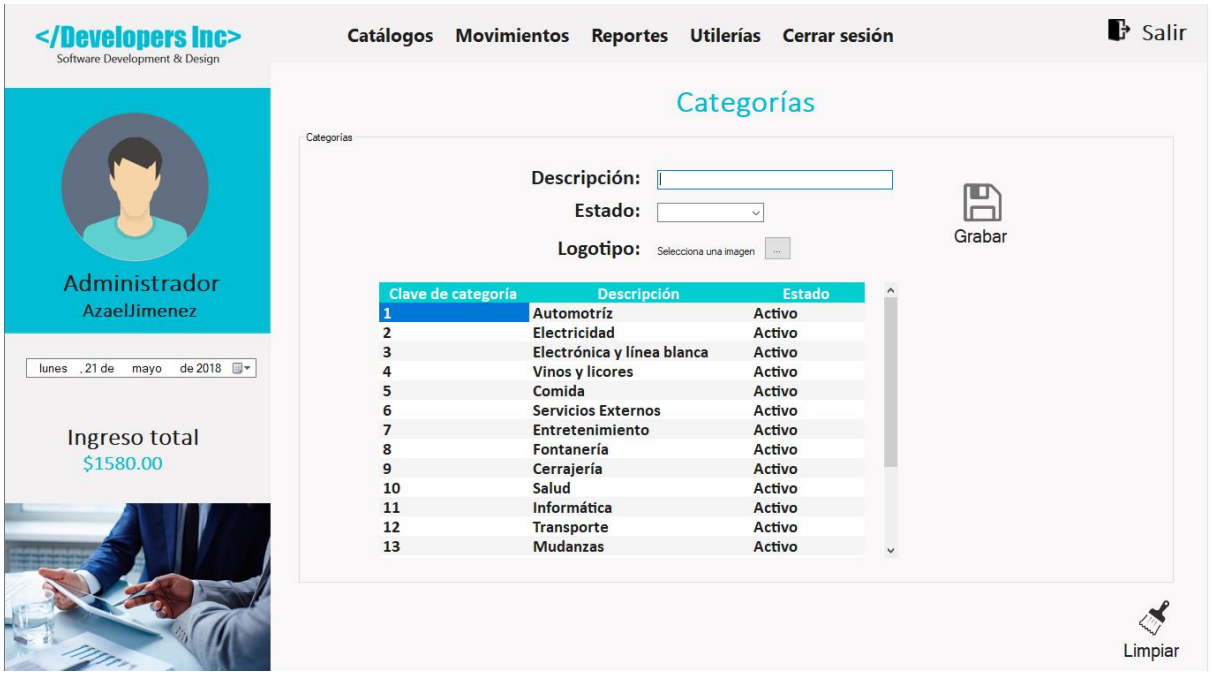

Figura 26. Interfaz de categorías.

6.1.6 Interfaz de la opción empleados

En la Figura 27 se visualiza la interfaz de empleados que se encuentra en la opción de catálogos, la cual muestra los distintos empleados registrados de los negocios vinculados, con todos sus datos.

Figura 27. Interfaz de empleados.

6.1.7 Interfaz de la opción servicios

En la Figura 28 se visualiza la interfaz para registrar servicios de un negocio vinculado.

Figura 28. Interfaz de servicio.

6.1.8 Interfaz de la opción negocios vinculados

En la Figura 29 se visualiza la interfaz de negocios vinculados.

Figura 29. Interfaz de negocios vinculados.

6.1.9 Interfaz de la opción movimientos

En la Figura 30 se visualiza la interfaz de las opciones de movimientos.

Figura 30. Interfaz de la pestaña movimientos.

6.1.10 Interfaz de la opción servicios solicitados

En la Figura 31 se visualiza la interfaz de servicios solicitados, la cual muestra una consulta con datos filtrados por fechas y por estados de servicio solicitado.

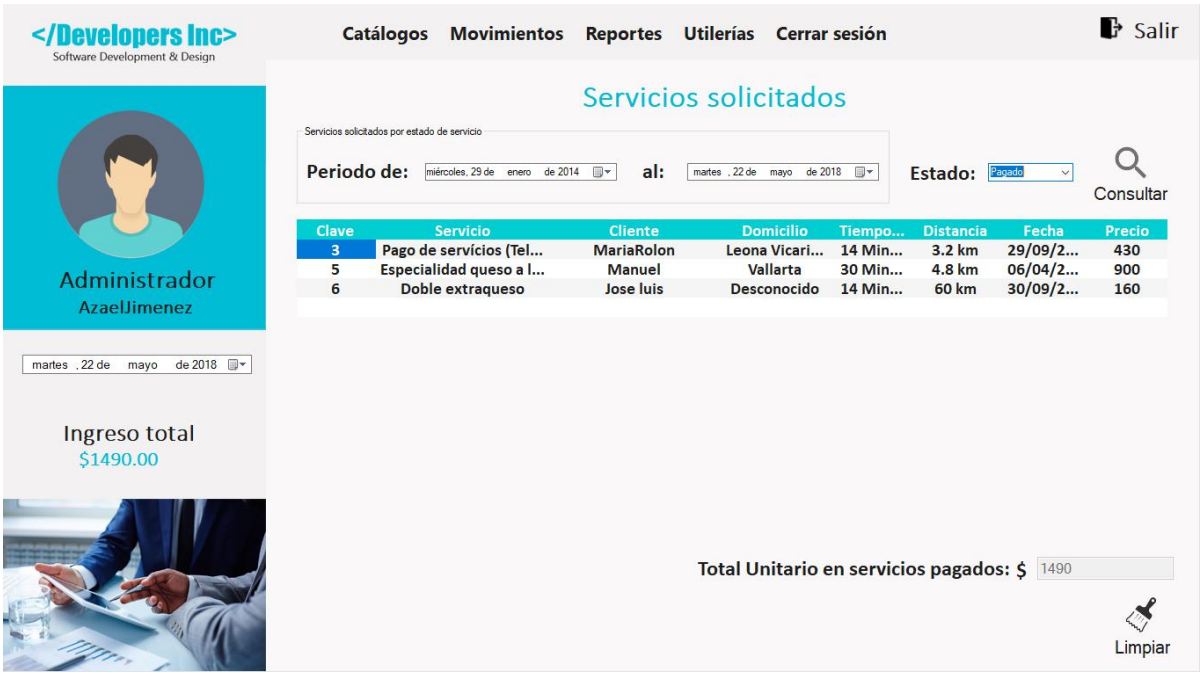

Figura 31. Interfaz de servicios solicitados.

6.1.11 Interfaz de la opción ingresos y monitoreo por fechas

En la Figura 32 se visualiza la interfaz de monitoreo de ingresos filtrados por fechas.

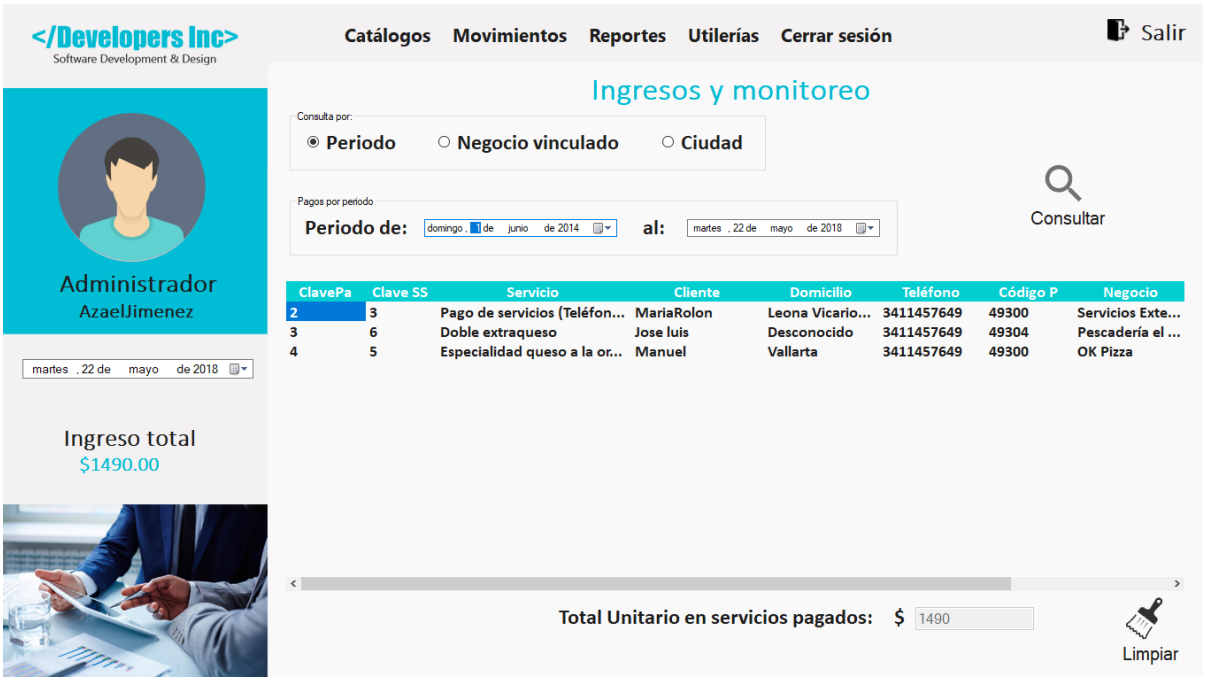

Figura 32. Interfaz de Ingresos y monitoreo por fechas.

6.1.12 Interfaz de la opción ingresos y monitoreo por negocios vinculados

En la Figura 33 se visualiza la interfaz de monitoreo de ingresos con datos filtrados por negocio vinculado.

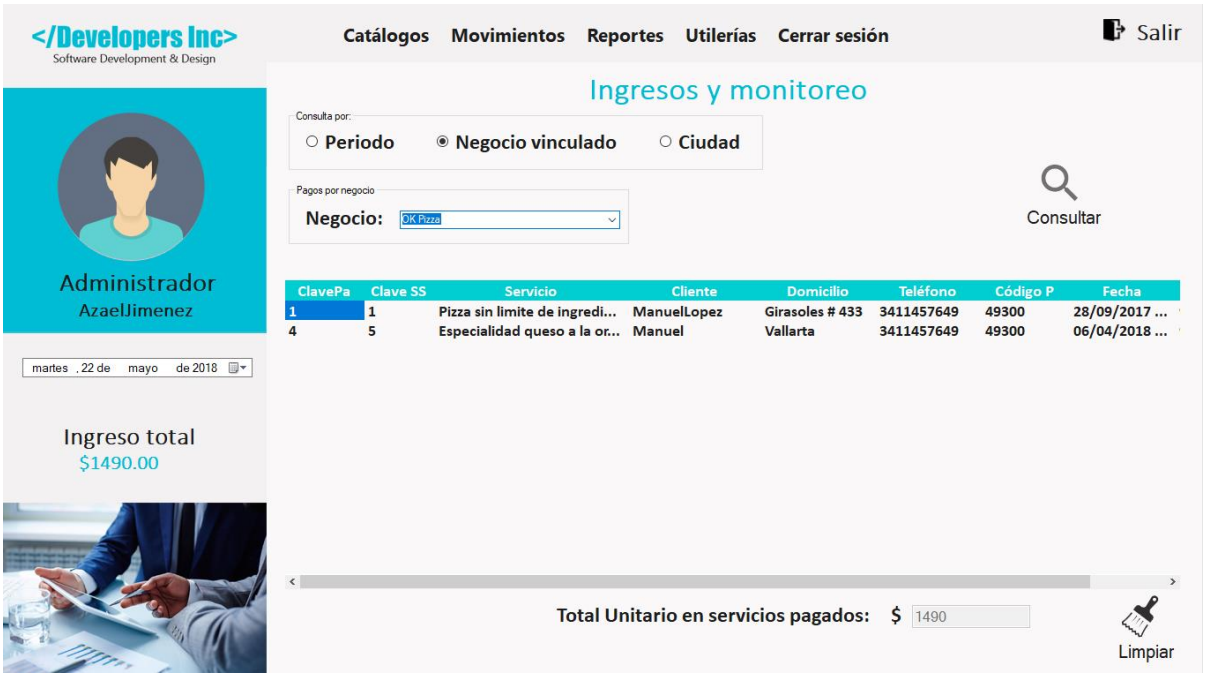

Figura 33. Interfaz de Ingresos y monitoreo por negocio.

6.1.13 Interfaz de la opción ingresos y monitoreo por ciudad

En la Figura 34 se visualiza la interfaz de ingresos y monitoreo con datos filtrados por ciudad.

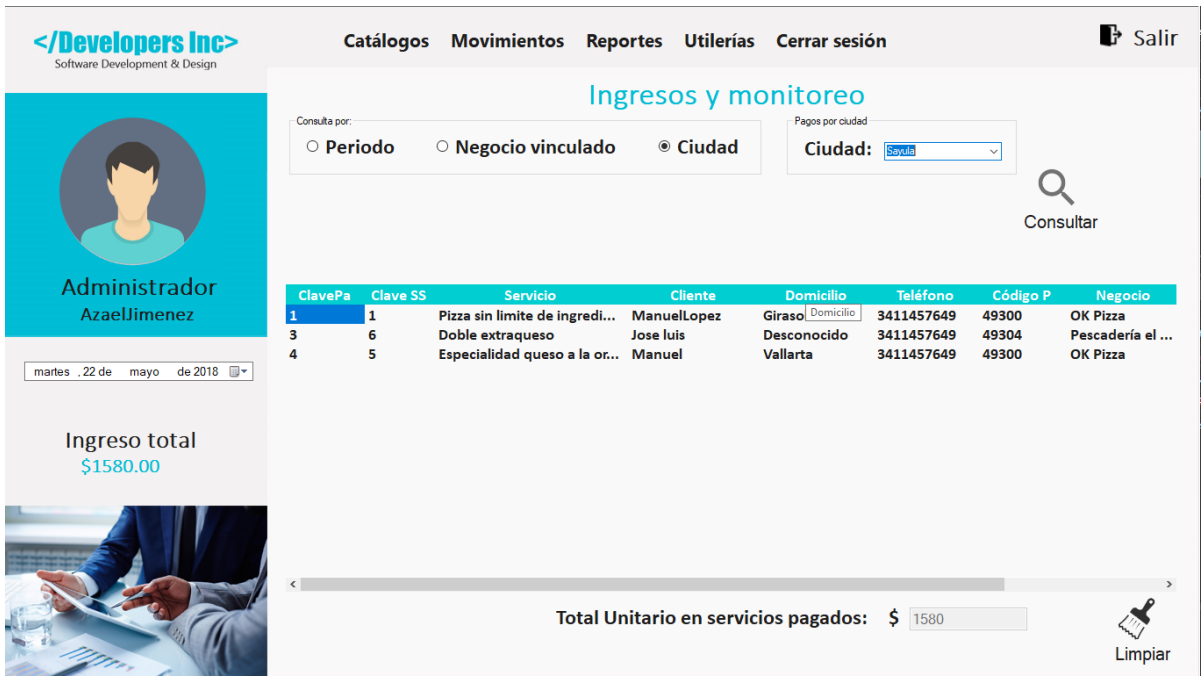

Figura 34. Interfaz de Ingresos y monitoreo por ciudad.

6.1.14 Interfaz de reporte de categorías de servicio

En la Figura 35 se visualiza la interfaz de reporte de categorías de servicio.

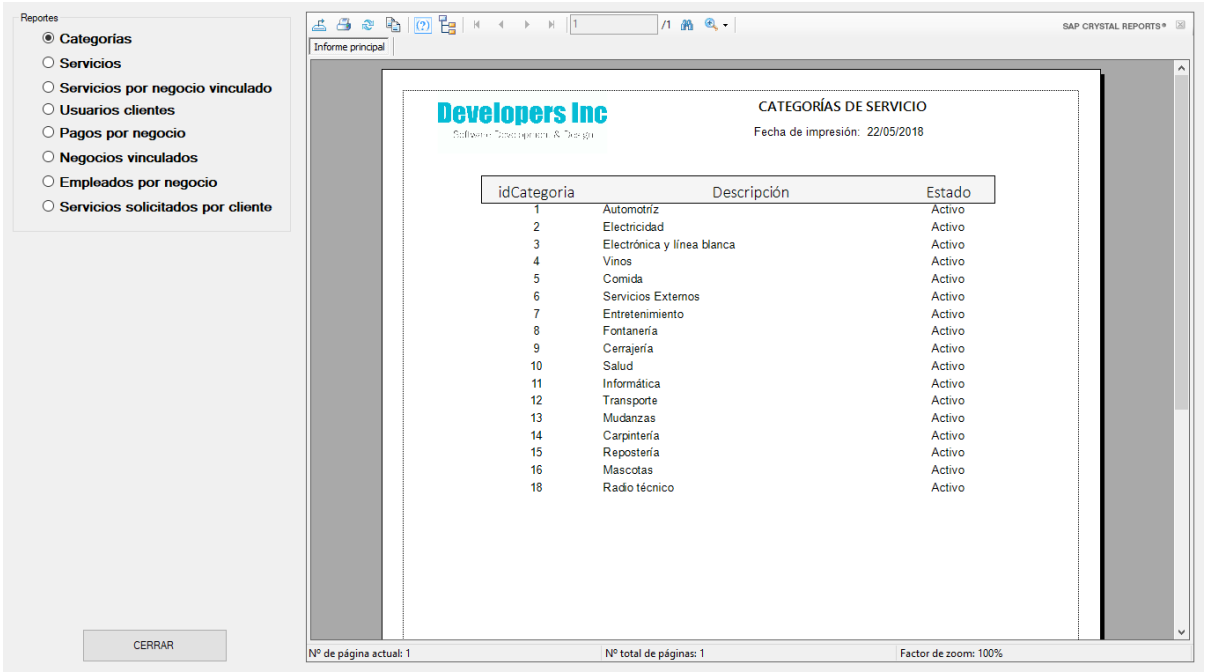

Figura 35. Interfaz de reporte de categorías de servicio.

6.1.15 Interfaz de reporte de servicios

En la Figura 36 se visualiza la interfaz de reporte de precios de servicios.

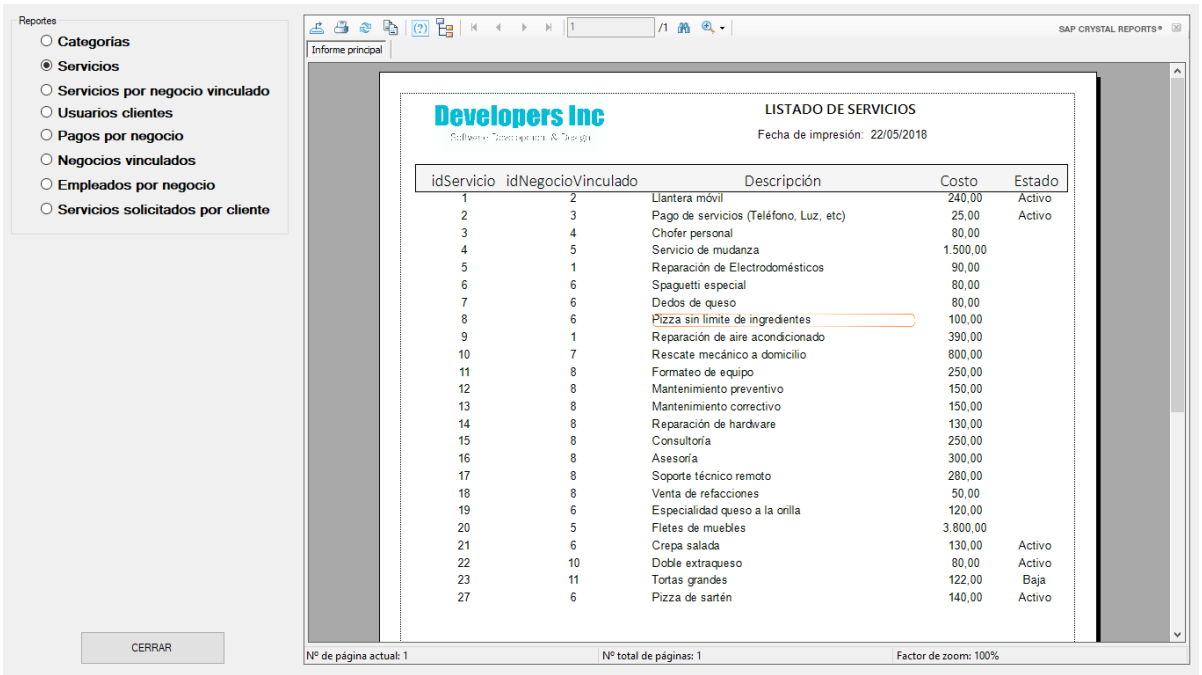

Figura 36. Interfaz de reporte de servicios.

6.1.16 Interfaz de reporte de servicios por negocio vinculado

En la Figura 37 se visualiza la interfaz de reporte de servicios por negocio vinculado el cual recibe un parámetro de entrada directamente de una ComboBox con tipo de dato String.

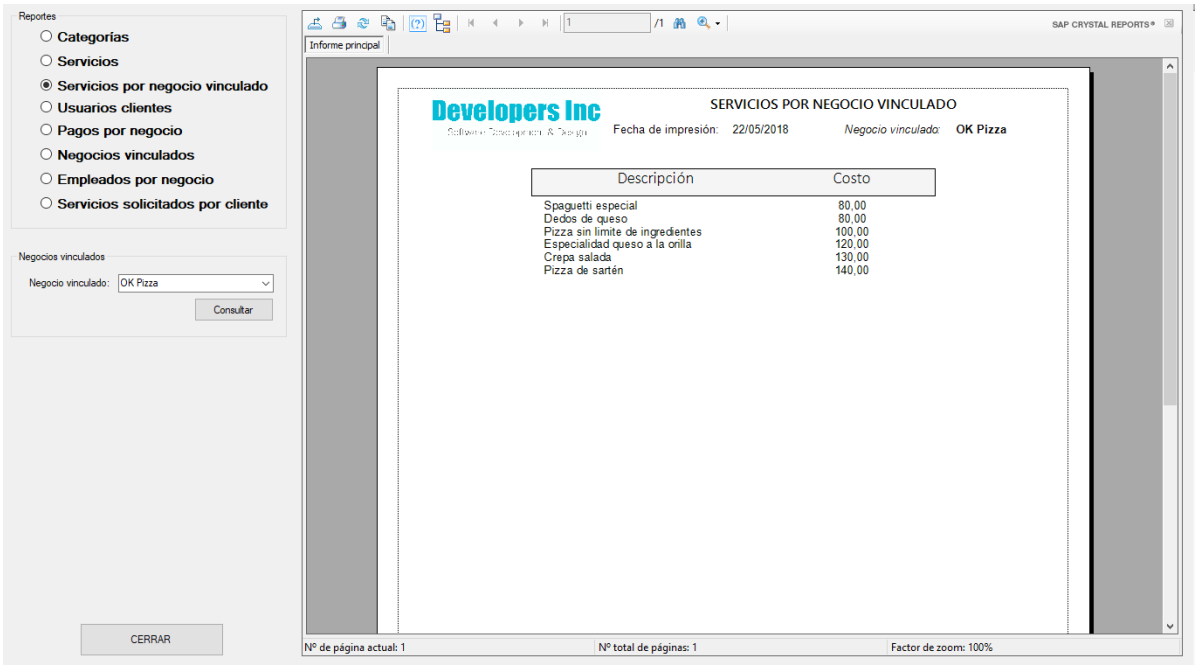

Figura 37. Interfaz de reporte de servicios por negocio vinculado.

6.1.17 Interfaz de reporte de usuarios clientes

En la Figura 38 se visualiza la interfaz de reporte de usuarios clientes.

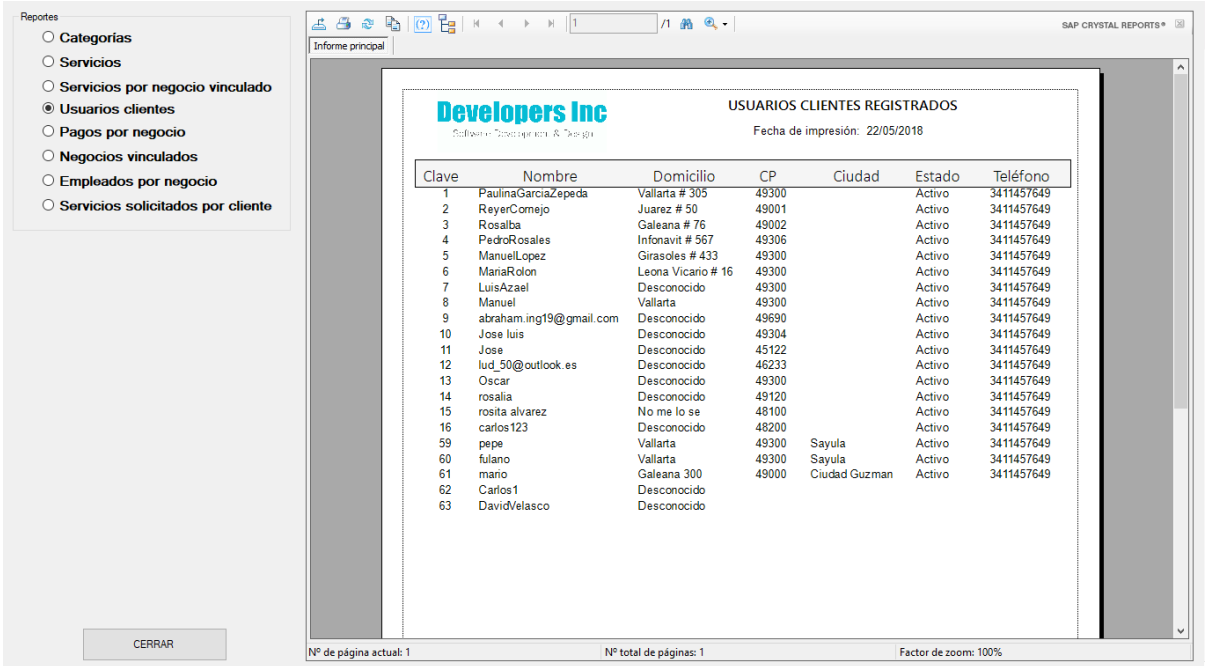

Figura 38. Interfaz de reporte de usuarios clientes.

6.1.18 Interfaz de reporte de negocios vinculados registrados

En la Figura 39 se visualiza la interfaz de reporte de negocios vinculados.

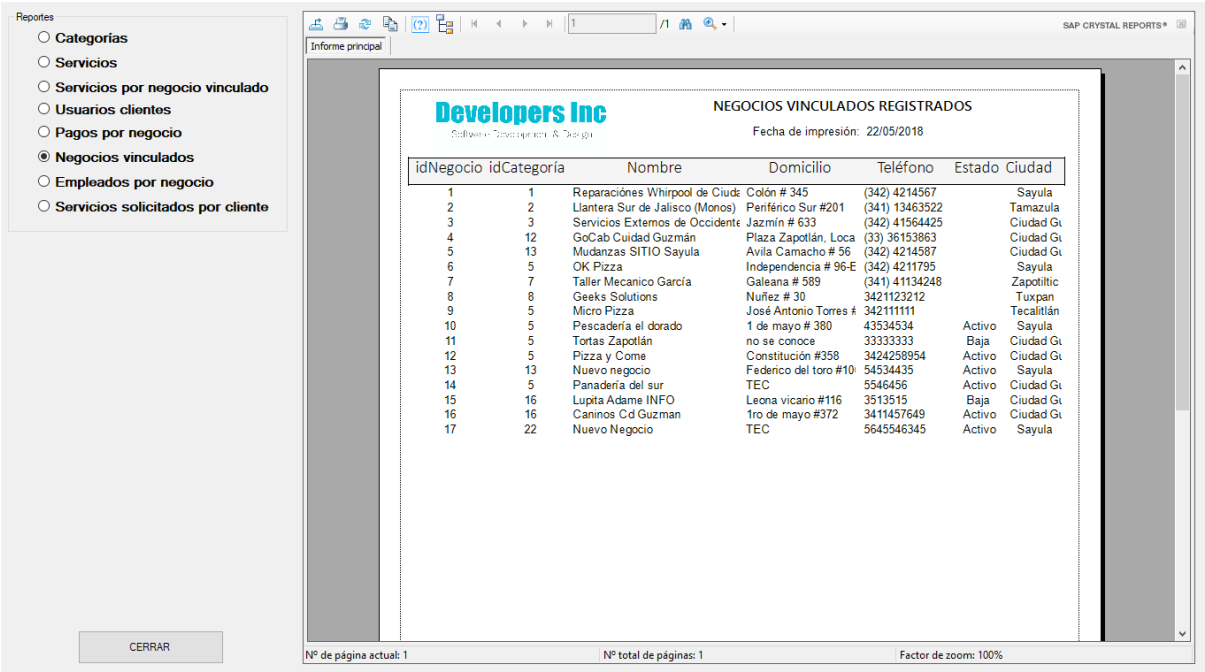

Figura 39. Interfaz de reporte de negocios vinculados.

6.1.19 Interfaz de reporte de empleados por negocio vinculado

En la Figura 40 se visualiza la interfaz de reporte de empleados por negocio vinculado el cual recibe un parámetro de entrada directamente de una ComboBox con tipo de dato String.

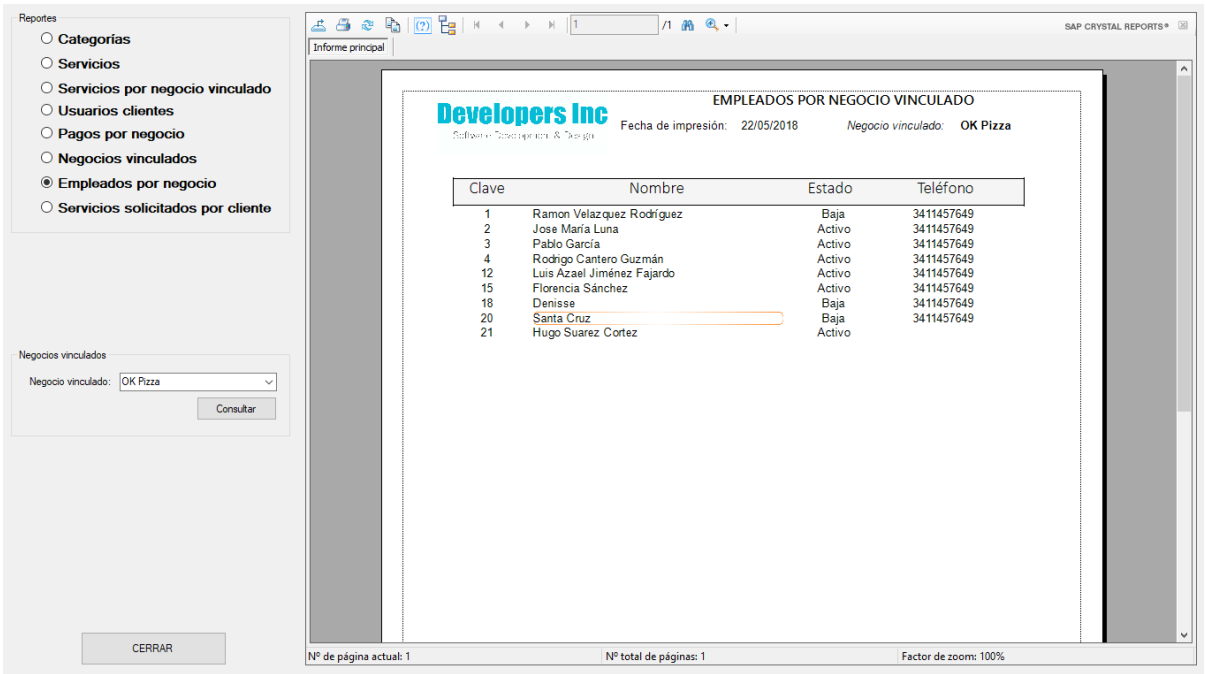

Figura 40. Interfaz de reporte de empleados por negocio vinculado.

6.1.20 Interfaz de reporte de servicios solicitados por cliente

En la Figura 41 se visualiza la interfaz de reporte de servicios solicitados por cliente el cual recibe un parámetro de entrada directamente de una ComboBox con tipo de dato String.

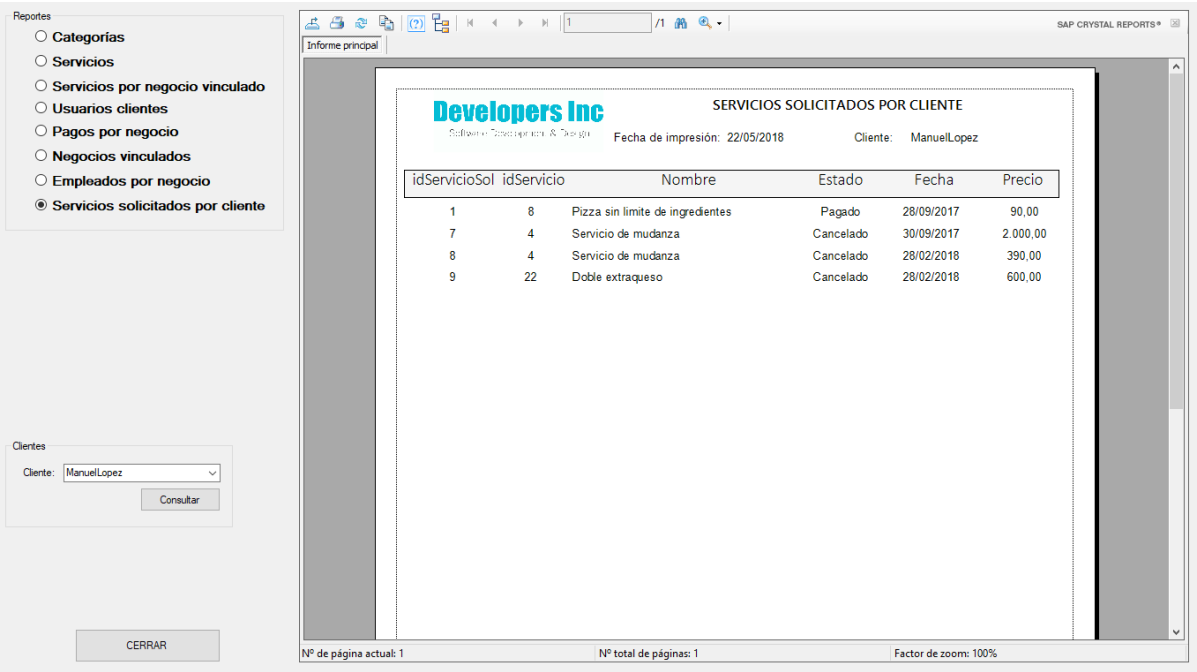

Figura 41. Interfaz de reporte de servicios solicitados por cliente.

6.2 Aplicación Web (rol de empleados de negocio)

6.2.1 Interfaz de inicio y bienvenida

En la Figura 42 se visualiza la interfaz de inicio y bienvenida de la aplicación, el cual comienza con una breve explicación donde se iniciará la sesión del empleado vinculado al negocio.

Figura 42. *Interfaz de Inicio y bienvenida.*

6.2.2 Interfaz de Login

En la Figura 43 se visualiza la interfaz de Login, muy parecida a la de administrador, sin embargo, tiene acceso solo para empleados de negocios vinculados.

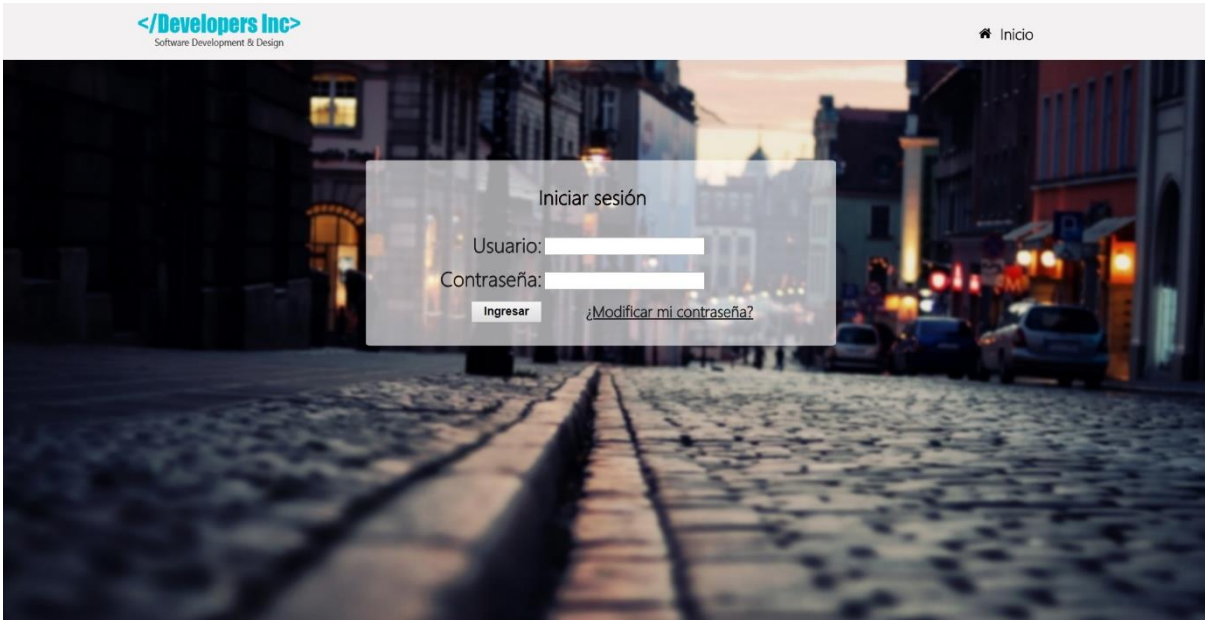

Figura 43. Interfaz de Login.

6.2.3 Interfaz principal

En la Figura 44 se visualiza la interfaz principal del negocio al que pertenece el empleado junto con su nombre de usuario, así como los servicios que ofrecen y su monto total de servicios solicitados por pagar al administrador.

Figura 44. Interfaz principal.

6.2.4 Interfaz principal con despliegue de consultas

En la Figura 45 se visualiza la interfaz la cual despliega las opciones de los servicios disponibles en consultas.

Figura 45. Interfaz principal con despliegue de consultas.

6.2.5 Interfaz de servicios solicitados del negocio

En la Figura 46 se visualiza la interfaz de servicios solicitados del negocio vinculado, la cual permite consultas por fechas y con posibilidad de editar y/o actualizar sus datos.

Figura 46. Interfaz de servicios solicitados del negocio.

6.2.6 Interfaz de servicios solicitados del negocio por rango de fechas

En la Figura 47 se visualiza la interfaz de servicios solicitados del negocio vinculado por fechas.

Figura 47. Interfaz de servicios solicitados del negocio por rango de fechas.

6.2.7 Interfaz de servicios solicitados del negocio por rango de fechas y por ciudad

En la Figura 48 se visualiza la interfaz de servicios solicitados del negocio vinculado por fechas y por ciudad, la cual permite consultas en un rango determinado de fechas o por ciudadpara el caso de que el negocio tenga sucursales en distintas ciudades o localidades.

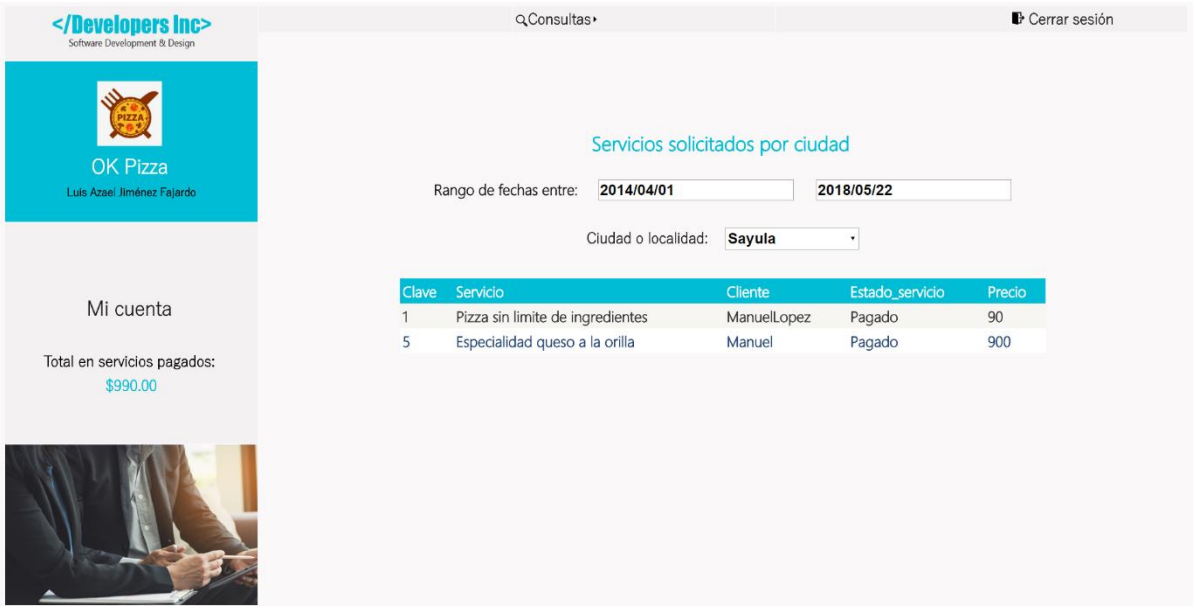

Figura 48. Interfaz de servicios solicitados del negocio por rango de fechas y por ciudad.

6.2.8 Interfaz de servicios solicitados del negocio por rango de fechas y por código postal

En la Figura 49 se visualiza la interfaz de servicios solicitados del negocio vinculado por fechas y por código postal.

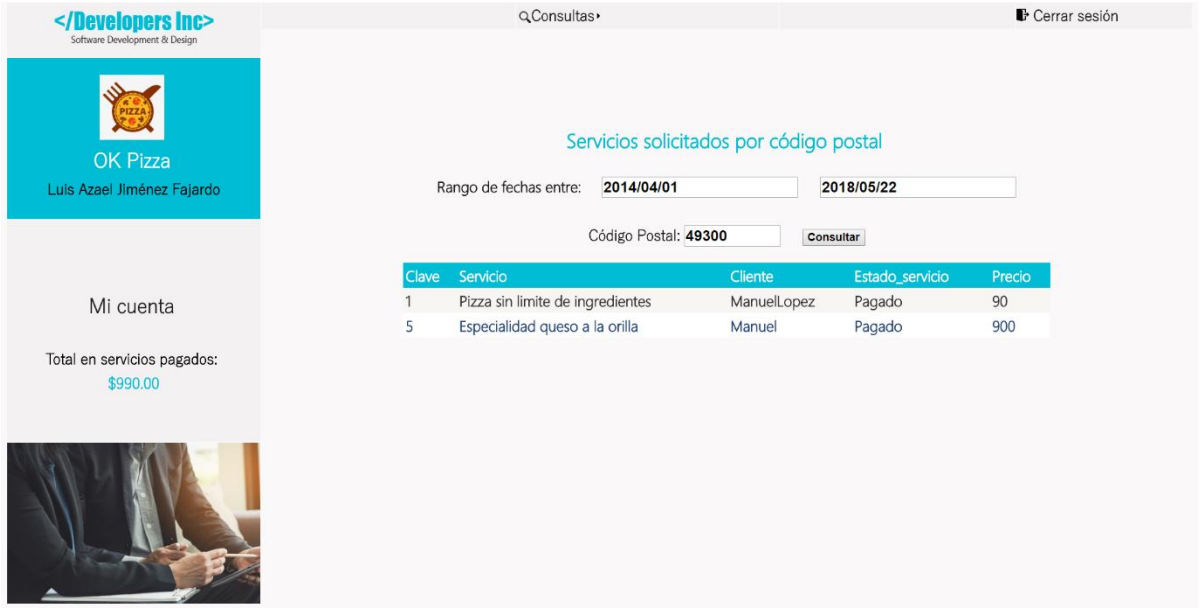

Figura 49. Interfaz de servicios solicitados del negocio por fechas y por código postal.

6.3 Aplicación móvil (rol de cliente)

6.3.1 Interfaz de bienvenida

En la Figura 50 se visualiza la interfaz bienvenida, la cual muestra el logo del desarrollador.

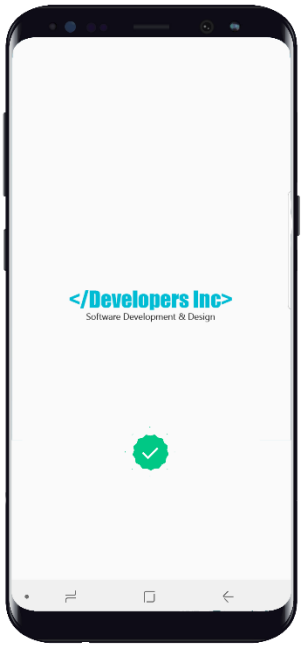

Figura 50. Interfaz de bienvenida.

6.3.2 Interfaz de Login

En la Figura 51 se visualiza la interfaz de Login, con distintas opciones para iniciar sesión.

Figura 51. Interfaz de Login.

6.3.3 Interfaz de inicio de sesión sin la opción de Facebook

En la Figura 52 se visualiza la interfaz de inicio de sesión tradicional.

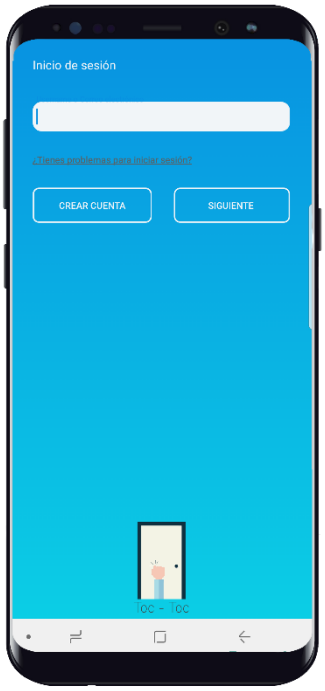

Figura 52. Interfaz de inicio de sesión sin opción de Facebook.

6.3.4 Interfaz de contraseña sin Login con Facebook

En la Figura 53 se visualiza la interfaz de ingreso de contraseña tradicional.

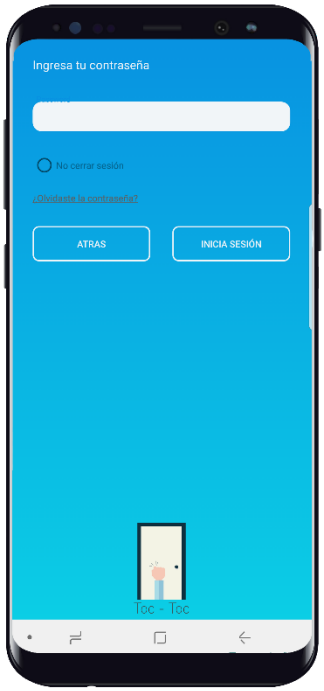

Figura 53. Interfaz de inicio de sesión sin opción de Facebook.

6.3.5 Interfaz de inicio de sesión con la opción de Facebook

En la Figura 54 se visualiza la interfaz de inicio de sesión ingresando con Facebook, la cual registrará nuevo cliente.

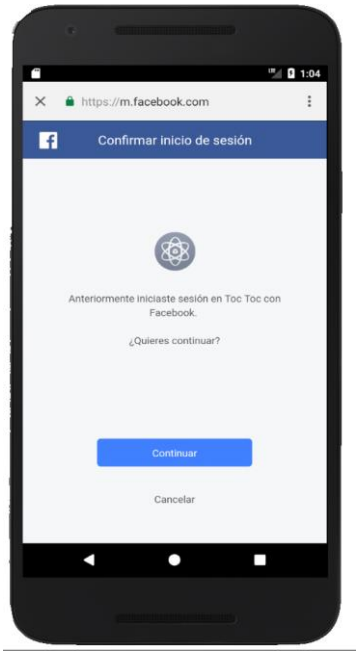

Figura 54. Interfaz de inicio de sesión con opción de Facebook.

6.3.6 Interfaz de crear cuenta directamente en la App

En la Figura 55 se visualiza la interfaz para crear una nueva cuenta como cliente.

Figura 55. Interfaz para crear una nueva cuenta como cliente.

6.3.7 Interfaz de ingresar contraseña nueva directamente en la App

En la Figura 56 se visualiza la interfaz para ingresar contraseña nueva como cliente.

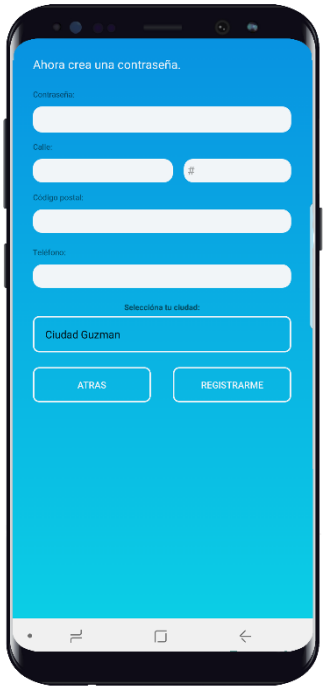

Figura 56. Interfaz para crear una nueva cuenta como cliente.

6.3.8 Interfaz principal con listado de categorías

En la Figura 57 se visualiza la interfaz principal que cuenta con un listado de las distintas categorías de negocio.

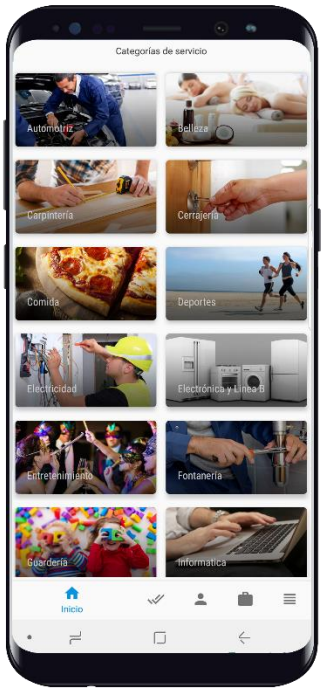

Figura 57. Interfaz principal con listado de categorías.

6.3.9 Interfaz del listado de negocios vinculados acorde a la categoría

En la Figura 58 se visualiza la interfaz con el listado de los distintos negocios vinculados.

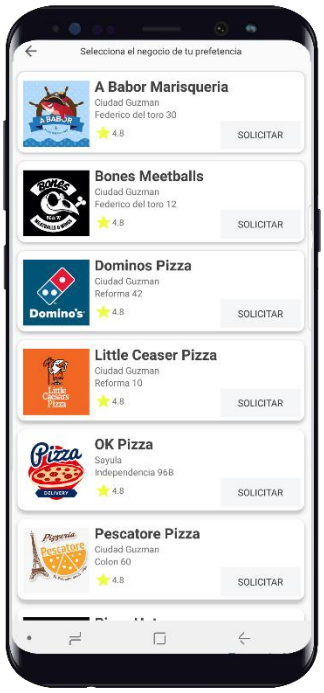

Figura 58. Interfaz con listado de negocios por categoría.

6.3.10 Interfaz de negocio seleccionado con mapa de ubicación

En la Figura 59 se visualiza la interfaz con mapa de ubicación del cliente.

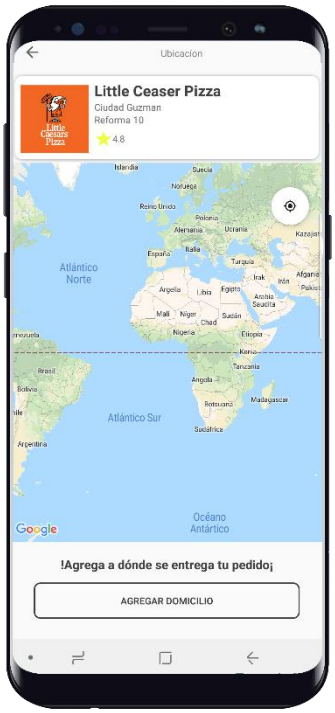

Figura 59. Interfaz de negocio seleccionado con mapa de ubicación.

6.3.11 Interfaz de negocio seleccionado con mapa de ubicación y barra de búsqueda

En la Figura 60 se visualiza la interfaz con mapa de ubicación, con los controles para confirmar ubicación y la barra de búsqueda.

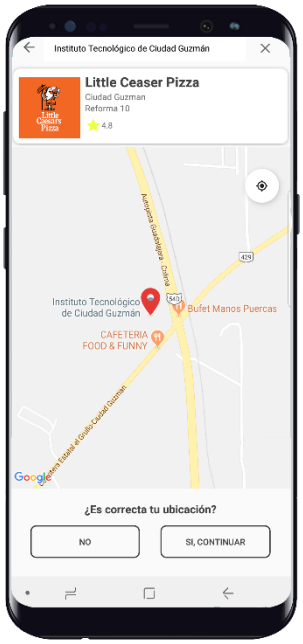

Figura 60. Interfaz de negocio seleccionado con mapa de ubicación y barra de búsqueda.

6.3.12 Interfaz de negocio seleccionado con sus respectivos servicios

En la Figura 61 se visualiza la interfaz del negocio seleccionado con sus datos y los servicios que ofrece dicho negocio.

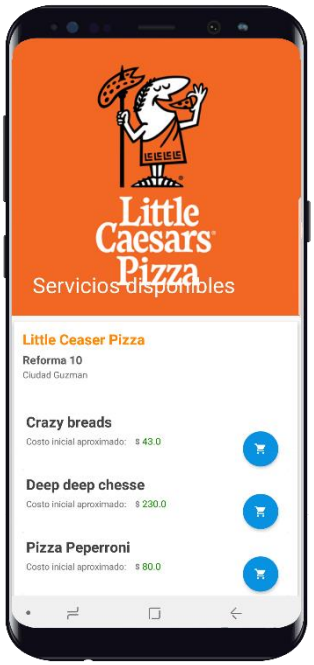

Figura 61. Interfaz de negocio seleccionado con sus respectivos servicios.

6.4 Reportes aplicación de escritorio

6.4.1 Interfaz de reporte de categorías de servicio

En la Figura 62 se visualiza la interfaz de reporte de categorías de servicio.

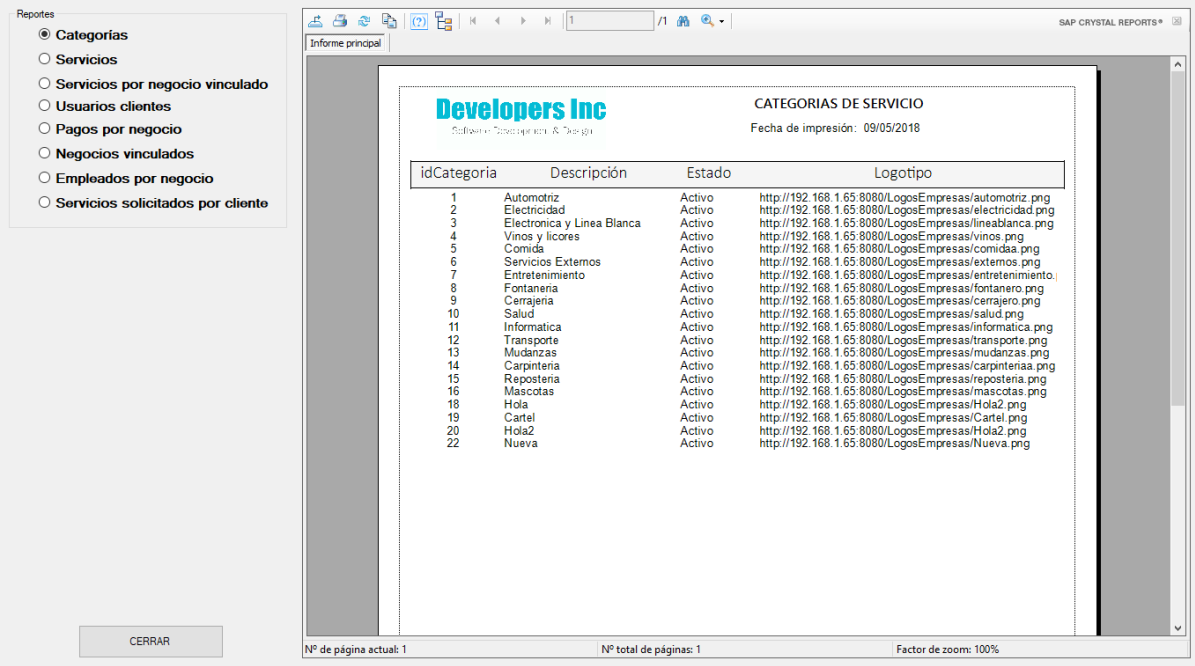

Figura 62. Interfaz de reporte de categorías de servicio.

6.4.2 Interfaz de reporte de servicios

En la Figura 63 se visualiza la interfaz de reporte de servicio.

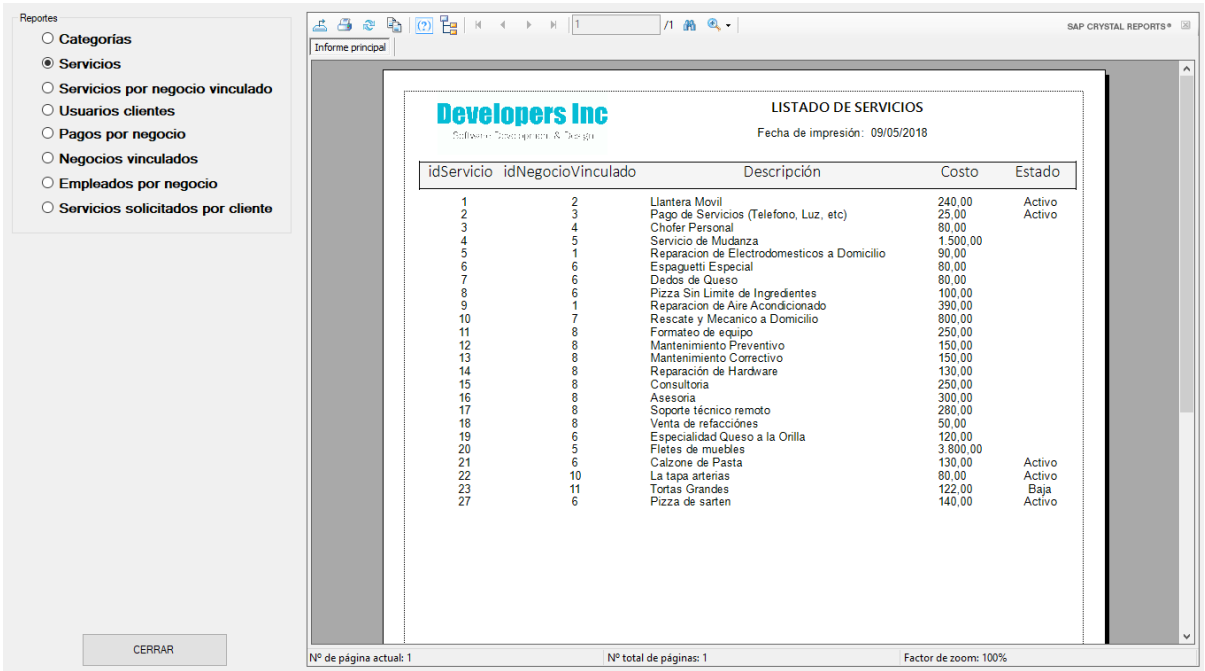

Figura 63. Interfaz de reporte de servicios.

6.4.3 Interfaz de reporte de servicios por negocio vinculado

En la Figura 64 se visualiza la interfaz de reporte de servicios por negocio vinculado el cual recibe un parámetro de entrada directamente de una ComboBox con tipo de dato String.

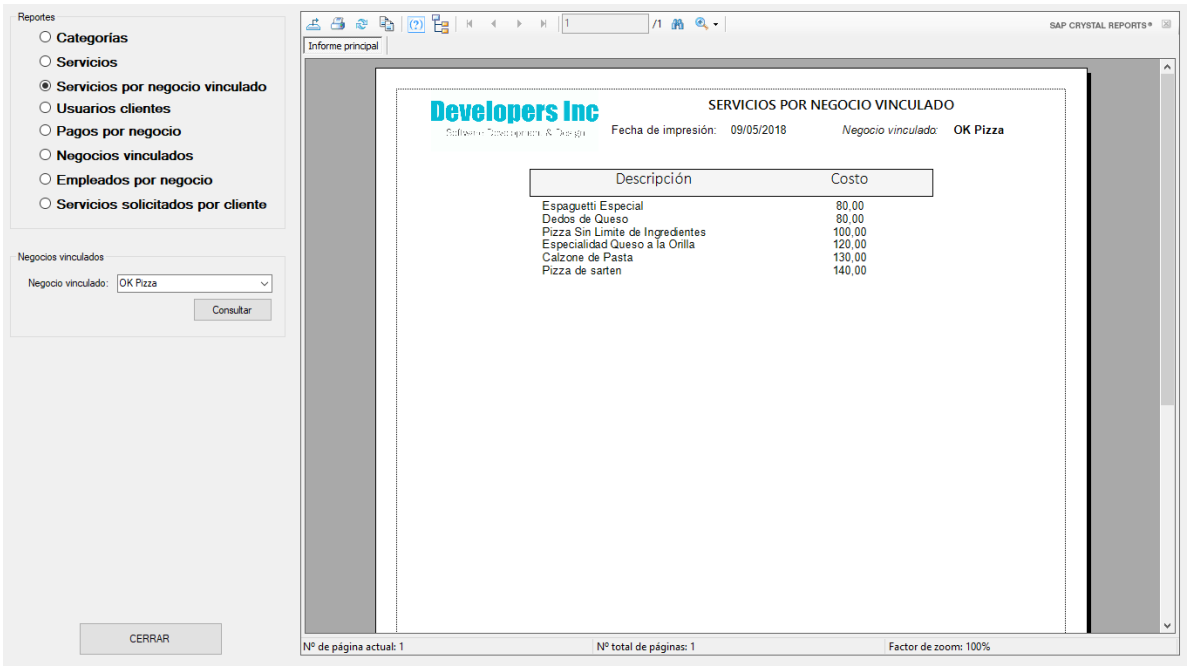

Figura 64. Interfaz de reporte de servicios por negocio vinculado.

6.4.4 Interfaz de reporte de usuarios clientes

En la Figura 65 se visualiza la interfaz de reporte de usuarios clientes.

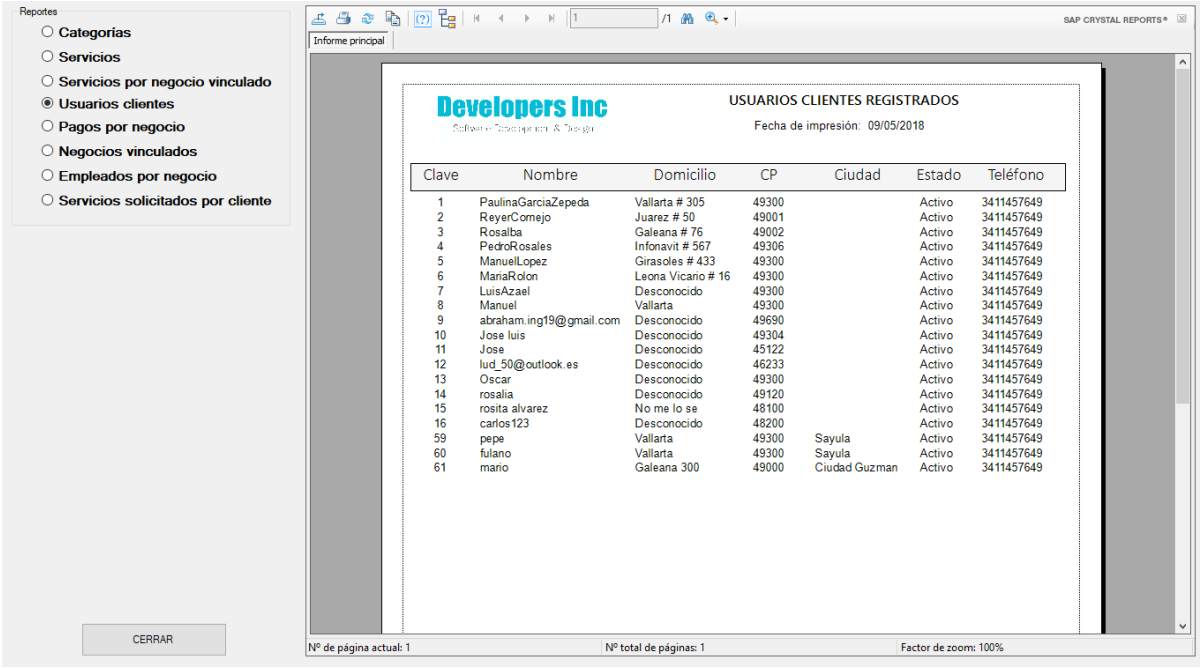

Figura 65. Interfaz de reporte de usuarios clientes.

6.4.5 Interfaz de reporte de negocios vinculados registrados

En la Figura 66 se visualiza la interfaz de reporte de negocios vinculados.

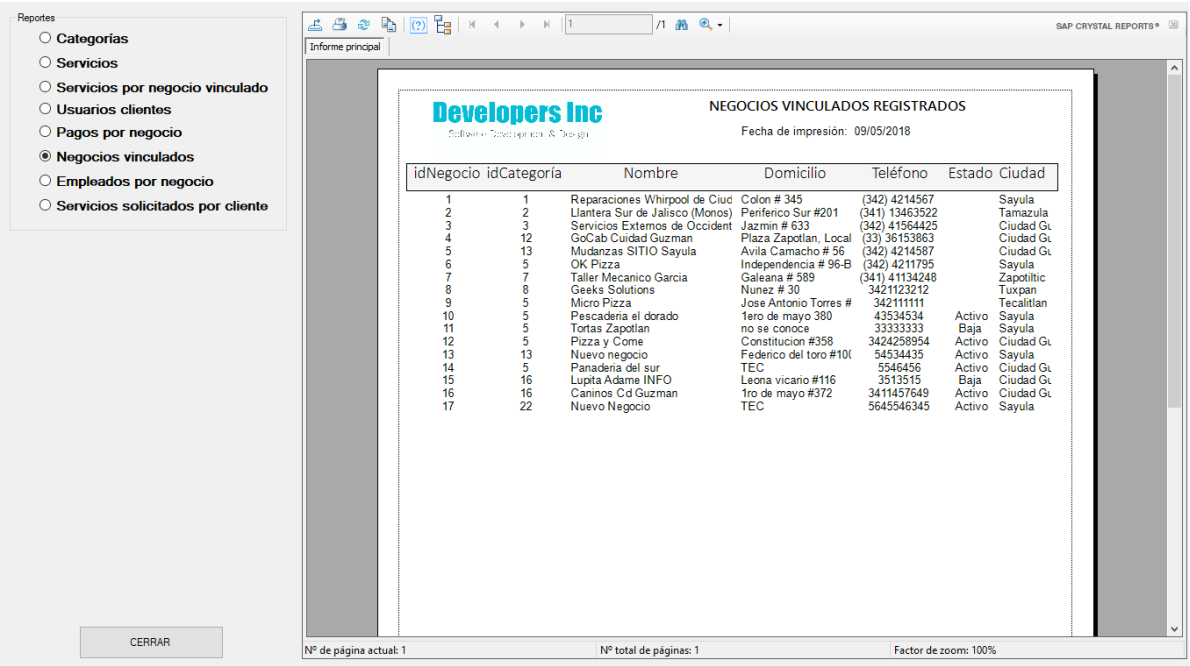

Figura 66. Interfaz de reporte de usuarios clientes.

6.4.6 Interfaz de reporte de empleados por negocio vinculado

En la Figura 67 se visualiza la interfaz de reporte de empleados por negocio vinculado el cual recibe un parámetro de entrada directamente de una ComboBox con tipo de dato String.

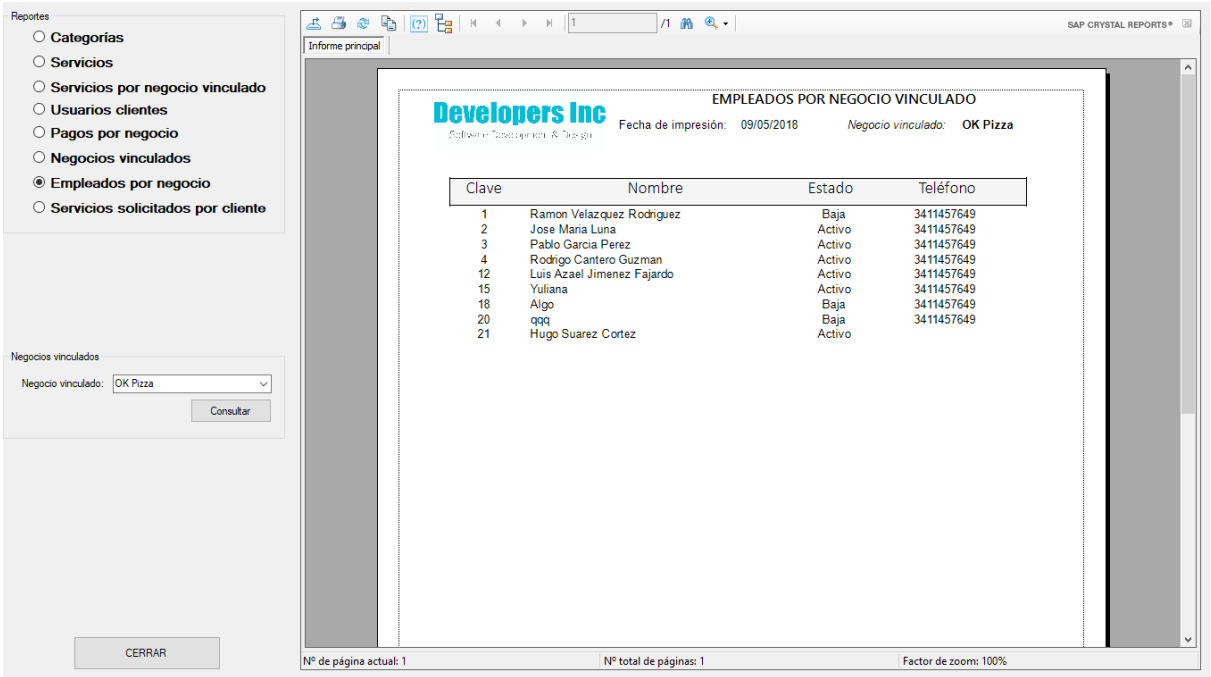

Figura 67. Interfaz de reporte de empleados por negocio vinculado.

6.4.7 Interfaz de reporte de servicios solicitados por cliente

En la Figura 68 se visualiza la interfaz de reporte de servicios solicitados por cliente el cual recibe un parámetro de entrada directamente de una ComboBox con tipo de dato String.

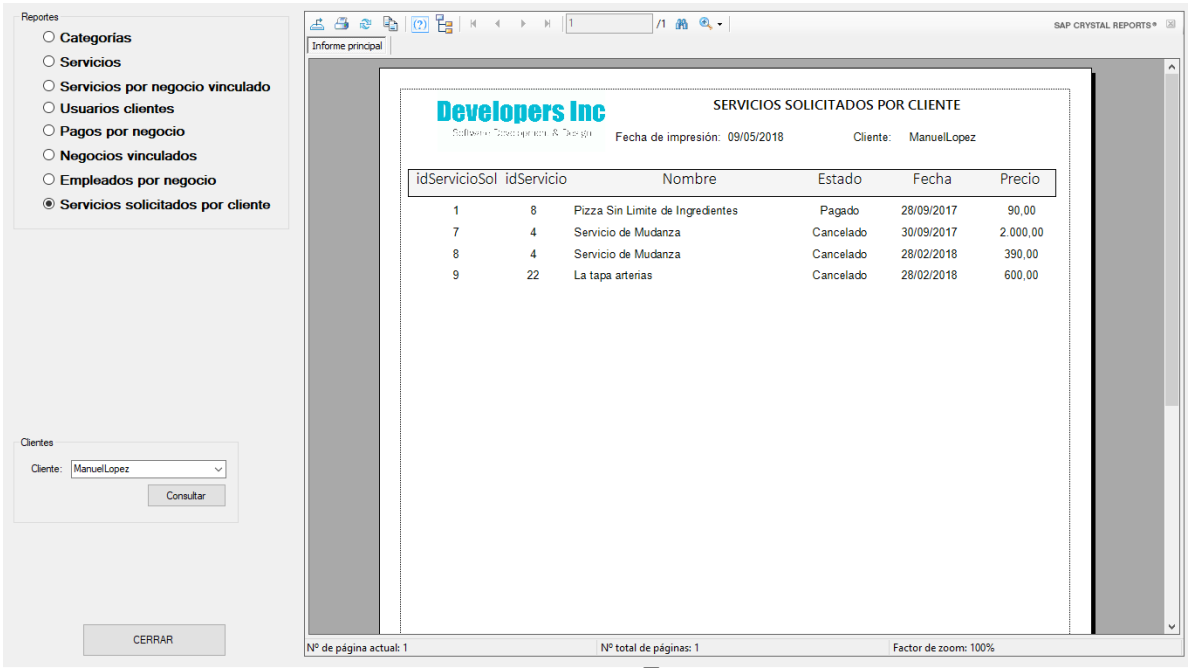

Figura 68. Interfaz de reporte de servicios solicitados por cliente.

7. DISCUSIÓN

7.1 Problema de diseño en DataGridView

Durante el diseño de la aplicación de escritorio, no se lograba establecer el color de fondo de los encabezados en los *DataGridView*, por lo que fue necesario buscar información relacionada con este problema.

En la Figura 69 se presenta el diseño previo a la corrección en el encabezado del *DataGridView*.

Figura 69. Interfaz previa de clientes sin color de fondo en encabezado.

Para resolver el problema, se aplicó la correccion a nivel diseño, con el siguiente código dentro del evento *Load()*.

```
Me.DG.EnableHeadersVisualStyles = False
Dim estiloDG As DataGridViewCellStyle = New DataGridViewCellStyle()
estiloDG.BackColor = Color.DarkTurquoise
estiloDG.ForeColor = Color.White
estiloDG.Alignment = DataGridViewContentAlignment.TopCenter
Me.DG.ColumnHeadersDefaultCellStyle = estiloDG
```

En la Figura 70 se muestra el diseño de la interfaz con los colores en el encabezado del *DataGridView* alusivos a la aplicación.

Figura 70. Interfaz de clientes con el problema de diseño corregido.

7.2 Implementación de Login con la API de Facebook

El tema de la seguridad e integridad de los datos se ha ido incrementando en los últimos años, por distintos ataques cibernéticos. Por lo anterior, se ha decidido incluir una de las mejores herramientas de seguridad e integración de datos con la posibilidad de iniciar sesión con Facebook, debido a su alto régimen de confidencialidad.

Dentro de la aplicación se incluyó en la opción de Login, iniciar sesión por medio de Facebook, implementando una API de la manera siguiente:

```
implementation 'com.facebook.android:facebook-login:[4,5)'
```

El código anterior se incluye en el *build.gradle* en la parte de módulos de la aplicación móvil. Asi mismo, se incluye en la parte del *Manifest* el siguiente código:

```
<meta-data          android:name="com.facebook.sdk.ApplicationId"
    android:value="@string/facebook_App_id"/>
    android:name="com.facebook.CustomTabActivity"
    android:exported="true">
    <intent-filter>
        <action   android:name="android.intent.action.VIEW"   />
        <category
android:name="android.intent.category.DEFAULT"                />
        <category
```

```
android:name="android.intent.category.BROWSABLE"                />
        <data  android:scheme="@string/fb_login_protocol_scheme"
/>
    </intent-filter>
</activity>
```

El código anterior arrojó un problema en las líneas siguientes:

```
android:value="@string/facebook_App_id"
```

```
<data android:scheme="@string/fb_login_protocol_scheme" />
```

Ya que en estos atributos se necesita un identificador único para el desarrollador, el cual se obtiene en la página oficial de Facebook for developers.

7.3 Problema de elementos dinámicos en Android

Durante el diseño de la aplicación móvil se detectó un problema al agregar nuevos elementos en tiempo de ejecución. Se corrigió realizando elementos dinámicos automáticos en relación a la base de datos, el cual su función es añadir nuevos elementos a la aplicación si en la base de datos se agrega un nuevo registro. Esto evitará rehacer la codificación en la aplicación móvil en caso de que se agreguen nuevos registros de categorías y negocios vinculados.

En el siguiente código se muestran los distintos elementos como la URL donde se encuentra el código PHP para llamar a la consulta de la base de datos, así como el adaptador de contenido y la lista dinámica los cuales son necesarios para llamar al método de creación dinámica.

```
private              static              final              String
PRODUCT_URL="http://192.168.1.65:8080/TocToc/consultaDinamicaCa
tegorias.php";
RecyclerView                                          recyclerView;
ProductAdapterCategorias                                   adapter;
List<ProductCategorias> productList;
```

En el siguiente código se muestra el ciclo que recibe la consulta desde la base de datos por medio del archivo PHP y asigna a las variables necesarias los datos que llevará cada elemento nuevo. En este caso, se hace referencia al nombre y logotipo de una categoría en específico. Posteriormente, un adaptador de contenido va agregando directamente a la lista dinámica, recibiendo dentro de la clase llamada *IngresoUsuario.class*. Esta clase recibe del contexto del adaptador el conjunto de los parámetros de contenido para mostrarlos en pantalla.

```
for(int          i=0;          i<products.length();          i++)
{
    JSONObject     productObject=     products.getJSONObject(i);
    String       Nombre=       productObject.getString("Nombre");
    String         Logotipo=productObject.getString("Logotipo");
    ProductCategorias    product=new    ProductCategorias(Nombre,
Logotipo);
    productList.add(product);
```

```
}
adapter=    new    ProductAdapterCategorias(IngresoUsuario.this,
productList);
recyclerView.setAdapter(adapter);
```

7.4 Problemas de identificador de botón en la lista dinámica

Como es lógico al momento de implementar los elementos dinámicos no es posible saber que identificador tienen los nuevos elementos, o al menos que nombre lleva cada elemento que se agrega automáticamente. Lo anterior resultó ser un serio problema para realizar los escuchadores y codificar cada nuevo botón. Así que la solución fue realizar un contenedor con vistas el cual actualiza el contenido de la lista dinámica y asigna una posición. Este contenedor con vistas se llama *onBindViewHolder()* de la clase *RecyclerView* con el método *adapter()*.

El siguiente código muestra la funcionalidad del contenedor así como la asignación de la posición a cada elemento, para posteriormente en el método *onBindViewHolder()* realizar el *OnClick()* como escuchador, dentro del contenedor que se denominó *holder*.

```
@Override
public  void  onBindViewHolder(final  ProductViewHolder  holder,
final                        int                          position)
{
    final     Product     product=     productList.get(position);
    holder.tvNombre.setText(product.getNombre());
    holder.tvLogo.setText(product.getLogotipo());

Glide.with(mCtx).load(product.getLogotipo()).into(holder.imagen
);
    holder.btOrder.setOnClickListener(new
View.OnClickListener()                                          {
        @Override
        public          void          onClick(View          view)
        {
if(holder.tvNombre.getText().toString().equals(product.getNombr
e()))
  {
   Toast.makeText(mCtx,              "Presionaste"              +
holder.tvNombre.getText().toString(),
Toast.LENGTH_SHORT).show();
  }

 }
    });
```

7.5 Problema de reportes en VB

El diseño de la base de datos se realizó en MySQL, lo cual generó distintos problemas en el apartado de conexiones, proveedor de datos y procedimientos almacenados.

Una de las soluciones aplicadas fue realizar conexiones directamente desde Orígenes de datos ODBC (32 bits), la cual requirió la instalación del conector MySQL Connector/ODBC y localizado en página oficial de MySQL.

Los reportes fueron diseñados con el software de SAP Crystal Reports en su versión 2013 el cual se obtuvo de su página oficial.

Fue necesario importar las siguientes clases para procesar reportes:

```
Imports Microsoft.Reporting.WinForms
Imports CrystalDecisions.Shared
```

Para visualizar los reportes sin parámetros fue necesario invocar los archivos con extensión .rpt por medio de la siguiente instrucción:

```
Me.CrystalReportViewer1.ReportSource=  =  "[RUTA  DEL  ARCHIVO
.RPT]"
Me.Show()
```

7.6 Problema de implementación de reportes con parámetros

Para obtener reportes con parámetros fue necesario generar un código que permitiera recibir los parámetros tanto para el procedimiento almacenado para filtrar datos como para ser desplegado ese valor dentro del informe.

```
Dim Parametros As New ParameterFields()
Dim PrimerParametro As New ParameterField()
Dim myDiscreteValue As New ParameterDiscreteValue()
PrimerParametro.ParameterValueType                              =
ParameterValueKind.StringParameter
PrimerParametro.ParameterFieldName = "cliente"
myDiscreteValue.Value = cboCliente.Text
PrimerParametro.CurrentValues.Add(myDiscreteValue)
Parametros.Add(PrimerParametro)
CrystalReportViewer1.ParameterFieldInfo = Parametros
Me.CrystalReportViewer1.ReportSource = "[Ruta del archivo .RPT]"
```

La palabra *"cliente"* se refiere al nombre del parámetro del procedimiento almacenado en la base de datos. De igual forma es necesario indicar el valor que tomará el parámetro en el sistema, el cual se asigna desde la *cboCliente*.

En la Figura 71 visualiza el reporte de "*SERVICIOS POR NEGOCIO VINCULADO"* con el parámetro cliente es llamado *"Geeks solutions"*.

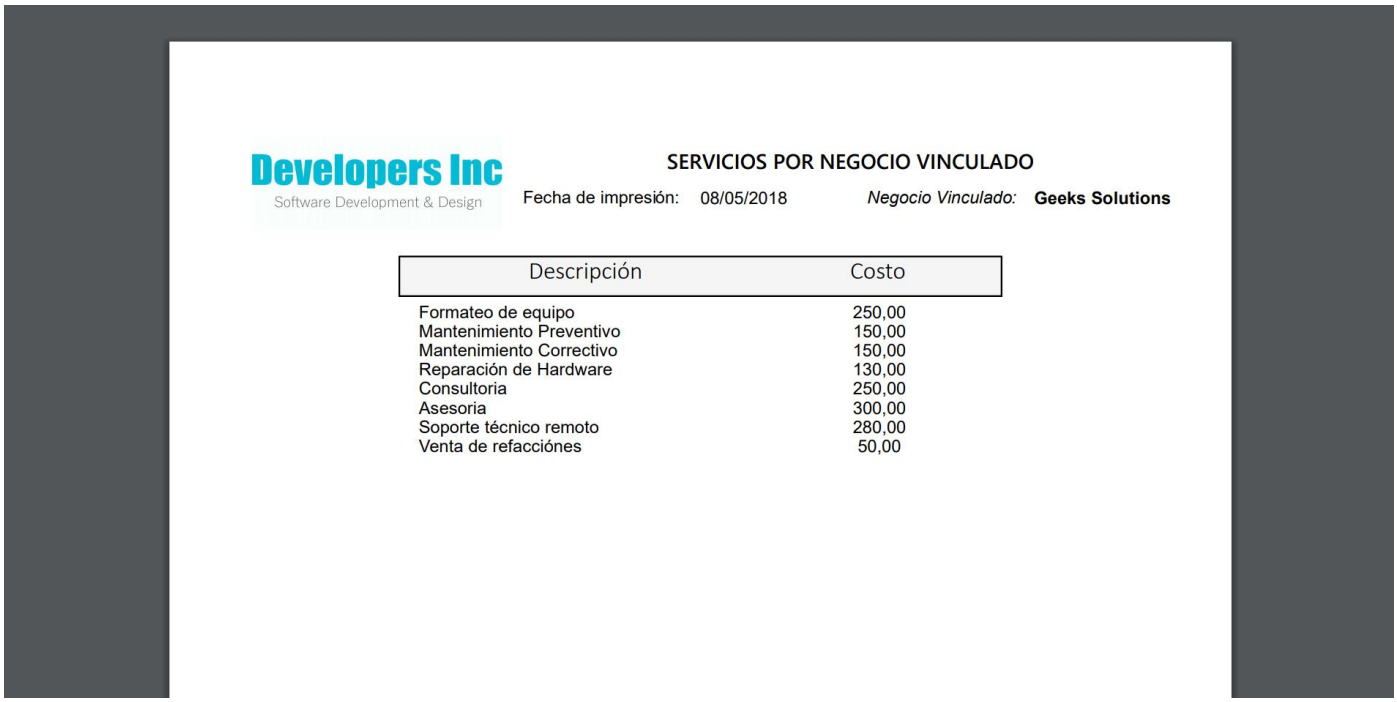

Figura 71. Interfaz de reporte de servicios por negocio vinculado con parámetro.

7.7 Problema para cargar imágenes en el servidor

La carga de imágenes es considerada en la aplicación de escritorio para que el administrador de la plataforma asigne los logotipos por categoría y las imágenes que identifican a los negocios vinculados. Estos archivos son copiados a la url destino en el servidor. Para solucionar este problema fue necesario implementar un elemento la clase *OpenFileDialog* de la siguiente manera:

```
My.Computer.FileSystem.CopyFile("RUTA_DE_ORIGEN",
"RUTA_DESTINO", True)
```

El código que se implementó ya adaptado al sistema fue el siguiente, el cual obtiene la ruta de origen de la clase OpenFileDialog con la propiedad FileName.

```
My.Computer.FileSystem.CopyFile(OpenFileDialog1.FileName,
"http://Localhost/LogosEmpresas/" & txtDescripcion.Text & ".png", True)
```

7.8 Problema de consulta de datos con tildes

Se presentó el problema al consultar datos que incluían acentos o tildes, ya que el cotejamiento de la base de datos requiere específicamente el del idioma español y regiones de América Latina el cual es utf8_spanish_ci. Por otra parte, también fue necesario modificar el archivo PHP que contiene las consultas ya que envía el formato JSON directamente a Android por medio de Web Services. El código que se presenta a continuación, muestra la modificación que se realizó en el archivo PHP de consultas en el cual se añadió la función *utf8_encode()*:

```
$temp [ 'Descripcion' ] = utf8_encode($Descripcion);
```

En Android se incluyó el siguiente código que permite adaptar el JSON recibido al cotejamiento que se requiere:

```
connection.setRequestMethod("POST");
connection.setRequestProperty("Content-Type",
"Application/json; charset=utf-8");
connection.setDoOutput(true);
```

7.9 Problema a nivel diseño con RecyclerView

Otro problema que se presentó durante el desarrollo de la aplicación móvil fue que los componentes asignados a *RecyclerView* trazados en el diseño no se comportaban de manera correcta en tiempo de ejecución en el archivo principal con extensión XML, ya que se duplicaban los datos de la empresa en cada uno de los servicios que ofrece. Para resolver este problema fue necesario dentro de la clase principal llamada *ConcluirServicio.class* indicar un nuevo componente de tipo *RecyclerView* el cual se inicializa con *recyclerViewNegocio* y *recyclerViewServicios* dentro del archivo XML correspondiente a la clase principal.

El código que se presenta a continuación muestra la manera correcta de incluir distintos componentes, creando dos archivos XML.

```
recyclerView=(RecyclerView)findViewById(R.id.recyclerViewNegocio);
recyclerView2=(RecyclerView)findViewById(R.id.recyclerViewServicios);
```

Posteriormente, dentro de la clase secundaria llamada *ProductAdapterConcluirServicio.class*, es asignado el layout con nombre *list_layout_negocio_servicio_concluido.XML* para visualizar el diseño previamente realizado:

```
View view =
inflater.inflate(R.layout.list_layout_negocio_servicio_concluido, null);
```

Lo mismo sucede con la clase *ProductAdapterServicios.class*, la cual llama al layout *list_layout_servicios.XML*:

```
View view = inflater.inflate(R.layout.list_layout_servicios,
null);
```

Para agregar el negocio vinculado a la interfaz de *servicios por negocio*, es necesario agregar los datos de la consulta que arroja el archivo PHP por medio de JSON, el siguiente código ubicado en la clase principal *ConcluirServicio.class* permite agregar cada registro a los componentes del archivo *list_layout_negocio_servicio_concluido.XML*.

```
Product product = new Product(Nombre, Ciudad, Domicilio,
Logotipo);
productList.add(product);
```

Una vez agregados los datos a los componentes correspondientes, se envía el adaptador de contenido con la lista de datos directamente al *RecyclerView* antes mencionado:

```
adapter = new
ProductAdapterConcluirServicio(ConcluirServicio.this,
productList);
recyclerView.setAdapter(adapter);
```

Lo mismo sucede con los datos de los servicios del negocio, pero en este caso se llama al archivo PHP que contiene el nombre del servicio, así como su precio inicial, y se agregan a los componentes correspondientes dentro del archivo *list_layout_servicios.XML*.

```
ProductServicios product = new ProductServicios(nombreServicio,
precioServicio);
productList2.add(product);
```

Una vez agregados los datos a los componentes correspondientes, el nuevo adaptador de contenido llamado *adapter2* se encarga de adaptar la lista de datos al nuevo *RecyclerView* mencionado previamente:

```
adapter2= new ProductAdapterServicios(ConcluirServicio.this,
productList2);
recyclerView2.setAdapter(adapter2);
```

De esta manera, en la Figura 72 se muestra el resultado final.

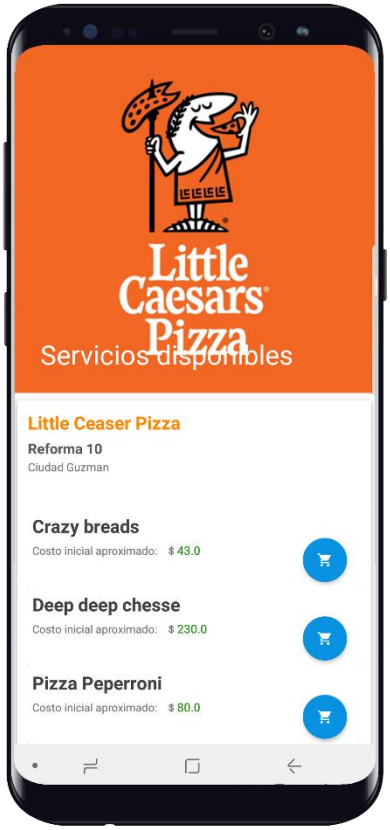

Figura 72. Interfaz de negocio seleccionado con sus respectivos servicios.

8. CONCLUSIONES

La población va creciendo a un ritmo acelerado y por ello los negocios necesitan diversificar más los servicios que ofrecen para satisfacer la demanda. Los negocios actuales en su mayoría no cuentan con sistemas de gestión, por lo anterior, puede haber pérdidas significativas si la competencia atiende a los clientes. Además, para que una ciudad sea ciudad inteligente, se requiere que las empresas o negocios tengan sistemas informáticos automatizados actualizados a la nueva tecnología.

El proyecto desarrollado permitirá brindar y generar empleo a más personas, y por lo tanto una mejor economía a nivel regional.

Por otra parte, el aprendizaje en el manejo de múltiples lenguajes de programación para realizar esta plataforma, abre un panorama tanto profesional como laboral donde se puede aprovechar todos los conocimientos en distintos ámbitos, dando soluciones rápidas a problemas informáticos reales.

Con este proyecto aprendí a detectar requerimientos, distintas ideas y formas de pensar, ya que se tienen que contemplar aspectos tanto funcionales como de diseño a la hora de codificar para que la plataforma sea amigable, intuitiva y fácil de manejar. Esta es la clave para que un sistema pueda funcionar de manera adecuada y tener más negocios que quieran unirse a la plataforma ofreciendo mejor atención a los clientes.

9. REFERENCIAS BIBLIOGRÁFICAS Y VIRTUALES

Android (2018). Developer Android. Recuperado el 31 de Julio de https://developer.android.com

Arias, M. A. (2013). Introducción a PHP, México (2013), México: IT Campus Academy

ASP (2018). Información general sobre ASP.NET. Recuperado el 31 e Julio de https://msdn.microsoft.com/es-es/library/4w3ex9c2(v=vs.100).aspx

Barrio, M. (2018). Internet de las cosas, Madrid, España: REUS

Berzal, F., Cortijo, F. J. y Cubero, J.C. *(2007).* Desarrollo profesional de aplicaciones Web con ASP.NET, España: ikor Consulting

Bouskela, M., Casseb, M., Bassi, S., De Luca, C. y Facchina, M. (2016). La ruta hacia las Smart Cities: Migrando de una gestión tradicional a la ciudad inteligente. Banco Interamericano de Desarrollo (BID) Recuperado el 29 de Diciembre de 2017 de https://publications.iadb.org/bitstream/handle/11319/7743/La-ruta-hacia-las-smart-cities-Migrando-de-una-gestion-tradicional-a-la-ciudad-inteligente.pdf.

Del Sole, A. (2017). Visual Studio, Morrisville , USA: Syncfusion

De Seta, L. (2008). Introducción a los servicios Web RESTful. Recuperado el 31 de Julio de 2018 de https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-Web-restful

Dimes,T. (2016). PHP, Inglaterra: Babelcube Inc.

Facebook. (2018). Facebook for developers, recuperado el 14 de Enero de 2018 de https://developers.facebook.com/

Fossati, M. (2017). Todo sobre Visual Basic, recuperado el 31 de Julio de 2018 de https://books.google.com.mx/books?id=GdwxDwAAQBAJ&printsec=frontcover&dq=visual+basic&hl=es&sa=X&ved=0ahUKEwj7zOH6kLTcAhUDUK0KHYcxAtoQ6AEIKzAA#v=onepage&q=visual%20basic&f=false

Gironés, J. T. (2015), El gran libro de android, Alfaomega/marcombo: México.

Google (2018). Google APIs for Android. Recuperado el 31 de Julio de 2018 de https://developers.google.com/android/reference/com/google/android/gms/maps/GoogleMap

Google (2018). Google Maps Platform. Recuperado y https://developers.google.com/maps/documentation/android-sdk/start

JSON (2018). Introducción a JSon. Recuperado el 31 de Julio de 2018 de https://www.json.org/json-es.html

Koekkoek, H. (2011). Distimo Publication Full Year 2011. Recuperado de Distimo Publication: http:// www.distimo.com/download/publication/ Distimo_Publication_-_Full_Year_2011/EN/ archive/

Méndez, M. (2017). Cultural and Smart City, España: Dykinson

Moreno (2014). Estudio de perspectivas y estrategias de desarrollo y difusión de aplicaciones móviles. Recuperado el 11 de Septiembre de 2018 de https://amiti.org.mx/wp-content/uploads/2013/10/RE_Estudio-APPS.pdf

Navarrete, T. (2006). El lenguaje JavaScript. Recuperado el 31 de Julio de 2018 de http://www.dtic.upf.edu/~tnavarrete/fcsig/javascript.pdf

MySQL (2018). Chapter 1 Introduction to MySql Connector/NET. Recuperado el 31 de Julio de 2018 de https://dev.mysql.com/doc/connector-net/en/connector-net-introduction.html

Piñar, J. L. (2017). Smart cities Derecho y técnica para una ciudad más habitable, España: REUS

Quiñonez, E., Ureña, Y.C. y Carruyo N. (2016). SMART CITY: FUTURISTIC VISION OF THE KNOWLEDGE SOCIETY IN SUCRE DEPARTMENT-COLOMBIA. Revista Científica Electrónica de Ciencias Gerenciales / Scientific e-journal of Management Science, num 35 (año 12) pág. 3-18.

Ramírez, E. Contreras, O. y Contreras, C. (2014) . Programación Móvil, recuperado el 31 de Julio de 2018 de http://programacionmovilufps.blogspot.com/2014/11/que-es-la-programacion-la-programacion.html

Revelo, J. (2015). Realizar peticiones Http con la librería Volley en Android. Recuperado el 31 de Julio de 2018 de http://www.hermosaprogramacion.com/2015/02/android-volley-peticiones-http/

Rodríguez J. J. (2003). Introducción a la programación. Teoría y práctica: teoría y práctica, España: Editorial Club Universitario

SAP Store (2018). SAP Crystal Reports 2013, recuperado el 5 de Enero de 2018 de https://www.sapstore.com/solutions/99013/SAP-Crystal-Reports-2013

10. ANEXOS

10.1 Código de registro de un nuevo negocio vinculado en sistema de escritorio en el rol de administrador.

```
Imports MySql.Data
Imports MySql.Data.Types
Imports MySql.Data.MySqlClient
Public Class NegociosVinculados

    Private Sub NegociosVinculados_Load(sender As Object, e As EventArgs) Handles
MyBase.Load
        DG.Refresh()
        If ConexionGlobal() Then
            'MessageBox.Show("Conectado")
        Else
            'MessageBox.Show("No Conectado")
        End If
        ''----------------------------------------
        Dim Comando As New MySqlCommand("select * from UsuariosEmpleados", Conexion)
        Dim Lector As MySqlDataReader
        Lector = Comando.ExecuteReader

        While Lector.Read()
            cboNombreEmpleado.Items.Add(Lector(1))
        End While
        Lector.Close()
        ''----------------------------------------
        Comando.CommandText = "select * from Categorias"
        Lector = Comando.ExecuteReader

        While Lector.Read()
            cboDescripcion.Items.Add(Lector(1))
        End While
        Lector.Close()
        ''----------------------------------------
        Comando.CommandText = "select * from NegociosVinculados"
```

```
        Lector = Comando.ExecuteReader
        While Lector.Read()
            DG.Rows.Add(Lector(0),   Lector(2),   Lector(3),   Lector(4),   Lector(5),
Lector(6), Lector(7), Lector(9), Lector(12))
        End While
        Lector.Close()

        cboEdo.Items.Add("Activo")
        cboEdo.Items.Add("Baja")
        DG.RowsDefaultCellStyle.BackColor = Color.WhiteSmoke
        DG.AlternatingRowsDefaultCellStyle.BackColor = Color.White
        btNuevo.Enabled = True
        btAceptar.Enabled = True

        Me.DG.EnableHeadersVisualStyles = False

        ' estilo para las cabeceras
        Dim styCabeceras As DataGridViewCellStyle = New DataGridViewCellStyle()
        styCabeceras.BackColor = Color.DarkTurquoise
        styCabeceras.ForeColor = Color.White
        styCabeceras.Alignment = DataGridViewContentAlignment.TopCenter

        ' asignar estilo al grid
        Me.DG.ColumnHeadersDefaultCellStyle = styCabeceras
        ''problemas a priorizar
    End Sub
    Private Sub cboDescripcion_SelectedIndexChanged(sender As Object, e As EventArgs)
Handles cboDescripcion.SelectedIndexChanged
        Dim R As String
        R = "SELECT * FROM Categorias WHERE Descripcion= '" & cboDescripcion.Text & "'"
        Dim Comando As New MySqlCommand(R, Conexion)
        Dim Lector As MySqlDataReader
        Lector = Comando.ExecuteReader
        Lector.Read()
        txtidCategoria.Text = Lector(0)
        Lector.Close()
    End Sub
```

```
    Private Sub cboNombreEmpleado_SelectedIndexChanged(sender As Object, e As EventArgs)
Handles cboNombreEmpleado.SelectedIndexChanged
        Dim R As String
        R   =   "SELECT   *   FROM   UsuariosEmpleados   WHERE   NombreUsuario=   '"   &
cboNombreEmpleado.Text & "'"
        Dim Comando As New MySqlCommand(R, Conexion)
        Dim Lector As MySqlDataReader
        Lector = Comando.ExecuteReader
        Lector.Read()
        txtidEmpleado.Text = Lector(0)
        Lector.Close()
    End Sub

    Private Sub btAceptar_Click(sender As Object, e As EventArgs) Handles btAceptar.Click
        DG.Rows.Clear()
        Dim R As String
        Dim contador As Integer
        Dim logo As String


        ''Guardar Imagen en Servidor----------------------------------------------------
----------------------------------------------
        My.Computer.FileSystem.CopyFile(OpenFileDialog1.FileName,
"C:\xampp\htdocs\LogosEmpresas\" & txtNombreNegocio.Text & ".png", True)
        tvImagen.Text = txtNombreNegocio.Text & ".png"
        logo   =   "http://192.168.1.65:8080/LogosEmpresas/"   &   txtNombreNegocio.Text   &
".png"
        ''-------------------------------------------------------------------------------
----------------------------------------------


        contador = 1
        R = "insert into NegociosVinculados (idCategoria, Nombre, Domicilio, Telefono,
RFC, Correo, Porcentaje, Observacion, Estado, ContadorCalificacion, Logotipo, Ciudad)
values (" & txtidCategoria.Text & ",'" & txtNombreNegocio.Text & "','" & txtDomicilio.Text
& "','" & txtTelefono.Text & "','" & txtRFC.Text & "','" & txtCorreo.Text & "','" &
txtPorcentaje.Text & "','" & txtObservacion.Text & "','" & cboEdo.Text & "'," & contador
& ",'" & logo & "','" & txtCiudad.Text & "')"
        Dim Comando As New MySqlCommand(R, Conexion)
        Comando.ExecuteNonQuery()



        Dim Lector As MySqlDataReader
```

```
        Comando.CommandText = "select * from NegociosVinculados"
        Lector = Comando.ExecuteReader
        While Lector.Read()
            DG.Rows.Add(Lector(0), Lector(2), Lector(3), Lector(4), Lector(5),
Lector(6), Lector(7), Lector(9), Lector(12))
        End While
        Lector.Close()
        MsgBox("Guardado correctamente", MsgBoxStyle.OkOnly)
    End Sub

    Private Sub PictureBox2_Click(sender As Object, e As EventArgs) Handles
PictureBox2.Click
        Me.Dispose()
    End Sub

    Private Sub btNuevo_Click(sender As Object, e As EventArgs) Handles btNuevo.Click
        btAceptar.Enabled = True
        btNuevo.Enabled = False
        btModificar.Enabled = False
        btOKI.Enabled = False
        GBCat.Enabled = True
        GBEmpleado.Enabled = True
        GBNeg.Enabled = True
    End Sub

    Private Sub btOKI_Click(sender As Object, e As EventArgs) Handles btOKI.Click
        btOKI.Visible = False
        cboClaveNegocio.Visible = True
        cboClaveNegocio.Enabled = True
        tvCve.Visible = True
        txtNombreNegocio.Visible = False
        btOKI.Enabled = False
        btModificar.Visible = True
        btModificar.Enabled = True
        txtDomicilio.Enabled = False
        txtTelefono.Enabled = False
        txtCiudad.Enabled = False
        txtRFC.Enabled = False
        txtCorreo.Enabled = False
```

```
        txtPorcentaje.Enabled = False
        btAceptar.Visible = False
        btNuevo.Visible = False
        Label15.Visible = False
        Label14.Visible = False
        MsgBox("Solo puedes modificar las observaciones y el estado", MsgBoxStyle.OkOnly)
        Dim Comando As New MySqlCommand("select NegociosVinculados.idNegocioVinculado
from NegociosVinculados", Conexion)
        Dim Lector As MySqlDataReader
        Lector = Comando.ExecuteReader
        While Lector.Read()
            cboClaveNegocio.Items.Add(Lector(0))
        End While
        Lector.Close()
    End Sub

    Private Sub btModificar_Click(sender As Object, e As EventArgs) Handles
btModificar.Click
        ''Actualizar
        DG.Rows.Clear()
        Dim R As String
        R = "update NegociosVinculados set Observacion = '" & txtObservacion.Text & "',
Estado= '" & cboEdo.Text & "' where idNegocioVinculado= " & cboClaveNegocio.Text
        'MsgBox(R)
        Dim Comando As New MySqlCommand(R, Conexion)
        Comando.ExecuteNonQuery()

        Dim Lector As MySqlDataReader
        Comando.CommandText = "select * from NegociosVinculados"
        Lector = Comando.ExecuteReader
        While Lector.Read()
            DG.Rows.Add(Lector(0), Lector(2), Lector(3), Lector(4), Lector(5),
Lector(6), Lector(7), Lector(9), Lector(12))
        End While
        Lector.Close()
        MsgBox("Modificado correctamente", MsgBoxStyle.OkOnly)
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
```

```
        OpenFileDialog1.InitialDirectory = "C:\Users\luki2\OneDrive\Imágenes"
        If OpenFileDialog1.ShowDialog() = Windows.Forms.DialogResult.OK Then
            ImgLogo.Image = Image.FromFile(OpenFileDialog1.FileName)
            tvRuta.Text = OpenFileDialog1.FileName
        End If
    End Sub
End Class
```

10.2 Código para realizar consultas de servicios solicitados de un negocio en sistema Web en el rol de usuario empleado.

```
Imports MySql.Data.MySqlClient
Public Class ServiciosSolicitados
    Inherits System.Web.UI.Page
    Dim auxi, estadoSet As String
    Dim Lectura As MySqlDataReader

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Load
        ConexionGlobal()
        'nombreUsuarioFinal
        'nombreUsuarioFinal

        If Not Page.IsPostBack Then
            txtTotales.Text = "$ 0.00"
            txtNombreEmpleado.Text = nombreUsuarioFinal
            txtEmpresaa.Text = nombreEmpresaFinal
            '''-------------------------------------------------------------------------
-------

            tvCkave.Visible = True
            cboClave.Visible = True
            ''cboEstados.Items.Clear()
            cboEstados.Items.Add("Atendido")
            cboEstados.Items.Add("Pagado")
            cboEstados.Items.Add("Cancelado")
            Conexion.Close()
```

```
            Conexion.Open()


            ''Mostrar Imagen de empresa-----------------------------------------------
-----------------------------------------------
            Dim R As String
            R = "select NegociosVinculados.Logotipo from NegociosVinculados where
NegociosVinculados.Nombre='" & nombreEmpresaFinal & "'"
            Dim comandoLogo As New MySqlCommand(R, Conexion)
            Dim lectorLogo As MySqlDataReader
            lectorLogo = comandoLogo.ExecuteReader
            lectorLogo.Read()
            Dim logo As String
            logo = lectorLogo(0)
            imgEmpresaa.ImageUrl = "\cargas\" & logo
            ''-------------------------------------------------------------------------
-----------------------------------------------
            Conexion.Close()


            ''Calcular el total
            Dim X As String
            Conexion.Open()
            X = "select sum(serviciossolicitados.Precio) from serviciossolicitados inner
join servicios on serviciossolicitados.idServicio= servicios.idServicio inner join
negociosvinculados on servicios.idNegocioVinculado=
negociosvinculados.idNegocioVinculado where serviciossolicitados.idEstado=4 and
negociosvinculados.Nombre='" & nombreEmpresaFinal & "'"
            Dim comandoSuma As New MySqlCommand(X, Conexion)
            Dim lectorSuma As MySqlDataReader
            lectorSuma = comandoSuma.ExecuteReader
            If (lectorSuma.Read()) Then
                ''Convert = lectorSuma(0)
                txtTotales.Text = "$" & lectorSuma(0) & ".00"

            Else
                txtTotales.Text = "$ 0.00"
            End If
            Conexion.Close()
```

```
        End If
        'Conexion.Close()
    End Sub

    Private Sub cboEstados_SelectedIndexChanged(sender As Object, e As EventArgs) Handles
cboEstados.SelectedIndexChanged
        ConexionGlobal()
        If (cboEstados.Text = "Atendido") Then
            claveEstadoFinal = 3

        ElseIf (cboEstados.Text = "Pagado") Then
            claveEstadoFinal = 4

        ElseIf (cboEstados.Text = "Cancelado") Then
            claveEstadoFinal = 5
        End If
    End Sub

    Protected Sub Unnamed3_Click(sender As Object, e As EventArgs)
        Response.Redirect("MiCuenta.aspx")
    End Sub

    Protected Sub Unnamed1_Click(sender As Object, e As EventArgs)
        Response.Redirect("Login.aspx")
    End Sub

    Protected Sub btOK_Click(sender As Object, e As EventArgs) Handles btOK.Click
        ConexionGlobal()
        Dim valor As String
        Dim clave As Integer
        valor = cboClave.Text
        clave = Integer.Parse(valor)

        Dim Comand As MySqlCommand = Conexion.CreateCommand()
        Dim lector6 As MySqlDataReader
        Comand.CommandText = "call ProcePSS (" & clave & ")"
        lector6 = Comand.ExecuteReader
        lector6.Read()
```

```
        txtPrecio.Text = lector6(0)
        estadoSet = lector6(1)
        lector6.Close()

        If estadoSet.Equals("Pagado") Then
            MsgBox("El servicio a sido pagado, no es posible modificarlo",
MsgBoxStyle.OkOnly)
            tvEstados.Visible = False
            cboEstados.Visible = False
            txtPrecio.Visible = False
            tvPrecio.Visible = False
            btActualizar.Visible = False
        ElseIf estadoSet.Equals("Cancelado") Then
            MsgBox("El servicio a sido cancelado, no es posible modificarlo",
MsgBoxStyle.OkOnly)
            tvEstados.Visible = False
            cboEstados.Visible = False
            txtPrecio.Visible = False
            tvPrecio.Visible = False
            btActualizar.Visible = False
        Else
            tvEstados.Visible = True
            cboEstados.Visible = True
            txtPrecio.Visible = True
            tvPrecio.Visible = True
            btActualizar.Visible = True
        End If
    End Sub

    Protected Sub btnConsultar_Click(sender As Object, e As EventArgs) Handles
btnConsultar.Click
        Dim fecha1, fecha2 As String
        fecha1 = txtFecha1.Text
        fecha2 = txtFecha2.Text
        cboClave.Items.Clear()
        ''------------------------------------------------------------------------
--
        Dim Comando6 As MySqlCommand = Conexion.CreateCommand()
        Comando6.CommandText = "select serviciossolicitados.idServicioSolicitado as
Clave, servicios.Nombre as Servicio, usuariosclientes.NombreUsuarioCliente as Cliente,
```

```
usuariosclientes.DomicilioActivo as Domicilio, serviciossolicitados.Tiempo as
Tiempo_estimado, serviciossolicitados.Distancia, estados.DescripcionEdo as Estado,
serviciossolicitados.Fecha, serviciossolicitados.Precio from serviciossolicitados INNER
join servicios on serviciossolicitados.idServicio=servicios.idServicio INNER join
usuariosclientes on
serviciossolicitados.idUsuarioCliente=usuariosclientes.idUsuarioCliente inner join
estados on serviciossolicitados.idEstado=estados.idEstado inner join negociosvinculados
on servicios.idNegocioVinculado=negociosvinculados.idNegocioVinculado where
negociosvinculados.Nombre='" & nombreEmpresaFinal & "' and serviciossolicitados.Fecha
BETWEEN '" & fecha1 & "' and '" & fecha2 & "'"
        DGNegocio.DataSource = Comando6.ExecuteReader
        DGNegocio.DataBind()
        Conexion.Close() ''''''''''
        Conexion.Open()

        Lectura = Comando6.ExecuteReader
        While (Lectura.Read())
            cboClave.Items.Add(Lectura(0))
        End While
        Lectura.Close()
    End Sub

    Protected Sub Menu1_MenuItemClick(sender As Object, e As MenuEventArgs)

    End Sub



    Protected Sub btActualizar_Click(sender As Object, e As EventArgs) Handles
btActualizar.Click
        ConexionGlobal()
        MsgBox(cboEstados.Text)
        If (cboEstados.Text = "Atendido") Then
            claveEstadoFinal = 3

        ElseIf (cboEstados.Text = "Pagado") Then
            claveEstadoFinal = 4

        ElseIf (cboEstados.Text = "Cancelado") Then
            claveEstadoFinal = 5
        End If
```

```
        Dim R As String
        Dim precio As Double
        Dim valorPrecio As String
        valorPrecio = txtPrecio.Text
        precio = Integer.Parse(valorPrecio)
        R = "update ServiciosSolicitados set idEstado = " & claveEstadoFinal & ", Precio
= " & precio & " where idServicioSolicitado= " & cboClave.Text
        'MsgBox(claveEstadoFinal)
        Dim Comando As New MySqlCommand(R, Conexion)
        Comando.ExecuteNonQuery()
        Conexion.Close()



        ''Calcular el total
        Dim X As String
        Conexion.Open()
        X = "select sum(serviciossolicitados.Precio) from serviciossolicitados inner join
servicios on serviciossolicitados.idServicio= servicios.idServicio inner join
negociosvinculados on servicios.idNegocioVinculado=
negociosvinculados.idNegocioVinculado where serviciossolicitados.idEstado=4 and
negociosvinculados.Nombre='" & nombreEmpresaFinal & "'"
        Dim comandoSuma As New MySqlCommand(X, Conexion)
        Dim lectorSuma As MySqlDataReader
        lectorSuma = comandoSuma.ExecuteReader
        If (lectorSuma.Read()) Then
            ''Convert = lectorSuma(0)
            txtTotales.Text = "$" & lectorSuma(0) & ".00"
        Else
            txtTotales.Text = "$ 0.00"
        End If
        Conexion.Close()

        tvEstados.Visible = False
        cboEstados.Visible = False
        txtPrecio.Visible = False
        tvPrecio.Visible = False
        btActualizar.Visible = False

    End Sub
```

```
End Class
```

10.2.1 Código en HTML:

```
<%@ Page Language="vb" AutoEventWireup="false" CodeBehind="ServiciosSolicitados.aspx.vb" Inherits="TocToc_Web_Page.ServiciosSolicitados" %>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"/>
    <meta http-equiv="Expires" content="0" />
   <meta http-equiv="Pragma" content="no-cache" />
   <title></title>
    <style type="text/css">
        .auto-style128 {
            width: 291px;
        }
        .auto-style159 {
           width: 257px;
            height: 71px;
        }
        .auto-style160 {
            width: 104px;
            height: 104px;
        }
        .auto-style185 {
            height: 41px;
            }
        .auto-style193 {
            height: 8px;
            width: 291px;
        }
        .auto-style194 {
            width: 302px;
            height: 8px;
        }
```

```
    .auto-style195 {
        height: 8px;
        width: 270px;
    }
    .auto-style196 {
        height: 8px;
        width: 201px;
    }
    .auto-style197 {
        width: 291px;
        height: 352px;
    }
    .auto-style198 {
        width: 291px;
        height: 256px;
    }
    .auto-style65 {
       width: 100%;
    }
    .auto-style199 {
        margin-left: 36px;
    }
    </style>
 <script type="text/javascript">
    {
        if (history.forward(1))
            location.replace(history.forward(1))
    }
    </script>

<script type="text/javascript">
    {
        if (history.forward(1))
            location.replace(history.forward(1))
    }
    </script>
<link href="styles/jquery-ui.css" rel="stylesheet" />
```

```
    <script src="Scripts/jquery-1.10.2.min.js"></script>
    <script src="Scripts/jquery-ui.js"></script>
        <script>
            $(function () {
                $('#txtFecha1').datepicker(
                    {
                        dateFormat: 'yy/mm/dd',
                        changeMonth: true,
                        changeYear: true,
                        yearRange: '1950:2100'
                    });
            })
        </script>
    <script>
        $(function () {
            $('#txtFecha2').datepicker(
                {
                    dateFormat: 'yy/mm/dd',
                    changeMonth: true,
                    changeYear: true,
                    yearRange: '1950:2100'
                });
        })
        </script>
</head>
<body style="background-color:#faf9f9";>
    <form id="form1" runat="server">
        <div>
            <table class="auto-style65">
        <tr>
            <td class="auto-style128" rowspan="2" style="text-align: center; background-color: #F3F1F2;">
                <img alt="" src="arriba.png" class="auto-style159" />
            </td>
            <td class="auto-style193" style="background-color: #F3F1F2; text-align: right;"></td>
            <td class="auto-style194" style="background-color: #F3F1F2; font-family: 'Yu Gothic Light'; font-size: x-
large; font-weight: bold;">
                <asp:Menu ID="Menu1" runat="server" ForeColor="#1D1D1D" Orientation="Horizontal">
```

```
                <DynamicMenuItemStyle ForeColor="#00BCD5" Font-Bold="True" Font-Size="X-Large" />
                <Items>
                  <asp:MenuItem Text="Consultas" Value="Consultas" ImageUrl="~/lupaR.png">
                    <asp:MenuItem Text="Servicios por periodo" Value="Servicios por estado">
                      <asp:MenuItem Text="Por periodo" Value="Por periodo" NavigateUrl="~/ConsultaPorFecha.aspx" ImageUrl="~/cargas/calendario.png"></asp:MenuItem>
                      <asp:MenuItem Text="Por ciudad" Value="Por ciudad" NavigateUrl="~/ConsultaPorCiudad.aspx" ImageUrl="~/cargas/ciudad.png"></asp:MenuItem>
                      <asp:MenuItem Text="Por código postal" Value="Por código postal" NavigateUrl="~/ConsultaPorCodigoPostal.aspx" ImageUrl="~/cargas/cp.png"></asp:MenuItem>
                    </asp:MenuItem>
                    <asp:MenuItem Text="Servicios solicitados" Value="Servicios solicitados" NavigateUrl="~/ServiciosSolicitados.aspx"></asp:MenuItem>
                  </asp:MenuItem>
                </Items>
              </asp:Menu>
        </td>
        <td class="auto-style195" style="background-color: #F3F1F2"></td>
        <td class="auto-style196" style="background-color: #F3F1F2; font-family: 'Yu Gothic Light'; font-size: x-large; font-weight: bold;">
          <asp:Image ImageUrl="~/cerrar.png" runat="server" CssClass="auto-style201"/>
          <asp:LinkButton Text="Cerrar sesión" runat="server" Font-Underline="False" ForeColor="#1D1D1D" OnClick="Unnamed1_Click" CssClass="auto-style201" />
        </td>
      </tr>
      <tr>
        <td class="auto-style185" colspan="4"></td>
      </tr>
      <tr>
        <td class="auto-style198" style="text-align: center; background-color: #00BCD4; font-family: 'Yu Gothic Light'; font-size: large;">
          <asp:Image ID="imgEmpresaa" runat="server" CssClass="auto-style160" />
          <br />
          <asp:Label Text="Empresa" runat="server" Font-Size="XX-Large" ForeColor="White" Font-Bold="True" ID="txtEmpresaa" />
          <br />
          <asp:Label Text="Usuario" runat="server" ID="txtNombreEmpleado" Font-Size="Large" Font-Bold="True" />
        </td>
        <td colspan="4" rowspan="3" style="position: static; vertical-align: top; text-align: center;">
```

```
<table class="auto-style65" style="font-family: 'Yu Gothic Light'; color: #1D1D1D; font-size: 23px">
    <tr>
        <td class="auto-style94" style="text-align: center; font-family: 'Yu Gothic UI Light'; font-size: 26px; color: #1D1D1D;">
            <br />

            <asp:Label Text="Servicios solicitados de mi empresa" runat="server" Font-Bold="True" Font-Size="XX-Large" Font-Underline="False" ForeColor="#00BCD5" />
            <br />
            <br />
            <asp:Label Text="Rango de fechas entre: " runat="server" Font-Bold="True" Font-Size="X-Large" />
                <asp:TextBox runat="server" ID="txtFecha1" CssClass="auto-style97" style="margin-left: 27px" Font-Bold="True" Font-Size="X-Large" />
                <asp:TextBox runat="server" ID="txtFecha2" CssClass="auto-style97" style="margin-left: 29px" Font-Bold="True" Font-Size="X-Large" />
                <asp:Button Text="Consultar" runat="server" ID="btnConsultar" CssClass="auto-style95" Width="102px" style="margin-left: 26px" Font-Bold="True" Font-Size="Large" />
        </td>
    </tr>
</table>
<br />
<table style="font-family: 'Yu Gothic UI Light'; font-size: medium; text-align: center;" class="auto-style65">
    <tr>
        <td  style="text-align: center; font-family: 'Yu Gothic UI Light'; font-size: 20px;">
            <asp:Label Text="Clave de servicio solicitado a actualizar: " runat="server" ID="tvCkave" Visible="False" Font-Bold="True" Font-Size="X-Large" />
            <asp:DropDownList ID="cboClave" runat="server" Height="34px" Width="143px" Visible="False" Font-Bold="True" Font-Size="X-Large"></asp:DropDownList>
            <asp:Button ID="btOK" runat="server" CssClass="auto-style199" Text="OK" Width="89px" Font-Bold="True" Font-Size="Large" />
        </td>
    </tr>
    <tr>
        <td>
            <br />
            <br />
            <br />
        </td>
```

```
            </tr>
            <tr>
                <td class="auto-style91">
                <asp:GridView ID="DGNegocio" runat="server" CellPadding="4" ForeColor="#1D1D1D" GridLines="None" Width="92%" Font-Size="X-Large" Height="100%" HorizontalAlign="Center" Font-Bold="True">
                    <AlternatingRowStyle BackColor="White" ForeColor="#284775" />
                    <EditRowStyle BackColor="#999999" />
                    <FooterStyle BackColor="#1D1D1D" Font-Bold="True" ForeColor="White" Font-Size="X-Large" />
                    <HeaderStyle BackColor="#00BCD5" Font-Bold="True" ForeColor="White" HorizontalAlign="Left" />
                    <PagerStyle BackColor="#284775" ForeColor="White" HorizontalAlign="Center" />
                    <RowStyle BackColor="#F7F6F3" ForeColor="#333333" HorizontalAlign="Left" VerticalAlign="Middle" />
                    <SelectedRowStyle BackColor="#E2DED6" Font-Bold="True" ForeColor="#333333" />
                    <SortedAscendingCellStyle BackColor="#E9E7E2" />
                    <SortedAscendingHeaderStyle BackColor="#506C8C" />
                    <SortedDescendingCellStyle BackColor="#FFFDF8" />
                    <SortedDescendingHeaderStyle BackColor="#6F8DAE" />
                </asp:GridView>
                    <br />
                    <br />
                    <table style="font-family: 'Yu Gothic UI Light'; font-size: 20px; text-align: center; color: #1D1D1D; " class="auto-style65">
            </table>
            <br />
            <table style="font-family: 'Yu Gothic UI Light'; font-size: 20px; text-align: center; color: #1D1D1D;" class="auto-style65">
                <tr>
                    <td style="text-align: center;">
                        <asp:Label Text="Estado : " runat="server" CssClass="auto-style69" ID="tvEstados" Visible="False" Font-Bold="True" />
                        <asp:DropDownList ID="cboEstados" runat="server" Height="22px" Width="144px" Visible="False" AutoPostBack="True"></asp:DropDownList>
                    </td>
                    <td style="text-align: center;" class="auto-style105">
                        <asp:Label Text="Precio estimado al momento:" runat="server" ID="tvPrecio" Visible="False" Font-Bold="True" />
                        <asp:TextBox ID="txtPrecio" runat="server" Width="128px" Visible="False"></asp:TextBox>
```

```
                </td>
                <td style="text-align: left;">
                    <asp:Button Text="Actualizar" runat="server" ID="btActualizar" Visible="False" Width="126px" Font-Bold="True" Font-Size="Large" />
                </td>
            </tr>
        </table>
            </td>
        </tr>
    </table>
    </td>
</tr>
        <tr>
    <td class="auto-style197" style="text-align: center; font-family: 'Yu Gothic Light'; font-size: x-large; background-color: #F3F1F2;">
        <br />
        <asp:LinkButton Text="Mi cuenta" runat="server" Font-Underline="False" ForeColor="#1D1D1D" OnClick="Unnamed3_Click" Font-Bold="True" Font-Size="XX-Large" />
         <br />
        <br />
        <asp:Label Text="Total en servicios pagados:" runat="server"  Font-Size="X-Large" Font-Bold="True" />
        <br />
        <asp:Label Text="Total" runat="server" ID="txtTotales" Font-Bold="True" ForeColor="#00BCD5" Font-Size="X-Large" />
    </td>
</tr>
        <tr>
    <td class="auto-style198" style="background-image: url('abajo.jpg')"></td>
</tr>
</table>
</div>
</form>
</body>
</html>
```

10.3 Código Android para mostrar las distintas categorías existentes en la base de datos por medio de consumo de Web Services en sistema móvil en el rol de usuario cliente.

```
import android.animation.Animator;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.support.annotation.NonNull;
import android.support.design.internal.NavigationMenuView;
import android.support.design.widget.BottomNavigationView;
import android.support.design.widget.Snackbar;
import android.support.v7.App.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.App.AppCompatDelegate;
import android.support.v7.widget.CardView;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.ViewAnimationUtils;
import android.view.ViewParent;
import android.view.WindowManager;
import android.widget.TextView;
import android.support.v7.widget.RecyclerView.ViewHolder;
import android.widget.Toast;

import com.android.volley.Request;
import com.android.volley.Response;
import com.android.volley.VolleyError;
import com.android.volley.toolbox.StringRequest;
import com.android.volley.toolbox.Volley;
import com.facebook.AccessToken;

import org.json.JSONArray;
import org.json.JSONException;
import org.json.JSONObject;
import org.w3c.dom.Text;

import java.nio.channels.Selector;
import java.util.ArrayList;
import java.util.List;

public class IngresoUsuario extends BaseActivity
{
   private static final String PRODUCT_URL
="http://10.32.150.52:8080/TocToc/consultaDinamicaCategorias.php";
```

```
    RecyclerView recyclerView;
    ProductAdapterCategorias adapter;
    List<ProductCategorias> productList;


    @Override
    int getContentViewId() {
        return R.layout.activity_ingreso_usuario;
    }

    @Override
    int getNavigationMenuItemId() {
        return R.id.cmdInicio;
    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        AppCompatDelegate.setCompatVectorFromResourcesEnabled(true);
        setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
        super.onCreate(savedInstanceState);

        productList= new ArrayList<>();

        recyclerView=(RecyclerView)findViewById(R.id.recyclerView);
        recyclerView.setHasFixedSize(true);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));

        //DATOS DE BD
        loadProducts();
    }
    private void loadProducts()
    {
        StringRequest stringRequest = new StringRequest(Request.Method.GET, PRODUCT_URL, new
Response.Listener<String>()
        {
            @Override
            public void onResponse(String response)
            {
                try
                {
                    JSONArray products = new JSONArray(response);

                    for(int i=0; i<products.length(); i++)
                    {
                        JSONObject productObject= products.getJSONObject(i);

                        String Descripcion= productObject.getString("Descripcion");
```

```
                String Logotipo=productObject.getString("Logotipo");


                ProductCategorias product = new ProductCategorias(Descripcion, Logotipo);
                productList.add(product);

            }
            adapter= new ProductAdapterCategorias(IngresoUsuario.this, productList);
            recyclerView.setAdapter(adapter);

        } catch (JSONException e)
        {
            e.printStackTrace();
        }
      }
    }, new Response.ErrorListener()
    {
      @Override
      public void onErrorResponse(VolleyError error)
      {
        Snackbar.make(recyclerView, error.getMessage(), Snackbar.LENGTH_SHORT).show();
      }
    });
    Volley.newRequestQueue(this).add(stringRequest);
  }
}
```

10.3.1 Código XML en la parte de diseño.

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  xmlns:design="http://schemas.android.com/apk/res-auto"
  xmlns:card_view="http://schemas.android.com/apk/res-auto"
  android:orientation="vertical"
  android:id="@+id/activity_ingreso_usuario"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:background="@color/colorTitulos"
  tools:context="com.example.luki2.toctoc.IngresoUsuario">

  <LinearLayout

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

```
android:orientation="vertical">


<android.support.v4.widget.NestedScrollView
   android:layout_width="match_parent"
   android:layout_height="match_parent">

   <LinearLayout
      android:id="@+id/LayoutContenedor"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:foregroundGravity="right"
      android:orientation="vertical">

      <android.support.v7.widget.CardView
         android:id="@+id/CardNeg5"
         android:layout_width="match_parent"
         android:layout_height="match_parent"
         design:cardCornerRadius="20dp"
         android:layout_marginBottom="10dp"
         android:layout_marginLeft="10dp"
         android:layout_marginRight="10dp"
         android:layout_marginTop="2dp"
         android:clickable="true"
         android:stateListAnimator="@animator/animacion"
         design:cardElevation="8dp">

         <LinearLayout
            android:id="@+id/layoutCard"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:background="#fff"
            android:orientation="horizontal">

            <LinearLayout
               android:id="@+id/layoutImagen"
               android:layout_width="100dp"
               android:layout_height="match_parent"
               android:orientation="vertical">

               <TextView
                  android:id="@+id/tvLogo"
                  android:layout_width="1dp"
                  android:layout_height="1dp"
                  android:visibility="invisible"                    />

               <ImageView
```

```
                android:id="@+id/imgCategoria"
                android:layout_width="90dp"
                android:layout_height="90dp"
                android:layout_gravity="center"
                android:layout_marginLeft="3dp"
                android:layout_marginTop="4dp"                                    />

            </LinearLayout>

            <LinearLayout
                android:id="@+id/layoutTextos"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:layout_marginLeft="5dp"
                android:layout_marginTop="6dp"
                android:orientation="vertical">

                <TextView
                    android:id="@+id/tvCategoria"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:fontFamily="sans-serif-condensed"
                    android:text="Categoria"

android:textAppearance="@style/TextAppearance.Compat.Notification.Info"
                    android:textColor="#434343"
                    android:textSize="24sp"
                    android:textStyle="bold"
                    android:typeface="sans"                                       />

                <RelativeLayout
                    android:id="@+id/layoutStars"
                    android:layout_marginTop="6dp"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:orientation="horizontal">

                    <Button
                        android:id="@+id/btCategoria"
                        android:background="@drawable/campana"
                        android:layout_width="60dp"
                        android:layout_height="40dp"
                        android:layout_alignParentEnd="true"
                        android:layout_alignParentTop="true"
                        android:layout_gravity="center"
                        android:layout_marginBottom="5dp"
                        android:layout_marginEnd="20dp"
```

```
                    android:stateListAnimator="@animator/animacion"
                    android:textColor="@color/colorTextos"                    />
                </RelativeLayout>
            </LinearLayout>
        </LinearLayout>
    </android.support.v7.widget.CardView>
</LinearLayout>
</android.support.v4.widget.NestedScrollView>

</LinearLayout>
</android.support.design.widget.CoordinatorLayout>
```