ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UN PROTOTIPO DE GESTIÓN DE EVENTOS BASADO EN UNA ARQUITECTURA ORIENTADA A SERVICIOS PARA SEGURIDAD PRIVADA

TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y REDES DE INFORMACIÓN

FERNANDO DANIEL CADENA VILLARREAL

fernando.cadena@epn.edu.ec

DIRECTOR: RAÚL DAVID MEJÍA NAVARRETE, M.Sc.

david.mejia@epn.edu.ec

Quito, julio 2019

AVAL

Certifico que el presente trabajo fue desarrollado por Fernando Daniel Cadena Villarreal, bajo mi supervisión.

RAÚL DAVID MEJÍA NAVARRETE
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo, Fernando Daniel Cadena Villarreal, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración dejo constancia de que la Escuela Politécnica Nacional podrá hacer uso del presente trabajo según los términos estipulados en la Ley, Reglamentos y Normas vigentes.

FERNANDO DANIEL CADENA VILLARREAL

DEDICATORIA

Dedico este trabajo a mis padres Robert y Mery quienes estuvieron pendientes desde muy pequeño y anhelaban ver finalmente que su hijo culmine una etapa de su vida.

A mi hermana Melanie a quien le tengo un gran cariño por apoyarme y exigirme inconscientemente ser un ejemplo para ella.

A mi abuelito Hernán que admiro mucho por ser una persona de lucha y que a pesar de sus años sigue esforzándose día a día.

AGRADECIMIENTO

A Dios por bendecirme cada día y guiarme por el camino correcto.

A mis padres por brindarme la oportunidad de estudiar la carrera que deseaba, por su apoyo incondicional en cada momento de dificultad, y por mostrarme con el ejemplo a perseverar para lograr mis objetivos.

A mi director de Trabajo de Titulación Ing. David Mejía MSc. por su paciencia, orientación y guía en el desarrollo de este trabajo de titulación.

A mis amigos con los cuales compartí grandes momentos y lecciones.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE CÓDIGOS

RESUMEN

Este Trabajo de Titulación presenta un prototipo de gestión de eventos basado en una arquitectura orientada a servicios para seguridad privada, como una solución a los problemas de almacenamiento y procesamiento de eventos registrados por emergencias de los sistemas de alarma.

El prototipo está conformado por: una base de datos, un servicio web WCF (*Windows Communication Foundation*), una aplicación Android y una aplicación de escritorio.

En la base de datos se almacena toda la información sobre los eventos, usuarios, personas y clientes implicados en las emergencias de los sistemas de alarma. El servicio web WCF ofrece funcionalidades a las aplicaciones mediante interfaces; permitiendo que una interfaz maneje las solicitudes de la aplicación Android y otra interfaz maneje las solicitudes de la aplicación de escritorio.

La aplicación Android dispone de mecanismos para leer códigos QR (*Quick Response*) mediante la cámara del dispositivo Android y, emplea el GPS (*Global Positioning System*) para capturar la ubicación del dispositivo, dichos mecanismos permiten realizar el registro de eventos. La aplicación de escritorio para Windows provee mecanismos para la gestión de eventos, usuarios, personas y clientes.

Este documento se encuentra organizado de la siguiente manera:

En el primer capítulo se presenta un marco teórico sobre: los modelos arquitectónicos, la metodología de desarrollo de software SCRUM, el modelo relacional de las bases de datos, se define brevemente a los servicios web WCF, a los códigos QR y GPS, para finalizar, el desarrollo de aplicaciones en Android y en Windows. El segundo capítulo presenta el proceso actual en casos de emergencias, las historias de usuario, el diseño e implementación del prototipo. El tercer capítulo presenta los resultados de las pruebas de funcionamiento y las encuestas de validación efectuadas con el personal de la compañía SEDYM. El cuarto capítulo presenta las conclusiones y recomendaciones obtenidas a lo largo del desarrollo de este Trabajo de Titulación.

Finalmente, como anexos se incluyen: la ficha de la entrevista para la obtención de requerimientos, las historias de usuario, el *script* para la creación de la base de datos, el proyecto de la aplicación de escritorio, el proyecto de la aplicación Android, el proyecto del servicio web WCF, la encuesta de validación y los manuales de usuario.

PALABRAS CLAVE: Android, Códigos QR, GPS, SCRUM, WCF, Windows Forms

ABSTRACT

The current Project develops a prototype of management events based on service-oriented architecture of the private security that presents a solution to the problems of storage and processing of events entered by emergency alarm systems.

The prototype consists of: a database, a WCF (Windows Communication Foundation) web service, an Android application, a desktop application for Windows.

In the database, all information about events, users, people and clients involved in the emergencies of the alarm system is stored. The WCF web service offers functionality to applications through interfaces, allowing one interface that handles the requests of the Android application and another interface that handles the requests of the desktop application.

The Android application has mechanisms to read QR (Quick Response) codes through the camera of the Android device and, using GPS (Global Positioning System) to capture the location of the device, these mechanisms allow to record events. The desktop application for Windows provides mechanisms for the management of events, users, people and clients.

This document is organized as follows:

In the first chapter presents concepts about the architectural models, the software development methodology SCRUM, the relational model of databases, WCF web services ares defined, then QR codes and GPS. Finally, it is about the development applications on Android and Windows. The second chapter presents the current process in cases of emergencies, the user stories, the design and implementation of the prototype. The third chapter presents the results of the performance tests and the validation surveys are realized to the personnel of the SEDYM company. The fourth chapter presents the conclusions and recommendations obtained throughout the development of this Degree work.

Finally, the appendices include: the interview form for obtaining requirements, the user stories, the script for creating the database, the desktop application project, the Android application project, the WCF web service project, the validation survey and user manuals.

KEYWORDS: Android, QR Codes, GPS, SCRUM, WCF, Windows Forms

1. INTRODUCCIÓN

El sector de la seguridad tiene una tendencia hacia la integración con los recursos tecnológicos, lo cual cambia el espectro de servicios de la Seguridad Privada [1].

Los recursos tecnológicos como las tecnologías móviles se han convertido en un sector a tomar en cuenta ya que han posibilitado facilitar actividades cotidianas que antes se realizaban de forma manual y que actualmente pueden ser automatizadas; por otro lado, los servicios web han permitido un manejo estructurado de la información, logrando que las aplicaciones no manipulen directamente los datos.

Al apoyarse en tecnologías móviles y sobre estas en el desarrollo de aplicaciones Android, se puede contar con mecanismos de escaneo de códigos QR[1] y GPS[2], que en conjunto servirán en el presente trabajo para realizar un ingreso de datos.

El presente Proyecto Integrador se enfoca en el desarrollo de un sistema distribuido conformado por: una base de datos SQL (*Structured Query Language*)[3] que almacena toda la información del prototipo, un servicio web WCF[4], una aplicación Android y una aplicación de escritorio.

Este sistema distribuido[5] proporcionará información sobre eventos ingresados por emergencias en los sistemas de alarma[6], mediante el cual se podrán registrar los eventos en una aplicación Android y, gestionar (ingresar, modificar, eliminar y visualizar) la información a través de la aplicación de escritorio.

1.1 OBJETIVOS

El objetivo general de este Proyecto Integrador es: desarrollar un prototipo de gestión de eventos basado en una arquitectura orientada a servicios para seguridad privada.

Los objetivos específicos de este Proyecto Integrador son:

a) Analizar el proceso actual que se realiza en caso de eventos de sistemas de alarma.

[1] **Código QR:** permite almacenar información como texto, enlaces a páginas web, los cuales se pueden exponer de manera impresa o en pantalla.
[2] **GPS:** sistema que indica la posición de un dispositivo, persona u vehículo sobre la tierra.
[3] **SQL:** lenguaje estándar para el almacenamiento, manipulación y recuperación de información de bases de datos relacionales.
[4] **WCF:** tecnología de Microsoft que permite intercambiar datos entre varios softwares.
[5] **Sistemas Distribuidos:** sistemas con componentes de hardware y software que se comunican a través de una red.
[6] **Sistemas de Alarma:** serie de equipos electrónicos que se encargan de advertir situaciones de peligro.

b) Diseñar los componentes que conformarán el prototipo sobre eventos de sistemas de alarma: aplicación Android, aplicación de escritorio, servicio web y servidor de base de datos.

c) Implementar los componentes del prototipo sobre eventos de sistemas de alarma.

d) Analizar los resultados de las pruebas ejecutadas en el prototipo de gestión de eventos.

1.2 ALCANCE

El Presente Proyecto Integrador está conformado por los siguientes componentes: una base de datos, un servicio web WCF, una aplicación Android y una aplicación de escritorio para Windows. En la Figura 1.1 se muestra un esquema del prototipo.

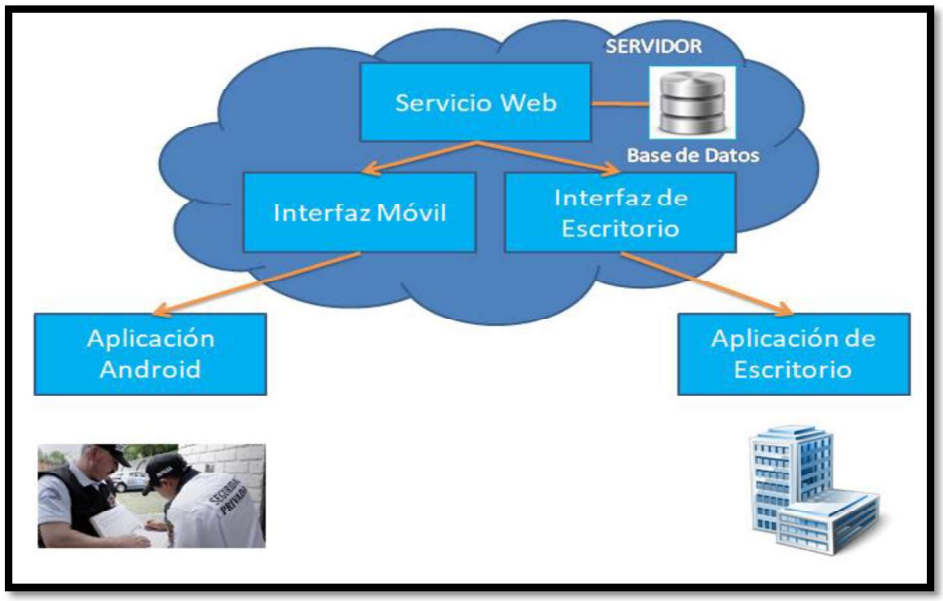


Figura 1.1 Esquema del prototipo

La base de datos SQL será la encargada de almacenar la información de los usuarios, personas, clientes, y eventos registrados por emergencias de los sistemas de alarma.

Por usuarios, se refiere al personal de la compañía de seguridad, las personas son los propietarios o autorizados de los inmuebles, los clientes hacen referencia a los inmuebles.

Los eventos almacenarán la información asociada a la asistencia del personal de la compañía de seguridad a un inmueble por emergencia médica, pánico o robo.

Los usuarios una vez autenticados podrán utilizar las aplicaciones Android y de escritorio.

El servicio web WCF ofrece funcionalidades a las aplicaciones mediante dos interfaces: una interfaz que manejará las peticiones de la aplicación Android y una interfaz que manejará las peticiones de la aplicación de escritorio en Windows.

La aplicación Android permitirá que el personal de una compañía de seguridad ingrese eventos por emergencias en los sistemas de alarma, para lo cual se realizará un mecanismo de ingreso de eventos, a través de este mecanismo el personal de la compañía de seguridad podrá registrar eventos a la base de datos cuando se produzcan distintas emergencias en la propiedad del cliente.

Adicionalmente en la aplicación Android se utilizarán dos mecanismos para validar la presencia del personal de la compañía de seguridad en la propiedad del cliente:

- El primer mecanismo se basará en el escaneo de un código QR. Se generarán dos códigos QR por cada cliente, que estarán ubicados en la propiedad del cliente. Estos códigos QR servirán para validar que el registro del evento sea perteneciente al cliente indicado por la aplicación.
- El segundo mecanismo se basará en capturar la información sobre la posición reportada por el GPS del dispositivo celular y corroborar con la ubicación almacenada de cada cliente en la base de datos [2].

La aplicación de escritorio para Windows permitirá ingresar, modificar, eliminar y visualizar: usuarios, personas, clientes y eventos.

Acerca de los eventos se podrán realizar reportes en la aplicación de escritorio según distintos parámetros como periodos de tiempo, por clientes, por personas autorizados, también por tipos de eventos, estados de los eventos, usuarios, como resultado el prototipo generará un archivo con la información almacenada en la base de datos.

Para establecer los requerimientos que el prototipo deberá cumplir se realizarán entrevistas al personal de la compañía SEDYM CIA LTDA. De igual manera se realizarán encuestas de evaluación de uso al personal de la misma compañía para conocer la satisfacción al emplear el prototipo.

1.3 MARCO TEÓRICO

En esta sección se presentan conceptos empleados en el desarrollo del prototipo de gestión de eventos, que incluye la definición de las arquitecturas orientada a servicios y cliente/servidor, la metodología de desarrollo de software SCRUM, las bases de datos SQL y los servicios web WCF [3].

Además, se presenta de forma resumida la estructura de los códigos QR y la captura de la ubicación a través del GPS. Por último, se trata sobre el desarrollo de aplicaciones en Android y de escritorio en Windows.

1.3.1 MODELOS ARQUITECTÓNICOS

Los modelos arquitectónicos describen la manera en que los componentes de los sistemas interactúan entre ellos, así como su vinculación a la red de computadores. Un modelo arquitectónico de un sistema distribuido permite una abstracción de las funciones de los componentes de los sistemas.

Un ejemplo de cómo se logra en principio esta abstracción es al clasificar los procesos entre servidores y clientes, misma clasificación indica qué responsabilidades realiza cada uno, lo cual permite la valoración del trabajo y el impacto en caso de fallos [3].

Estilo Arquitectural: Un estilo arquitectural o patrón arquitectónico, es un conjunto de principios que dan forma a una aplicación.

Según [4] el beneficio más relevante es que los estilos arquitectónicos proveen un lenguaje común, permiten las comunicaciones independientemente de las tecnologías, que incluyen patrones y principios junto con un grupo de restricciones, pero sin entrar en detalle sobre las tecnologías a emplear.

Entre algunos estilos arquitecturales se encuentran SOA (*Service Oriented Architecture)* y Cliente-Servidor.

Estilo SOA: La arquitectura orientada a servicios es capaz de proveer las funcionalidades de una aplicación como una forma de servicio a través de una red de comunicaciones, y permite a las aplicaciones, consumir la funcionalidad como un servicio usando contratos y mensajes.

Los servicios tienen una débil asociación porque estos ocupan interfaces basadas en estándares, las cuales pueden ser invocadas, publicadas y descubiertas [4].

Los servicios en SOA se fundamentan en brindar un esquema y una interacción por medio de mensajes con una aplicación mediante el uso de interfaces, dejando de lado los componentes y objetos.

Se debe mencionar que los usuarios u otros servicios son capaces de acceder a servicios locales o servicios remotos mediante una red de comunicaciones.

Los principios del estilo arquitectónico SOA se detallan a continuación [4]:

- Servicios Distribuibles: Los servicios son libres de ubicarse en cualquier lugar de una red, local o externa, tomando en cuenta siempre que deben contarse con los protocolos de comunicación preestablecidos.
- Servicios con débil relación: Por su característica de autonomía, se puede aseverar que los servicios son capaces de ser actualizados sin afectar a las aplicaciones que lo usan, siempre y cuando siga existiendo la compatibilidad preestablecida.
- Servicios de compartición de esquema y contrato: Para comunicarse los servicios comparten esquemas y contratos, mas no clases descritas en las aplicaciones.

El estilo arquitectónico SOA es adecuado en ambientes donde se debe permitir una comunicación por medio de mensajes entre aplicaciones independientemente de la plataforma en que estos funcionen.

Entre las aplicaciones que usan SOA están: aplicaciones para la compartición de información, manejo de procesos o servicios específicos de una industria y combinación de información con múltiples fuentes de origen.

Estilo Cliente/Servidor [4]**:** El estilo cliente/servidor describe a los sistemas distribuidos que comprenden a un sistema separado el cliente del servidor, los cuales se conectan a través de una red de comunicaciones.

De manera general el estilo arquitectural cliente/servidor define las relaciones entre uno o varios clientes y uno o varios servidores, en donde los clientes inician una o varias peticiones, esperan sus respuestas y procesan las respuestas.

El servidor por otro lado autoriza al usuario, procesa las peticiones y retorna un resultado.

Además, es capaz de enviar respuestas en varios protocolos y formatos de datos para comunicar al cliente.

Algunos ejemplos que utilicen el estilo cliente/servidor son: un servidor de impresión, un servidor web y un cliente que realice solicitudes por un navegador.

1.3.2 METODOLOGÍA DE DESARROLLO SCRUM

SCRUM no corresponde a siglas, sino proviene de un concepto deportivo utilizado en el rugby, vinculado a la rápida recuperación del juego ante una infracción menor [5].

Es una metodología de desarrollo ágil que tiene un enfoque de colaboración de equipos en proyectos, en la cual se definen artefactos y roles que forman la estructura mínima para lograr un adecuado desempeño [6].

SCRUM emplea una teoría de control empírico con tres ejes principales: transparencia, inspección y adaptación [6]:

- Transparencia: Provee una garantía en el desarrollo del proceso sobre las cosas que afecten el producto final.
- Inspección: Apoya a descubrir cambios no deseados del proceso.
- Adaptación: Establece ajustes necesarios para reducir la afectación de los cambios en el proceso.

1.3.3 ROLES SCRUM

En la Tabla 1.1 se presentan los roles y su descripción según SCRUM [7].

Tabla 1.1 Roles de la metodología SCRUM

Rol	Descripción
Product Owner (PO)	Representa a las personas con interés en desarrollar el proyecto y obtener un producto. Entre sus obligaciones están: establecer los requerimientos del proyecto a desarrollar, acordar los cambios en los requerimientos y prioridades de entrega del proyecto, así como validar el producto final.
SCRUM *Master* (SM)	Es quien dirige el proyecto. Responsable de impartir el proceso SCRUM, a todos los involucrados en el proyecto, de ser necesario. Comprueba que se sigan las prácticas de SCRUM. Sus principales cargos son: revisar el estado de las tareas, detectar conflictos y problemas que detengan el desarrollo de SCRUM y solucionar los mismos.
SCRUM *Team* (ST)	Encargados de transformar los requerimientos en funcionalidades del proyecto.

1.3.4 EVENTOS SCRUM

Los eventos en SCRUM se utilizan con el fin de realizar reuniones entre los distintos involucrados del proyecto.

En las reuniones se definen objetivos, se inspecciona los avances, se realizan cambios, todo esto con el fin de mejorar el desarrollo de un proyecto.

Sprint: En SCRUM los trabajos generalmente se planifican en ciclos denominados *sprints* que se definen como iteraciones a desarrollarse en breves periodos de tiempo.

El ST gestiona sobre un conjunto de funcionalidades, cuáles serán asignadas a cada *sprint* anteponiendo las funciones de mayor necesidad, para su realización al principio del proyecto.

Al dividir en *sprints* a proyectos grandes se logra distribuir el trabajo, permitiendo hacer más discernible el desarrollo. Al terminar un *sprint* se debe obtener un producto de *software* entregable al PO [7].

Un *sprint* consiste de los siguientes elementos [6]:

- Planeación del *Sprint*: Para realizar la planeación del *sprint* se realiza una reunión, entre todos los involucrados del proyecto con la intención de definir parámetros acerca de los entregables y su realización, más específicamente del diseño del sistema y la aproximación sobre la cantidad de trabajo.
- Reunión Diaria del *Sprint*: Reunión que dura pocos minutos, se realiza como herramienta de apoyo para conocer los avances, los problemas presentados y las funcionalidades a realizarse en días posteriores.
- Revisión del *Sprint*: La revisión del *sprint* se realiza al concluir con el *sprint* de manera que el PO pueda conocer qué funcionalidades se realizaron, y cuáles no; el ST da a conocer los problemas y las soluciones que se han usado para el avance del proyecto.
- Retrospectiva del *Sprint*: La retrospectiva del *sprint* es una reunión interna del ST, para identificar cómo estuvo la comunicación, el proceso y las herramientas en el *sprint* finalizado, con el fin de generar un plan de mejora para el próximo *sprint*.

1.3.5 ARTEFACTOS SCRUM

Son subproductos de las tareas que proveen una dirección y transparencia al ST; entre ellos están [7]:

Product Backlog: El *Product Backlog* es un documento de alto nivel como guía que tiene una lista de tareas que deben realizarse para el desarrollo de un proyecto.

Cuenta con la información general acerca de las funcionalidades de un sistema, además puede contener información acerca del código y nombre de la historia de usuario. Muchas veces se encuentra ordenado por los valores como prioridad o esfuerzo de cada tarea.

Es dinámico y se modifica conforme se presenten nuevas necesidades del producto. El PO es el encargado de definir el contenido.

Sprint Backlog: Es una lista de actividades que se desarrollarán en un *sprint*, determina que actividades se ejecutan por cada persona del ST y el periodo de tiempo en finalizarlas. Al hacer uso del *sprint backlog* en un proyecto, es más discernible encontrar las actividades con retardo de tiempo o problemas, con el fin de resolverlos en el menor tiempo posible.

1.3.6 ARTEFACTOS EMPLEADOS EN LAS METODOLOGÍAS DE DESARROLLO DE SOFTWARE

Es importante definir las historias de usuario utilizadas ampliamente para el desarrollo de software.

Historias de Usuario: Son especificaciones de las funcionalidades de un proyecto, producto de la cooperación entre el PO y el SM, con la implementación se contribuye al PO.

Se caracterizan por ser lo suficientemente delimitadas para poder implementarlas, detallan las exigencias del PO.

Existen varias maneras de definir una historia de usuario. Por lo cual se utiliza una estructura de las historias de usuario en las que se pueden contar con ítems como: el código de la historia, el actor, el desarrollador, el nombre del *sprint*, el nombre de la historia de usuario, el número de *sprint* en el que se realizará la historia, breve descripción de la historia de usuario y los comentarios de funcionalidades adicionales.

- Código: Código único que identifica a las historias de usuario. En la Tabla 1.2 se presenta una descripción más detallada del código de la historia de usuario.
- Actor: Nombre de la persona que podrá ejecutar el requerimiento descrito en la historia de usuario.
- Desarrollador: Nombre de la persona que desarrollará el requerimiento descrito en la historia de usuario.
- *Sprint*: Define el nombre del *sprint*, permite asociar los requerimientos del prototipo, de tal forma que el desarrollador encuentre actividades comunes en cada uno de los *sprints*.
- Nombre: Detalle del nombre de la historia de usuario.
- Número de *Sprint:* Número del *sprint* en el que se desarrollará la historia de usuario.
- Descripción: Presenta la definición de lo que realiza la historia de usuario.

- Comentarios: Indica información complementaria para el desarrollo de la historia de usuario y funcionalidades adicionales.

Tabla 1.2 Descripción del campo Código de la Historia de Usuario

HU	01
Abreviación utilizada para describir a la Historia de Usuario.	Número de la Historia de Usuario.

En la Tabla 1.3 se presenta un ejemplo de una historia de usuario [8].

Tabla 1.3 Ejemplo de Historia de Usuario

Código: HU-00	**Actor:** Administrador
Desarrollador: Fernando Cadena	**Sprint:** Gestión de Usuarios
Nombre: Ingresar Usuarios	**Número de Sprint:** 1
Descripción: Como administrador quiero poder ingresar la información de los usuarios como: nombre, apellido, alias[7], contraseña, correo y tipo de usuario, para establecer los usuarios que ingresarán al sistema.	
Comentarios: Se requiere que los alias tengan un valor único.	

1.3.7 BASES DE DATOS

Las Bases de Datos son maneras de almacenar información en archivos de una computadora [9].

Es una colección estructurada de datos, que tienen una definición común. Las bases de datos se diseñan para cumplir con ciertos requerimientos de información de una empresa, organización o persona.

Proveen funciones útiles como la compartición, la integridad y eliminación de la duplicidad de la información, otras funcionalidades de usar bases de datos son: la manipulación de la

[7] **Alias:** Nombre alternativo que se le da a una persona.

información, el acceso a la información por varios usuarios y el acceso de manera simultánea [9] [10].

1.3.8 SISTEMAS DE GESTIÓN DE BASES DE DATOS

Un Sistema de Gestión de Bases de Datos (SGBD) consiste en una agrupación de datos relacionados entre sí, como también de un grupo de programas que accederán a los datos. El objetivo de usar un SGBD es brindar un almacenamiento y recuperación eficiente acerca de la información de la base de datos.

Está diseñado para soportar y gestionar una gran cantidad de información, la gestión implica desde el almacenamiento de información hasta mecanismos para el manejo de la información [9].

Entre algunos gestores de bases de datos se encuentran: Microsoft SQL Server, Microsoft Access, Oracle, PostgreSQL, MySQL y varios más.

1.3.9 MICROSOFT SQL SERVER

SQL Server es un sistema de base de datos profesional del modelo relacional desarrollado por Microsoft. Es un sistema que administra y analiza bases de datos relacionales [11].

SQL Server emplea el lenguaje estándar SQL, es capaz de soportar múltiples conexiones de clientes. Utiliza SQL Server Management Studio como herramienta de administración y desarrollo de aplicaciones aprovechando su interfaz gráfica [12].

1.3.10 MODELO DE DATOS

El modelo de datos es parte principal en la estructura de una base de datos, el cual define la descripción de los datos, relaciones, semántica y ligaduras de consistencia.

Existen modelos de datos tales como: modelo entidad-relación y modelo relacional.

Modelo entidad-relación: El modelo entidad-relación (E-R) realiza una correspondencia entre el mundo real compuesto por un conglomerado de objetos básicos denominados entidades y las relaciones existentes entre objetos.

Las entidades son objetos del mundo real distinguibles de otros objetos, y para describirlos de mejor manera se utilizan un conjunto de atributos. Por otro lado las relaciones son asociaciones entre algunas entidades [9].

En la Figura 1.2 se muestra un ejemplo práctico de un diagrama entidad-relación. Los componentes del diagrama entidad-relación son:

- Rectángulos: Representan una entidad.
- Elipses: Representan atributos.
- Rombos: Representan relaciones entre entidades.
- Líneas: Concatenan atributos con entidades y entidades con relaciones.

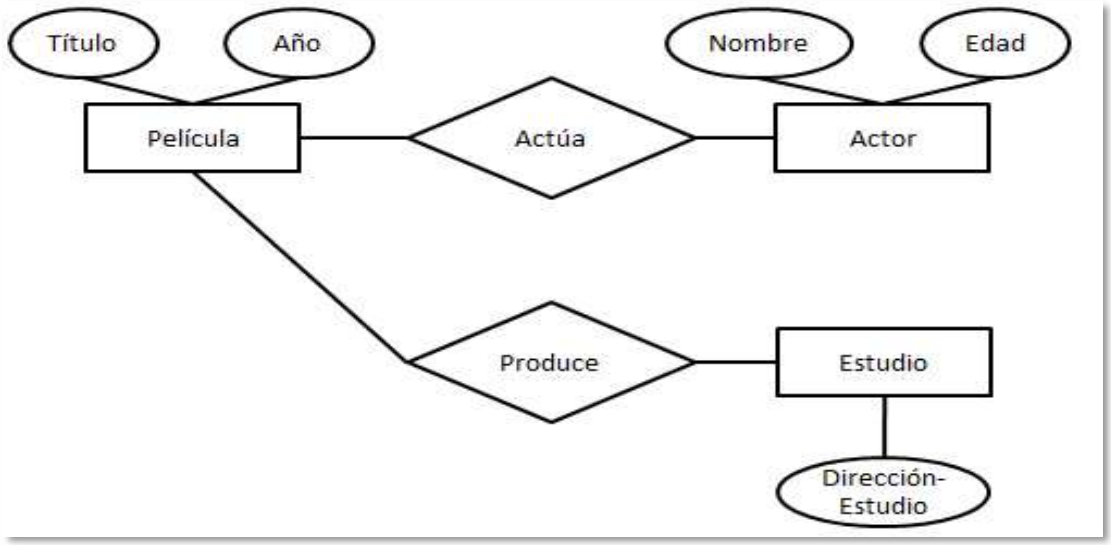


Figura 1.2 Ejemplo de un diagrama entidad-relación

Modelo relacional: Modelo ampliamente utilizado por su consistente manera de presentar los datos. Se forma por un conjunto de tablas, cada una con un nombre distinto.

Una tabla es un conjunto de datos representado en una matriz bidimensional, las tablas contienen un conjunto de filas. Una fila tiene un conjunto de valores, que forman una entrada de la tabla.

En la Figura 1.3 se presenta el ejemplo de una tabla llamada `Cuenta` donde se pueden observar las filas compuestas de los valores: `número-cuenta`, `nombre-sucursal` y `saldo`.

número-cuenta	*nombre-sucursal*	*saldo*
C-101	Centro	500
C-102	Navacerrada	400
C-201	Galapagar	900
C-215	Becerril	700
C-217	Galapagar	750
C-222	Moralzarzal	700
C-305	Collado Mediano	350

Figura 1.3 Tabla cuenta [9]

1.3.11 ACCESO A DATOS

Muchas aplicaciones funcionan sobre bases de datos. Por tal motivo, es importante trabajar con tecnologías de acceso a datos, una de ellas es Entity Framework.

Entity Framework: Entity Framework ofrece la posibilidad de crear aplicaciones que accedan a bases de datos con un mayor nivel de abstracción llevando al nivel conceptual.

En este nivel Entity Framework acepta código independientemente del motor de almacenamiento que provenga de esquemas relacionales [13].

El fin de llegar a una programación conceptual es minimizar las diferencias entre modelos orientados a objetos y modelos relacionales, para esto se recurre a un patrón de diseño para el modelado de datos que se divide en tres partes [13]:

- Modelo conceptual: Define entidades y relaciones del sistema modelado.
- Modelo lógico: Describe las bases de datos relacionales: sus tablas, relaciones, columnas, etc.
- Modelo físico: Define información específica sobre almacenamiento, índices en base al motor de bases de datos aplicado.

Es posible utilizando LINQ (*Language-Integrated Query*) o más específicamente LINQ to Entities conocer las entidades del modelo conceptual de Entity Framework.

LINQ: Es un lenguaje de consulta integrado, que posibilita expresar operaciones de consulta en lenguajes de programación como C# sobre datos de distintas fuentes como: objetos en memoria, bases de datos relacionales o documentos XML.

Una consulta se refiere a una expresión que toma información de un origen de datos. Las operaciones de consulta en LINQ se componen de las siguientes acciones [13]:

- Creación de los datos: El Código 1.1 presenta el ejemplo para la creación de una lista de datos; en la línea 19 se crea un objeto de tipo lista `Usuario` que almacenará los usuarios; en las líneas 20 y 21 se agregan dos nuevos usuarios a la lista, con los siguientes atributos (de izquierda a derecha): `id`, `nombre`, `alias` y `contraseña`.

```
19  List<Usuario> usuarios = new List<Usuario>();
20  usuarios.Add(new Usuario(1, "Daniel", "Dan", "987654321"));
21  usuarios.Add(new Usuario(2, "Luis", "luis", "987654321"));
```

Código 1.1 Creación de una lista de datos

- Creación de la consulta: El Código 1.2 muestra el ejemplo donde se realiza la consulta usando LINQ; la línea 24 almacena en una variable la información de la consulta; la línea 25 usa la cláusula `from` para definir la lista de datos; la línea 26 utiliza la cláusula `where` para definir los parámetros de la consulta; la línea 27 emplea la cláusula `select` para escoger los elementos a retornar de la consulta.

```
24  var consultaUsuario =
25                  from variableauxiliar in usuarios
26                  where variableauxiliar.Nombre == "Luis"
27                  select variableauxiliar;
```

Código 1.2 Creación de la consulta

- Resultado de la consulta: El Código 1.3 mostrará el resultado de la información de la consulta; la línea 29 utiliza la instrucción `foreach` para recorrer los datos de la consulta anterior, el valor de cada usuario se lo captura mediante la variable de iteración `usuario`; la línea 30 mostrará por consola el atributo `Nombre` de los usuarios.

```
29  foreach (var usuario in consultaUsuario) {
30      Console.WriteLine(usuario.Nombre);
31  }
```

Código 1.3 Resultado de la consulta

1.3.12 SERVICIOS WEB

Los servicios web proveen un estándar de interoperabilidad entre diferentes aplicaciones, que pueden ejecutarse en una variedad de plataformas o *frameworks*[8]. Las aplicaciones intercambian datos para ofertar servicios, y permitir presentar información de manera dinámica a los clientes. El WC3 (*World Wide Web Consortium*)[9] define a los servicios web como un sistema que soporta interacciones máquina a máquina mediante una red [14].

Los estándares de los servicios web se componen de:

- XML (*eXtensible Markup Language XML*): Es un formato estándar que se basa en texto y representa información de manera estructurada como: documentos, datos, configuración, transacciones, facturas, entre otras. Diseñado para almacenar y transportar datos, legibles por humanos y máquinas, cuenta con un amplio uso en la distribución de datos a través de Internet.

[8] **Frameworks**: Estructura de software de componentes modificables para el desarrollo o implementación de una aplicación.
[9] **WC3:** Comunidad internacional que desarrolla estándares web.

- SOAP (*Simple Object Access Protocol*): Protocolo liviano, se basa en XML, para el paso de información en ambientes distribuidos. Proporciona una funcionalidad de empaquetar mensajes, y se ubica en el núcleo de los servicios web. En el Código 1.4 se presenta el esqueleto de un mensaje SOAP, que cuenta con un elemento raíz `<Envelope>`, la cabecera `<Header>` utilizada para llevar información relacionada con la aplicación, el cuerpo `<Body>` que abarca la información para el destinatario final y un subelemento `<Fault>` para notificación de errores.

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/..."
 soap:encodingStyle="http://www.w3.org/...">

  <soap:Header>
    información especifica de la aplicación, como la autenticación
  </soap:Header>

  <soap:Body>
    mensaje destinado al punto final de la comunicación
    <soap:Fault>
      para indicar mensajes de error
    </soap:Fault>
  </soap:Body>

</soap:Envelope>
```

Código 1.4 Esqueleto de un mensaje SOAP [13]

- WSDL (*Web Services Definition Language*): Permite realizar una descripción estructurada de las comunicaciones, definiendo una gramática XML para efectuar dicha descripción de servicios web como colecciones de extremos de comunicación que intercambian mensajes. Los solicitantes del servicio utilizan a WSDL para enlazar con el servicio web.
- UDDI (*Universal Discovery, Description and Integration*): es un registro que permite a los solicitantes de servicio localizar servicios web. Se utiliza los servicios UDDI para publicar, descubrir, compartir o interactuar con los servicios web [15].

En la Figura 1.4 se muestra un ejemplo de la integración de los estándares de los servicios web.

Una vez finalizadas las funcionalidades de un servicio web, se necesita que estás sean consumidas por los usuarios, para ello se realiza el registro utilizando WSDL en un repositorio UDDI, posterior a ello, el usuario hará búsquedas en el repositorio hasta

localizar el WSDL publicado, es posible hacer la petición SOAP (XML) vía HTTP (*Hypertext Transfer Protocol*)[10] al servicio web, el servicio web retorna una respuesta SOAP [15].

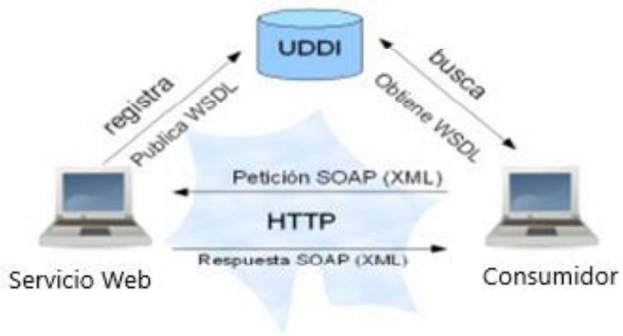


Figura 1.4 Dinámica de un Servicio Web [15]

Tipos de Servicios Web: Un servicio web es un componente de software, al cual se puede acceder mediante una red. Para implementar estos servicios se diferenciará entre dos tipos de servicios web [14]:

- Servicios web basados en SOAP (*Simple Object Access Protocol*): Los servicios web SOAP, o servicios web *big*, emplean mensajes XML para la comunicación bajo el protocolo SOAP, protocolo que define la comunicación de datos entre un cliente y servicio de distintos procesos. La utilización de un servicio SOAP es más estructurada, se utilizan contratos que describen la interfaz ofertada por el servicio web, WSDL describe las especificaciones del contrato, también hay que observar los requerimientos aplicables no funcionales como son: transacciones, estados, seguridad y coordinación [14].
- Servicios Web basados en RESTful (*Representational State Transfer Web Services*): Los servicios Web RESTful son aplicables en ambientes con un gran número de clientes, debido al balanceo de carga, el soporte para el almacenamiento caché y la posibilidad de transmitir mensajes ligeros con la notación JSON (*JavaScript Object Notation*)[11]. Se adaptan más fácilmente a HTTP que los servicios SOAP, RESTful provee un tipo de abstracción que no utiliza mensajes XML o WSDL. Estos servicios web RESTful se pueden consumir de manera más ágil, por ejemplo, a través de un navegador web [16].

[10] **HTTP:** Protocolo de comunicación para sistemas de información que puede usarse a través de Internet.
[11] **JSON:** Formato de texto ligero de intercambio de datos, que es independiente de un lenguaje de programación.

1.3.13 WINDOWS COMMUNICATION FOUNDATION (WCF)

WCF es un *framework* que facilita la comunicación de distintas aplicaciones en varias plataformas. Un servicio trata sobre una aplicación que envía y recibe mensajes con varias aplicaciones, por lo que resulta ideal aplicarlo en ambientes distribuidos. Debido a que los servicios son autónomos y comparten esquemas (datos) y contratos (funcionalidad), pues no se tratan de clases o tipos, sino de envío y recepción de mensajes [15].

En definitiva, los clientes consumen servicios y los servicios ofertan soluciones mediante el envío y recepción de mensajes. WCF provee el servicio a través de mensajes que contienen una cabecera y un cuerpo, basados en XML o JSON.

Implementar un servicio WCF: Un servicio es una aplicación que muestra sus extremos, donde cada extremo puede contener varias operaciones de servicio. El extremo ofrece la comunicación con el servicio.

Un extremo se compone de: una dirección, un enlace y un contrato [13].

- La dirección: Indica dónde se encuentra el servicio utilizando una URL (*Uniform Resource Location*)[12].
- Un enlace: Establece la forma de comunicación con el servicio, algunos ejemplos son `BasicHttpBinding`[13] y `WsHttpBinding`[14]. Los enlaces de WCF son capaces de especificar si se usará un protocolo como HTTP o FTP, también de mecanismos de seguridad tales como: autenticación de Windows o usuarios y contraseñas.
- Un contrato: Define las operaciones establecidas basadas en una interfaz del servicio WCF.

Definir un contrato: La primera actividad para crear el servicio web es establecer un contrato. Los contratos se relacionan con una interfaz que lleva el atributo `ServiceContract`. En la interfaz se indican las operaciones que expondrá el servicio.

Las operaciones en cambio son implementadas mediante métodos definidos en la clase del servicio web y llevan el atributo `OperationContract`. De no llevar las operaciones este atributo, no se podrá exponer a los clientes.

[12] **URL:** Secuencia de caracteres estandarizados que denominan recursos en Internet para ser localizados.
[13] **BasicHttpBinding:** Usa HTTP para comunicarse y XML como mecanismo de codificación.
[14] **WsHttpBinding:** Tiene más características que BasicHttpBinding como WS-Security (no envía en texto plano los datos),

En el Código 1.5 se muestra el ejemplo de un contrato de servicio.

```
[ServiceContract(Namespace ="http://websol.aut.uah.es/ServicioWeb/")]
public interface IServWebWCFConverTemps
{
    //Operaciones ofrecidas por el servicio
    [OperationContract]
    double ConvCentAFahr(double gradosCent);

    [OperationContract]
    double ConvFahrACent(double gradosFahr);

}
```

Código 1.5 Ejemplo de contrato de servicio [13]

El atributo `ServiceContract` hace referencia a la clase `ServiceContractAtribute` para señalar que la interfaz `IServWebWCFConverTemps` establece un contrato de servicio en una biblioteca WCF. El `NameSpace` detalla el espacio de nombres del servicio, usado para diferenciarse de otros servicios. `OperationContract` hace referencia a la clase `OperationContractAttribute` para señalar que los métodos presentados (`ConvCentAFahr` y `ConvFahrACent`) serán expuestos por el servicio WCF [13].

En la Figura 1.5 se muestra el explorador de soluciones de la biblioteca de servicios WCF creada.

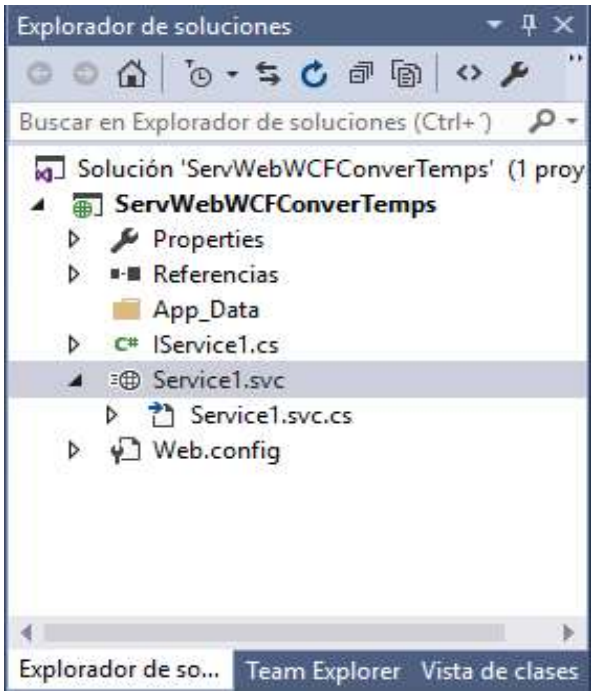


Figura 1.5 Explorador de soluciones de una biblioteca de servicios WCF [13]

Los archivos usados en la biblioteca de servicios WCF se describen a continuación [13]:

- `Service1.svc`: Punto de entrada al servicio WCF.
- `Service1.svc.cs`: Comprende a la clase utilizada para crear el servicio (es la clase que define la interfaz).
- `IService1.cs`: Interfaz donde se definen las operaciones ofrecidas por el servicio WCF.

Una vez se compila y ejecuta la biblioteca de servicios WCF usando Visual Studio. El usuario podrá ingresar a la dirección `localhost:50217/Service1.svc` donde se aloja el servicio (la dirección cambia según la configuración de la biblioteca de servicios WCF).

En la Figura 1.6 se muestra la página web de la dirección anterior que tiene al servicio WCF alojado.

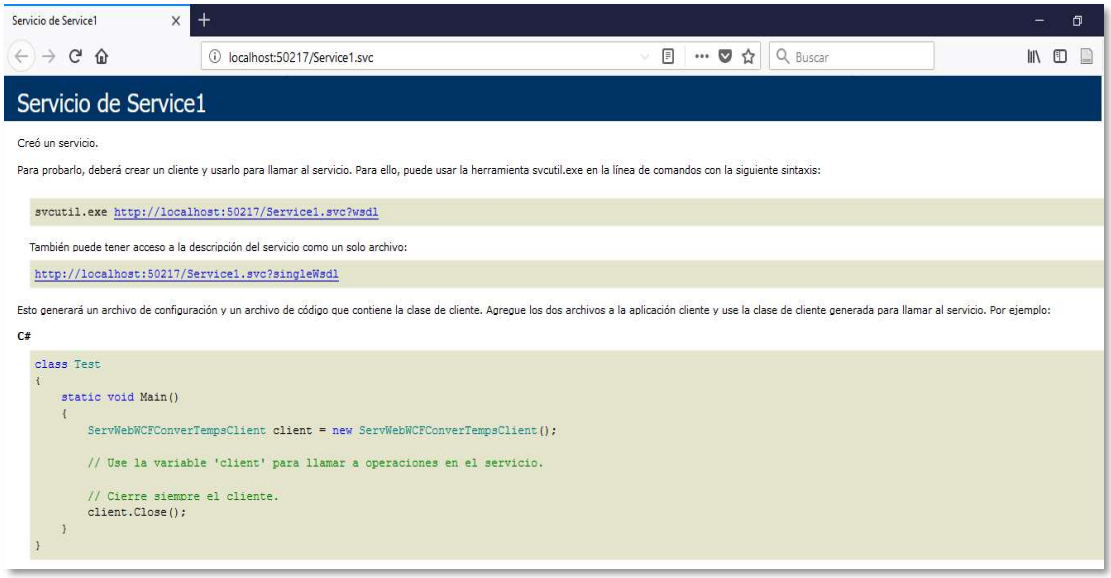


Figura 1.6 Vista de la página web del servicio WCF

1.3.14 DESARROLLO DE APLICACIONES EN ANDROID

Android es un sistema operativo multidispositivo, en un principio pensado para dispositivos celulares, pero actualmente se ejecuta en tabletas, televisores, computadores, cámaras fotográficas, entre otros dispositivos. Android es un software de código libre, gratuito y multiplataforma [17].

Este sistema operativo permite crear aplicaciones con acceso a los recursos de los dispositivos móviles como: GPS, llamadas, agenda, cámara, etc.

El sistema operativo Android se basa en una arquitectura de 4 niveles, que se muestran en la Figura 1.7 [18]:

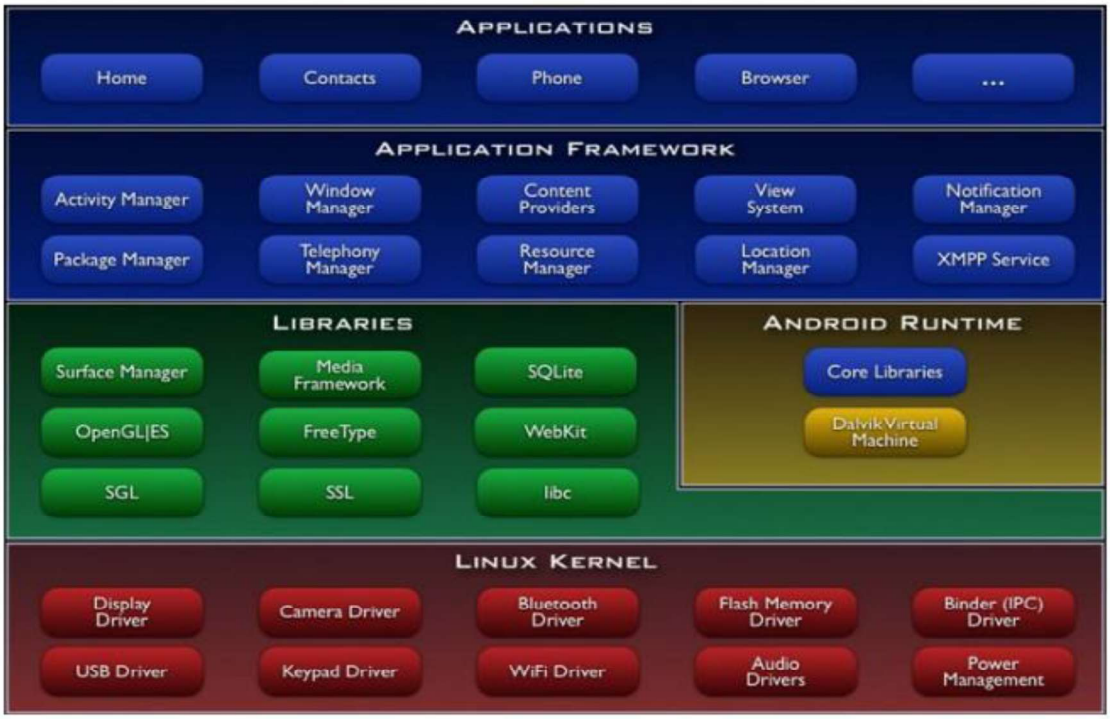


Figura 1.7 Arquitectura de Android [19]

Linux Kernel: Está encargado de funciones fundamentales como: gestión de *drivers*[15], seguridad, comunicaciones, multiproceso, etc.

Libraries: Las librerías son desarrolladas en C y C++ proporcionan a Android gran parte de sus capacidades, entre algunas librerías están: *Surface Manager*[16], que cuenta con diferentes elementos de navegación de pantalla y gestiona las ventanas activas de la aplicaciones en todo momento; SQLite, que crea y gestiona las bases de datos relacionales; Webkit, que brinda un motor que utilizan las aplicaciones de tipo navegador, desde la versión 4.4 se cambió por Chromium/Blink base del navegador Chrome de Google.

Android Runtime: El entorno de ejecución se encuentra al mismo nivel que las librerías. Está compuesto por las *Core Libraries*, que son librerías con una gran variedad de clases de Java.

[15] **Drivers:** Programa informático que enlaza al sistema operativo con un dispositivo periférico.
[16] **Surface Manager:** Librería que maneja el acceso para representar gráficamente en 2D y 3D.

El entorno de ejecución se fundamentaba en el uso de una máquina virtual como lo tenía Java, debido a las limitaciones de procesamiento y memoria se creó la máquina virtual Dalvik, que funcionó hasta Android 5.0 luego se cambió a ART (*Android Runtime*).

Application Framework: El *framework* de aplicaciones es el conjunto de herramientas que sirve para desarrollar aplicaciones, entre las API (*Application Programming Interface*)[17] utilizadas en esta capa están [19]:

- *Activity Manager:* Conjunto de las API que administran el ciclo de vida de las aplicaciones en Android.
- *Window Manager:* Gestiona las ventanas de las aplicaciones, usa la librería *Surface Manager*.
- *Telephone Manager:* Contiene las API relacionadas a funciones base del teléfono (llamada, mensaje, etc.).
- *Content Providers:* Posibilita la compartición de datos entre las aplicaciones Android. Un ejemplo, es el observar datos de contactos por otras aplicaciones.
- *View System:* Oferta una variedad de elementos para crear interfaces de usuario (GUI), como botones, listas, tamaño de ventanas, o el manejo de estas interfaces mediante la opción táctil o teclado.
- *Location Manager:* Proporciona información de localización y posicionamiento a las aplicaciones.
- *Notification Manager:* Informa al usuario sobre alertas de una aplicación que se muestra en la barra de estado.
- XMPP (*Extensible Messaging and Presence Protocol*)*:* conjunto de las API que usan este protocolo para envió y recepción de mensajes basados en XML, utilizado para la mensajería instantánea e información de presencia.

Los desarrolladores pueden acceder a cualquiera de las API del *framework* y utilizarlas en nuevas aplicaciones.

Mediante esta arquitectura se minimiza la redundancia de componentes, pues las aplicaciones pueden usar las capacidades de otras aplicaciones.

Applications: Nivel que está conformado por aplicaciones fundamentales ya instaladas en un dispositivo Android, pueden ser: correo, mensajería, agenda, mapas, navegador, contactos, y demás aplicaciones que el usuario agrega [18].

[17] **API:** Conglomerado de constantes, funciones y protocolos que permiten programar aplicaciones.

Todas las aplicaciones se sirven de los servicios, las API y las librerías de niveles antes descritos.

Las aplicaciones Android están conformadas por varios tipos de bloques. Los bloques de Android pueden clasificarse de la siguiente manera [18] [19]:

View: El componente *View* o Vista es un elemento que conforma la interfaz de usuario[18] de las aplicaciones, como ejemplo: botones o cuadros de texto.

Layout: El *Layout* se conforma por un grupo de vistas. Existen varios tipos de *layouts* que se clasifican según la organización del grupo de vistas de forma: lineal, en cuadrícula o definiendo la posición de cada vista. La forma de generar un *layout* usualmente es mediante un documento XML.

Activity: Un *Activity* o Actividad se compone de varios elementos de visualización, que en conjunto forman la pantalla de una aplicación, siendo el objetivo crear una interfaz de usuario.

La actividad cuenta con una parte lógica que es un archivo `.java` y una parte visual asociada a un *layout*; existe la posibilidad, que un componente *activity* no contenga la parte visual.

Fragment: Un *Fragment* o Fragmento se compone por varias vistas, como su nombre lo indica el fragmento crea una parte de la interfaz de usuario. Uno o varios fragmentos son utilizados en una actividad, para reducir el uso de varias actividades o para mantener un área de visualización común.

Los fragmentos se introdujeron a partir de la llegada de las tabletas, las cuales tenían mayor espacio para visualizar las aplicaciones, espacio el cual ofrecía mayor dinamismo en los diseños de las interfaces de usuario.

Un ejemplo sería: el mostrar un menú en un lateral de la pantalla y presentar la información en el resto de la pantalla.

Broadcast Receiver: El componente *Broadcast Receiver* recepta anuncios *broadcast*[19] y realiza una tarea ante los anuncios. Entre algunos anuncios a indicarse se encuentra los generados por el sistema como son: llamada, nivel de batería bajo y mensaje entrante. Además de otros anuncios generados por aplicaciones.

[18] **Interfaz de Usuario:** Medio por el cual una persona interactúa con un software o hardware determinado.
[19] **Broadcast:** Modo de transmisión de información múltiple y simultáneo.

Broadcast Receiver no contempla una interfaz de usuario, aunque puede utilizar la API *Notification Manager*, para informar al usuario sobre cualquier información utilizando la barra de estado del dispositivo móvil.

Service: Un *Service* o Servicio es un proceso de una aplicación ejecutada en segundo plano, que puede no tener una interfaz de usuario activa.

Este proceso ocurre mientras otras aplicaciones se encuentran ejecutándose en la pantalla principal. Utilizado para mantener activo el proceso de una aplicación, sin importar que el usuario ejecute otra actividad.

Intent: Un *Intent* se define como la posibilidad de realizar una acción, algunos ejemplos son: el envío de mensajes de texto o el envío de correos electrónicos.

Se puede definir otras acciones cómo: iniciar un *activity*, iniciar un *service* o comunicarse con un *service.*

El lanzar un *Intent* también provee que una aplicación delegue el trabajo a otra. Un ejemplo es: cuando se abre una URL desde alguna aplicación, el sistema provee un navegador web ejecute la acción requerida.

Android Manifest: Para complementar el desarrollo de aplicaciones Android es importante tratar sobre el archivo `AndroidManifest.xml`.

`AndroidManifest.xml` es un archivo fundamental en las aplicaciones Android, define los requerimientos para que el sistema pueda desplegarse y las configuraciones para que una aplicación se ejecute sin problemas.

Una de sus principales funciones es la descripción de los componentes de una aplicación como: actividades, servicios y proveedores de contenido. Además en el archivo, se puede establecer permisos para que la aplicación se integre con normalidad al sistema Android y sus aplicaciones por defecto [18].

Android Studio: Android Studio es el IDE (*Integrated Development Environment*)[20] oficial para el desarrollo de Android, que incluye todas las herramientas necesarias para la creación de aplicaciones en Android, está basado en IntelliJ IDEA[21], un editor de códigos y herramientas.

[20] **IDE:** Software que facilita a programadores el desarrollo de aplicaciones.
[21] **IntelliJ IDEA:** Es un entorno de desarrollo integrado para crear programas informáticos.

Los proyectos en Android Studio tienen uno o más módulos conformados con archivos de código fuente y archivos de recursos. Existen algunos tipos de módulos como son:

- Módulos de aplicaciones para Android
- Módulo de bibliotecas
- Módulo de Google App Engine

Por defecto, en Android Studio, los archivos del proyecto se visualizan en la vista de proyectos de Android, tal como se indica en la Figura 1.8.

La vista se construye por medio de módulos que ofrecen el acceso a archivos de origen de un proyecto. La aplicación cuenta con las carpetas:

- `manifests`: donde se ubica el archivo `AndroidManifest.xml`.
- `java`: en el cual se encuentran los archivos de código fuente de Java, así como también el código de prueba *JUnit*[22].
- `res`: conformado por todos los recursos, tales como: documentos XML, imágenes de mapa de bits, entre otros.

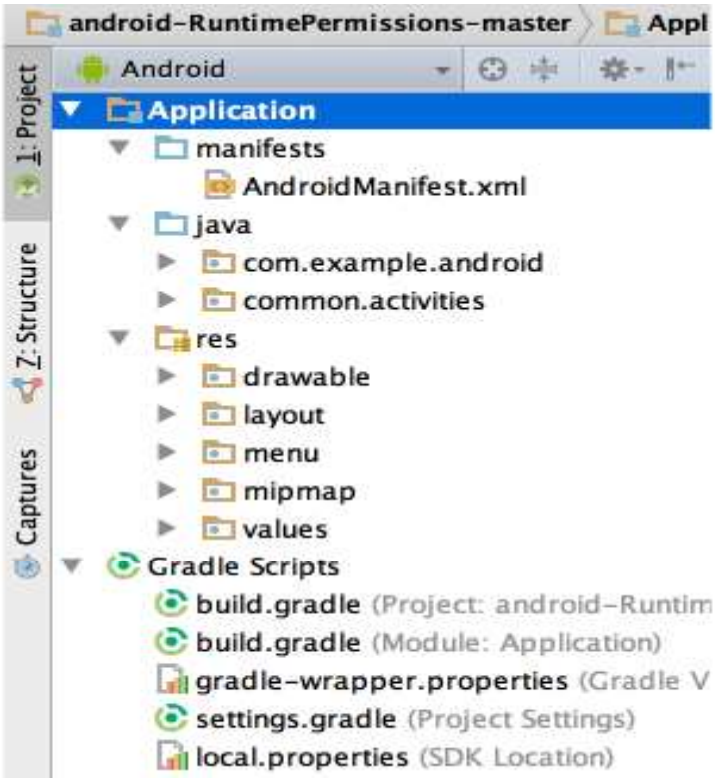


Figura 1.8 Vista de proyectos en Android Studio [20]

[22] **JUnit:** Herramienta que realiza pruebas unitarias en Java.

La ventana principal de Android Studio y sus áreas lógicas se muestran en la Figura 1.9.

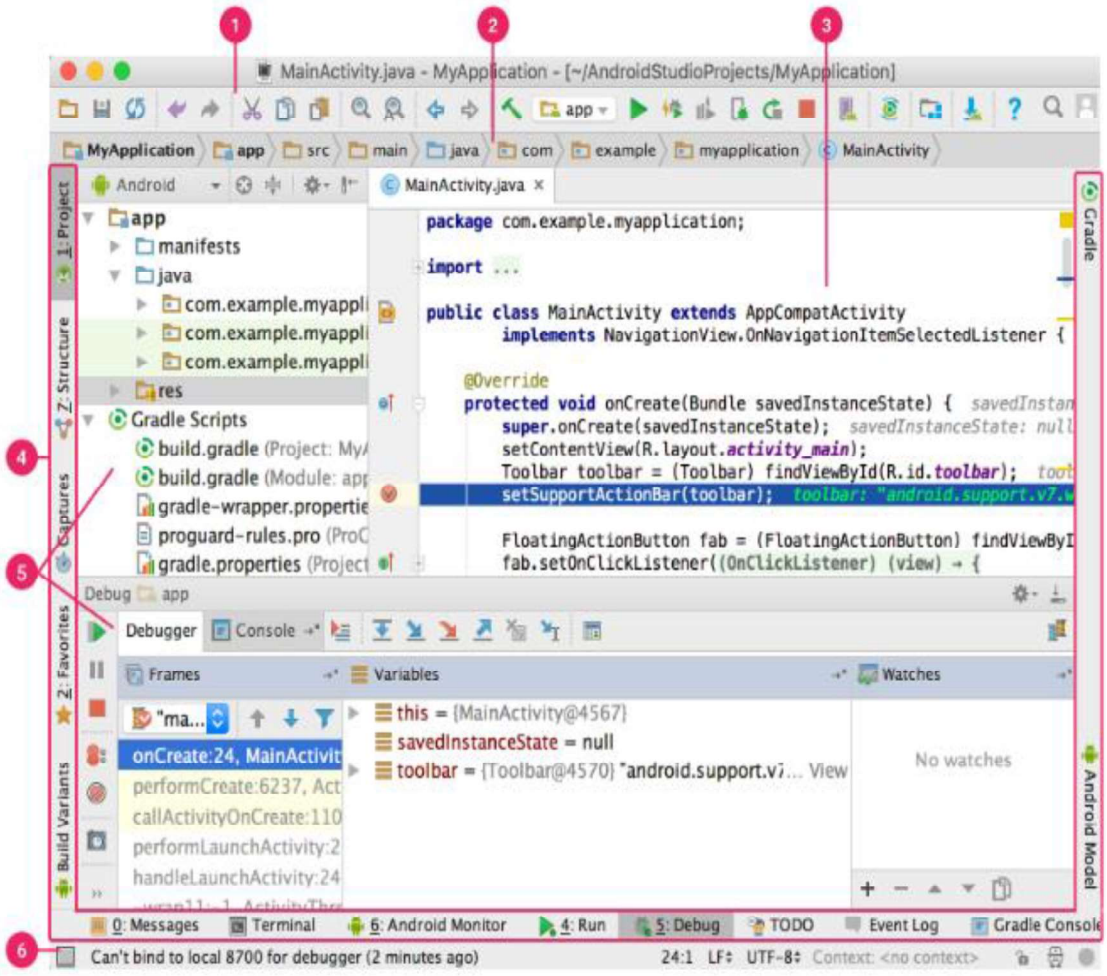


Figura 1.9 Ventana principal de Android Studio [20]

1 Barra de Herramientas: En la cual se realiza un gran número de acciones, entre ellas: la ejecución de la aplicaciones y la inicialización de herramientas de Android.

2 Barra de Navegación: Colabora con la exploración de un proyecto, permitiendo editar los archivos disponibles.

3 Ventana del Editor: Espacio donde es posible crear y editar código. La visualización dependerá del archivo seleccionado.

4 Barra de la Ventana de Herramientas: Ubicada en la parte lateral izquierda del IDE, conformada por botones que abren o cierran ventanas de herramientas individuales.

5 Ventanas de Herramientas: Donde se accede a tareas determinadas, tales como: administración de proyectos, búsquedas, controles de versión, entre otras.

6 Barra de Estado: Indica el estado del proyecto y del IDE, donde se visualiza todas las advertencias o mensajes.

1.3.15 CÓDIGOS QUICK RESPONSE

Los códigos *Quick Response* (QR) son formas de representar y almacenar información en una matriz de puntos bidimensional [21].

En Ecuador, la inclusión de los códigos QR se puede observar en aplicaciones de cine para ingresar a una función, tal como lo realiza Supercines[23]; o en productos como el plátano de exportación que permite a extranjeros redirigir a páginas para conocer un poco más del país [22].

El uso de los códigos QR permite simplificar el proceso de ingreso en el caso del cine o mostrar la información asociada para el caso del plátano de exportación.

Estructura de los Códigos QR: Un código QR se lo denomina símbolo, estos símbolos se forman por cuadros negros o blancos a su vez denominados módulos, los módulos toman los valores de 0 o 1 binario respectivamente.

El símbolo contiene dos grandes bloques de módulos: los patrones de función y la región de codificación [21].

Por cada símbolo hay un conjunto de módulos los cuales no tienen datos codificados, son más bien utilizados para decodificar, estos son los llamados patrones de función [21]:

- Patrón de Posición: Se encuentra con un color verde, existen tres de estos patrones en el símbolo, ubicados en las esquinas exceptuando la esquina inferior derecha, funcionan para determinar la orientación del símbolo.
- Patrón de Alineamiento: Le sobrepone un color naranja, sirve para detectar la posición cuando los módulos se han desplazado.
- Patrón de Sincronización: Útil para sincronizar los patrones de posición del símbolo previniendo eventuales distorsiones.
- Separador: Está de color amarillo, su función es dividir los patrones de posición del símbolo.

La región de codificación se sitúa en las regiones no ocupadas por patrones de función, esta región se conforma por:

- Datos y Corrección de errores: La información propia del código junto con la información útil para corregir errores.

[23] **Supercines:** Cadena de cines de Ecuador.

- Información del Formato: Se encuentra de color rojo, encargada de guardar la información del grado de corrección de errores.
- Información de la Versión: Se encuentra de color azul e indica la cantidad de información que el símbolo almacena.

En la Figura 1.10 se distingue la estructura de un código QR.

Figura 1.10 Estructura de un código QR [23]

1.3.16 GPS

El GPS (Sistema de Posicionamiento Global) en un principio se desarrolló con fines militares, aunque hoy en día es usado abiertamente por civiles. En la plataforma Android se encuentra un sistema de posicionamiento con dos tecnologías, ya integradas al sistema Android [18].

- Sistemas de localización global basado en gps, cuando se dispone de visibilidad directa con los satélites.
- Sistemas de localización basado en información de torres de telefonía celular o puntos de acceso a internet.

API de localización de Android: En primer lugar, se debe dar paso al acceso de información de localización, que se encuentra por razones de seguridad restringido por defecto. Para permitir el uso de localización se debe registrar permisos en el archivo `AndroidManifest.xml`. La aplicación debe contar con los permisos `ACCESS_FINE_LOCATION`[24], `ACCESS COARSE LOCATION`[25] e `INTERNET`[26].

[24] **Access_Fine_Location:** Localización precisa proporcionada por el GPS.
[25] **Access_Coarse_Location:** Localización imprecisa proporcionada por las torres de telefonía celular y Wifi.
[26] **Internet:** Permiso para tener acceso a internet por la aplicación.

Se crea una actividad que emplea la API de localización. En el Código 1.6 se presenta un fragmento del método para definir el objeto que provee los servicios de ubicación del dispositivo.

En la línea 47 se crea el objeto `locationManager` que provee los servicios de ubicación, este objeto se encuentra en el método `onCreate` de la actividad de ejemplo.

```
47      LocationManager locationManager = (LocationManager) getSystemService(Context.LOCATION_SERVICE);
```

Código 1.6 Método que define el objeto `locationManager`

El Código 1.7 muestra el método `onClick` del botón `toggleGPSUdpates` para empezar a capturar la ubicación. En las líneas 58 a 62 se verifica que se encuentren los permisos necesarios en el archivo `AndroidManifest.xml`.

En las líneas 63 y 64 se ejecuta el método `requestLocationUpdates` para iniciar la captura de la ubicación del dispositivo, que tiene como argumentos: el proveedor de localización, el tiempo mínimo de actualización en milisegundos, la distancia mínima de actualización en metros y el objeto `locationListenerGPS` que realiza una acción cuando la ubicación ha cambiado.

```
57      public void toggleGPSUpdates(View view) {
58          if (ActivityCompat.checkSelfPermission(
59              context: this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED
60              && ActivityCompat.checkSelfPermission( context: this, Manifest.permission.ACCESS_COARSE_LOCATION)
61                  != PackageManager.PERMISSION_GRANTED) {
62          }
63          locationManager.requestLocationUpdates(
64              LocationManager.GPS_PROVIDER,  minTime: 2 * 20 * 1000,  minDistance: 10, locationListenerGPS);
65      }
```

Código 1.7 Método que inicia la localización

En el Código 1.8 se muestra la funcionalidad desarrollada para el objeto `locationListenerGPS`, el cual se ejecuta al ser llamado por el objeto `locationManager` anterior.

Para el funcionamiento se utilizará únicamente el método `onLocationChanged` el cual ocurre cuando la ubicación cambia.

En las líneas 69 y 70 se captura la longitud y latitud; en las líneas 71 a 73 se crea un nuevo hilo de ejecución; en las líneas 74 y 75 se asignan los valores obtenidos en los *textbox*. Finalmente, en la línea 76 se muestra un mensaje para advertir la captura de la ubicación [24].

```
67      private final LocationListener locationListenerGPS = new LocationListener() {
68          public void onLocationChanged(Location location) {
69              longitudeGPS = location.getLongitude();
70              latitudeGPS = location.getLatitude();
71              runOnUiThread(new Runnable() {
72                  @Override
73                  public void run() {
74                      longitudeValueGPS.setText(longitudeGPS + "");
75                      latitudeValueGPS.setText(latitudeGPS + "");
76                      Toast.makeText( context: MainActivity.this,  text: "GPS Proveedor actualizado", Toast.LENGTH_SHORT).show();
77                  }
78              });
79      }
```

Código 1.8 Método para capturar la longitud y latitud

Para verificar el funcionamiento de la captura de la ubicación, se establecen valores para latitud y longitud en el emulador Genymotion. En la Figura 1.11 se observa una captura de pantalla del emulador con los valores establecidos.

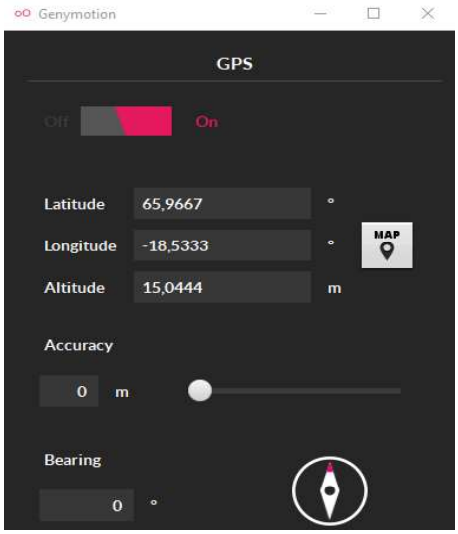


Figura 1.11 Simular valores de latitud y longitud mediante Genymotion

En la Figura 1.12 se observa una captura de pantalla del *layout* que muestra los valores de longitud y latitud capturados; por lo tanto se puede apreciar que el funcionamiento del método para capturar la ubicación funciona correctamente.

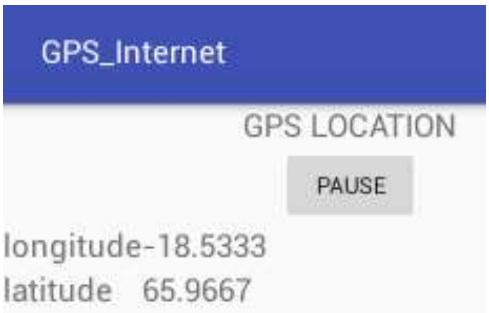


Figura 1.12 Resultado de la localización por GPS.

1.3.17 DESARROLLO DE APLICACIONES DE ESCRITORIO

Las aplicaciones de escritorio permiten juntar las herramientas disponibles de un computador en una aplicación.

Las aplicaciones permiten actualizar o visualizar información, así como comunicarse con equipos externos a una red. Los usuarios aprovechan las interfaces gráficas para realizar las tareas previstas por cada aplicación.

Existen un gran número de lenguajes de programación para desarrollar aplicaciones de escritorio como son: C, C#, Java y Visual Basic; a su vez existe también varios IDE los cuales proveen las herramientas para el desarrollo de aplicaciones, entre algunos están: MonoDevelop, SharpDevelop y Visual Studio.

Lenguaje C#: El lenguaje de programación C# es orientado a objetos y ofrece a desarrolladores la posibilidad de crear varias aplicaciones que corren en .NET Framework[27].

C# es desarrollado y estandarizado por Microsoft. Utilizando este lenguaje se pueden crear aplicaciones para: servicios web, aplicaciones clientes para Windows, aplicaciones cliente-servidor, aplicaciones con bases de datos, entre otras [25].

C# permite el desarrollo de: atributos, que llevan información en tiempo de ejecución; comentarios adjuntos al código o LINQ para realizar mecanismos de consulta sobre un origen de datos.

Visual Studio: Es un IDE que corre en sistemas operativos Windows, puede emplear lenguajes de programación como: C#, C++, Visual Basic, Java y PHP, como otros para la web, entre ellos: ASP.NET y Django.

Todo con la finalidad de crear aplicaciones para transmitir o almacenar información, así como, conceder la comunicación entre aplicaciones.

Entre las tareas que brinda se encuentran: el desarrollo, compilación, depuración, prueba, mejora del rendimiento de aplicaciones y colaboración del código en tiempo real [26].

La ventana principal de Visual Studio 2015 y sus áreas se muestran en la Figura 1.13.

[27] **.NET Framework:** Conjunto de interfaces de programación de aplicaciones y una biblioteca de código compartida usada para el desarrollo de aplicaciones.

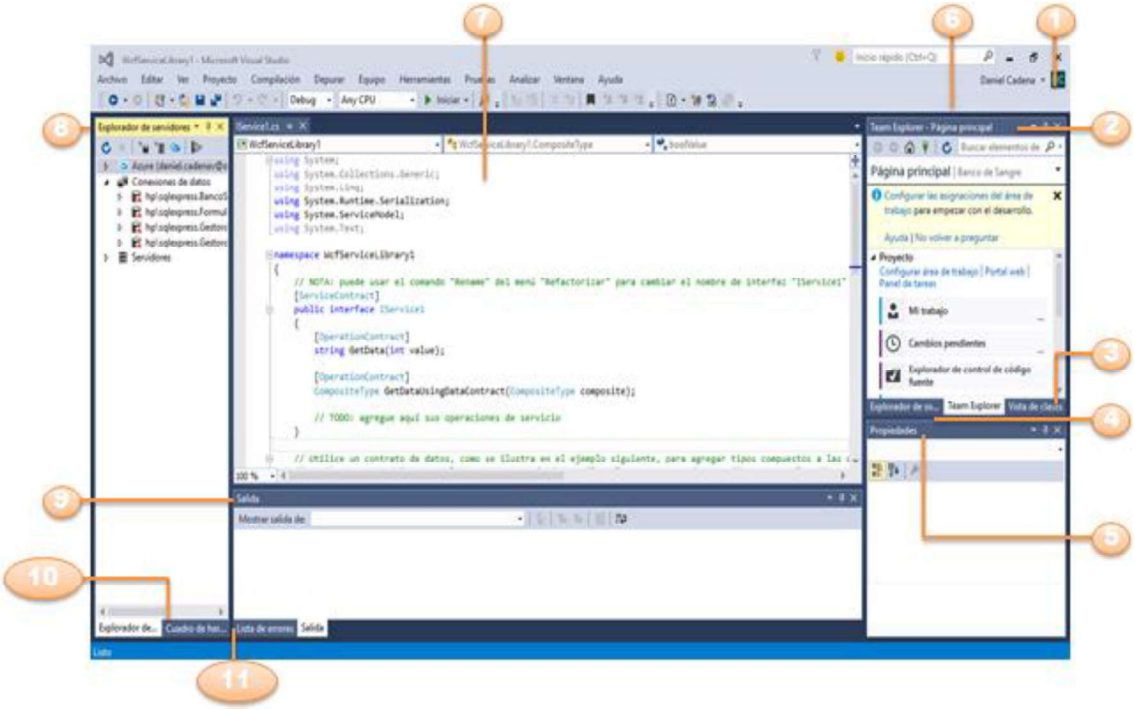


Figura 1.13 Ventana principal de Visual Studio

1 Iniciar sesión: La primera vez que se ejecute Visual Studio, pedirá una cuenta de Microsoft para iniciar el uso del IDE.

2 Team Explorer: Ideal para ver el avance del proyecto, así como para permitir la colaboración de usuarios externos, utilizando tecnologías de control de versión como Git.

3 Vista de clases: Permite un acceso sencillo a los componentes de cada una de las clases del código.

4 Explorador de soluciones: Posibilita observar, navegar y administrar los archivos del código, este explorador de soluciones ofrece una mejor organización del código, asociando los archivos en soluciones y proyectos.

5 Propiedades: Muestra las propiedades de los objetos seleccionados.

6 Inicio Rápido: Ventana para buscar rápidamente comandos, herramientas o características de Visual Studio.

7 Ventana Editor: Donde se encuentra el editor de código, permite visualizar y editar el código fuente, además de diseñar las interfaces de usuario.

8 Explorador de Servidores: Capaz de observar y administrar los recursos de Azure[28], conexiones de Bases de Datos, etc.

[28] **Azure:** Conjunto de servicios en la nube que permite administrar e implementar aplicaciones desarrolladas por Microsoft.

9 Ventana Salida: Superficie en la cual se muestran notificaciones, mensajes de error y depuración, otras advertencias del compilador, o el estado del proyecto.
10 Cuadro de Herramientas: Muestra componentes para agregar a los proyectos de Visual Studio.
11 Lista de errores: Visualiza errores, advertencias o mensajes del proyecto una vez que sea compilado.

Windows Forms: Los Windows Forms o Formularios Windows, son áreas visuales que forman las interfaces de usuario de una aplicación. Las interfaces se componen de controles como: botones, etiquetas, cuadros de texto, etc.

Los controles se programan para tener una respuesta a acciones del usuario como: al ingresar texto dentro de los cuadros o hacer clics con el ratón sobre los botones [25] [27].

Controles: Los interfaces de usuario se componen de controles o componentes entre los más comunes se encuentran:

- `Label`: Presenta imágenes o texto no editable por el usuario que use la aplicación.
- `Textbox`: Proporciona al usuario un ingreso de datos a través del teclado, o presenta información que puede o no ser editable.
- `Button`: Realiza una determinada tarea al hacer clic con el ratón.
- `ComboBox`: Presenta una lista de elementos, de la cual puede escoger uno el usuario, por medio de un clic del ratón o a través del teclado digitando dentro del cuadro.
- `ListBox`: Presenta una lista de elementos, de la cual pueden escogerse uno o varios elementos por el usuario.
- `Panel`: Donde pueden contenerse y organizarse otros controles.

Propiedades: Un formulario cuenta con las propiedades que definirán su texto, fuente, entre otros detalles, los más comunes son:

- `AutoScroll`: Valor que permite o no utilizar las barras de desplazamiento.
- `FormBorderStyle`: Establece que estilo tendrá el borde del formulario, puede ser: simple, tridimensional o ninguno.
- `Font`: Establece la fuente de texto para el formulario, así como una fuente que utilizarán todos los controles que se encuentren en el formulario.
- `Text`: Utilizado para agregar el texto que se muestra en la barra del título del formulario.

Métodos: Los métodos de los formularios realizan una tarea, entre algunos métodos están:

- `Close`: Cierra el formulario, liberando los recursos usados por este como la memoria que utilizan los controles que integra el formulario.
- `Hide`: Oculta el formulario, pero no lo cierra ni libera los recursos.
- `Show`: Hace visible a un formulario.

Eventos: Un evento se puede entender como un mensaje que un objeto envía a otro objeto [13]. Entre los principales eventos de los formularios y controles están:

- `Load`: Se produce antes de que se muestre el formulario por primera vez.
- `Clic`: Se produce al hacer clic con el ratón sobre un control.
- `MouseDown`: Se produce cuando el puntero del ratón está sobre un control.

Delegados: Los manejadores de eventos que se conectan a los eventos de un determinado control mediante objetos, son llamados delegados [27]. Los controles tienen delegados establecidos para manejar los eventos.

1.4 RELACIÓN CON TRABAJOS SIMILARES DEL ÁREA

Se presenta una comparativa de trabajos similares en la carrera de Ingeniería en Electrónica y Redes de Información con este proyecto integrador.

En [28] se desarrolló un sistema distribuido basado en SOA y *Model-View-Controller* (MVC) para el despliegue de Información Médica de Pacientes, en el cual se utilizó: la metodología SCRUM, un servicio web WCF, una base de datos y MVC para el diseño arquitectónico del software.

En este proyecto integrador se aplican herramientas similares al trabajo anterior, como son: la metodología SCRUM para el desarrollo del software, el uso de un servicio web WCF para permitir la transferencia de la información y una base de datos SQL para almacenar la información; no se aplica el diseño arquitectónico MVC. Las diferencias parten en el ámbito de desarrollo, para este proyecto es, la gestión de eventos sobre emergencias de sistemas de alarma. Es importante mencionar que en este proyecto integrador se desarrolla una aplicación móvil para gestionar los eventos.

En [29] se desarrolló un sistema distribuido para el manejo de la información de asistencia en tierra a aeronaves, se utilizó para manejar datos de vuelos y listar servicios adicionales asociados al vuelo. El cual usó: un servicio web SOAP, una base de datos ubicada en el dispositivo móvil y un aplicativo móvil.

Este proyecto integrador desarrolla un aplicativo móvil, el cual no utiliza una base de datos local, sino gestiona los datos mediante un servicio web Restful conectado con una base de datos SQL. También se desarrolla en el proyecto integrador una aplicación de escritorio, la cual gestiona los datos utilizando el mismo servicio web.

2. METODOLOGÍA

El presente Trabajo de Titulación se desarrollará utilizando una investigación aplicada, debido a que se realizará un prototipo de gestión de eventos para seguridad privada.

La recolección de información se realizará mediante entrevistas al personal de la compañía de seguridad privada SEDYM CIA LTDA. Con la información recolectada, se describe la situación actual acerca de los procesos en casos de emergencia de la compañía, así mismo se presentan diagramas que detallan cada uno de los procesos.

Empleando los conocimientos sobre: la metodología de desarrollo de software SCRUM, las bases de datos relacionales, los servicios web WCF, las aplicaciones Android y las aplicaciones de escritorio en Windows, se conformará un prototipo de gestión de eventos. Se establecerán las historias de usuario, la arquitectura del prototipo y las herramientas utilizadas para el desarrollo. Siguiendo a la metodología SCRUM se presentarán el *product backlog*, los roles, la planificación de los *sprints* y las tareas de cada *sprint*.

El diseño del prototipo se conformará por: el diagrama entidad-relación y el diagrama relacional para la base de datos; los diagramas con las clases e interfaces del servicio web; los diagramas de clases, los diagramas de secuencia, la realización de *sketches*, los diagramas de actividades y la realización de las interfaces de usuario para la aplicación Android y la aplicación de escritorio.

La implementación ejecutará la codificación del prototipo, validación de datos y el alojamiento del servicio web WCF en el servidor web IIS (*Internet Information Services*).

Al finalizar con la implementación del prototipo, se realizarán las pruebas de funcionamiento de los componentes del prototipo y la corrección de errores. Se harán las encuestas de validación al personal de la compañía de seguridad SEDYM CIA LTDA. El nombre del prototipo es: Security Events.

2.1. SITUACIÓN ACTUAL

Se conoce que el proceso llevado por parte de la compañía de seguridad incluye a ciertas personas (clientes, supervisores y operadores de la central de monitoreo), así como los procesos en casos de emergencia reportados mediante: un botón de pánico, un botón de emergencia médica o por activaciones de robo de los sistemas alarmas, además de toda la documentación que implica el registro de estas emergencias.

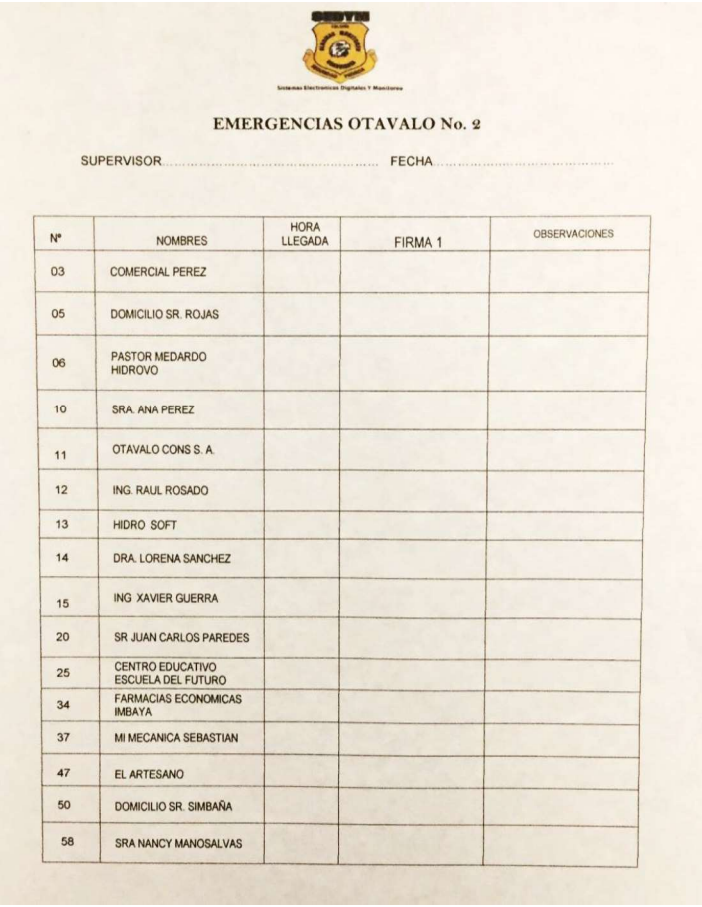
Actualmente todos los procesos se los registra de forma manual en una hoja de observaciones, por lo cual al realizar reportes o llevar un control de los eventos por cada

cliente se dificulta, como también complica la identificación de los distintos parámetros propios del evento.

Por lo expuesto anteriormente, el proceso puede volverse complejo de gestionar tomando en cuenta que la información se genera diariamente, y que al cabo de algunos meses las hojas de observaciones se almacenan en archivadores.

En la Figura 2.1 se presenta la hoja de observaciones en caso de emergencia con la siguiente información:

- Supervisor: Nombre del supervisor.
- Fecha: Fecha del día que suscitaron las emergencias.
- N°: Número de cuenta del cliente.
- Nombres: Nombre del cliente.
- Hora Llegada: Hora de llegada a la emergencia.
- Firma 1: Firma de la persona autorizada.
- Observaciones: Información acerca de la emergencia.



EMERGENCIAS OTAVALO No. 2

SUPERVISOR..... FECHA.....

Nº	NOMBRES	HORA LLEGADA	FIRMA 1	OBSERVACIONES
03	COMERCIAL PEREZ			
05	DOMICILIO SR. ROJAS			
06	PASTOR MEDARDO HIDROVO			
10	SRA. ANA PEREZ			
11	OTAVALO CONS S. A.			
12	ING. RAUL ROSADO			
13	HIDRO SOFT			
14	DRA. LORENA SANCHEZ			
15	ING. XAVIER GUERRA			
20	SR JUAN CARLOS PAREDES			
25	CENTRO EDUCATIVO ESCUELA DEL FUTURO			
34	FARMACIAS ECONOMICAS IMBAYA			
37	MI MECANICA SEBASTIAN			
47	EL ARTESANO			
50	DOMICILIO SR. SIMBAÑA			
58	SRA NANCY MANOSALVAS			

Figura 2.1 Hoja de observaciones en casos de emergencia

2.1.1 PROCESOS

Con base en las entrevistas realizadas al personal de la compañía SEDYM, en la Figura 2.2 se presenta el diagrama de actividades del proceso actual de cómo actúan el operador y supervisor en un caso de emergencia cuando un cliente presiona el botón de pánico en los sistemas de alarma.

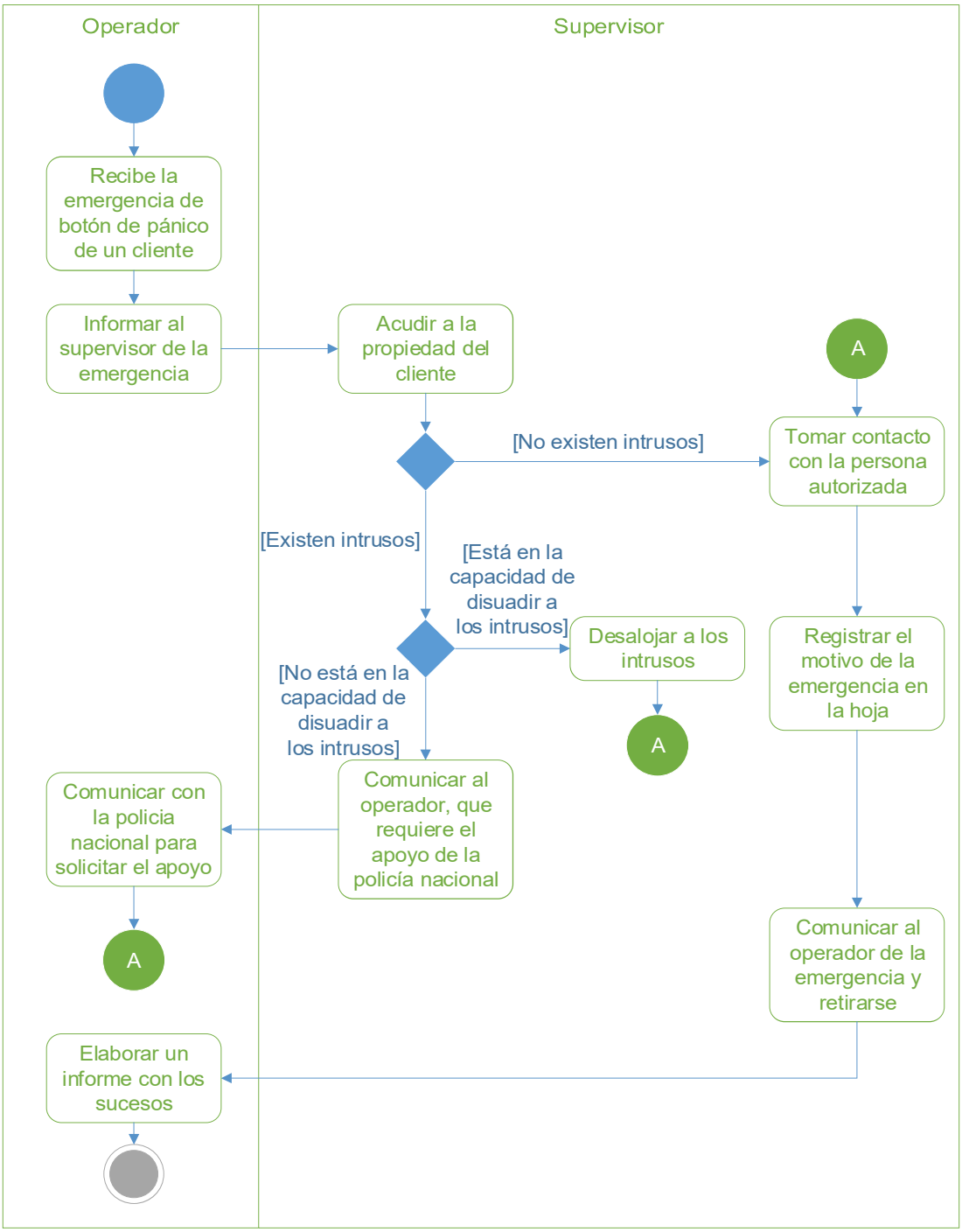


Figura 2.2 Diagrama de actividades al presionar un botón de pánico

En la Figura 2.3 se muestra el diagrama de actividades del proceso actual de cómo actúan el operador y supervisor en un caso de emergencia cuando un cliente presiona el botón de emergencia médica en los sistemas de alarma.

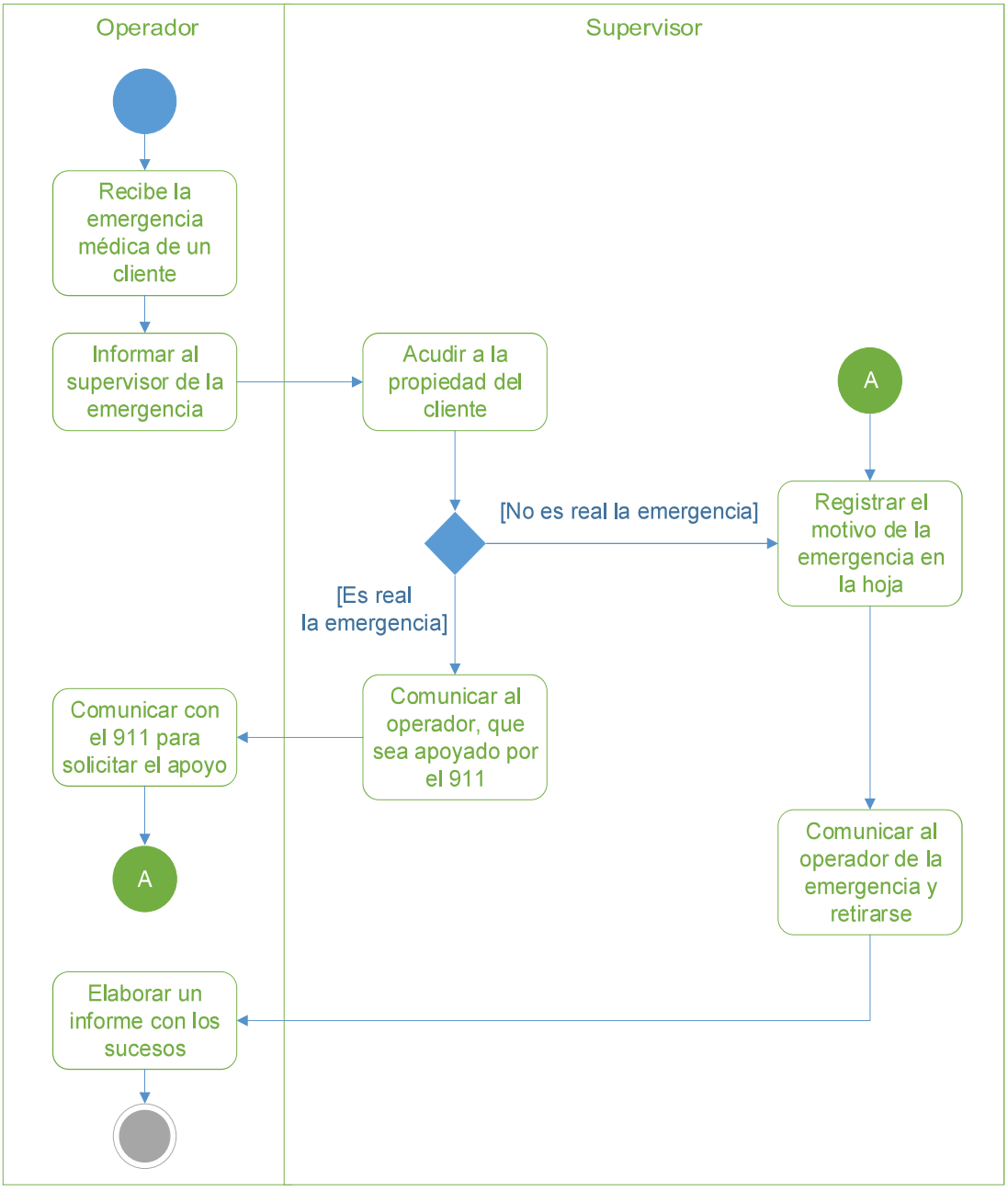


Figura 2.3 Diagrama de actividades al presionar un botón de emergencia médica

En la Figura 2.4 se presenta el diagrama de actividades del proceso actual de acción en un caso de emergencia cuando un cliente tiene una activación de robo del sistema de alarma.

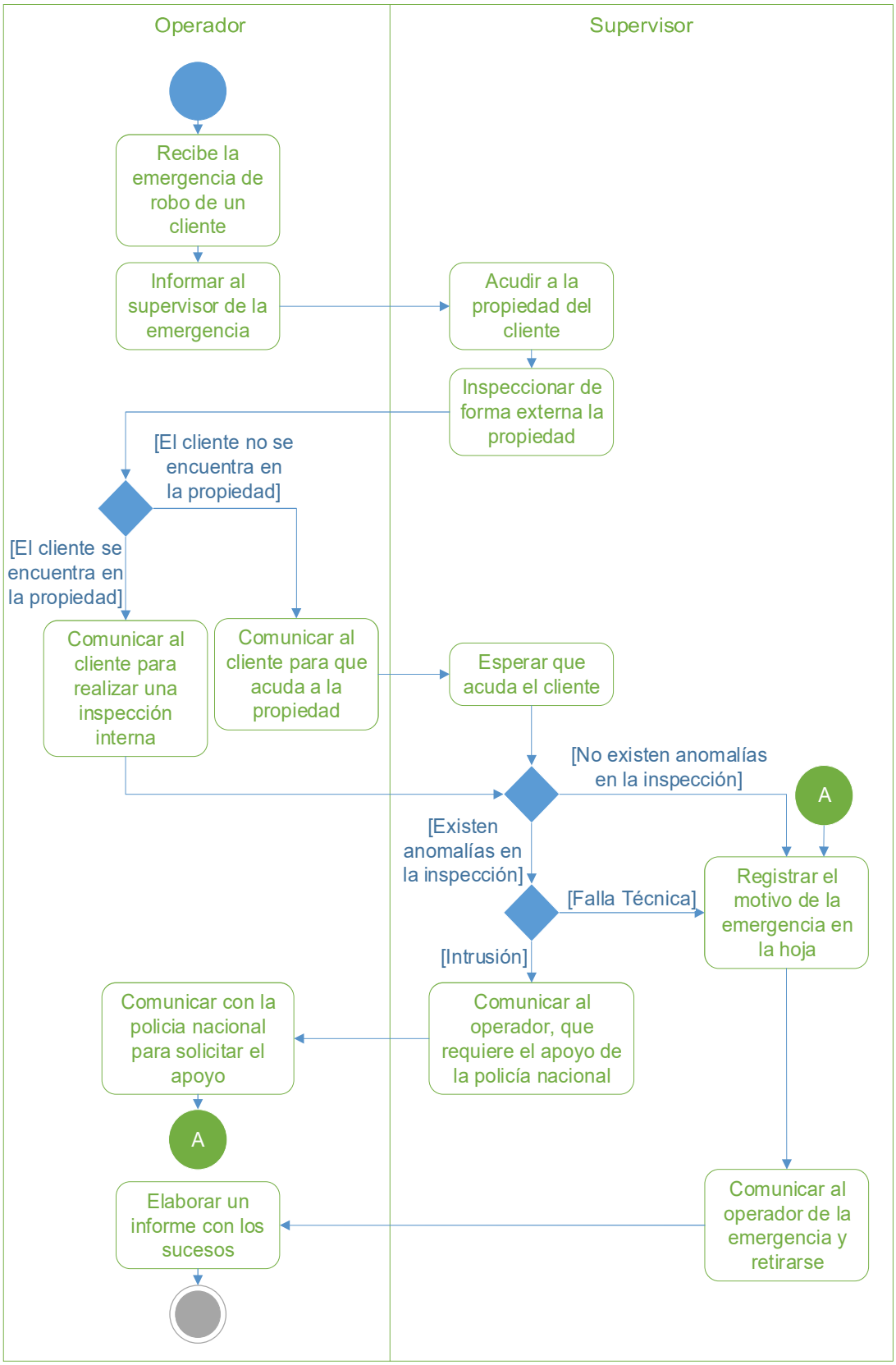


Figura 2.4 Diagrama de actividades en caso de emergencia por activación de robo

2.1.2 REQUERIMIENTOS FUNCIONALES

Los requerimientos se obtuvieron a través del refinamiento de la información de las entrevistas realizadas al personal de la compañía SEDYM. La ficha usada para la entrevista se encuentra en el Anexo A.

- El usuario administrador estará registrado en el prototipo.
- Iniciar sesión en la aplicación Android y la aplicación de escritorio.
- Cerrar sesión en la aplicación Android.
- Salir en la aplicación de escritorio.
- Permitir la creación, visualización y eliminación de eventos, la eliminación comprende una ocultación de los eventos en el prototipo, más no una eliminación de la información de los eventos.
- Permitir la creación, visualización y eliminación de tipos de eventos.
- Crear tres estados de los eventos: enviado, en desarrollo y finalizado.
- Permitir la creación, modificación, visualización y eliminación de usuarios, la eliminación comprende una ocultación de los usuarios en el prototipo, más no una eliminación de la información de los usuarios.
- Crear tres tipos de usuarios: administrador, operador y supervisor.
- Permitir la creación, modificación, visualización y eliminación de clientes, la eliminación comprende una ocultación de los clientes en el prototipo, más no una eliminación de la información de los clientes.
- Permitir la búsqueda de clientes según: nombre del cliente, número de cuenta y ciudad.
- Permitir la creación, visualización y eliminación de ciudades de clientes.
- Permitir la creación, visualización y eliminación de tipos de clientes.
- Permitir la creación, visualización y eliminación de planes de clientes.
- Permitir la creación, modificación, visualización y eliminación de las zonas de los sistemas de alarma.
- Permitir la creación, modificación, visualización y eliminación de personas, la eliminación comprende una ocultación de las personas en el prototipo, más no una eliminación de la información de los personas.
- Permitir la búsqueda de personas según: el nombre de la persona y el número de cuenta del cliente.
- Permitir la creación, visualización y eliminación de roles de personas.
- Permitir la creación, visualización y eliminación de responsabilidades.

- Permitir la búsqueda de eventos y exportación a Excel por: fecha de inicio y fin de envío, llegada y salida, asistencia policial, supervisión, tipo, estado, persona autorizada, supervisor y operador.
- Modificar la URL de conexión al servicio web en la aplicación Android y la aplicación de escritorio.
- Permitir la habilitación de usuarios, personas, clientes y eventos ocultos del prototipo.

2.1.3 REQUERIMIENTOS NO FUNCIONALES

- La información del prototipo estará almacenada en una base de datos.
- La información almacenada en la base de datos deberá ser actualizada en las aplicaciones.
- Las contraseñas de los usuarios deberán almacenarse usando un algoritmo de seguridad.
- Las contraseñas de los usuarios no deberán visualizarse en las aplicaciones.
- La aplicación Android y la aplicación de escritorio deberán contar con mensajes informativos cuando se ingrese información.
- La aplicación Android y la aplicación de escritorio deberán consumir el servicio WCF.
- Para obtener un mensaje de respuesta del servicio WCF se deberá previamente validar el id y token del usuario, exceptuando únicamente las funcionalidades de autenticación.
- El prototipo deberá contar con manuales de usuario para la aplicación Android y la aplicación de escritorio.

2.1.4 HISTORIAS DE USUARIO

Las historias de usuario fueron desarrolladas usando el formato del capítulo uno, donde también se encuentra la definición de cada ítem.

Todas las historias de usuario se presentan en el Anexo B.

En la Tabla 2.1 se muestra un ejemplo de una historia de usuario para realizar la autenticación en la aplicación de escritorio, que requerirá el ingreso del alias y la contraseña de cada usuario para ingresar en el prototipo.

Además, se requiere la implementación de una funcionalidad para cambiar la contraseña al primer inicio de sesión o cuando sea restablecida la contraseña.

Tabla 2.1 Historia de Usuario para la autenticación

Código: HU-01	**Actor:** Administrador, Operador
Desarrollador: Fernando Cadena	**Sprint:** Usuarios
Nombre: Autenticación aplicación de escritorio	**Número de Sprint:** 3
Descripción: Como administrador u operador quiero colocar mi alias y contraseña para poder ingresar al sistema.	
Comentarios: Se requiere modificar mi contraseña cuando inicie sesión por primera vez o cuando un administrador restablezca mi contraseña.	

2.2. ARQUITECTURA DEL PROTOTIPO

La arquitectura se indica en la Figura 2.5. Esta arquitectura tiene en el lado del cliente: la aplicación Android y la aplicación de escritorio, en el lado del servidor: el servicio WCF y la base de datos.

Las aplicaciones cliente y el servidor realizan el intercambio de mensajes a través de solicitudes y respuestas, los mensajes utilizan los métodos del protocolo HTTP y el formato JSON.

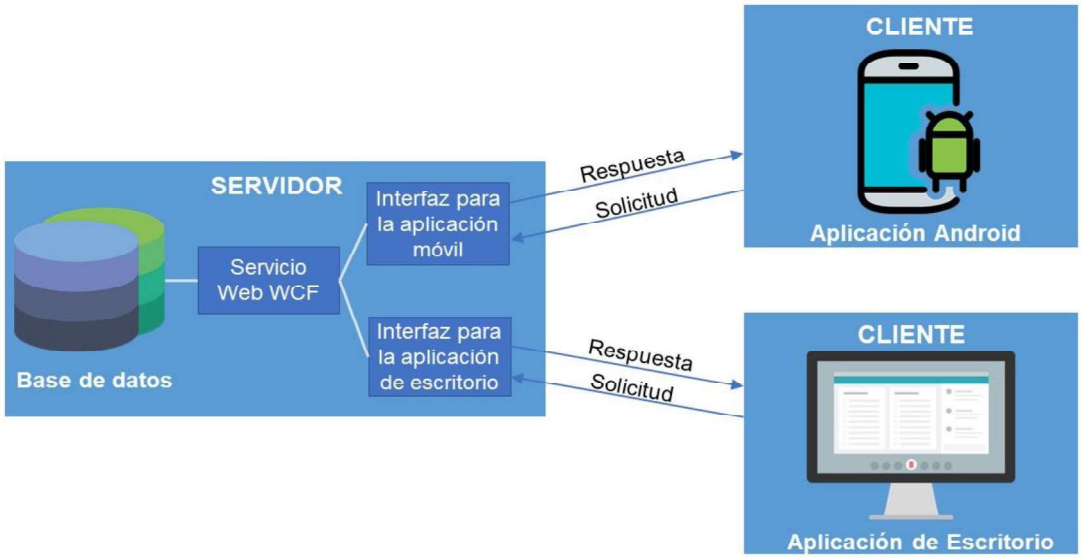


Figura 2.5 Arquitectura del Prototipo

2.3. HERRAMIENTAS DE DESARROLLO

Es necesario identificar las herramientas de desarrollo que permitirán la codificación de la base de datos, el servicio web WCF, la aplicación Android y la aplicación de escritorio.

Base de datos: La base de datos se desarrollará utilizando el IDE SSMS (SQL Server Management Studio) 2014 y usará el lenguaje SQL para la creación de la base de datos, tablas y atributos.

El IDE sirve para acceder, manejar y almacenar la información de SQL Server a través de una interfaz gráfica [30]. La base de datos es de tipo relacional.

Servicio web: El servicio web se desarrollará en el IDE Visual Studio 2015 y usará el lenguaje de programación C#. El servicio web aprovechará los componentes del IDE para realizar las consultas a la base de datos usando de LINQ. El servicio web es de tipo WCF Restful.

Aplicación de escritorio: La aplicación de escritorio se desarrollará en el IDE Visual Studio 2015 y usará el lenguaje de programación C#. El IDE ayuda a crear, depurar y editar código para realizar una aplicación. A través de los Windows Forms se generará las interfaces que tendrá la aplicación.

Aplicación Android: La aplicación Android se desarrollará en el entorno de desarrollo integrado Android Studio 3.1.4 y usará el lenguaje de programación Java.

El IDE ofrece varias funcionalidades como [20]: un entorno para realizar aplicaciones de todos los dispositivos Android, la posibilidad de realizar cambios durante la ejecución de la aplicación y herramientas para verificar el rendimiento y la usabilidad.

2.4. PRODUCT BACKLOG

En la Tabla 2.2 y la Tabla 2.3 se presenta el *Product Backlog*. Se detallan los campos que contiene el *Product Backlog*:

- Código de la Historia: Código único utilizado para diferenciar una historia de usuario.
- Nombre: Título de la historia de usuario.

Tabla 2.2 *Product Backlog* (Parte I de II)

Código de la Historia	Nombre
HU-01	Autenticación aplicación de escritorio

Tabla 2.3 *Product Backlog* (Parte II de II)

HU-02	Gestión de Cuentas de Usuarios
HU-03	Gestión de Cuentas de Clientes
HU-04	Gestión de Zonas del Cliente
HU-05	Gestión de Cuentas de Personas
HU-06	Gestión de Responsabilidad de las Personas
HU-07	Gestión de Eventos
HU-08	Búsqueda de Eventos y Exportación a Excel
HU-09	Configuración de Parámetros
HU-10	Autorizaciones
HU-11	Autenticación aplicación Android
HU-12	Registro y Listado de Eventos
HU-13	Códigos QR
HU-14	Localización
HU-15	Visualizar Clientes

2.4.1 ROLES DE SCRUM

En la Tabla 2.4 se muestran los roles que se han definido en el desarrollo del prototipo, agregando una breve descripción de las funciones de cada rol.

Tabla 2.4 Roles de SCRUM

Rol	Persona	Descripción
Product Owner (PO)	Representante legal de la compañía SEDYM Cía. Ltda.	Persona que establece los requerimientos del prototipo, acuerda las prioridades de entrega y valida el producto final.
SCRUM *Master* (SM)	Raúl David Mejía Navarrete, M.Sc.	Responsable de comprobar el estado de las actividades del prototipo, de su correcta ejecución y asistir en los problemas que en el desarrollo se presenten.
SCRUM *Team* (ST)	Fernando Daniel Cadena Villarreal	Encargado de realizar las funcionalidades descritas en los requerimientos del prototipo.

2.4.2 PLANIFICACIÓN DE LOS SPRINTS

En base al *product backlog* se procede a planificar los *sprints*, se formaron seis *sprints* para el desarrollo del prototipo.

En la Tabla 2.5 se muestra la planificación de los *sprints*, se agregan las columnas con el número del *sprint* y el nombre del *sprint*, también se organizan las historias de usuario según características similares.

Tabla 2.5 Planificación de los *Sprints*

No. Sprint	Sprint	Código	Historias de Usuario
1	Autenticaciones	HU-01	Autenticación aplicación de escritorio
		HU-11	Autenticación aplicación Android
2	Usuarios	HU-02	Gestión de Cuentas de Usuarios
3	Clientes	HU-03	Gestión de Cuentas de Clientes
		HU-04	Gestión de Zonas del Cliente
		HU-15	Visualizar Clientes
4	Personas	HU-05	Gestión de Cuentas de Personas
		HU-06	Gestión de Responsabilidad de las Personas
5	Eventos	HU-07	Gestión de Eventos
		HU-08	Búsqueda de Eventos y Exportación a Excel
		HU-12	Registro y Listado de Eventos
		HU-13	Códigos QR
		HU-14	Localización
6	Configuraciones	HU-09	Configuración de Parámetros
		HU-10	Autorizaciones

***Sprint* 1 - Autenticaciones:** En cada *sprint* se agrega la columna componente para identificar si se realizará en la: la aplicación de escritorio o en la aplicación Android. También se agrega la columna tareas con la descripción de las actividades. En la Tabla 2.6 se muestran las tareas planificadas para el *sprint* 1.

Tareas las cuales son la autenticación y el cambio de contraseña en la aplicación de escritorio y en la aplicación Android.

Tabla 2.6 *Sprint* 1

Código	Nombre	Componente	Tareas
HU-01	Autenticación aplicación de escritorio	Aplicación de Escritorio	Creación de la interfaz de usuario para la autenticación
			Implementar código para la autenticación
			Creación de la interfaz de usuario para el cambio de contraseña
			Implementar código para el cambio de contraseña
HU-11	Autenticación aplicación Android	Aplicación Android	Creación de la interfaz de usuario para la autenticación
			Implementar código para la autenticación
			Creación de la interfaz de usuario para el cambio de contraseña
			Implementar código para el cambio de contraseña

***Sprint* 2 – Usuarios:** En la Tabla 2.7 se muestran las tareas planificadas para el *sprint* 2. Se realizarán el CRUD[29] de usuarios y la validación de datos del usuario para la aplicación de escritorio.

Tabla 2.7 *Sprint* 2

Código	Nombre	Componente	Tareas
HU-02	Gestión de Cuentas de Usuarios	Aplicación de Escritorio	Creación de las interfaces de usuario para la gestión de usuarios
			Implementar código para la gestión de usuarios
			Validación de campos para ingresar y modificar un usuario

***Sprint* 3 – Clientes:** En la Tabla 2.8 se muestran las tareas planificadas para el *sprint* 3, se realizará el CRUD de clientes y zonas. Además, se incluye la visualización de un mapa para ubicar la dirección de un cliente en la aplicación de escritorio, y otro mapa para mostrar las ubicaciones de los clientes en la aplicación Android.

[29] **CRUD:** Crear, leer, actualizar y eliminar.

Tabla 2.8 *Sprint* 3

Código	Nombre	Componente	Tareas
HU-03	Gestión de Cuentas de Clientes	Aplicación de Escritorio	Creación de un código QR
			Creación de la interfaz de usuario para la gestión de clientes
			Implementar código para la gestión de clientes
			Implementar código para buscar clientes
			Validación de campos para ingresar y modificar clientes
			Creación de la interfaz de usuario para visualizar el mapa
			Implementar código para visualizar el mapa
HU-04	Gestión de Zonas del Cliente		Creación de la interfaz de usuario para la gestión de zonas
			Implementar código para la gestión de zonas
			Validación de campos para ingresar, modificar y eliminar zonas
HU-15	Visualizar Clientes	Aplicación Android	Creación de la interfaz de usuario para ubicar clientes
			Implementar código para visualizar clientes

***Sprint* 4 – Personas:** En la Tabla 2.9 y la Tabla 2.10 se muestran las tareas planificadas para el *sprint* 4 que son el CRUD y la validación de datos de las personas y responsabilidades.

Tabla 2.9 *Sprint* 4 (Parte I de II)

Código	Nombre	Componente	Tareas
HU-05	Gestión de Cuentas de Personas	Aplicación de Escritorio	Creación de la interfaz de usuario para la gestión de personas
			Implementar código para la gestión de personas
			Validación de campos para ingresar y modificar personas
			Implementar código para buscar personas

Tabla 2.10 *Sprint* 4 (Parte II de II)

HU-06	Gestión de Responsabilidad de las Personas	Aplicación de Escritorio	Creación de la interfaz de usuario para la gestión de responsabilidades
			Implementar código para la gestión de responsabilidades
			Validación de campos para ingresar y eliminar responsabilidades

***Sprint* 5 – Eventos:** En la Tabla 2.11 y la Tabla 2.12 se presentan las tareas planificadas para el *sprint* 5, como el CRUD y la validación de datos de los eventos, la búsqueda y exportación de eventos para la aplicación de escritorio. Los mecanismos de captura de códigos QR, captura de localización y el registro de eventos en la aplicación Android.

Tabla 2.11 *Sprint* 5 (Parte I de II)

Código	Nombre	Componente	Tareas
HU-07	Gestión de Eventos	Aplicación de Escritorio	Creación de la interfaz de usuario para la gestión de eventos
			Implementar código para la gestión de eventos
			Validación de campos para ingresar y modificar eventos
			Asignación de permisos según el tipo de usuario
HU-08	Búsqueda de Eventos y Exportación a Excel	Aplicación de Escritorio	Creación de la interfaz de usuario para buscar eventos
			Implementar código para buscar eventos
			Implementar código para exportar eventos a Excel
			Implementar código para visualizar eventos
HU-12	Registro y Listado de Eventos	Aplicación Android	Creación de las interfaces de usuario para navegar en la aplicación y listar eventos
			Implementar código para visualizar eventos
			Creación de la interfaz de usuario para registrar eventos
			Implementar código para registrar eventos

Tabla 2.12 *Sprint* 5 (Parte II de II)

HU-13	Códigos QR		Creación de la interfaz de usuario para escanear códigos QR
			Implementar código para capturar los códigos QR
			Implementar código para comparar los códigos QR
HU-14	Localización		Implementar código para tomar la ubicación
			Implementar código para comparar las ubicaciones

***Sprint* 6 – Configuraciones:** En la Tabla 2.13 se muestran las tareas planificadas para el *sprint* 6. Las tareas se componen de la configuración de parámetros y las autorizaciones de los eventos, usuarios, clientes y personas para la aplicación de escritorio.

Tabla 2.13 *Sprint* 6

Código	Nombre	Componente	Tareas
HU-09	Configuración de Parámetros	Aplicación de Escritorio	Creación de la interfaz de usuario para configurar parámetros
			Implementar código para la gestión de parámetros
			Validación de campos para el ingreso de datos de roles de personas, planes, tipos, ciudades de clientes y tipos de eventos
			Implementar código para modificar la URL de conexión al servicio web
HU-10	Autorizaciones		Creación de la interfaz de usuario para habilitar eventos, usuarios, clientes y personas
			Implementar código para habilitar eventos, usuarios, clientes y personas

2.5. DISEÑO E IMPLEMENTACIÓN

Para la consecución del prototipo se desarrollan distintos componentes, a continuación se describen estos componentes.

2.5.1 BASE DE DATOS

Para almacenar la información del prototipo se crea la base de datos. Se diseña el diagrama entidad-relación usando el formato descrito en el capítulo uno.

Diseñar el diagrama Entidad-Relación: En la Figura 2.6 y la Figura 2.7 se muestra el diagrama entidad-relación conformado por los entidades, atributos y relaciones.

En el diagrama se determinan las entidades: `Usuario`, `Tipo Usuario`, `Evento`, `Tipo Evento`, `Estado Evento`, `Cliente`, `Ciudad Cliente`, `Tipo Cliente`, `Plan Cliente`, `Zona del Sistema de Alarma`, `Persona` y `Rol de la Persona`.

Los atributos de la entidad `Usuario` son: `idUsuario`, `nombreUsuario`, `apellidoUsuario`, `aliasUsuario`, `telefonoUsuario`, `cédula`, `correo`, `hashContraseña`, `token`, `primerInicioSesion` y `disponibilidadUsuario`.

Para describir las relaciones se toma como ejemplo la entidad `Usuario`: La relación entre la entidad `Usuario` y `Tipo Usuario` se crea debido a que varios usuarios tienen un solo tipo de usuario, por tanto, la cardinalidad es de muchos a uno (N:1).

La relación entre la entidad `Usuario` y `Evento` se crea debido a que varios usuarios pueden registrar varios eventos, por tanto, la cardinalidad es de muchos a muchos (N:N). Primero se determina que la entidad `Usuario` se utiliza para identificar los actores del prototipo, varios elementos de la entidad `Usuario` se relacionan con un solo `Tipo Usuario`.

Por otro lado, varios elementos de la entidad `Usuario` se relacionan con varios elementos de la entidad `Evento`.

La entidad `Evento` es la que recoge la mayoría de información del prototipo. Varios elementos de la entidad `Evento` se relacionan con un solo `Tipo Evento`, un `Estado Evento` y una `Persona`. También varios elementos de la entidad `Evento` se relacionan con varios elementos de la entidad `Zona del Sistema de Alarma`.

Por otra parte, varios elementos de la entidad `Persona` se relacionan con un `Rol de la Persona`. Dado a que el prototipo permite la creación de clientes, varios elementos de la entidad `Persona` pueden relacionarse con varios elementos de la entidad `Cliente`.

Los elementos de la entidad `Cliente` tienen relacionado una `Ciudad Cliente`, un `Tipo Cliente` y un `Plan Cliente`, y que cada `Cliente` tiene varios elementos relacionados con la entidad `Zona del Sistema de Alarma`.

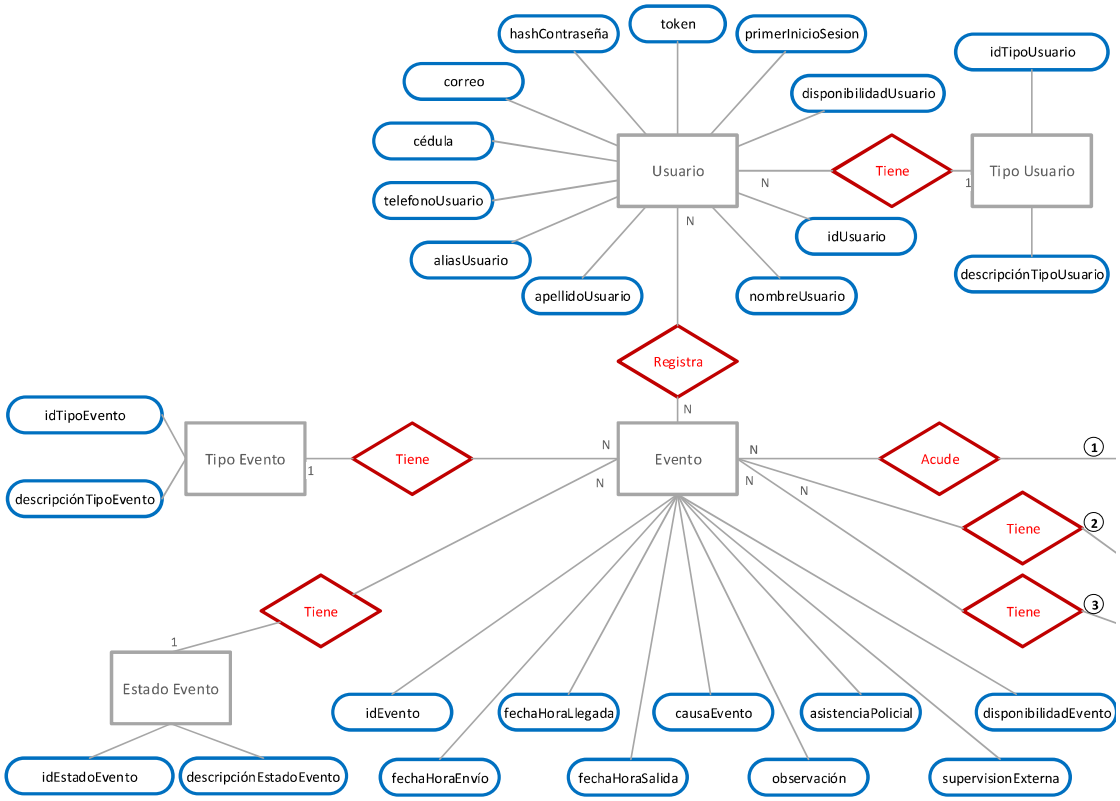


Figura 2.6 Diagrama Entidad-Relación (Parte I de II)

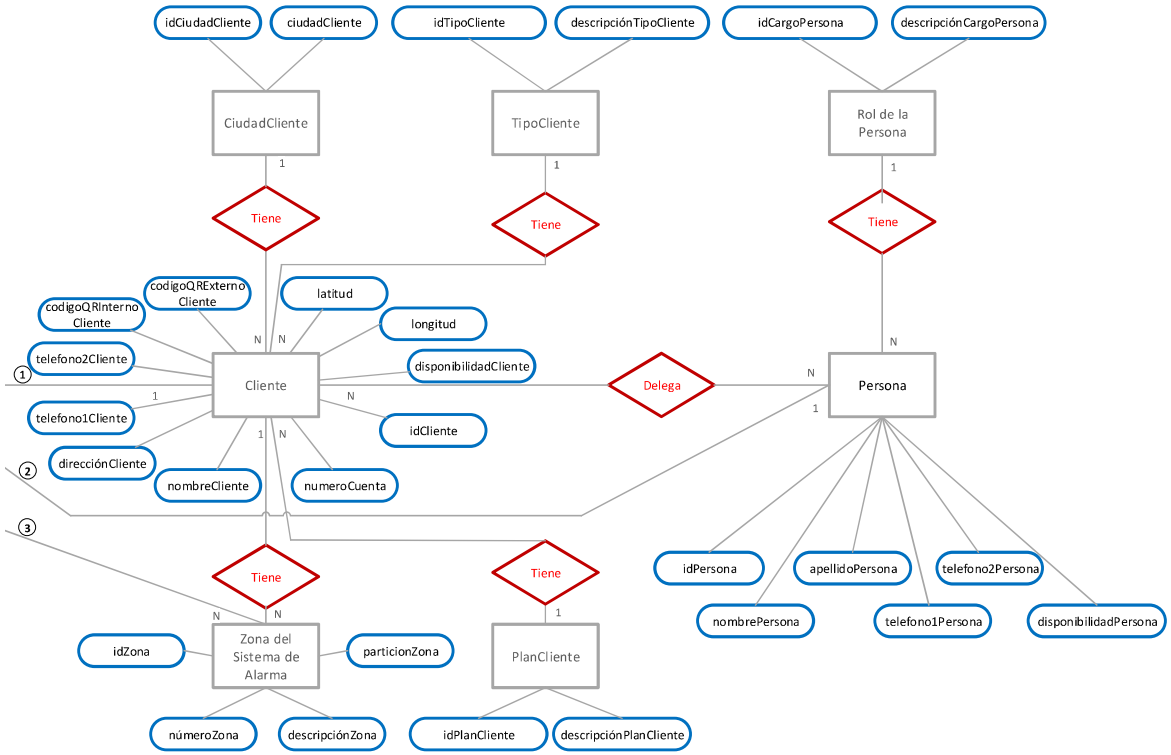


Figura 2.7 Diagrama Entidad-Relación (Parte II de II)

Diseñar el diagrama Relacional: Con base en el diagrama entidad-relación se genera el diagrama relacional que se muestra en la Figura 2.8, la Figura 2.9 y la Figura 2.10.

En el diagrama se definen las tablas y atributos. En los atributos se describen el tipo de datos y la propiedad `Nullable` que para todos los atributos no se la permite, exceptuando el atributo `token`, debido a que, si se asignará un `token` en la creación del usuario, este podría realizar las solicitudes al servicio WCF sin haber iniciado sesión, por tanto, el `token` se crea la primera vez que inicia sesión. Se describen las tablas principales del diagrama relacional:

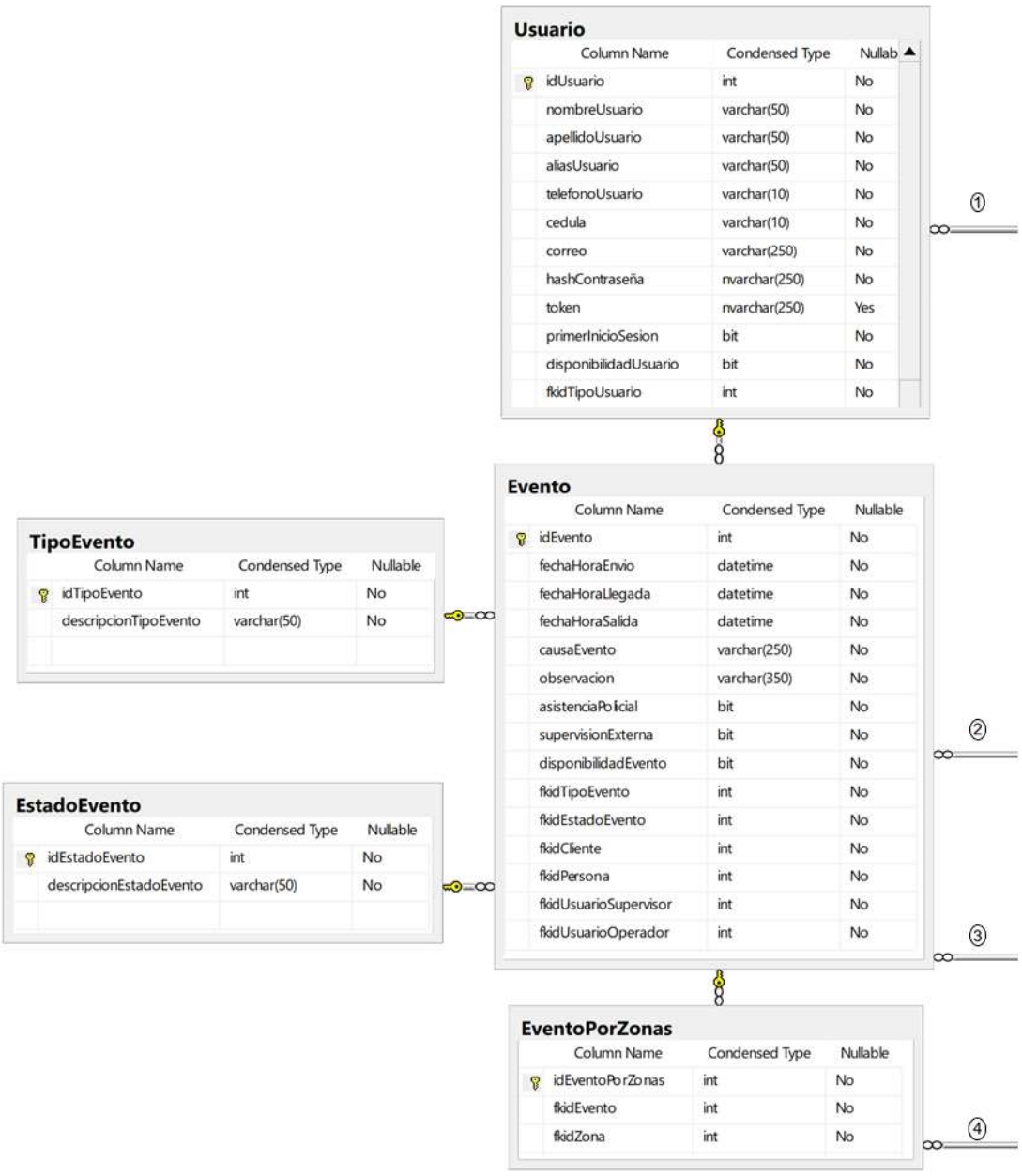


Figura 2.8 Diagrama Relacional (Parte I de III)

Figura 2.9 Diagrama Relacional (Parte II de III)

La tabla `Usuario` almacenará la información en relación con un usuario, como llave primaria está `idUsuario` que sirve como identificador de los registros, la llave foránea `fkidTipoUsuario` sirve para relacionarse con la tabla `TipoUsuario`, otros atributos son el nombre, apellido, alias, teléfono, cedula, correo, el hash de la contraseña almacena el resultado generado del algoritmo SHA-256[30] con la contraseña, el token y las variables

[30] **SHA-256:** Es un algoritmo de hash seguro que convierte un conjunto de datos en una serie de caracteres de longitud fija.

para identificar si es el primer inicio de sesión y si el usuario se encuentra disponible para realizar las funcionalidades del prototipo.

La tabla `Evento` guardará toda la información con respecto a los eventos, como llave primaria está `idEvento` que sirve para identificar los registros, tiene seis llaves foráneas `fkidTipoEvento`, `fkidEstadoEvento`, `fkidCliente`, `fkidPersona`, `fkidUsuarioSupervisor` y `fkidUsuarioOperador` para relacionarse respectivamente con las tablas `TipoEvento`, `EstadoEvento`, `Cliente`, `Persona` y `Usuario` dos veces, debido a que se relaciona para el usuario supervisor y para el usuario operador. Los demás atributos guardan las fechas de envío, llegada y salida, la causa del evento, la observación, si hubo asistencia policial, si la supervisión fue externa y si el evento se encuentra disponible en el prototipo.

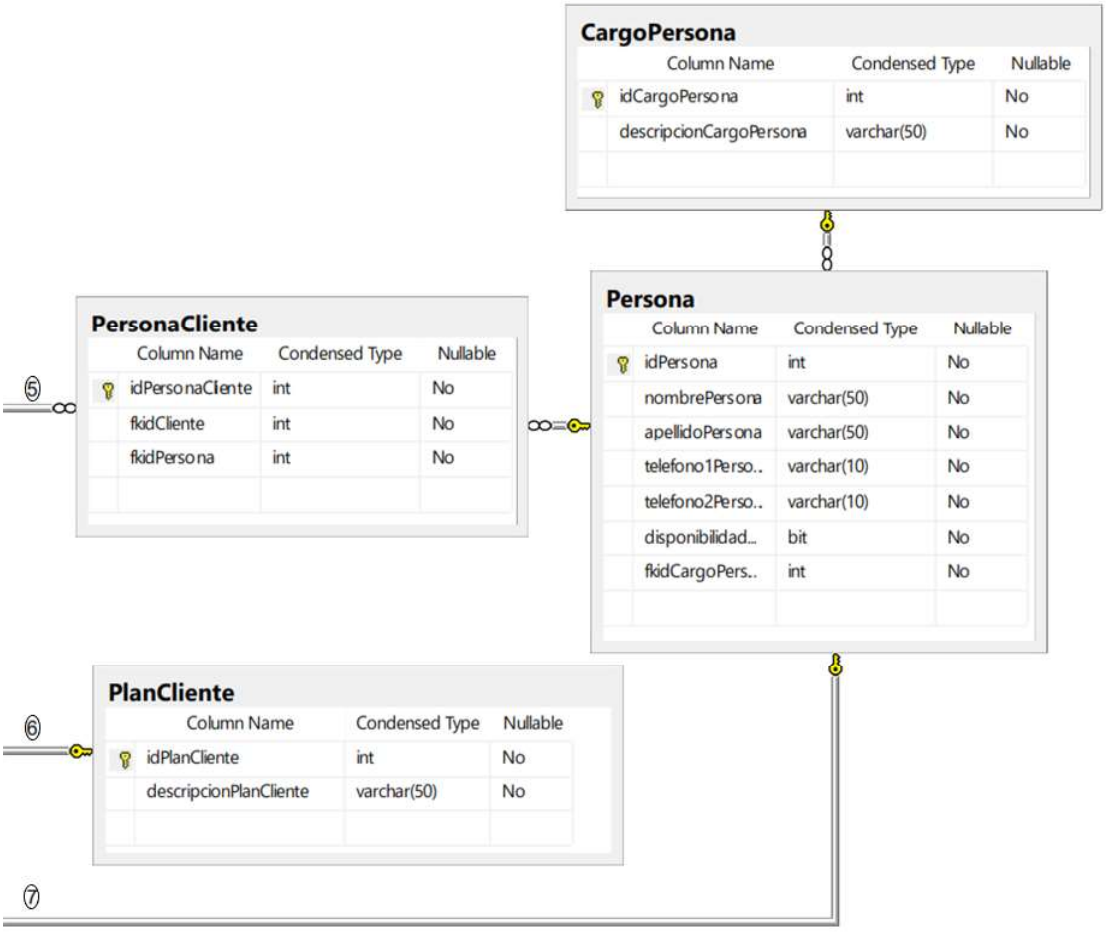


Figura 2.10 Diagrama Relacional (Parte III de III)

La tabla `Cliente` almacenará la información respecto a los clientes, como llave primaria está `idCliente` para identificar los registros, tiene tres llaves foráneas

`fkidPlanCliente`, `fkidTipoCliente` y `fkidCiudadCliente` sirve para relacionarse con las tabla `PlanCliente`, `TipoCliente` y `CiudadCliente`, otros atributos son el número de cuenta, nombre, dirección, teléfono principal, teléfono secundario, la información relacionada con el código QR interno y el código QR externo, la latitud y longitud de la propiedad, y si el cliente se encuentra disponible en el prototipo.

La tabla `Persona` contendrá la información de las personas, como llave primaria está `idPersona` para identificar los diferentes registros, la llave foránea `fkidCargoPersona` sirve para relacionarse con la tabla `CargoPersona`, otros atributos son el nombre, apellido, teléfono principal, teléfono secundario y la disponibilidad de la persona en el prototipo.

La tabla `PersonaCliente` es una tabla intermedia que permitirá la relación entre varios clientes y varias personas, como llave primaria está `idPersonaCliente` para identificar los registros, tiene dos llaves foráneas `fkidCliente` y `fkidPersona` que sirven para relacionarse con las tablas `Cliente` y `Persona` respectivamente.

Definir los actores del prototipo: Para el prototipo se han establecido tres actores, cada uno tendrá determinados privilegios.

Los actores se describen en la Tabla 2.14 y la Tabla 2.15.

Tabla 2.14 Actores del Prototipo (Parte I de II)

Actor	Descripción
Administrador	El administrador cuenta con los privilegios para: Aplicación de escritorio: • Gestionar (crear, modificar, eliminar y visualizar) las cuentas de usuarios (administrador, operador y supervisor). • Gestionar cuentas de clientes y zonas de los sistemas de alarma. • Gestionar cuentas de personas y responsabilidades. • Ingresar, eliminar y visualizar eventos y realizar búsquedas de los eventos según criterios definidos en las historias de usuario. • Configuración de parámetros. • Habilitación de eventos, usuarios, clientes y personas. Aplicación Android: • Configuración de parámetros. • Visualizar la ubicación de los clientes.

Tabla 2.15 Actores del Prototipo (Parte II de II)

Operador	El operador cuenta con los privilegios para: Aplicación de escritorio: • Ingresar cuentas de clientes y gestionar zonas de los sistemas de alarma. • Ingresar y modificar cuentas de personas y responsabilidades. • Ingresar y visualizar eventos, y realizar búsquedas de los eventos.
Supervisor	El supervisor cuenta con los privilegios para: Aplicación Android: • Registrar los eventos en casos de emergencia, ya sea por pánico, emergencia médica o activación del sistema de alarma. • Visualizar la ubicación de los clientes.

Crear la base de datos: Para la creación de la base de datos se realiza un *script* con los comandos necesarios, la base a su vez contiene tablas y atributos. El *script* se lo ejecuta en el gestor de base de datos SQL Server 2014 Management Studio.

En el Código 2.1 se presenta un fragmento del *script* para la creación de la base de datos y sus tablas; en la línea 2 se crea la base de datos `PrototipoSeguridad`; la línea 3 selecciona la base de datos. Entre las líneas 6 a 15 se crea la tabla `Persona` y sus atributos, en la línea 7 se define la llave primaria `idPersona` que será única y autoincremental. Entre las líneas 8 y 12 se definen los atributos obligatorios; en las líneas 14 y 15 se establece la llave foránea `fkidCargoPersona` para relacionarse con la tabla `CargoPersona` que tiene las restricciones de actualizar la llave foránea en la tabla `Persona` si se actualiza la llave primaria de la tabla `CargoPersona`. Además de no eliminación de un registro de la tabla `CargoPersona` si se encuentra relacionado con un valor de llave foránea de la tabla `Persona`.

```
2  CREATE DATABASE PrototipoSeguridad
3  use PrototipoSeguridad
4  go
5  ---------------------------------CREACIÓN-DE-TABLAS----------------------
6  CREATE TABLE Persona (
7      idPersona int primary key identity(1,1),
8      nombrePersona varchar(50) NOT NULL,
9      apellidoPersona varchar(50) NOT NULL,
10     telefono1Persona varchar(10) NOT NULL,
11     telefono2Persona varchar(10) NOT NULL,
12     disponibilidadPersona bit NOT NULL,
13
14     fkidCargoPersona int NOT NULL foreign key references
15     CargoPersona (idCargoPersona) on update cascade on delete no action,
16 );
```

Código 2.1 Creación de la Base de Datos y Tabla `Persona`

El *script* con todo el código para la creación de la base de datos, tablas y atributos se encuentra en el Anexo C.

2.5.2 SERVICIO WEB

El servicio web es el encargado de conectar las solicitudes de la aplicación Android y la aplicación de escritorio con la base de datos, para lo cual se establece en el servicio web una interfaz y clase que maneje las solicitudes de la aplicación Android y otra interfaz y clase que maneje las solicitudes de la aplicación de escritorio.

Definir las clases e interfaces del servicio: En la Figura 2.11 se exhiben la interfaz `IAplicacionMovil` donde se especifican las operaciones del servicio con sus respectivas URI (*Uniform Resource Identifier*)[31] y la clase `AplicacionMovil` que permitirá desarrollar las funcionalidades de las operaciones.

Posteriormente, se describen los métodos más relevantes de `AplicacionMovil`.

- `AutenticacionUsuario`: permite validar las credenciales del usuario y retorna el objeto del usuario autenticado.

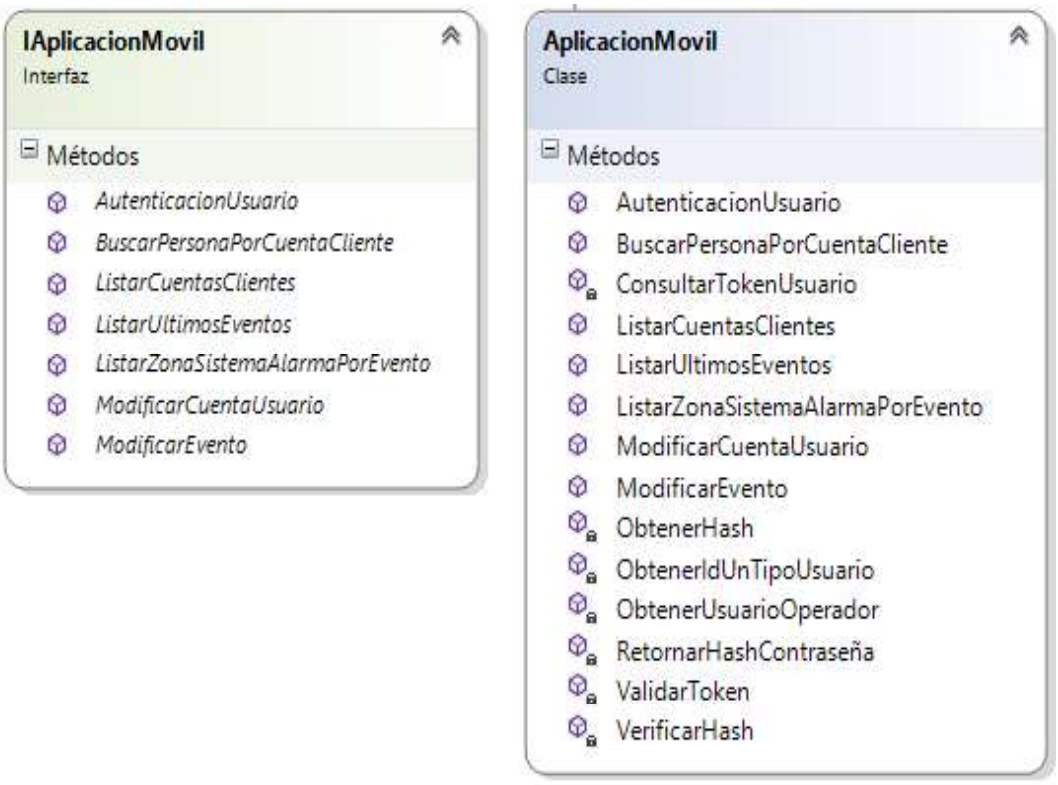


Figura 2.11 Clase e Interfaz del servicio web para la aplicación Android

[31] **URI:** Es una secuencia de datos que permite identificar un recurso en una red.

- `BuscarPersonaPorCuentaCliente`: el argumento de entrada es el número de cuenta y retorna el objeto de la persona, que coincida con el argumento.
- `ListarUltimosEventos`: los argumentos de entrada son el alias de usuario y la disponibilidad de los eventos y retorna una lista de objetos eventos, que coincidan con los argumentos.
- `ModificarEvento`: como argumentos de entrada está la información del evento como el id, fecha de envío, llegada, salida, causa del evento, observación, asistencia policial, supervisión externa, disponibilidad del evento, e identificadores del tipo del evento, estado del evento, cliente, persona, usuario supervisor y operador, retorna un valor entero para informar la modificación del evento.
- `ValidarToken`: los argumentos de entrada son el token por validar y el id del usuario, y retorna un valor `true` si el token del argumento de entrada coincide con el token almacenado para el usuario, caso contrario retorna un `false`.

En la Figura 2.12 y la Figura 2.13 se muestra la interfaz `IAplicacionEscritorio` donde se especifican las operaciones del servicio con sus respectivas URI y la clase `AplicacionEscritorio` que permitirá desarrollar las funcionalidades de las operaciones.

Posteriormente, se describen los métodos más relevantes de `AplicacionEscritorio` y todavía no descritos de la clase anterior.

- `DisponibilidadCuentaCliente`: permite cambiar el campo disponibilidad de un cliente, los argumentos de entrada son el id del cliente y el valor de disponibilidad, y retorna un valor entero para informar el cambio del campo.
- `EliminarCargoPersona`: permite eliminar un registro de la tabla `CargoPersona`, el argumento de entrada es el id del cargo de la persona, y retorna un valor entero para informar la eliminación.
- `IngresarCargoPersona`: permite ingresar un registro en la tabla `CargoPersona`, el argumento de entrada es la descripción del cargo de la persona, y retorna un valor entero para informar sobre el ingreso.
- `ObtenerCargosPersonas`: retorna una lista de todos los objetos del cargo de la persona.
- `VerificarHash`: como argumentos de entrada está la contraseña y el hash almacenado de un usuario, para la contraseña se vuelve a generar el hash y se compara con el hash almacenado, retorna un valor booleano para informar el resultado de la comparación.

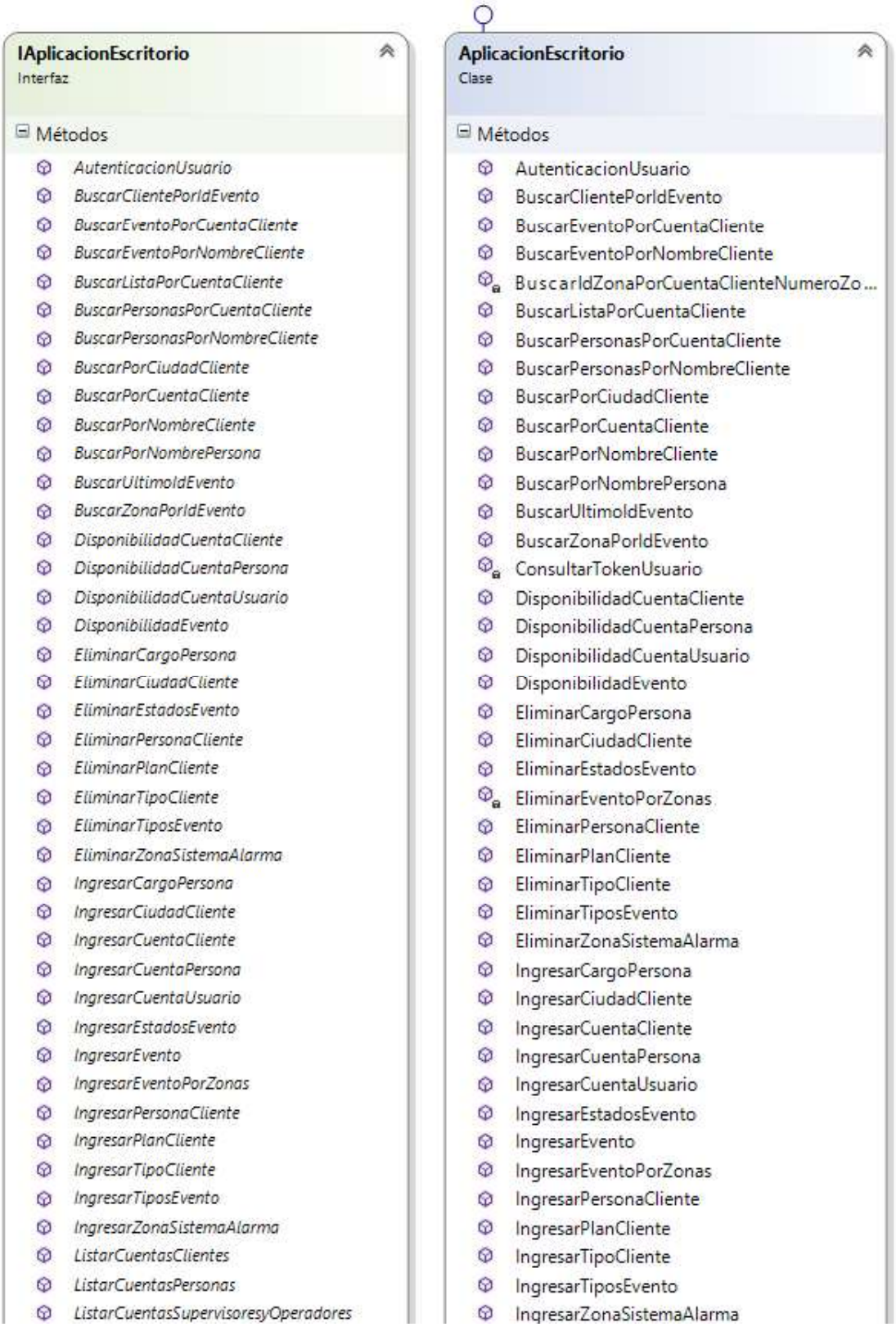


Figura 2.12 Clase e Interfaz del servicio web para la aplicación de escritorio (Parte I de II)

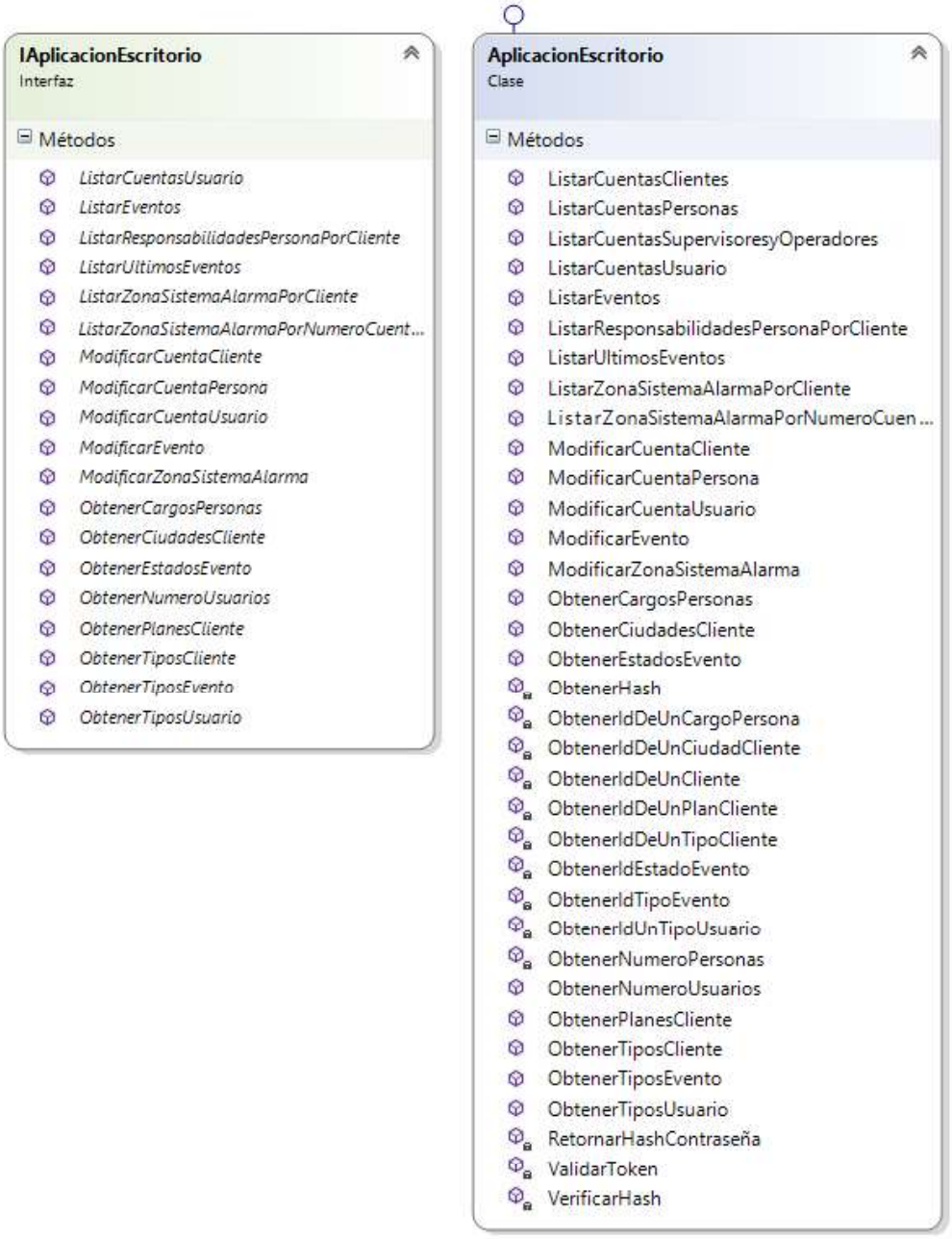


Figura 2.13 Clase e Interfaz del servicio web para la aplicación de escritorio (Parte II de II)

Crear el servicio web: A través del IDE Visual Studio 2015 se crea una aplicación de servicios WCF; en la aplicación de servicios se crean las interfaces y las clases que manejan las solicitudes de la aplicación Android y la aplicación de escritorio.

En la Figura 2.14 se presenta el explorador de soluciones de Visual Studio en el cual se encuentra la aplicación de servicios WCF `ServicioWCF` conformada por:

- `Properties`: Las propiedades de la aplicación de servicios WCF que almacena valores como la URL del servicio web.
- `Referencias`: Donde se almacenan las referencias que se usarán en el proyecto.
- `AplicacionEscritorio.svc` y `AplicacionMovil.svc`: Puntos de entrada al servicio WCF
- `AplicacionEscritorio.svc.cs` y `AplicacionMovil.svc.cs`: Las clases del servicio WCF
- `BDDPrototipoSeguridadEF`: La conexión a la base de datos usando Entity Framework.
- `IAplicacionEscritorio.cs` y `IAplicacionMovil.cs`: Las interfaces del servicio WCF.
- `Packages.config`: El archivo de configuración que mantiene una lista de los paquetes referenciados en el proyecto.
- `Web.config`: La configuración web es un documento XML que contiene la información sobre seguridad, los servicios y la cadena de conexión a la base de datos.

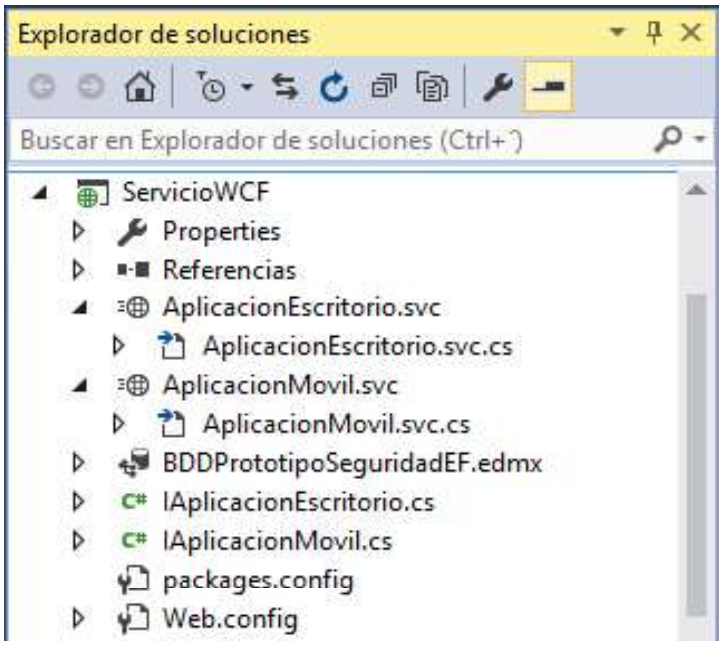


Figura 2.14 Vista del Explorador de Soluciones de la aplicación de servicios WCF

En el Código 2.2 se observa un contrato de operación del servicio web que permite ingresar un tipo de cliente a la base de datos con el parámetro descripción del tipo de cliente. En la línea 114 se especifica el atributo `OperationContract` para definir que el método será expuesto a través del servicio WCF.

En la línea 115 se especifica el atributo `WebInvoke` que determina a qué método HTTP (POST) responde la operación; en la línea 116 se detalla el formato de datos de la solicitud; en la línea 117 se detalla el formato de datos de la respuesta.

La línea 118 establece el estilo del cuerpo de los mensajes que se envían desde y hacia la operación; en la línea 119 se detalla la URI para reconocer al método.

En la línea 120 se define el método `IngresarTipoCliente`, que tiene como argumento de entrada la `descripcionTipoCliente` y cuyo retorno será un entero para informar acerca del éxito en el ingreso.

```
114     [OperationContract]
115     [WebInvoke(Method = "POST",
116                RequestFormat = WebMessageFormat.Json,
117                ResponseFormat = WebMessageFormat.Json,
118                BodyStyle = WebMessageBodyStyle.Bare,
119                UriTemplate = "/IngresarTipoCliente/{descripcionTipoCliente}")]
120     int IngresarTipoCliente(string descripcionTipoCliente);
```

Código 2.2 Definición de la operación `IngresarTipoCliente`

En el Código 2.3 se muestra la implementación del método anterior, en la línea 891 se crea una variable que recogerá el resultado de la validación del Id y token.

En las líneas 894 y 895 se extraen de la cabecera el Id y token; en la línea 132 se utiliza el método para validar los tokens, si la validación es correcta se retornará un valor `true`, caso contrario un valor `false`.

Se continúa a la línea 903 si la validación es correcta, donde se crea un objeto `db` que representa las entidades de la base de datos generadas por Entity Framework; la línea 906 crea el objeto de tipo cliente a almacenar en la base de datos.

En la línea 909 se establece el atributo del objeto utilizando el argumento de entrada; utilizando el objeto `db` se agrega el objeto de tipo cliente a la base de datos y se guardan cambios (líneas 911 y 914).

Finalmente, se retorna el resultado de la agregación del objeto (línea 915). Si la validación es incorrecta se continúa a la línea 920 que retorna un valor entero de -1 para indicar que no se ingresó el tipo de cliente.

Además, entre las líneas 923 y 926 se captura la excepción producida por errores con Entity Framework, y en las líneas 927 a 930 se capturan los errores producidos en la ejecución de la aplicación.

```
886     public int IngresarTipoCliente(string descripcionTipoCliente)
887     {
888         try
889         {
890             //Variable de validacion del token
891             bool validacionToken = false;
892
893             //Leer los valores del la cabecera Id y Token
894             string idUsuarioToken = WebOperationContext.Current.IncomingRequest.Headers["Id"];
895             string tokenHeader = WebOperationContext.Current.IncomingRequest.Headers["Token"];
896
897             //Método para Validar los tokens
898             validacionToken = ValidarToken(tokenHeader, idUsuarioToken);
899
900             //Condicional que evalúa la validacion del token
901             if (validacionToken)
902             {
903                 using (DBPrototipoSeguridadEntities db = new DBPrototipoSeguridadEntities())
904                 {
905                     //Creación del objeto tipoCliente
906                     TipoClienteEF tipoCliente = new TipoClienteEF();
907
908                     //Agregación del campo descripcionTipoCliente al objeto
909                     tipoCliente.descripcionTipoCliente = descripcionTipoCliente;
910
911                     db.TipoCliente.Add(tipoCliente);
912
913                     //Resultado de almacenar el objeto en la base de datos
914                     int resultadoCambios = db.SaveChanges();
915                     return resultadoCambios;
916                 }
917             }
918             else
919             {
920                 return -1;
921             }
922         }
923         catch (System.Data.Entity.Core.EntityException)
924         {
925             return -1;
926         }
927         catch (Exception ex)
928         {
929             return -1;
930         }
```

Código 2.3 Método para ingresar tipos de clientes en la base de datos

En el IIS, se procede a crear un sitio web como se observa en la Figura 2.15 que alojará al servicio WCF. En el sitio web se asigna el puerto para escuchar las peticiones 8200 de las aplicaciones, y se configura la ruta de acceso a los archivos del servicio web.

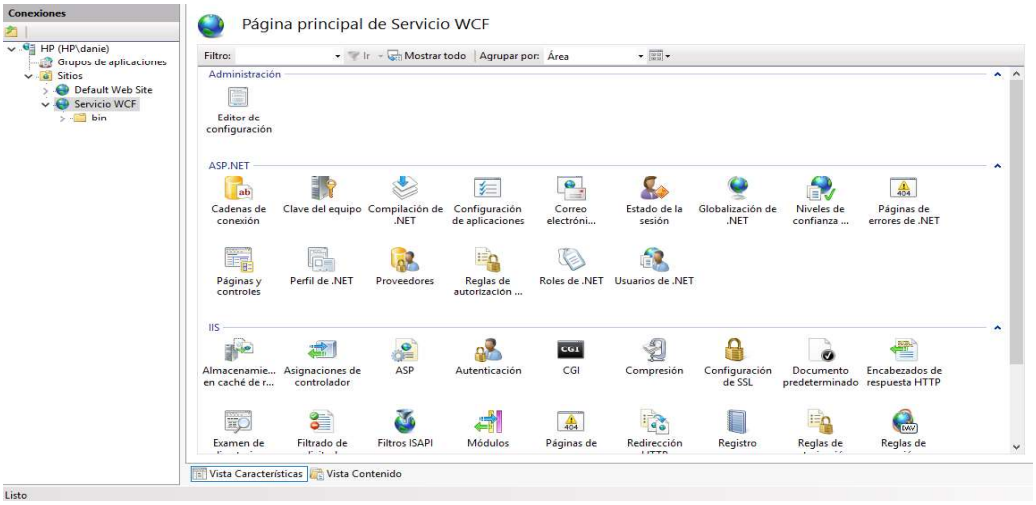


Figura 2.15 Página principal del sitio web

2.5.3 APLICACIÓN DE ESCRITORIO

Para gestionar la información del prototipo se crea la aplicación de escritorio. Se diseña el diagrama de clases UML referenciándose en el diagrama relacional de la base de datos, también el diagrama de secuencia del prototipo con la aplicación, así como las *sketches* principales y un diagrama de actividades.

Diseñar el diagrama de clases UML: En la Figura 2.16 y la Figura 2.17 se muestra el diagrama UML que se conforma de las clases:

- `Conexion`: Permitirá realizar las solicitudes al servicio web a través de métodos definidos.
- `UsuarioApp`: Permitirá registrar la información de los usuarios.
- `TipoUsuarioApp`: Permitirá registrar la información de los tipos de usuarios como: administrador, operador y supervisor.
- `EventoApp`: Permitirá registrar toda la información de los eventos.
- `TipoEventoApp`: Permitirá almacenar la información de los tipos de eventos como: pánico, emergencia médica y robo.
- `EstadoEventoApp`: Permitirá manejar la información de los estados del evento como: enviado, en desarrollo y finalizado.
- `ClienteApp`: Permitirá registrar la información de los clientes.
- `CiudadClienteApp`: Permitirá registrar las ciudades de los clientes como: Quito, Ibarra, Cayambe y Otavalo.
- `TipoClienteApp`: Manejará los tipos de clientes como: domicilio, industria, entidad financiera y centro educativo.
- `PlanClienteApp`: Manejará los planes de clientes como: super-premium, premium y estándar.
- `ZonaSistemaAlarmaApp`: Permitirá registrar la información de las zonas de los clientes.
- `PersonaApp`: Permitirá registrar la información de las personas.
- `CargoPersonaApp`: Manejará los cargos de las personas como: gerente, secretaria, cajera, conserje, padre, madre e hijo.
- `PersonaClienteApp`: Permitirá relacionar a múltiples personas con múltiples clientes, sirve como intermediaria entre las clases `PersonaApp y ClienteApp`, sirve para que múltiples personas sean registradas como responsables de múltiples clientes.

- `EventoPorZonasApp`: Permitirá registrar a múltiples eventos con múltiples zonas activadas, sirve como intermediaria entre las clases `EventoApp` y `ZonaSistemaAlarmaApp`, sirve para que múltiples eventos sean registrados con múltiples zonas.

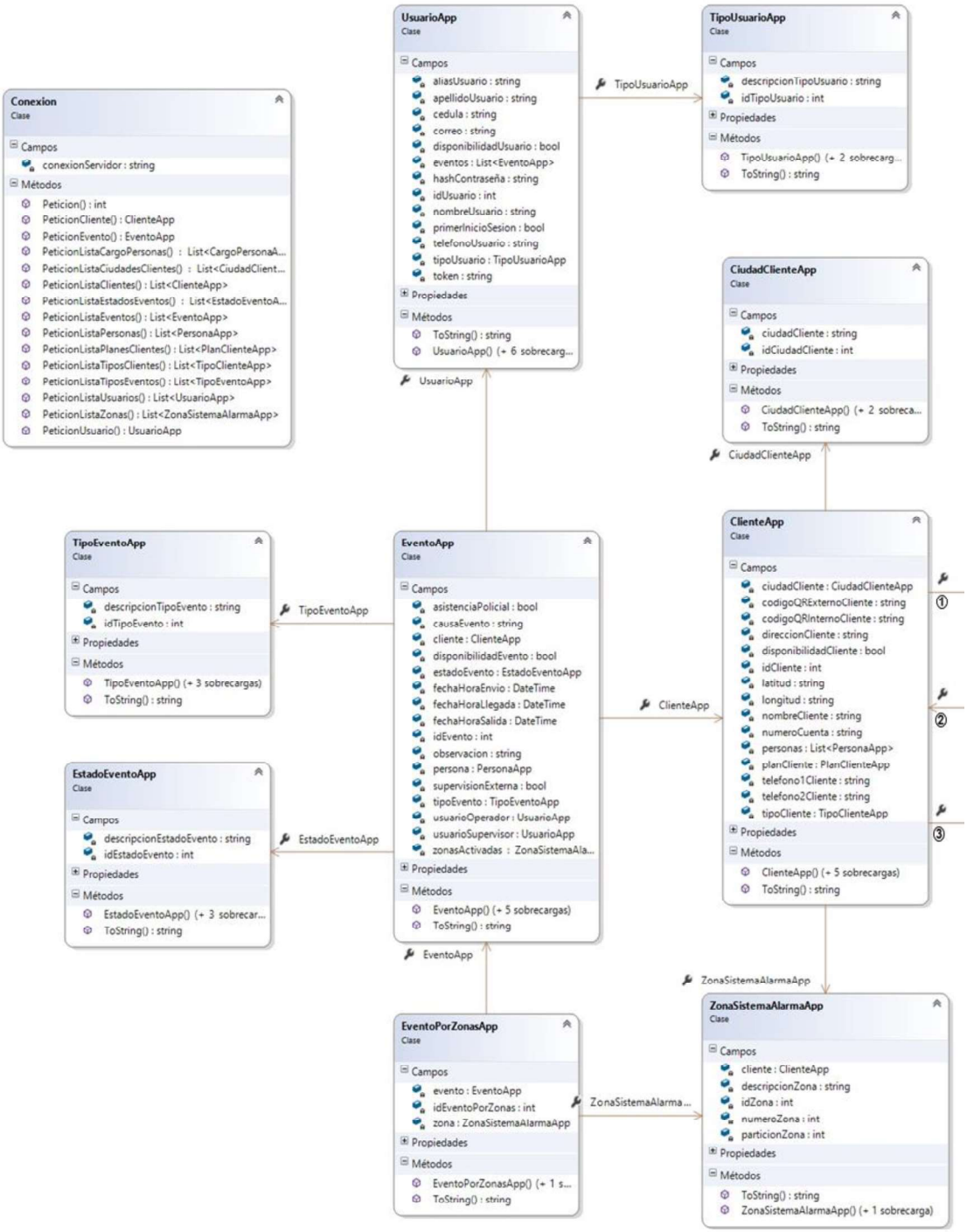


Figura 2.16 Diagrama de Clases de la aplicación de escritorio (Parte I de II)

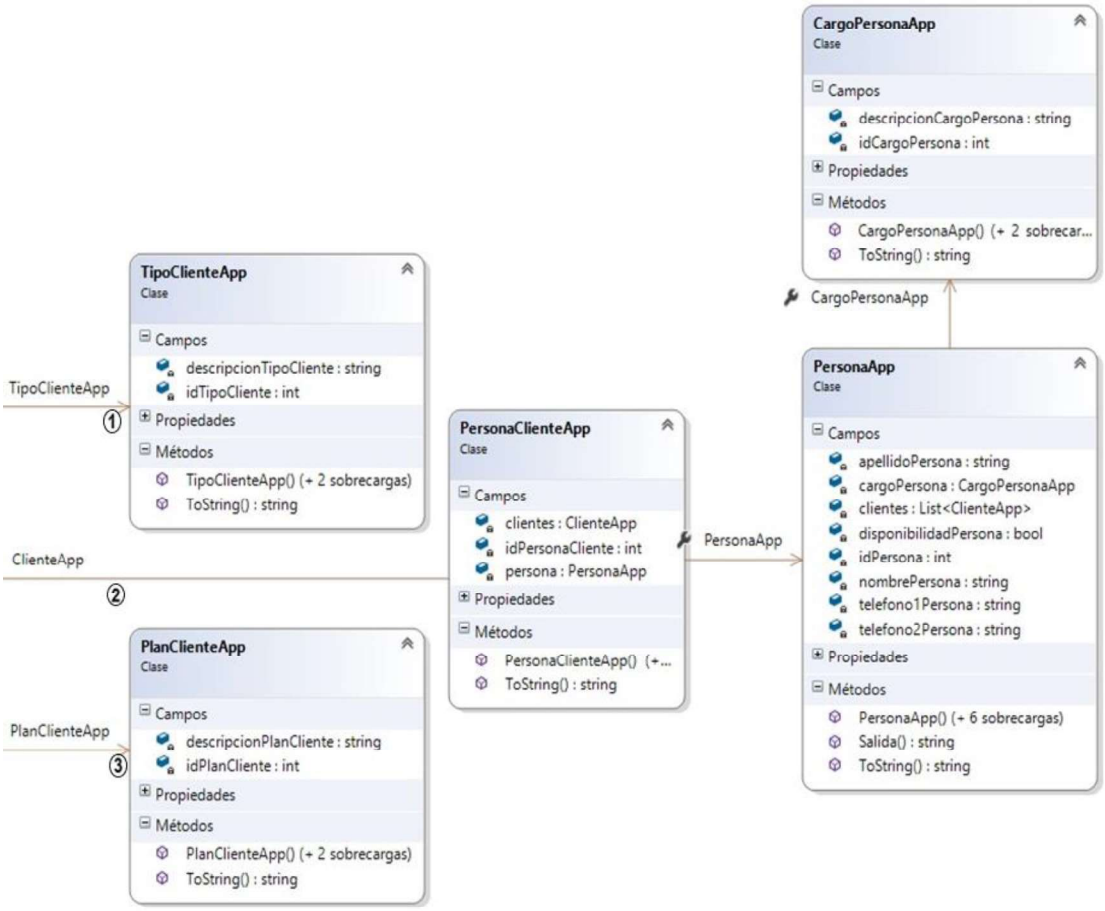


Figura 2.17 Diagrama de Clases de la aplicación de escritorio (Parte II de II)

Diseñar el diagrama de secuencia: En la Figura 2.18 se muestra el diagrama de secuencia del prototipo con la aplicación de escritorio, el cual muestra lo que se realiza cuando el usuario hace una solicitud desde la aplicación de escritorio hacia el servicio WCF.

Primero, el usuario utiliza una funcionalidad de la aplicación de escritorio mediante las interfaces de usuario para ingresar o solicitar datos.

La aplicación de escritorio es la encargada de generar la solicitud hacia el servicio WCF ubicado en el servidor.

Del lado del servidor se recepta la solicitud y se llama al método adecuado del servicio WCF para atender la solicitud, la información requerida se toma de la base de datos.

La base de datos retorna la información requerida al servicio WCF, que a su vez devuelve a la aplicación de escritorio del lado del cliente, para finalmente mostrar la información al usuario.

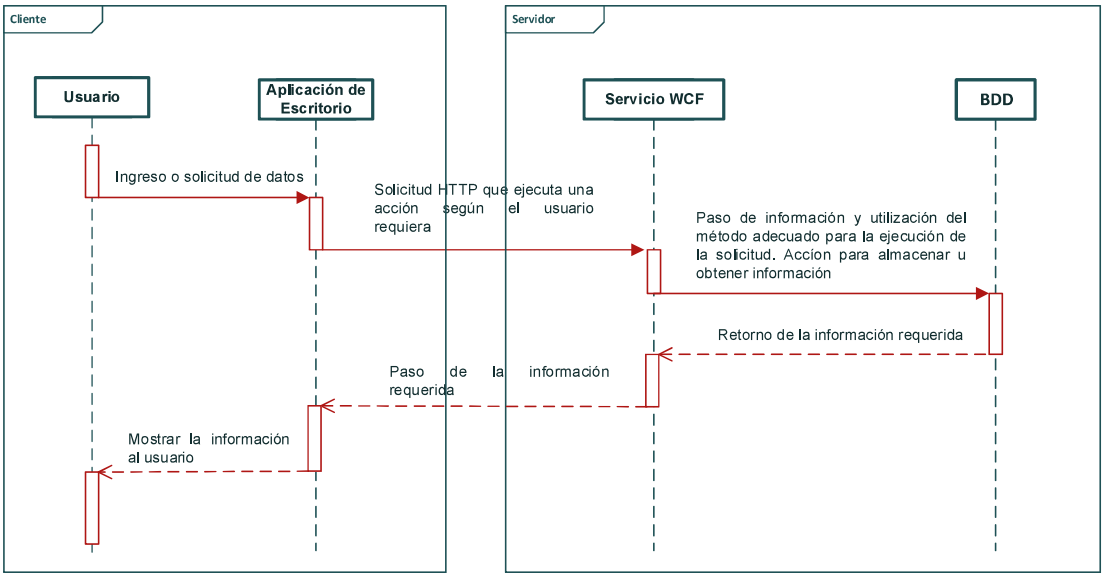


Figura 2.18 Diagrama de Secuencia del prototipo con la aplicación de escritorio

Diseñar las *sketches* principales: En la Figura 2.19 se muestra el *sketch* para autenticar usuarios, en el cual se ingresarán el alias y contraseña para iniciar sesión en la aplicación de escritorio.

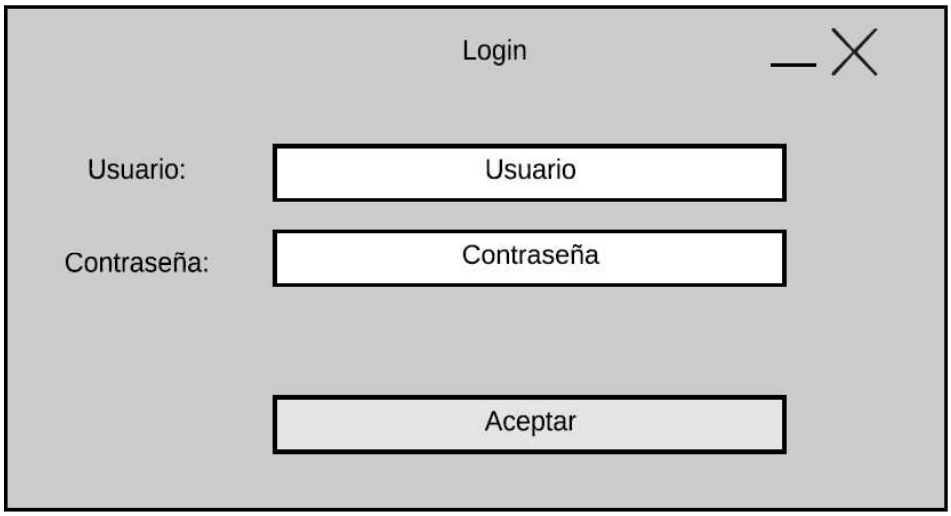


Figura 2.19 *Sketch* para autenticar usuarios

La Figura 2.20 presenta el *sketch* para gestionar eventos, que cuenta con una barra de color naranja en la parte izquierda para navegar a través de la aplicación.

En el recuadro uno se mostrarán los eventos de las últimas 24 horas; en la parte superior derecha habrá dos botones para ingresar o eliminar eventos. En el recuadro dos se muestra la información complementaria del evento, cliente, zonas y personas en función del evento seleccionado en el recuadro uno.

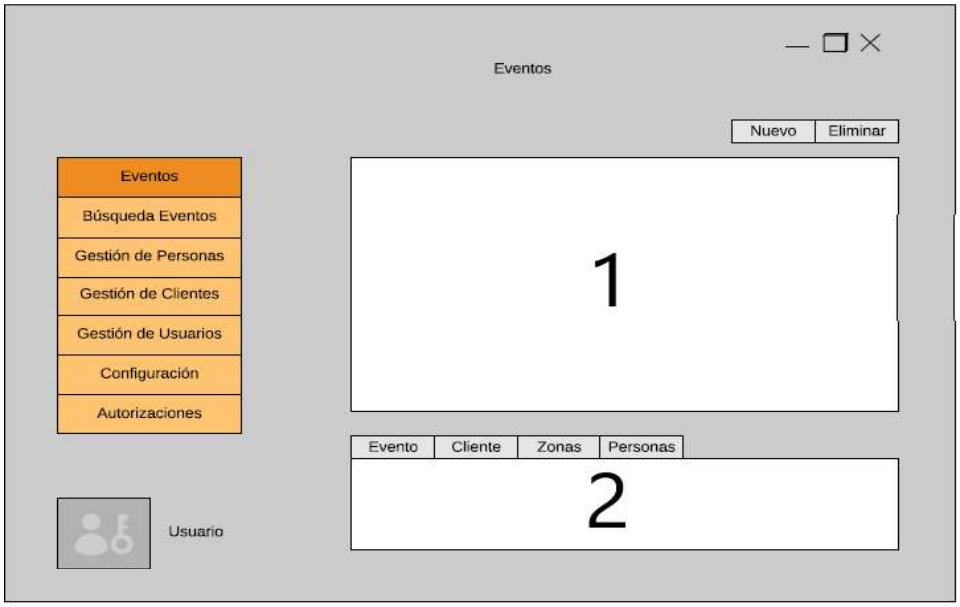


Figura 2.20 *Sketch* para gestionar eventos

En la Figura 2.21 se muestra el *sketch* para gestionar personas, el cual mantiene la barra de navegación del anterior *sketch*. En el recuadro blanco se visualizarán todas las personas registradas. En la parte superior al recuadro está la sección que realizará búsquedas de personas. En la parte superior habrá los botones para crear, modificar y eliminar personas. El botón `Responsabilidad` servirá para asociar a una persona la responsabilidad de uno o varios clientes.

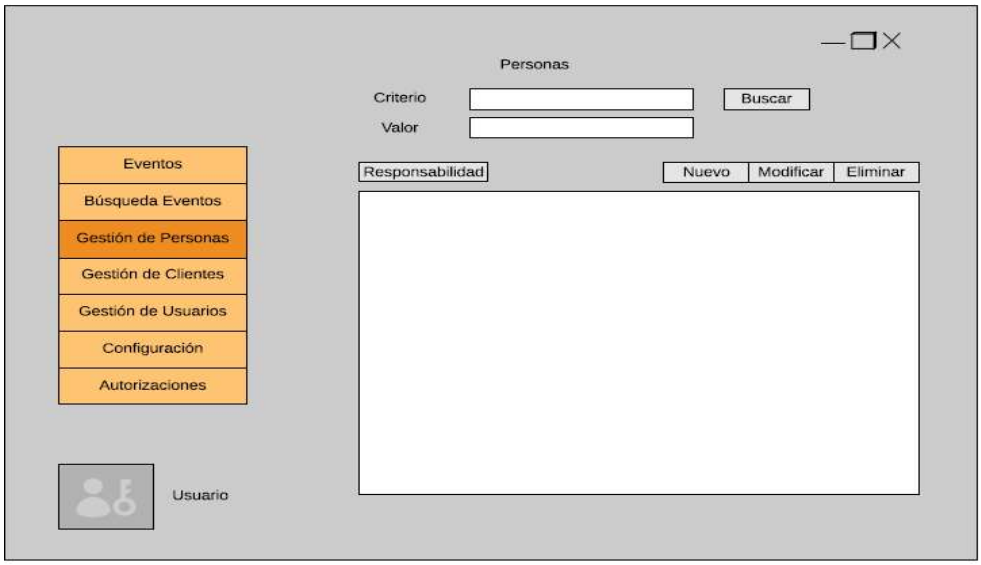


Figura 2.21 *Sketch* para gestionar personas

La Figura 2.22 muestra el *sketch* para ingresar y modificar los campos nombre, apellido, rol, teléfono y teléfono secundario de una persona.

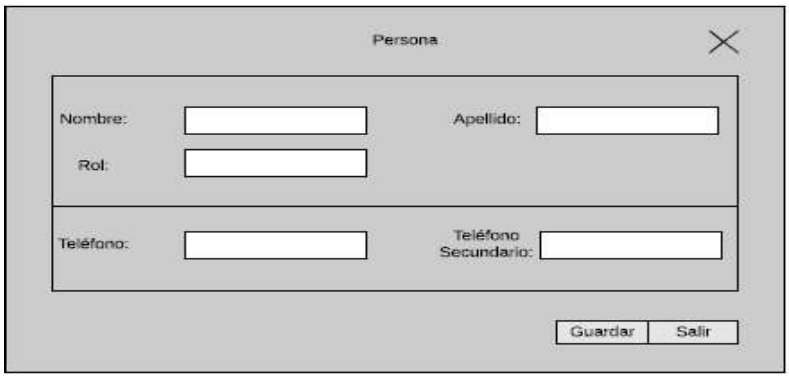


Figura 2.22 *Sketch* para ingresar y modificar personas

Diseñar el diagrama de actividades principal: La Figura 2.23 indica el diagrama de actividades cuando se crea un evento. El proceso inicia cuando el administrador u operador ingresa la información del evento. La aplicación validará la información ingresada, si es válida, la aplicación registrará el evento en la base de datos, caso contrario podrán nuevamente iniciar el proceso. Luego el administrador u operador ingresa las zonas activadas. La aplicación validará las zonas ingresadas, si es válida procede a registrar, caso contrario podrán nuevamente ingresar las zonas. Por último, el administrador u operador visualizarán el evento y así finaliza el proceso.

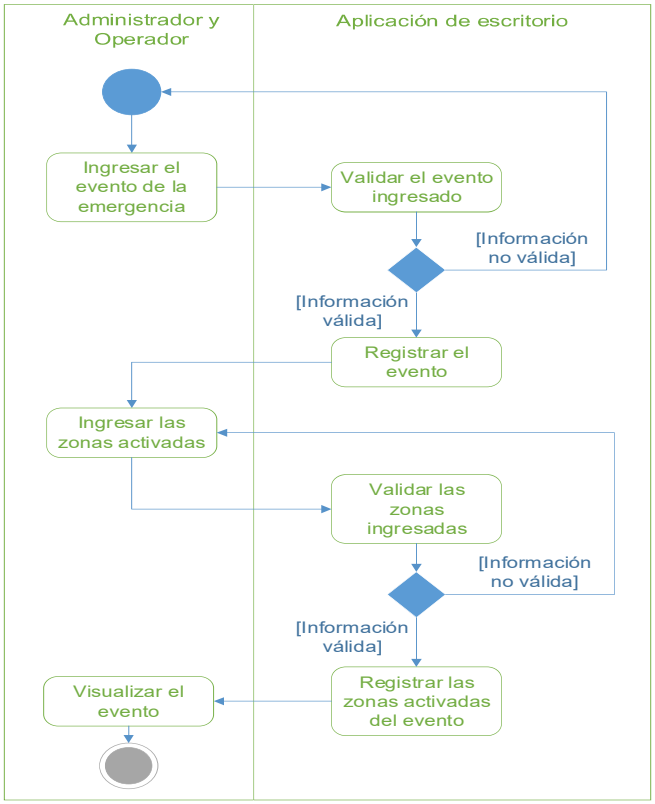


Figura 2.23 Diagrama de actividades al crear un evento

2.5.4 APLICACIÓN ANDROID

Para registrar los eventos del prototipo se crea la aplicación Android. Se diseña el diagrama de clases UML, también el diagrama de secuencia del prototipo con la aplicación, así como las *sketches* principales y un diagrama de actividades.

Diseñar el diagrama de clases UML: En la Figura 2.24, la Figura 2.25 y la Figura 2.26 se muestra el diagrama de clases UML para la aplicación Android, se describirán brevemente las clases diferentes a las mencionadas en la aplicación de escritorio:

- `UsuarioApp`, `TipoUsuarioApp`, `EventoApp`, `TipoEventoApp`, `EstadoEventoApp`, `ClienteApp`, `CiudadClienteApp`, `TipoClienteApp`, `PlanClienteApp`, `ZonaSistemaAlarmaApp`, `PersonaApp` y `CargoPersonaApp`.

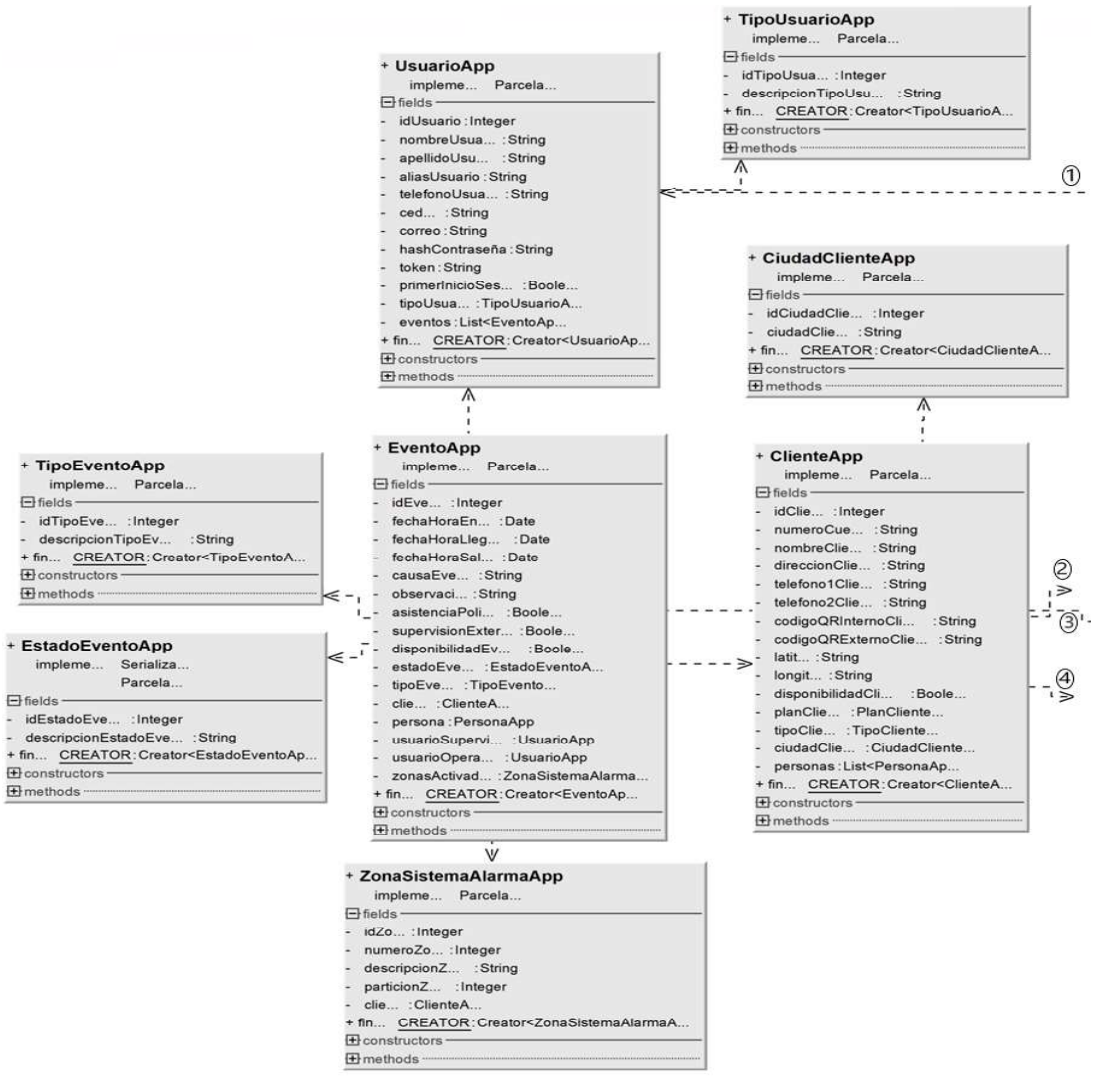


Figura 2.24 Diagrama de Clases de la aplicación Android (Parte I de III)

- `ConexionRestful`: Contiene los métodos que realizan las peticiones al servicio web.
- `Main`: Donde se ejecutan las funcionalidades como: la autenticación, la presentación de la barra lateral de navegación, el cambio de contraseña y la captura de la ubicación del dispositivo.
- `ListaEventos`: Gestionará la presentación de los eventos al usuario según los estados del evento: enviado, en desarrollo y finalizado.
- `AdaptadorEventoFinalizado`: Permitirá mostrar la información de los eventos con estado finalizado.
- `AdaptadorZonaActivada`: Permitirá mostrar la información de una zona del sistema de alarma.

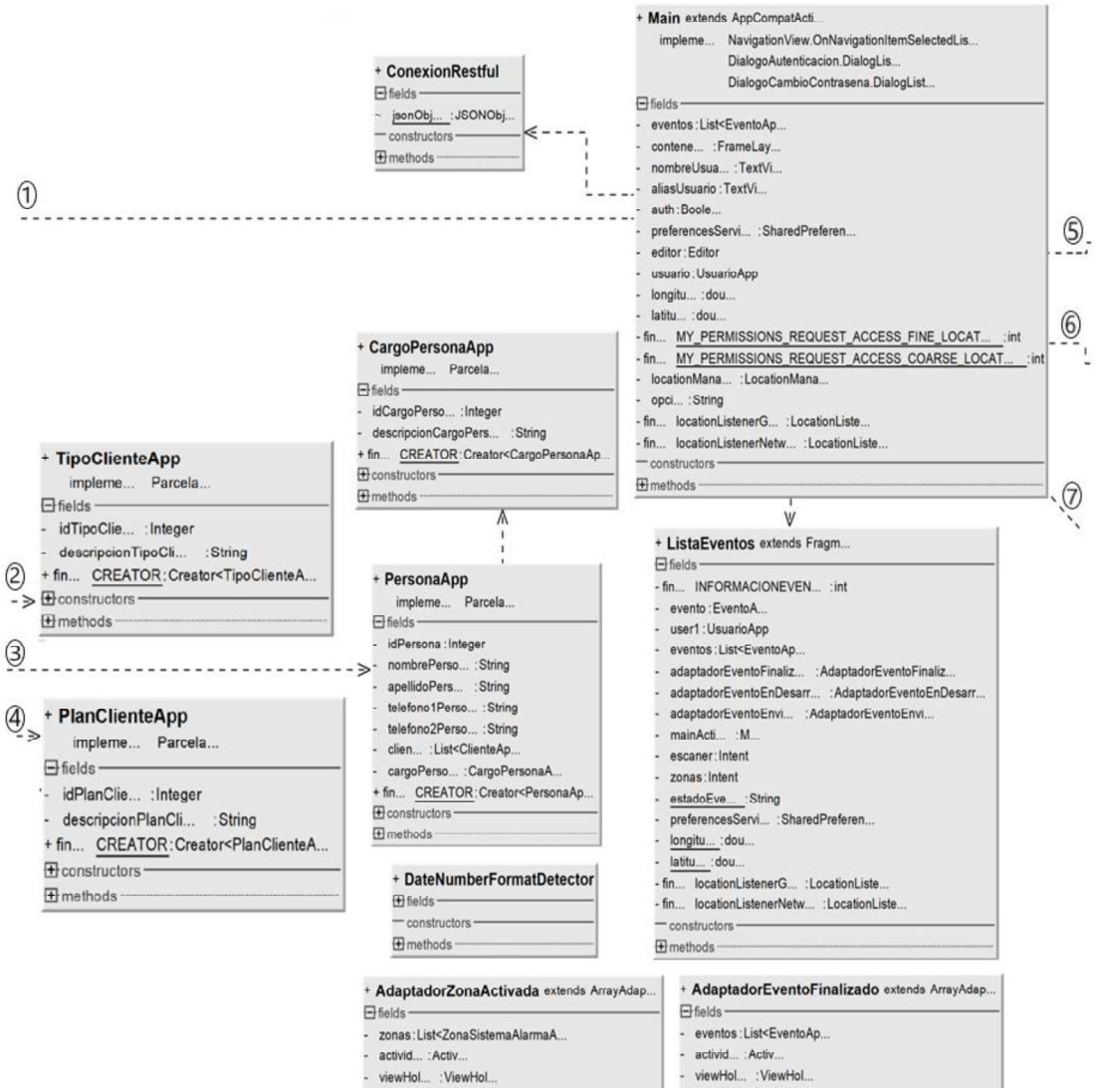


Figura 2.25 Diagrama de Clases de la aplicación Android (Parte II de III)

- `Preferencias`: Permitirá registrar la URL del servicio web, el tiempo de conexión y tiempo de lectura máximos.
- `Escaneo`: Permitirá ejecutar la funcionalidad para escanear un código QR a través de la cámara del dispositivo Android.
- `GoogleMapsFragmento`: Proporcionará la vista del mapa de Google, cuenta con funcionalidades como ubicar el dispositivo, cambiar el tipo de mapa y visualizar las direcciones de los clientes.

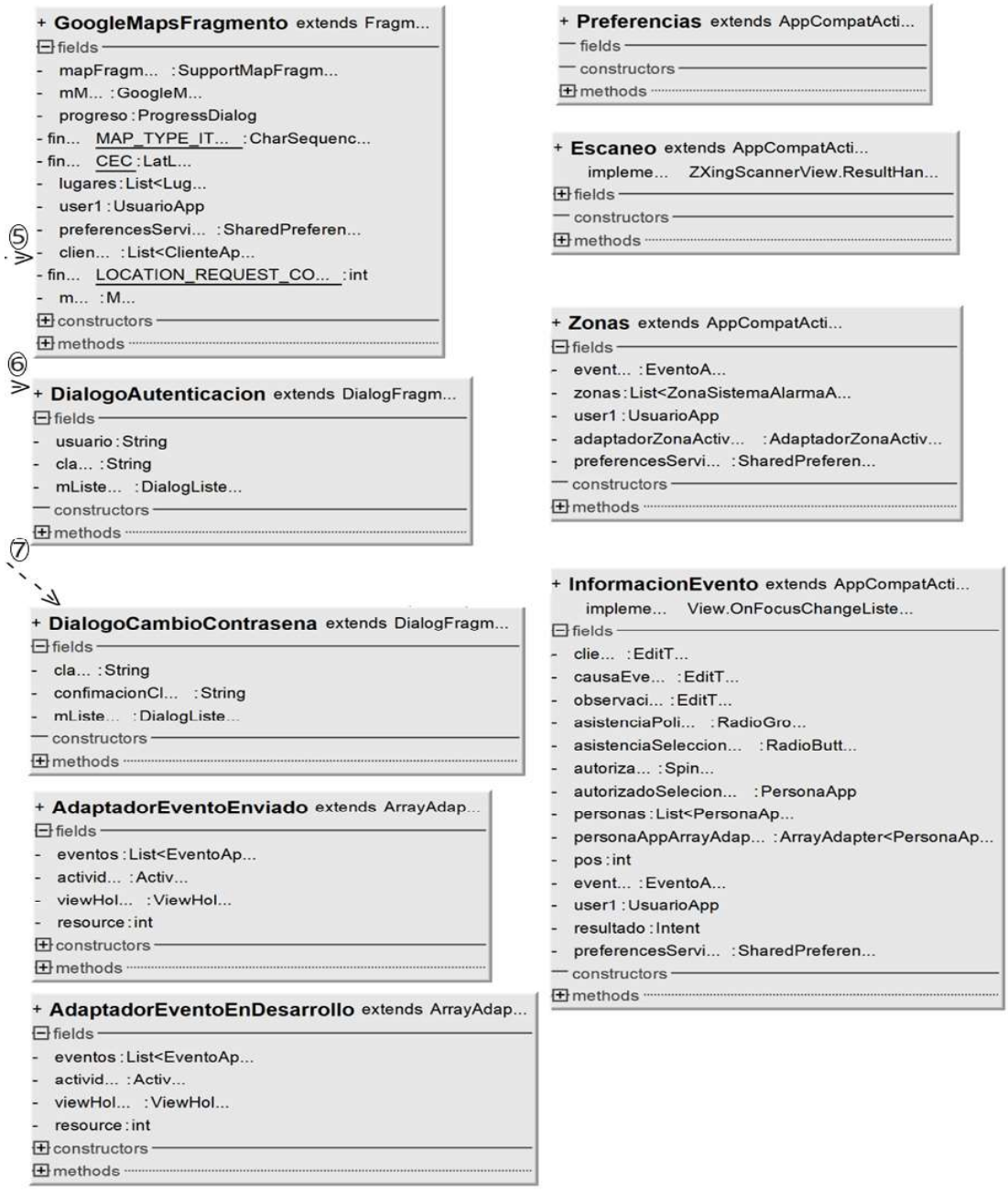


Figura 2.26 Diagrama de Clases de la aplicación Android (Parte III de III)

- `DateNumberFormatDetector`: Permitirá generar el formato de fecha para que los métodos del servicio web puedan procesar la información.
- `DialogoAutenticacion`: Permitirá la autenticación de un usuario en la aplicación Android.
- `AdaptadorEventoEnviado`: Permitirá mostrar la información de los eventos con estado enviado.
- `AdaptadorEventoEnDesarrollo`: Permitirá mostrar la información de los eventos con estado en desarrollo.
- `InformacionEvento`: Permitirá registrar la información de un evento como: causa del evento, observación, asistencia policial y autorizado.
- `Zonas`: Permitirá manejar las zonas activadas de un evento.
- `DialogoCambioContrasena`: Permitirá realizar el cambio de la contraseña de un usuario.

Diseñar el diagrama de secuencia: En base al diagrama de secuencia presentado en la Figura 2.18, se realiza el diagrama de secuencia del prototipo con la aplicación Android, como se observa en la Figura 2.27.

El cual presenta el único cambio suscitado en la solicitud HTTP de la aplicación Android al servicio WCF, esta solicitud se realiza asincrónicamente, debido a que es la manera en la que trabajan los dispositivos Android, para no ocupar una considerable cantidad de procesamiento.

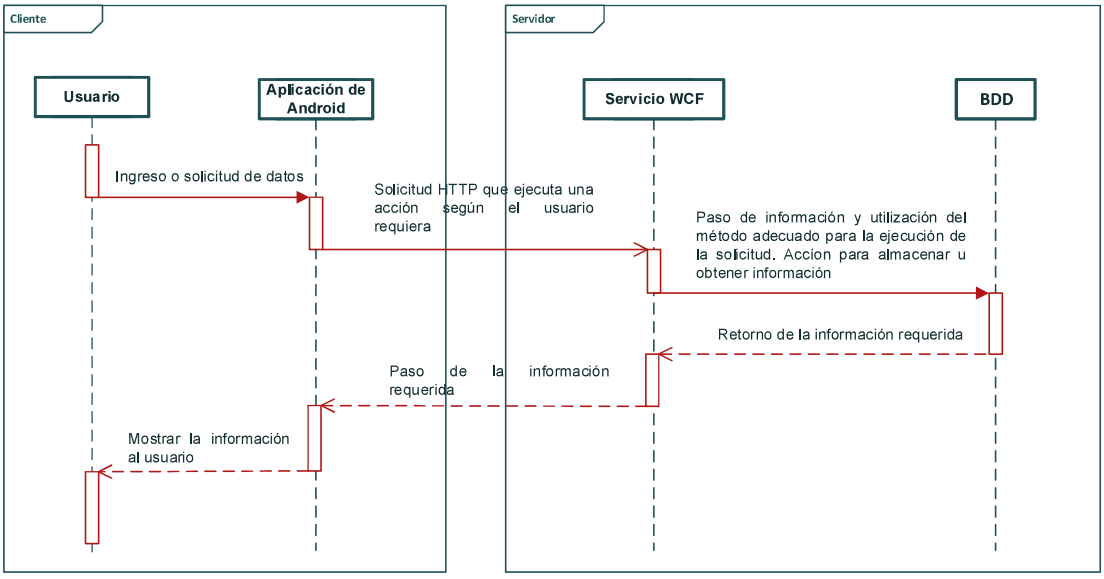


Figura 2.27 Diagrama de Secuencia del prototipo con la aplicación Android

Diseñar las *sketches* principales: La Figura 2.28 presenta el *sketch* para autenticar usuarios, donde el usuario rellena su alias y contraseña para ingresar a la aplicación Android.

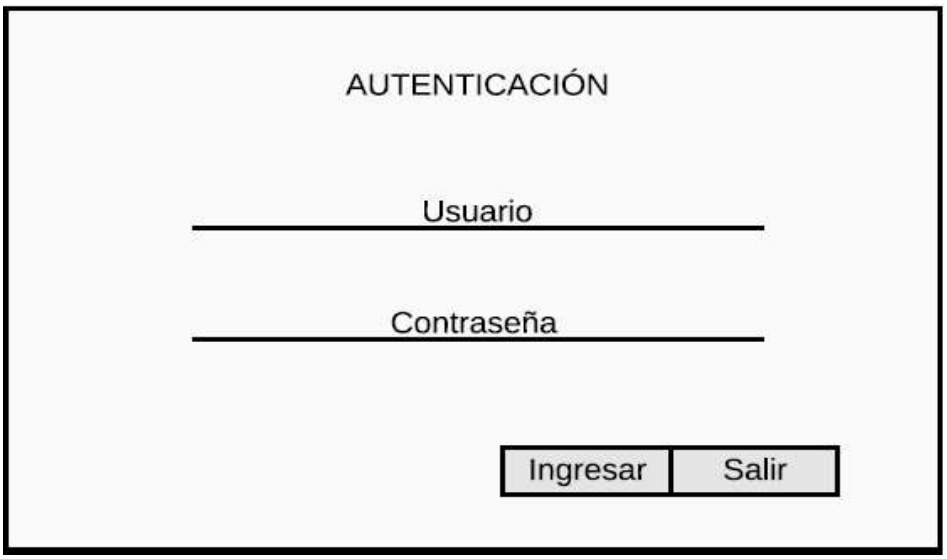


Figura 2.28 *Sketch* para autenticar usuarios

En la Figura 2.29 se muestra el *sketch* para listar los eventos, se observa una barra en la parte izquierda para navegar a través de la aplicación, en el recuadro blanco se mostrarán los eventos que coincidan con la opción seleccionada: enviados, en desarrollo o finalizados; también se podrá mostrar un mapa con las direcciones de los clientes al presionar sobre el botón `Ubicación`.

Figura 2.29 *Sketch* para listar los eventos

En la Figura 2.30 se muestra el *sketch* con un ejemplo de un evento enviado, se mantiene la barra en la parte izquierda para navegar a través de la aplicación, en el recuadro blanco se mostrarán la información correspondiente al cliente, la fecha y hora de envío, y el tipo de evento.

Figura 2.30 *Sketch* para mostrar la información del evento enviado

En la Figura 2.31 se muestra el *sketch* que mostrará el mapa para visualizar los clientes y el dispositivo móvil, adicional se mantiene la barra de navegación en la parte izquierda.

Figura 2.31 *Sketch* para mostrar el mapa

Diseñar el diagrama de actividades principal: La Figura 2.32 indica el diagrama de actividades cuando se registra un evento.

El proceso inicia cuando el supervisor lee el evento e ingresa la información de llegada al evento. La aplicación validará la información de llegada, si es válida, la aplicación registrará el evento en la base de datos, caso contrario podrá nuevamente ingresar la información de llegada. Luego el supervisor ingresa la información de salida del evento. La aplicación validará la información de salida, si es válida procede a registrar, caso contrario podrá nuevamente ingresar la información de salida. Por último, el supervisor visualizará el evento y así finaliza el proceso.

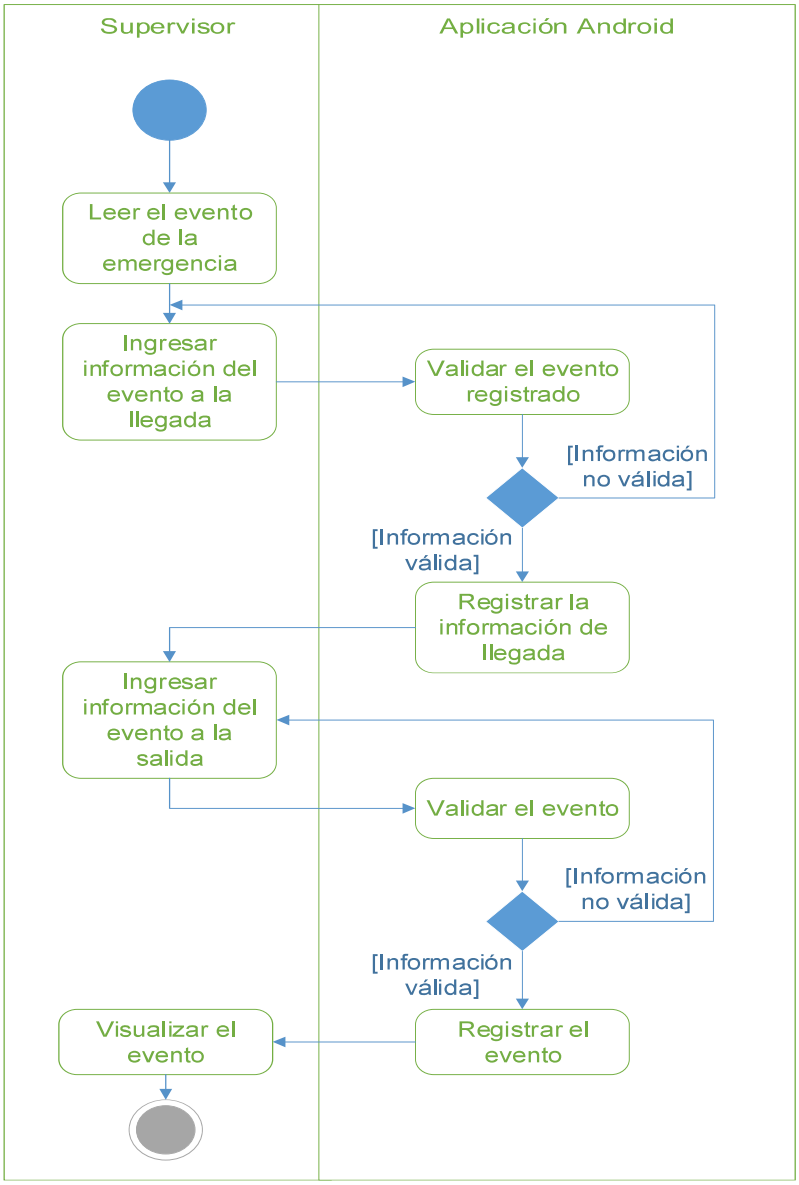


Figura 2.32 Diagrama de actividades para registrar un evento

2.6. SPRINT 1 – AUTENTICACIONES

En la implementación del *sprint* 1, se realizan las funcionalidades de autenticación para la aplicación de escritorio y aplicación Android.

Creación de la interfaz de usuario para la autenticación – aplicación de escritorio: En la Figura 2.33 se muestra la interfaz de usuario que permite la autenticación de un usuario, se colocarán las credenciales correspondientes al usuario para ingresar a la aplicación de escritorio.

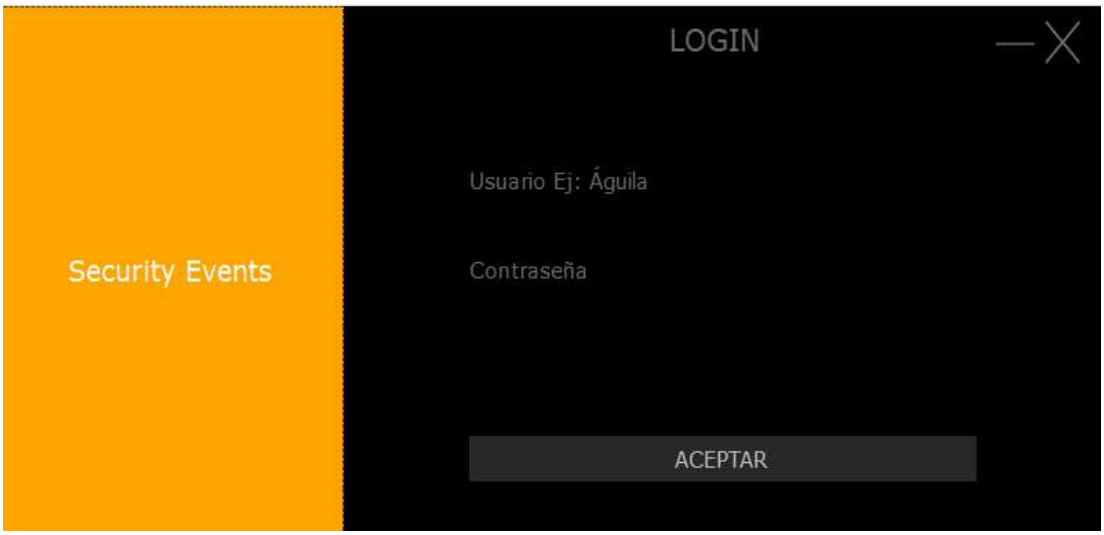


Figura 2.33 Interfaz de usuario para la autenticación

Implementar código para la autenticación – aplicación de escritorio: En el código 2.4 se muestra la implementación de la autenticación en la aplicación de escritorio, el método `PeticionUsuario` de la clase `Conexion` permite realizar las solicitudes de la aplicación de escritorio hacía el servicio web WCF.

Para realizar la autenticación de un usuario, previamente se instancia la clase `Conexion` y se llama a un método útil en este caso `PeticionUsuario` para realizar la funcionalidad de autenticar el usuario.

El método `PeticionUsuario` recibe como argumentos de entrada: la URL del método del servicio web, los parámetros con la información del usuario y el método HTTP sobre el cual se ejecutará la solicitud (línea 241).

En las líneas 244 a 262, se crea una lista llamada `parametrosOrganizados` que organiza los parámetros de entrada siguiendo un formato aceptado por el servicio WCF, en la lista se coloca después de cada variable `parametros` el carácter *slash*, exceptuando

la última variable (líneas 251 hasta 261); las líneas entre la 265 y 272 almacenan la lista `parametrosOrganizados` en una única variable.

Se realiza dicha funcionalidad para generar el formato necesario que deben tener las variables de entrada, para ejecutar la solicitud hacia el servicio web.

Esta funcionalidad se repite en todas las solicitudes realizadas desde la aplicación de escritorio.

```
240     //Método que realiza la petición al servicio web y retorna un objeto de tipo UsuarioA
241     public UsuarioApp PeticionUsuario(string url, List<string> parametros, string metodo)
242     {
243         //Instancia de una lista de string parametrosOrganizados
244         List<string> parametrosOrganizados = new List<string>();
245 
246         //Lazo que recorre los parametros para colocarles el separador "/"
247         //entre cada parametro, exceptuando el último parametro
248         for (int contador = 0; contador < parametros.Count(); contador++)
249         {
250             //En el ulitmo parámetro no se agrega el separador "/"
251             if (contador == (parametros.Count() - 1))
252             {
253                 parametrosOrganizados.Add(parametros[contador]);
254                 break;
255             }
256             //Se agrega el separador "/" después de cada parametro
257             else
258             {
259                 parametrosOrganizados.Add(parametros[contador]);
260                 parametrosOrganizados.Add("/");
261             }
262         }
263 
264         //Instancia de un StringBuilder elementos
265         StringBuilder elementos = new StringBuilder();
266 
267         //Toma los valores de la lista parametrosOrganizados y
268         //los almacena en la variable elementos
269         foreach (var auxiliar in parametrosOrganizados)
270         {
271             elementos.Append(auxiliar);
272         }
```

Código 2.4 Método para realizar una solicitud al servicio web (Parte I de II)

El Código 2.5 muestra la segunda parte del método `PeticionUsuario`; en las líneas 278 a 281 se realiza la solicitud HTTP apuntando a la URL formada por:

La conexión al servidor que tiene la IP y el puerto donde se encuentra el servicio web, la URL especificando el método del servicio web y los argumentos de entrada del método.

Además, se define el tipo de contenido de la solicitud en JSON (línea 280) y el método HTTP para realizar la solicitud (línea 281).

En la línea 284 se obtiene la respuesta del servidor; la línea 287 obtiene el cuerpo del mensaje de la respuesta; en la línea 290 se almacena todo el mensaje en una variable `json`.

Para finalmente deserializar la información de tipo JSON al objeto de tipo `UsuarioApp` (línea 295).

Finalmente, en la línea 298 se retorna el objeto del usuario obtenido.

```
274         //Realiza una solicitud de tipo http con los parametros
275         //conexionServidor que lleva la URL del servicio web
276         //contrato que lleva el nombre del método a usarse
277         //elementos que seran las variables para ingresar el usuario
278         HttpWebRequest request =
279                     (HttpWebRequest)WebRequest.Create(@conexionServidor + url + elementos);
280         request.ContentType = "text/json";
281         request.Method = metodo;
282
283         //Se recoje la respuesta de la solicitud
284         using (HttpWebResponse response = (HttpWebResponse)request.GetResponse())
285
286         //Se obtiene el cuerpo de la respuesta del servidor
287         using (StreamReader reader = new StreamReader(response.GetResponseStream()))
288         {
289             //Lee todos los caracteres del reader desde la posición inicial hasta la final
290             var json = reader.ReadToEnd();
291
292             //Se debe importar la Biblioteca Newtonsoft.Json para serializar
293             //y deserializar cualquier objeto de .Net a JSON y viceversa
294             //Deserialización de la cadena JSON en un objeto de tipo UsuarioApp
295             UsuarioApp resultadoPeticion = JsonConvert.DeserializeObject<UsuarioApp>(json);
296
297             //Retorna el objeto UsuarioApp
298             return resultadoPeticion;
299         }
300     }
```

Código 2.5 Método para realizar una solicitud al servicio web (Parte II de II)

El Código 2.6 muestra el método para autenticar usuarios; en la línea 244 se instancia la clase `Conexion` que permite realizar las solicitudes al servicio WCF utilizando sus métodos.

En las líneas 247 a 249 se crea una lista con las credenciales del usuario a autenticar; en las líneas 252 y 253 a través del objeto `conectar` se llama al método `PeticionUsuario`, y se pasan como argumentos: la URL del método del servicio web, las credenciales del usuario y el método GET para indicar al servicio web que se requiere obtener información.

En caso de ser exitosa la autenticación se retorna un objeto de tipo usuario, caso contrario se retorna un valor `NULL` (línea 256).

```
239    //Método que realiza una petición de Autenticacion al servidor,
240    //para permitir el ingreso de un usuario al prototipo.
241    public UsuarioApp Autenticacion()
242    {
243        //Instancia de la clase Conexion
244        Conexion conectar = new Conexion();
245
246        //Creación de la lista de string con el nombre de usuario y la contraseña
247        List<string> variables = new List<string>();
248        variables.Add(txt_Usuario.Text);
249        variables.Add(txt_Contraseña.Text);
250
251        //Petición al servidor para validar los datos ingresados
252        UsuarioApp usuarioResultado =
253                    conectar.PeticionUsuario("AutenticacionUsuario/", variables, "GET");
254
255        //Retorno del usuario autenticado
256        return usuarioResultado;
257    }
```

Código 2.6 Método para autenticar usuarios

Creación de la interfaz de usuario para el cambio de contraseña – aplicación de escritorio: La Figura 2.34, muestra la interfaz de usuario donde el usuario cambiará la contraseña predeterminada por una nueva en la aplicación de escritorio, esta interfaz se mostrará cuando el usuario inicie sesión por primera vez o cuando la contraseña del usuario sea modificada por un administrador del prototipo.

Figura 2.34 Interfaz de usuario para cambiar la contraseña

Implementar código para el cambio de contraseña – aplicación de escritorio: Al igual que en el Código 2.6 se debe obtener la instancia de la clase `Conexion`, luego se debe generar una lista con la nueva contraseña del usuario.

En el Código 2.7 se muestra un fragmento del método para cambiar la contraseña en la aplicación de escritorio, en la línea 71 se utiliza el objeto `conectar` que llama al método

`Peticion`, en la cual se pasan como argumentos: el objeto del usuario autenticado en la aplicación de escritorio, la URL del método del servicio web, la nueva contraseña del usuario y el método PUT para indicar al servicio web que se requiere realizar un cambio. Como resultado se obtiene un entero para indicar el éxito en la modificación.

```
70    //Petición al servidor para cambiar la clave de un usuario
71    int resultadoPeticion = conectar.Peticion(user, "ModificarCuentaUsuario/", variables, "PUT");
```

Código 2.7 Método para cambiar la contraseña

Creación de la interfaz de usuario para la autenticación – aplicación Android: En la Figura 2.35, se muestra la interfaz que realiza la autenticación de un usuario en la aplicación Android. Se debe introducir las credenciales correspondientes al usuario, para poder ingresar a la aplicación.

Figura 2.35 Interfaz de usuario para la autenticación

Implementar código para la autenticación – aplicación Android: En el Código 2.8 se presenta la clase asincrónica `Autenticacion` que realiza la autenticación de los usuarios en la aplicación Android.

En las líneas 496 a 500 se declaran las variables a utilizar en la clase, en las líneas 504 a 510 se extraen los valores de las preferencias como la URL del servicio web, el tiempo máximo de conexión y el tiempo máximo de lectura. Se desarrollan de la línea 515 a 670 los tres métodos que implementa la clase asincrónica, que serán definidos posteriormente.

```
492     //Clase que envia los datos a la clase ConexionRestful en segundo plano
493     public class Autenticacion extends AsyncTask <String, String, Boolean> {
494
495         //Declaración de Variables
496         Map<String, String> resultado = new HashMap();
497         private String resultadoJSON;
498         private String solicitudURL;
499         private StringBuilder errores;
500         private ProgressDialog progreso;
501
502         //El hostname, tiempo de conexión y tiempo de lectura se encuentran en
503         //las preferencias
504         String hostname =
505                 preferencesServicio.getString("hostname",
506                         "http://192.168.1.111:8200/AplicacionMovil.svc/");
507         Integer tiempoconexion =
508                 Integer.parseInt(preferencesServicio.getString("connect_timeout", "10"));
509         Integer tiempolectura =
510                 Integer.parseInt(preferencesServicio.getString(" read_timeout", "10"));
511
512         // Método que es el primero en ejecutarse,
513         // antes de iniciar el proceso en segundo plano
514         @Override
515         protected void onPreExecute() {...}
528
529         //Metodo que ejecuta el cuerpo de la clase
530         @Override
531         protected Boolean doInBackground(String... args) {...}
573
574         // Se ejecuta al completarse el método doInbackground
575         // Método que se ejecuta en la terminación del proceso en segundo plano
576         @Override
577         protected void onPostExecute(Boolean verificador) {...}
670     }
```

Código 2.8 Autenticación de usuarios en la aplicación Android

El Código 2.9 muestra el método `onPreExecute` que es el primero en ejecutarse de la clase `Autenticacion`, en la línea 511 se crea el objeto `progreso` para manejar un cuadro de diálogo; la línea 512 establece un mensaje para mostrar en el diálogo; en la línea 515 se inicializa la variable que recogerá los errores, se procede a mostrar el `ProgressDialog` (línea 518).

```
506     @Override
507     protected void onPreExecute() {
508         super.onPreExecute();
509
510         //Se instancia al ProgressDialog
511         progreso = new ProgressDialog( context: Main.this);
512         progreso.setMessage("CARGANDO ...");
513
514         //Se instancia la variable que almacenará los errores
515         errores = new StringBuilder();
516
517         //Muestra el ProgressDialog
518         progreso.show();
519     }
```

Código 2.9 Método `onPreExecute` para la autenticación de usuarios

En el Código 2.10 se desarrolla el método `doInBackground` que ejecuta el cuerpo de la clase en segundo plano donde se realiza la solicitud al servicio WCF, en las líneas 527 a 530 a través del objeto `ConexionRestful` se llama al método `connectREST` que contiene los argumentos: la URL del servicio web, el método del servicio web, el usuario, la contraseña, el tipo de contenido, la información extra a enviar, el método HTTP, el tiempo máximo de conexión y el tiempo máximo de lectura.

En las líneas de la 534 a la 548 se realiza un condicional con la respuesta del servidor para validar que la autenticación se realizó correctamente; en la línea 537 se obtiene el cuerpo de la respuesta en formato JSON; la línea 541 obtiene la URL de la solicitud al servicio. Al final en las líneas 550 a 555 se capturan los errores que pudieran suceder por fallos de conexión.

```
521     //Metodo que ejecuta el cuerpo de la clase
522     @Override
523     protected Boolean doInBackground(String... args) {
524         try{
525             //Conexión al servidor a través de la clase ConexionRestful
526             //usando el método connectREST
527             resultado = ConexionRestful.connectREST(hostname, args[0], args[2],
528                     args[3],"application/x-www-form-urlencoded;charset=UTF-8",
529                     "", args[1], tiempoconexion*1000,
530                     tiempolectura*1000);
531
532             //Condicional para conocer si la petición al
533             // servidor fue exitosa(si el code es 200) o no
534             if (Integer.parseInt(resultado.get("code")) == 200) {
535                 //El campo "body" del resultado regresa el mensaje
536                 // en formato JSON de la respuesta del servidor
537                 resultadoJSON = resultado.get("body");
538
539                 //El campo "URL" del resultado regresa la URL creada
540                 // para realizar la petición al servidor
541                 solicitudURL = resultado.get("URL");
542             }
543             else {
544                 //El campo "message" del resultado regresa
545                 // el mensaje de error correspondiente
546                 errores.append(resultado.get("message")).append("\n");
547                 return false;
548             }
549         }
550         catch(ConnectException ex){
551             //Impresion del Error
552             ex.printStackTrace();
553             errores.append("Fallo de Conexión al Servidor");
554             return false;
555         }
```

Código 2.10 Método `doInBackground` para la autenticación de usuarios

El Código 2.11 presenta el método `onPostExecute`, se ejecuta al terminar la tarea en segundo plano; en la línea 582 se instancia la clase `GsonBuilder` para construir un objeto de la clase `Gson`; en las líneas 587 a 595 se realiza una funcionalidad para lograr que los campos de tipo `date` sean legibles al usuario; la línea 599 instancia la clase `Gson` utilizando el objeto de la clase `GsonBuilder` para deserializar la respuesta del servicio WCF; la línea 602 deserializa la información de tipo JSON a un objeto de tipo `UsuarioApp`.

```
575     // Se ejecuta al completarse el método doInbackground
576     // Método que se ejecuta en la terminación del proceso en segundo plano
577     @Override
578     protected void onPostExecute(Boolean verificador) {
579         super.onPostExecute(verificador);
580         try{
581             //Instancia del objeto GsonBuilder que manejará la información recibida
582             GsonBuilder builder = new GsonBuilder();
583
584             //Registro de un adaptador que maneja los tipos de valores Date como
585             //Long, se realiza esta funcionalidad para obtener la fecha como
586             //un valor de datos legible
587             builder.registerTypeAdapter(Date.class, new JsonDeserializer<Date>() {
588                 public Date deserialize(JsonElement json, Type typeOfT,
589                                         JsonDeserializationContext context)
590                         throws JsonParseException {
591                     String s = json.getAsJsonPrimitive().getAsString();
592                     long l = Long.parseLong(s.substring(6, s.length() - 7));
593                     Date d = new Date(l);
594                     return d;
595                 }
596             });
597
598             //Instancia del objeto Gson
599             Gson gson = builder.create();
600
601             //Retorna un Objeto UsuarioApp con los elementos ya parseados.
602             UsuarioApp usuarioApp = gson.fromJson(resultadoJSON, UsuarioApp.class);
```

Código 2.11 Método `onPostExecute` para la autenticación de usuarios (Parte I de III)

El Código 2.12 en las líneas 606 a 622 valida la información obtenida en el anterior método `doInBackground`. Una vez que la información sea corroborada, se continuará en la línea 623 donde se almacena en la variable `auth` el estado de la autenticación; en la línea 624 se carga el usuario obtenido a un usuario local.

Entre las líneas 626 y 628 se define y muestra el `NavigationView` que es el menú lateral; en las líneas 631 a 633 se obtiene la información del usuario, en medio de las líneas 636 a 639 se define el `Header` del `NavigationView` incluyendo los elementos: `nombreUsuario` y `aliasUsuario`.

```
606             //Condicional que verifica el valor booleano del proceso anterior
607             if (!verificador) {
608                 lanzarDialogo("AUTH",
609                         "AUTENTICACIÓN", "ERROR: ".concat(errores.toString()));
610             }
611             //Condicional que verifica el valor del código
612             else if ((resultado.get("code") == null)||(resultado.get("code").isEmpty())) {
613                 lanzarDialogo("AUTH",
614                         "AUTENTICACIÓN", "ERROR: Servidor Caido");
615             }
616             //Condicional que verifica el valor del resultado en formato JSON
617             else if ((resultadoJSON == null) || (resultadoJSON.isEmpty())) {
618                 lanzarDialogo("AUTH",
619                         "AUTENTICACIÓN", "ERROR: Usuario o Clave Incorrectas");
620             }
621             //Condicional que realiza la autenticación
622             else {
623                 auth = true;
624                 usuario = usuarioApp;
625                 //Instancia del NavigationView que es el menu lateral
626                 NavigationView navigationView =
627                         (NavigationView) findViewById(R.id.nav_view);
628                 navigationView.setNavigationItemSelectedListener(Main.this);
629
630                 //Obtención de la información del usuario
631                 String nombre = usuario.getNombreUsuario().concat(" ")
632                                 .concat(usuario.getApellidoUsuario());
633                 String alias = usuario.getAliasUsuario();
634
635                 //Se muestra el header del NavigationView
636                 View headerView =
637                         navigationView.inflateHeaderView(R.layout.nav_header_main);
638                 nombreUsuario = headerView.findViewById(R.id.nombreUsuario);
639                 aliasUsuario = headerView.findViewById(R.id.alias);
```

Código 2.12 Método `onPostExecute` para la autenticación de usuarios (Parte II de III)

En el Código 2.13 en las líneas 628 y 629 se establecen los valores del usuario en las variables: `nombreUsuario` y `aliasUsuario` en el `Header` del `NavigationView`; las líneas 632 y 633 almacenan el estado de la autenticación en las preferencias, para mantener iniciada la sesión una vez que la aplicación se cierre.

Entre las líneas 635 a 642 se verifica si el valor que identifica el primer inicio de sesión es `true`, si es así se llama al método que permite modificar la contraseña de un usuario (líneas 636 hasta 638), caso contrario se ejecuta la clase `Hilo` que mostrará los eventos enviados del usuario autenticado (líneas 640 y 641).

Por último, entre las líneas 645 a 649 se captura la excepción producida cuando la aplicación usa una referencia de un objeto nulo, y en las líneas 650 a 654 se capturan los errores producidos en la ejecución de la aplicación, además de mostrarse un mensaje de acuerdo con la excepción; en la línea 656 se elimina el objeto del cuadro de diálogo.

```
627                 //Setea los valores en el header
628                 nombreUsuario.setText(nombre);
629                 aliasUsuario.setText(alias);
630
631                 //Editar las preferencias para mantener iniciada la sesión
632                 editor.putString("login", "true");
633                 editor.apply();
634
635                 if (usuario.getPrimerInicioSesion().equals(true)) {
636                     cambioContrasena("CONTRASENA",
637                             "CAMBIO DE CONTRASEÑA",
638                             "Ingrese una nueva contraseña");
639                 } else {
640                     new Hilo().execute("ListarUltimosEventos/", "GET",
641                             usuario.getAliasUsuario(), "ENVIADO");
642                 }
643             }
644         }
645         catch(NullPointerException exception){
646             exception.printStackTrace();
647             Toast.makeText(Main.this,
648                     exception.getMessage(), Toast.LENGTH_SHORT).show();
649         }
650         catch(Exception exception){
651             exception.printStackTrace();
652             Toast.makeText(Main.this,
653                     exception.getMessage(), Toast.LENGTH_SHORT).show();
654         }
655         //Elimina ProgressDialog.
656         progreso.dismiss();
657     }
658 }
```

Código 2.13 Método `onPostExecute` para la autenticación de usuarios (Parte III de III)

2.7. SPRINT 2 – USUARIOS

La implementación del *sprint* 2 consta del desarrollo de la gestión de usuarios para la aplicación de escritorio.

Creación de las interfaces para la gestión de usuarios – aplicación de escritorio: En la Figura 2.36 se presenta la interfaz para gestionar usuarios en la aplicación de escritorio.

En la parte izquierda se encuentra un panel para navegar por la aplicación de escritorio, en la parte inferior del panel se detalla el nombre y tipo del usuario conectado. En la interfaz se muestran todos los usuarios almacenados en el prototipo y distintos botones:

- Nuevo: Permite crear un nuevo usuario.
- Modificar: Permite cambiar la información de un usuario.
- Eliminar: Permite deshabilitar la cuenta de un usuario.

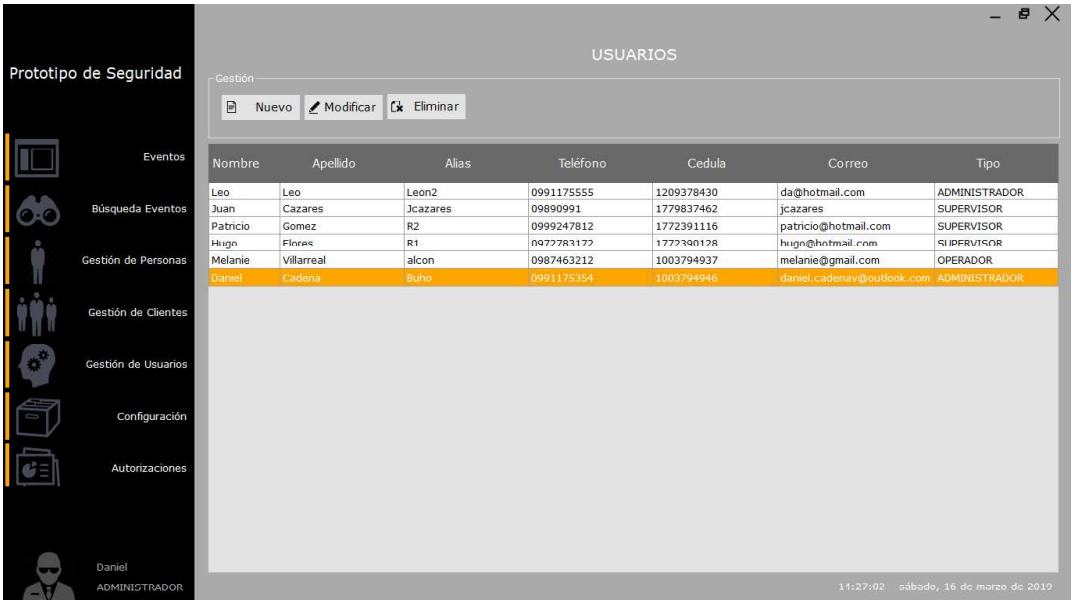


Figura 2.36 Interfaz de usuario para gestionar cuentas de usuarios

En la Figura 2.37 se muestra la interfaz para ingresar y modificar un usuario en la aplicación de escritorio, donde se ingresan los campos del usuario: nombre, apellido, alias, cédula, correo, contraseña, teléfono y tipo de usuario.

Los cambios se almacenan al presionar el botón `Guardar`, para salir sin guardar cambios se presiona el botón `Salir` o el icono superior derecho con la equis.

Figura 2.37 Interfaz de usuario para ingresar y modificar un usuario

Implementar código para la gestión de usuarios – aplicación de escritorio: Para la creación de un usuario se realiza la misma funcionalidad que en el Código 2.6 primero se

debe obtener la instancia de la clase `Conexion`, luego se debe generar una lista con la información del usuario recuperada de la interfaz.

El Código 2.14 muestra un fragmento del método para la creación de un usuario en la aplicación de escritorio, el cual en las líneas 237 y 238 a través del objeto `conectar` llama al método `Peticion`, el cual recibe como argumentos: el objeto del usuario autenticado, la URL del método del servicio web, la información del usuario y el método POST para indicar que se requiere agregar un usuario. Como resultado se obtiene un entero para indicar el éxito en la creación.

```
236    //Petición al servidor para ingresar una cuenta de usuario
237    int resultadoPeticion =
238                    conectar.Peticion(user, "IngresarCuentaUsuario/", variables, "POST");
```

Código 2.14 Creación de un usuario

El código para la modificación de un usuario es similar al Código 2.7 con la diferencia que, en lugar de colocar únicamente la contraseña, se coloca la información del usuario obtenida de la interfaz.

El Código 2.15 muestra un fragmento del método para eliminar un usuario en la aplicación de escritorio, en las líneas 296 y 297 se llama al método `Petición` del objeto `conectar`, los argumentos del método son: el objeto del usuario autenticado, la URL del método del servicio web, el Id y la disponibilidad del usuario, y el método PUT para indicar al servicio web que se requiere realizar un cambio.

Es importante mencionar que el usuario no se borra de la base de datos, sino se oculta de las aplicaciones cambiando el campo de disponibilidad. Como resultado se obtiene un entero para indicar el éxito en la eliminación.

```
295    //Petición al servidor para deshabilitar una cuenta de usuario
296    int resultadoPeticion =
297                    conectar.Peticion(user, "DisponibilidadCuentaUsuario/", variables, "PUT");
```

Código 2.15 Eliminación de un usuario

En el Código 2.16 se muestra el método para visualizar usuarios en la aplicación de escritorio, se realiza llamando al método `PeticionListaUsuarios` del objeto `conectar`, donde se ingresa: el objeto del usuario autenticado, la URL del método del servicio web, la variable disponibilidad del usuario y el método GET para indicar al servicio web que se requiere obtener información (líneas 89 y 90).

Como resultado se obtiene una lista de usuarios, los cuales se muestran en un componente `DataGridView` (línea 93).

```
88  //Petición al servidor para listar los usuarios de la base de datos
89  List<UsuarioApp> usuarios =
90                      conectar.PeticionListaUsuarios(user, "ListarCuentasUsuario/", variables, "GET");
91
92  //Método para cargar los usuarios en el DataGridView
93  CargarDataGridView(usuarios);
```

Código 2.16 Visualización de usuarios

Validación de campos para ingresar y modificar un usuario – aplicación de escritorio: El Código 2.17 muestra un fragmento del método para ingresar un usuario, donde se validará la información previamente a almacenar un usuario.

En las líneas 210 a 213 se verifica que todos los campos de la interfaz de usuario no estén ni vacíos ni nulos. En la línea 219 se llama a un método para validar el formato del ingreso de un correo; la línea 225 llama a un método para comprobar que la cédula ingresada sea correcta.

Una vez se han evaluado los campos con los anteriores condicionales, se continúa a la línea 231 y posteriores donde se realiza la funcionalidad para ingresar un usuario. De no cumplirse con alguna validación se mostrará un mensaje acorde a la validación no completada.

```
203 //Método que realiza una solicitud de IngresarCuentasUsuario al servicio web WCF, para guardar un
204 //usuario en la base de datos.
205 private void GuardarUsuario()
206 {
207     try
208     {
209         //Condicional que verifica que ningún campo este vacío o nulo
210         if (string.IsNullOrEmpty(txt_NombreUsuario.Text) || string.IsNullOrEmpty(txt_ApellidoUsuario.Text)
211             || string.IsNullOrEmpty(txt_AliasUsuario.Text) || string.IsNullOrEmpty(txt_TelefonoUsuario.Text)
212             || string.IsNullOrEmpty(txt_Cedula.Text) || string.IsNullOrEmpty(txt_Correo.Text)
213             || string.IsNullOrEmpty(txt_HashContraseña.Text) || string.IsNullOrEmpty(cbx_TipoUsuario.Text))
214         {
215             MessageBox.Show("Llene todos los campos ", "Advertencia",
216                                         MessageBoxButtons.OK, MessageBoxIcon.Warning);
217         }
218         //Condicional que evalúa un formato de mail válido
219         else if (!validarEmail(txt_Correo.Text))
220         {
221             MessageBox.Show("Ingrese un mail correcto ", "Advertencia",
222                                         MessageBoxButtons.OK, MessageBoxIcon.Warning);
223         }
224         //Condicional que evalúa una cédula válida
225         else if (!VerificarCedula(txt_Cedula.Text))
226         {
227             MessageBox.Show("Ingrese una cédula correcta ", "Advertencia",
228                                         MessageBoxButtons.OK, MessageBoxIcon.Warning);
229         }
230         //Condicional que guarda un usuario
231         else
```

Código 2.17 Validación de valores al ingresar y modificar un usuario

El Código 2.18 muestra la funcionalidad para validar el ingreso de únicamente letras en el *textbox* `NombreUsuario`, la validación se realiza en el evento `KeyPress`, que se produce cuando el usuario digita una tecla sobre el *textbox*.

Esta validación también se realizó para ingresar únicamente letras en el campo apellido. Realizando un cambio en la primer condición por `char.IsNumber` es posible realizar la validación para permitir únicamente números en los campos teléfono y cédula.

```
395     //Evento que valida únicamente el ingreso de letras en el TextBox
396     private void txt_NombreUsuario_KeyPress(object sender, KeyPressEventArgs e)
397     {
398         if (!(char.IsLetter(e.KeyChar)) && (e.KeyChar != (char)Keys.Back))
399         {
400             //Mensaje de advertencia
401             MessageBox.Show("Únicamente se permiten letras", "Advertencia",
402                             MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
403             e.Handled = true;
404             return;
405         }
406     }
```

Código 2.18 Validación en el *textbox* `NombreUsuario`

2.8. SPRINT 3 – CLIENTES

La implementación del *sprint* 3 consta del desarrollo de la gestión de clientes, la gestión de zonas de clientes para la aplicación de escritorio. La visualización de clientes para la aplicación Android. Se mostrará la información relevante del *sprint*.

Creación de un código QR: Para la generación de códigos QR existe una gran variedad de programas, entre ellos se pueden destacar: Kaywa QR Code, QRmaker Pro y QRCode Generator [31].

Para el ejemplo se utiliza el software Kaywa QR Code, que en la Figura 2.38 muestra una captura de pantalla donde se especifica el texto del título del trabajo de titulación, y al presionar sobre el botón `Generate`, se creará el código QR correspondiente.

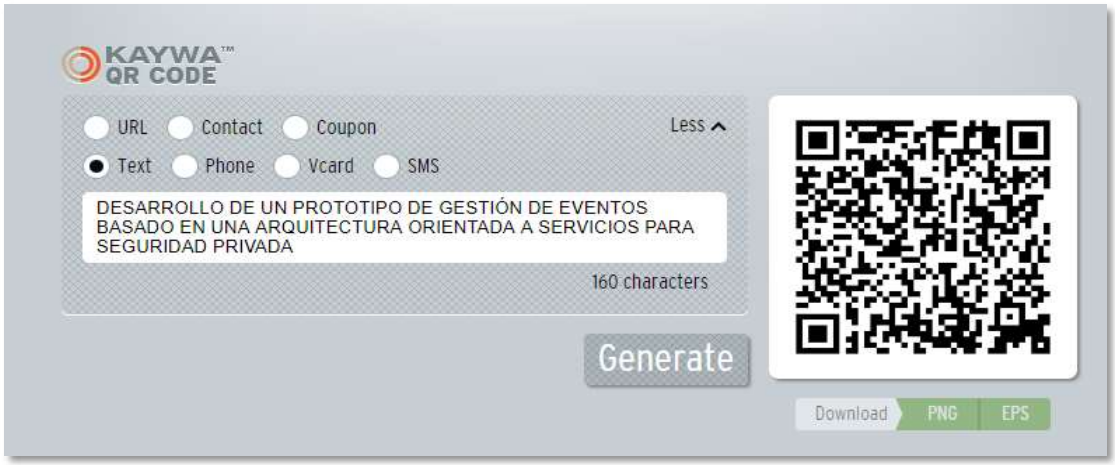


Figura 2.38 Creación de código QR usando Kaywa QR Code

Creación de la interfaz de usuario para la gestión de clientes – aplicación de escritorio: En la Figura 2.39 se presenta la interfaz de usuario para gestionar clientes en la aplicación de escritorio, el cual en su parte superior derecha permite realizar búsquedas según el criterio: nombre, cuenta y ciudad.

En la pantalla se muestran todos los clientes registrados, los botones ubicados en la parte superior izquierda tienen las funcionalidades de:

- Nuevo: Permite crear un nuevo cliente.
- Modificar: Permite cambiar la información de un cliente.
- Eliminar: Permite deshabilitar la cuenta de un cliente.
- Zonas: Permite agregar las zonas de los sistemas de seguridad de cada cliente.

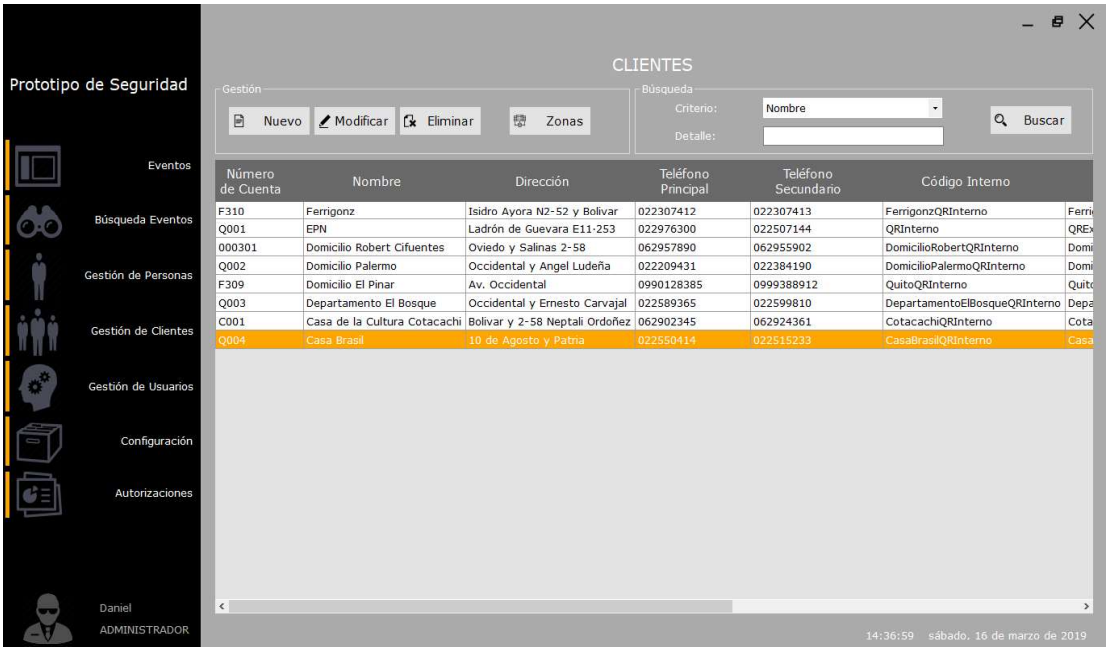


Figura 2.39 Interfaz de usuario para gestionar clientes

En la Figura 2.40 se muestra la interfaz de usuario para ingresar y modificar un cliente en la aplicación de escritorio, donde se almacena: número de cuenta, nombre, dirección, ciudad, tipo, plan, teléfono, teléfono secundario, código interno, código externo, latitud y longitud.

Los cambios se almacenan al presionar el botón `Guardar`, para salir sin guardar cambios se presiona el botón `Salir` o el icono con la equis. El botón `Mapa` abre un formulario donde se puede buscar la dirección de un cliente para obtener los valores de latitud y longitud. El botón `Códigos` genera el texto de los campos `Código Interno` y `Código Externo`.

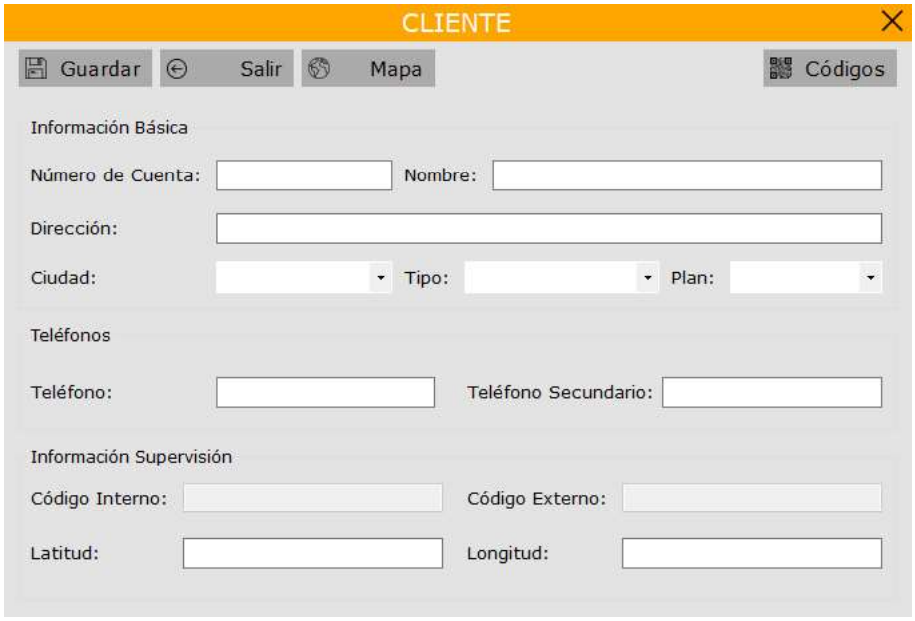


Figura 2.40 Interfaz de usuario para ingresar y modificar un cliente

Implementar código para buscar clientes – aplicación de escritorio: El Código 2.19 presenta la búsqueda de clientes en la aplicación de escritorio, según el criterio de nombre.

En las líneas 351 y 352 se realiza la búsqueda por nombre mediante el método `PeticionListaClientes` del objeto `conectar`, que recibe como argumentos: el objeto del usuario autenticado, el URL del método del servicio web, el nombre del cliente y el método GET para indicar al servicio web que se requiere obtener información, como resultado se obtiene una lista de clientes. En la línea 355 los clientes se muestran en un componente `DataGridView`.

El resto de búsquedas se realizan de manera similar únicamente cambian: el URL del método del servicio web y la información a buscar.

```
350    //Petición al servidor para buscar clientes por nombre
351    List<ClienteApp> clientes =
352                        conectar.PeticionListaClientes(user, "BuscarPorNombreCliente/", variables, "GET");
353
354    //Método para cargar el DataGridView con los clientes buscados
355    CargarDataGridView(clientes);
```

Código 2.19 Búsqueda de clientes por nombre

Creación de la interfaz de usuario para visualizar un mapa – aplicación de escritorio: La Figura 2.41 presenta la interfaz de usuario para visualizar el mapa en la aplicación de escritorio, donde se realiza una búsqueda según: provincia, ciudad, calle y código postal.

Al presionar el botón `Buscar` se dirige a la dirección ingresada, para mostrar los valores de latitud y longitud se debe presionar un clic sobre un punto exacto del mapa.

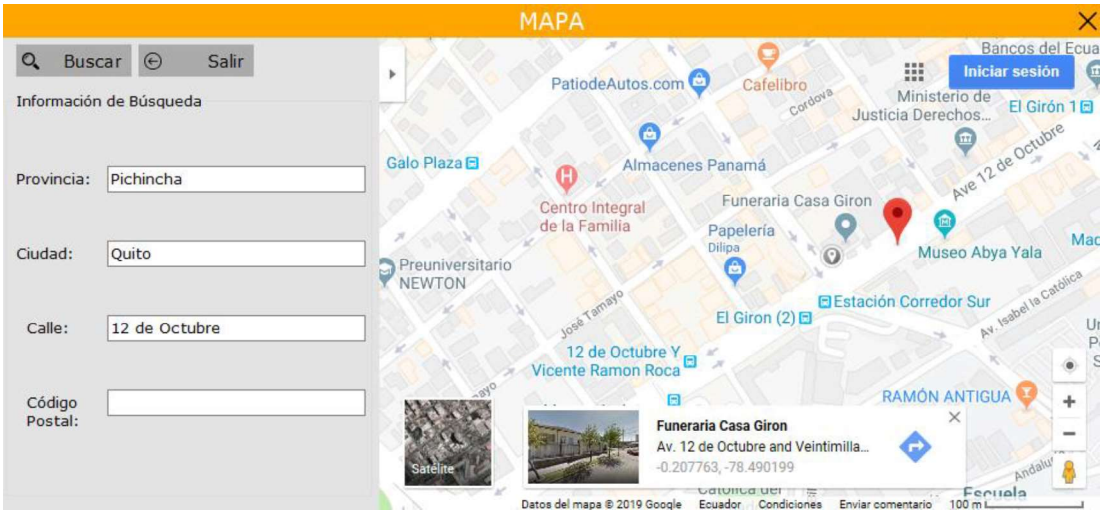


Figura 2.41 Interfaz de usuario para visualizar el mapa

Creación de la interfaz de usuario para la gestión de zonas – aplicación de escritorio: La Figura 2.42 muestra la interfaz de usuario para gestionar zonas en la aplicación de escritorio, en la interfaz se permite ingresar y modificar: número de zona, descripción de la zona y partición de la zona.

Para almacenar la información se presiona sobre el botón `Guardar`. Al presionar sobre el botón `Cancelar` se borra la selección del ítem y se permite almacenar una nueva zona. El botón `Eliminar` borra la zona seleccionada.

Figura 2.42 Interfaz de usuario para gestionar zonas

Implementar código para la gestión de zonas – aplicación de escritorio: En el Código 2.20 se presenta un fragmento del método para eliminar una zona en la aplicación de escritorio, en las líneas 352 y 353 se llama al método `Peticion`, el cual recibe como argumentos: el objeto del usuario autenticado, la URL del método del servicio web, el Id de la zona y el método `DELETE` para indicar al servicio que se requiere eliminar la información. La eliminación de una zona borra completamente el registro de la base de datos contrario a lo que se realiza en la eliminación de un usuario que únicamente ocultaba el registro de la aplicación.

```
351     //Petición al servidor para eliminar una zona
352     int resultadoPeticion =
353                     conectar.Peticion(user, "EliminarZonaSistemaAlarma/", variables, "DELETE");
```

Código 2.20 Eliminación de una zona

Creación de la interfaz de usuario para ubicar clientes – aplicación Android: La Figura 2.43 presenta la interfaz de usuario que mostrará las ubicaciones de los clientes en la aplicación Android, en la parte superior están los botones que sirven para (de izquierda a derecha):

- Primer Botón: Muestra el menú de navegación lateral.
- Segundo Botón: Lista a todos los clientes con una marca de color rojo.
- Tercer Botón: Limpia todos los clientes del mapa.
- Cuarto Botón: Permite cambiar entre distintos tipos de mapa: *road map*, *satellite*, *terrain* y *hybrid*.
- Quinto Botón: Se utiliza para cambiar las preferencias de la aplicación Android.

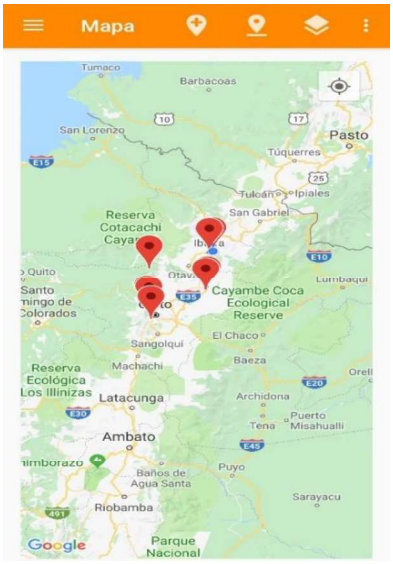


Figura 2.43 Interfaz de usuario para mostrar las ubicaciones de los clientes

Implementar código para ubicar clientes – aplicación Android: Previamente a ubicar clientes, se debe mostrar el mapa en la interfaz de la aplicación Android.

En el Código 2.21 se muestra un fragmento del método `onCreateView` que mostrará el mapa, en la línea 174 se comprueba si el objeto `mapFragment` está creado, objeto que es el manejador del fragmento para crear el mapa; en la línea 175 se crea el objeto; en la línea 176 se obtiene el mapa de manera asincrónica.

```
166     @Override
167     public View onCreateView(LayoutInflater inflater, ViewGroup container,
168                              Bundle savedInstanceState) {
169         View rootView = inflater.inflate(R.layout.content_main, container, false);
170         /*
171         Si el fragmento no esta creado, se invoca al manejador de fragmentos del mapa,
172         y se lanza en el FrameLayout del layout que estaba definido para la clase Main.
173          */
174         if (mapFragment == null) {
175             mapFragment = SupportMapFragment.newInstance();
176             mapFragment.getMapAsync(new OnMapReadyCallback() {
```

Código 2.21 Método `onCreateView` para ubicar clientes

De la misma manera que el Código 2.8 para la autenticación, también se implementa un código similar para la ubicación de clientes, exclusivamente se mostrarán los cambios realizados en los métodos `doInBackground` y `onPostExecute`.

El Código 2.22 muestra un fragmento del método `doInBackground`, en las líneas 380 y 381 agrega el ID y token del usuario en una variable `token`, misma que servirá para validar que la petición sea realizada por un usuario previamente autenticado.

En las líneas 385 a 387 a través del objeto `ConexionRestful` se llama al método `connectREST` que contiene los argumentos: la URL del servicio web, el método del servicio web, el tipo de contenido, la información extra a enviar, el método HTTP, el ID y token, el tiempo máximo de conexión y el tiempo máximo de lectura.

```
378             //Se llena el Map<String,String> token que contiene el Id
379             //y Token para acceder a los métodos del servicio WCF
380             token.put("Token", user1.getToken());
381             token.put("Id", user1.getIdUsuario().toString());
382
383             //Se realiza la petición al servidor a través de la clase
384             //ConexionRestful usando su método connectREST
385             resultado = ConexionRestful.connectREST(hostname, args[0],
386                     "application/x-www-form-urlencoded;charset=UTF-8", "",
387                     args[1], token, tiempoconexion*1000, tiempolectura*1000);
```

Código 2.22 Método `doInBackground` para ubicar clientes

El Código 2.23 muestra un fragmento del método `onPostExecute`, con las diferencias sobre la obtención de un objeto de tipo lista `clientes` en vez de `usuario`.

En las líneas 457 a 459 del objeto `clientes` se obtiene los valores para la latitud y longitud que se almacena en la constante `localizacion`.

Finalmente, en las líneas 463 a 465 usando `localizacion` y el nombre del cliente, se agregan al mapa utilizando una marca.

```
455     //Condicional que realiza el listado de clientes
456     else {
457         LatLng localizacion = UBICACION;
458         for (ClienteApp aux : clientes) {
459             localizacion = new LatLng(Double.parseDouble(aux.getLatitud()),
460                     Double.parseDouble((aux.getLongitud().toString())));
461
462             //Adiciona marcas en el mapa
463             mMap.addMarker(new MarkerOptions()
464                     .position(localizacion)
465                     .title(aux.getNombreCliente())
466             );
467         }
```

Código 2.23 Método `onPostExecute` para ubicar clientes

2.9. SPRINT 4 – PERSONAS

La implementación del *sprint* 4 cuenta con el desarrollo de la gestión de personas y la gestión de responsabilidades para la aplicación de escritorio.

Creación de la interfaz de usuario para la gestión de personas – aplicación de escritorio: En la Figura 2.44 se presenta la interfaz de usuario para gestionar personas en la aplicación de escritorio.

En la parte superior derecha se permite realizar búsquedas según el criterio: nombre y cuenta de cliente. En la interfaz se muestran todas las personas almacenadas en la base de datos. Los botones ubicados en la parte superior izquierda tienen las funcionalidades:

- Nuevo: Permite crear una nueva persona.
- Modificar: Permite cambiar la información de una persona.
- Eliminar: Permite deshabilitar la cuenta de una persona.
- Responsabilidad: Permite agregar los responsables de los clientes.

En el ejemplo se observan todas las personas cargadas de la base de datos.

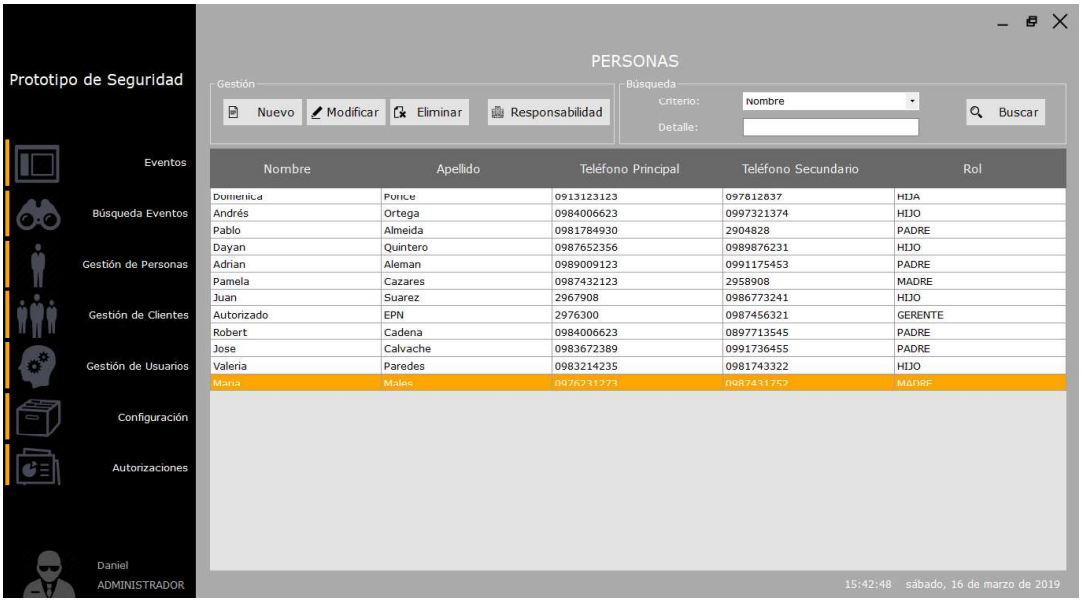


Figura 2.44 Interfaz de usuario para gestionar personas

La Figura 2.45 muestra la interfaz de usuario para ingresar y modificar una persona en la aplicación de escritorio, que registra: nombre, apellido, rol, teléfono y teléfono secundario, se almacena la información presionando el botón `Guardar`, o para salir sin guardar cambios se presiona el botón `Salir` o el icono con la equis.

Figura 2.45 Interfaz de usuario para ingresar y modificar una persona

Creación de la interfaz de usuario para la gestión de responsabilidades – aplicación de escritorio: La Figura 2.46 muestra la interfaz de usuario para ingresar y eliminar una responsabilidad en la aplicación de escritorio, en esta interfaz se carga el nombre de la persona seleccionada de la interfaz para gestionar personas, también se muestra una lista

desplegable en el campo `Responsable del Cliente` con todos los clientes, luego se deberá seleccionar un cliente y almacenar el cambio al presionar el botón `Guardar`.

Después de guardar, la responsabilidad se agrega a la lista mostrándose la información: cuenta y responsable del cliente; el botón `Eliminar` borra la responsabilidad seleccionada. En el ejemplo se indica que la persona Andrés es responsable del cliente Casa Brasil.

Figura 2.46 Interfaz de usuario para ingresar y eliminar una responsabilidad

2.10. SPRINT 5 – EVENTOS

En el *sprint* 5 se desarrolla la gestión de eventos, la búsqueda y exportación de los eventos para la aplicación de escritorio; el listado de los eventos, los mecanismos de escaneo de códigos QR y de localización para la aplicación Android.

Creación de la interfaz de usuario para la gestión de eventos – aplicación de escritorio: En la Figura 2.47 se indica la interfaz de usuario para gestionar eventos en la aplicación de escritorio, en la pantalla principal se cargan los eventos creados de las últimas 24 horas, en la parte inferior se puede visualizar la información de un evento seleccionado de la parte superior como: la información complementaria del evento, la información del cliente, las zonas activadas y las personas autorizadas.

En el extremo inferior derecho se observa la hora y fecha del día en curso. En la parte superior derecha se encuentra los botones de minimizar, maximizar y cerrar. Por último, sobre el recuadro donde se muestran los eventos están los componentes:

- Nuevo: Permite crear un nuevo evento.
- Eliminar: Permite deshabilitar un evento.
- Actualizar Evento: Refresca los eventos según el tiempo seleccionado.

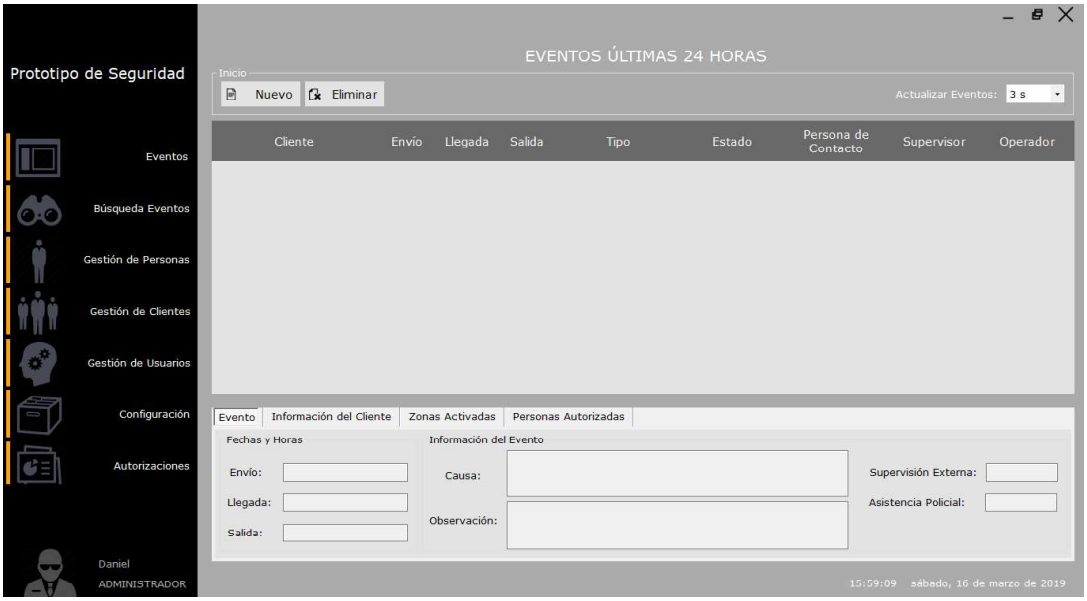


Figura 2.47 Interfaz de usuario para gestionar eventos

La Figura 2.48 indica la interfaz de usuario para ingresar un evento, donde se registra: la cuenta del cliente, el supervisor a enviar, el tipo del evento, nombre del cliente, el operador que ingresa el evento, envío con fecha y hora, y estado del evento, se almacena la información presionando el botón `Nuevo`, o para salir sin guardar cambios se presiona el botón `Salir`. Luego de guardar los cambios exitosamente se habilita la parte inferior que permite ingresar una o varias zonas activadas de la propiedad del cliente.

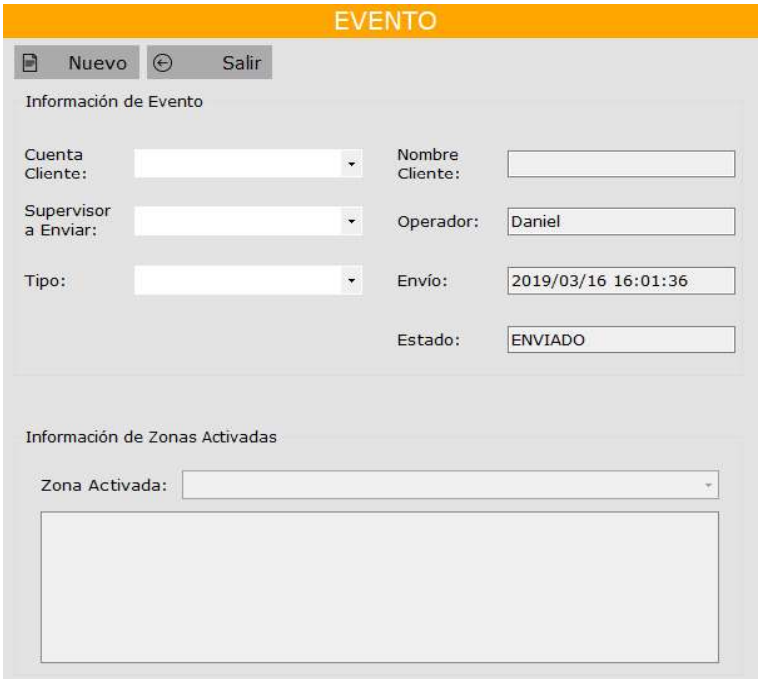


Figura 2.48 Interfaz de usuario para ingresar un evento

Creación de la interfaz de usuario para buscar eventos – aplicación de escritorio: La Figura 2.49 presenta la interfaz de usuario para realizar búsquedas de eventos en la aplicación de escritorio.

En el campo criterio por cliente se puede seleccionar la búsqueda según: nombre y número de cuenta, se ingresa el detalle en el campo inferior.

Adicional a los criterios anteriores se puede buscar según los parámetros: envío, llegada y salida con fecha y hora, asistencia policial, supervisión externa, tipo evento, estado evento, persona, supervisor y operador. Los eventos que coincidan con los criterios de búsqueda se cargan en la parte inferior, presionando el botón `Buscar`.

Existe la posibilidad de que los mismos eventos capturados se puedan exportar a un archivo en formato Excel presionando el botón `Exportar a Excel`.

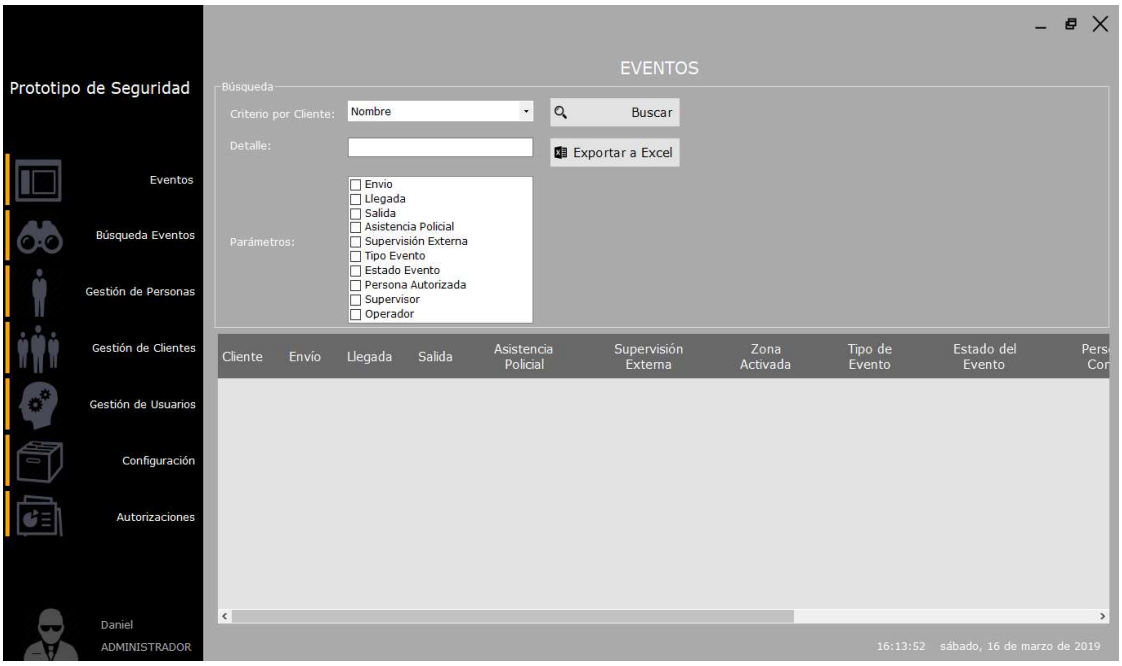


Figura 2.49 Interfaz de usuario para buscar eventos

Creación de la interfaces de usuario para navegar en la aplicación y listar eventos – aplicación Android: En la Figura 2.50 se presenta la interfaz de usuario para navegar en la aplicación Android, esta contiene una barra de navegación lateral en la parte izquierda, en la parte superior se detalla el nombre, apellido y alias del usuario autenticado.

En la parte inferior se muestran los distintos accesos a los eventos según su estado, y al final la opción ubicaciones clientes que muestra un mapa y permite observar a los clientes según una marca.

Figura 2.50 Interfaz de usuario para navegación en la aplicación Android

La Figura 2.51 muestra la interfaz de usuario para listar los eventos finalizados en la aplicación Android, en la captura de pantalla se puede apreciar un evento con los datos de: cliente, envío, llegada y salida con fechas y horas, autorizado, tipo de evento, causa y observación.

Figura 2.51 Interfaz de usuario para listar los eventos finalizados

Creación de la interfaz de usuario para escanear códigos QR – aplicación Android: En la Figura 2.52 se presenta la interfaz de usuario para escanear códigos QR en la aplicación Android, en la captura de pantalla se observa que la cámara del dispositivo Android está iniciada y se encuentra capturando la información de un código QR.

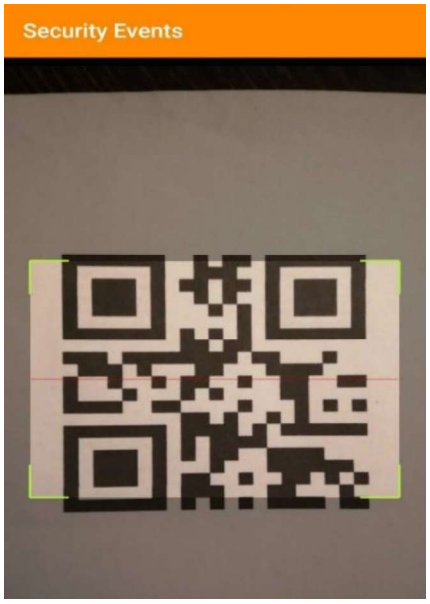


Figura 2.52 Interfaz de usuario para escanear códigos QR

Implementar código para capturar los códigos QR – aplicación Android: El Código 2.24 muestra cómo se inicia la cámara del dispositivo Android para capturar los códigos QR en la aplicación Android.

En la línea 115 se instancia la clase `ZXingScannerView` que contiene los métodos para iniciar la cámara y realizar acciones al escanear un código QR; la línea 116 indica que `escanerView` será la vista a mostrar; la línea 117 define un método que realizará una funcionalidad al capturar un código; la línea 118 inicia la cámara del dispositivo.

```
114        // Instancia de la clase ZXingScannerView
115        escanerView = new ZXingScannerView( context: this);
116        setContentView(escanerView);
117        escanerView.setResultHandler(this);
118        escanerView.startCamera();
```

Código 2.24 Método para capturar los códigos QR

En el Código 2.25 se presenta un fragmento del método implementado cuando se captura un código QR en la aplicación Android.

En la línea 131 se valida que el evento seleccionado previamente de la lista tenga una posición mayor a cero y contenga valores; en la línea 135 se compara que el código QR externo almacenado sea igual al valor escaneado del código QR, si la validación es exitosa la variable que indica la validación de un código QR y la variable que indica si la supervisión fue externa toman el valor de `true` (línea 139 y 142). Finalmente en las líneas 145 a 146 se muestra un mensaje que la validación fue correcta.

```
128     //Método para el escaneo de códigos QR
129     @Override
130     public void handleResult(Result result) {
131         if ((pos >= 0) && (evento1 != null)) {
132
133             //Condicional que compara el código QR Externo del Cliente
134             //almacenado con el capturado mediante la cámara
135             if (evento1.getCliente().getCodigoQRExternoCliente().toString()
136                                                     .equals(result.getText())) {
137
138                 //Setea la variable validaciónQR en true
139                 validacionQR = true;
140
141                 //Cambio a true el campo supervisionExterna del Evento
142                 evento1.setSupervisionExterna(true);
143
144                 //Mensaje del Resultado obtenido
145                 Toast.makeText(Escaneo.this, "Lectura de Código Exitoso ",
146                                                     Toast.LENGTH_LONG).show();
```

Código 2.25 Método que se realiza al capturar un código QR

Implementar código para tomar la ubicación – aplicación Android: El Código 2.26 muestra un fragmento del método para iniciar la ubicación del dispositivo Android, en las líneas 108 a 111 se establece el objeto `locationManager` para iniciar la ubicación del dispositivo.

Se utiliza el mismo objeto para dos proveedores de localización `GPS_PROVIDER` y `NETWORK_PROVIDER` que usarán la ubicación por GPS y por medio de Internet respectivamente.

El objeto llama al método `requestLocationUpdates` para recibir las actualizaciones de la ubicación que tiene como argumentos: el proveedor de localización, el tiempo mínimo de actualización en milisegundos, la distancia mínima de actualización en metros y los objetos `locationListenerGPS` y `locationListenerNetwork` que realizan una acción cuando se cambia la ubicación.

```
102         //Parámetros del locationManager:
103         //1)Nombre del proveedor de localización al que se quiere suscribir.
104         //2)Tiempo mínimo entre actualizaciones, en milisegundos.
105         //3)Distancia mínima entre actualizaciones, en metros.
106         //4)Instancia de un objeto LocationListener, se implementa para definir
107         //las acciones a realizar en cada actualización de la posición.
108         locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER,
109                                                 1 * 60 * 1000, 10, locationListenerGPS);
110         locationManager.requestLocationUpdates(LocationManager.NETWORK_PROVIDER,
111                                                 1 * 60 * 1000, 10, locationListenerNetwork);
```

Código 2.26 Ubicación del dispositivo Android

En el Código 2.27 se expone la clase `LocationListener` la cual se ejecuta al ser llamada por el objeto `locationManager` cuando se produce un cambio de ubicación, la clase `LocationListener` sobrescribe cuatro métodos, siendo el método `onLocationChanged` el único que se implementa, el método `onLocationChanged` se ejecuta cuando se cambia la ubicación del dispositivo; en las líneas 848 y 849 se almacenan la longitud y latitud que servirán para validar la ubicación; en las líneas 851 y 852 se muestra un mensaje al capturar la ubicación.

El ejemplo muestra la localización usando el proveedor de localización Internet; no obstante, el código con el proveedor GPS es semejante.

Es importante mencionar que los dos proveedores de localización funcionan en conjunto y almacenan los valores en las mismas variables, para obtener la ubicación independientemente del proveedor.

```
842     //Usada para recibir las notificaciones de la clase LocationManager cuando cambia
843     //la localización
844     private final LocationListener locationListenerNetwork = new LocationListener() {
845         //Método que se ejecuta cuando cambia la localización
846         public void onLocationChanged(Location location) {
847             try {
848                 longitude = location.getLongitude();
849                 latitude = location.getLatitude();
850
851                 Toast.makeText(Main.this , "Ubicación de la Red obtenida",
852                                                 Toast.LENGTH_SHORT).show();
853             }
854             catch (Exception exception) {
855                 exception.printStackTrace();
856                 Log.e("Error",exception.getMessage());
857             }
858         }
```

Código 2.27 Método para capturar la localización

Implementar código para comparar las ubicaciones – aplicación Android: El Código 2.28 muestra el método que valida la localización del dispositivo Android con la localización almacenada en la base de datos.

En las líneas 290 a 293 se recupera la latitud y longitud almacenadas en la base de datos; en las líneas 296 y 297 se obtiene la diferencia entre los valores de latitud y longitud almacenados con los capturados. En las líneas 300 y 301 los resultados obtenidos se multiplican por mil para manejar una validación con números que no sobrepasen de uno, usando tres decimales de aproximación.

En la línea 305 se verifica que el valor absoluto del resultado sea menor a uno, con eso se concluye que el dispositivo Android se encuentra en la misma posición que la almacenada en la base de datos, caso contrario el dispositivo se encuentra en una posición distinta.

Por último, en las líneas 306 y 307 se establece el valor de la variable que indica la validación de la ubicación en `true` y se muestra un mensaje de validación exitosa, caso contrario la variable que valida la ubicación toma el valor de `false` y se muestra un mensaje de error (líneas 311 a 313).

```
286     private void MetodoValidacionLocalizacion(){
287
288         //Conversion de los valores a double de latitud y longitud almacenados
289         //en la base de datos
290         Double latitudCliente =
291                 Double.parseDouble(evento1.getCliente().getLatitud());
292         Double longitudCliente =
293                 Double.parseDouble(evento1.getCliente().getLongitud());
294
295         //Resta de los valores latitud y longitud almacenados con los capturados
296         Double resultadoLatitud = latitudCliente - latitude;
297         Double resultadoLongitud = longitudCliente - longitude;
298
299         //Se multiplica los resultados de latitud y longitud por 1000
300         resultadoLatitud = resultadoLatitud * 1000;
301         resultadoLongitud = resultadoLongitud * 1000;
302
303         //Condicional que evalúa que los resultados de latitud y longitud sean menor
304         //o igual a uno, para concluir si es una validación exitosa o fallida
305         if ((Math.abs(resultadoLatitud) <= 1) && (Math.abs(resultadoLongitud) <= 1)) {
306             validacionGPS = true;
307             Toast.makeText(Escaneo.this, "Validación exitosa de posición",
308                                                     Toast.LENGTH_LONG).show();
309         }
310         else{
311             validacionGPS = false;
312             Main.mensaje(Escaneo.this, "Error de Posición",
313                                         "La ubicación no es la correcta");
314         }
315     }
```

Código 2.28 Comparación de ubicación capturada con ubicación almacenada

Creación de la interfaz de usuario para registrar eventos – aplicación Android: La Figura 2.53 muestra la interfaz de usuario para registrar eventos en la aplicación Android, donde se ingresa: causa del evento, observación, asistencia policial y autorizado.

Existen tres botones en la parte superior (de izquierda a derecha):

- Primer Botón: Permite regresar al *layout* al listado de eventos en desarrollo.
- Segundo Botón: Almacena la información del evento.
- Tercer Botón: Salir sin guardar cambios.

Figura 2.53 Interfaz de usuario para registrar eventos

2.11. SPRINT 6 – CONFIGURACIONES

En el *sprint* 6 se desarrolla las configuraciones de parámetros y la habilitación de eventos, personas, clientes y usuarios.

Creación de la interfaz de usuario para configurar parámetros – aplicación de escritorio: La Figura 2.54 muestra la interfaz de usuario para configurar parámetros en la aplicación de escritorio, como el agregar o eliminar: el rol de las personas, los planes, tipos y ciudades de los clientes, y los tipos de eventos. Además, es posible modificar la URL de conexión al servicio web WCF.

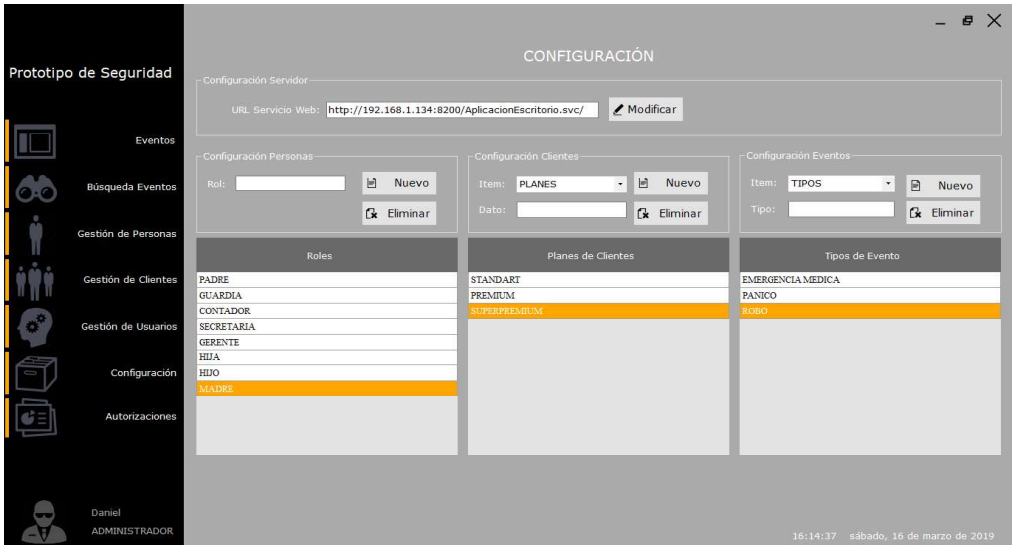


Figura 2.54 Interfaz de usuario para configurar parámetros

Implementar código para modificar la URL de conexión al servicio web – aplicación de escritorio: En el Código 2.29 se muestra un fragmento del método para modificar la URL del servicio web en la aplicación de escritorio.

En la línea 1004 se abre el archivo de configuración donde se almacena la URL; en la línea 1007 se obtiene la configuración de la aplicación.

En la línea 1010 se modifica la cadena que tiene la URL con el nuevo valor de la URL colocado en el *textbox*.

La línea 1013 almacena los cambios; la línea 1016 actualiza la configuración de la aplicación de escritorio para reflejar los cambios.

Finalmente, en las líneas 1019 a 1021 se mostrará un cuadro de dialogo con un mensaje de modificación correcta.

```
997   //Evento click del botón Modificar que almacena el valor del textbox txt_urlServicioWeb en la
998   //variable conexionServidor del App.config
999   private void btn_Modificar_Click(object sender, EventArgs e)
1000  {
1001      try
1002      {
1003          //Abre el archivo de configuración actual, con un nivel de usuario de configuración
1004          var configFile = ConfigurationManager.OpenExeConfiguration(ConfigurationUserLevel.None);
1005
1006          //Obtiene la configuración de la aplicación.
1007          var settings = configFile.AppSettings.Settings;
1008
1009          //Establece el valor del textbox en la variable conexionServidor
1010          settings["conexionServidor"].Value = txt_urlServicioWeb.Text;
1011
1012          //Se modifica el archivo de configuración
1013          configFile.Save(ConfigurationSaveMode.Modified);
1014
1015          //Actualiza la sección de configuración de la aplicación
1016          ConfigurationManager.RefreshSection(configFile.AppSettings.SectionInformation.Name);
1017
1018          //Mensaje de cambio correcto
1019          DialogResult resultado = new DialogResult();
1020          Form mensaje = new MessageBoxForm("Se modificó la URL del servicio web con éxito", "Correcto");
1021          resultado = mensaje.ShowDialog();
1022      }
```

Código 2.29 Modificación de la URL del servicio web

Creación de la interfaz de usuario para la habilitación de registros – aplicación de escritorio: La Figura 2.55 presenta la interfaz de usuario para habilitar: eventos, usuarios, personas y clientes en la aplicación de escritorio.

En el recuadro de la parte inferior se muestra la información anulada correspondiente al criterio de habilitación. El botón `Habilitar` modificará el valor de disponibilidad para poder utilizar el registro en las funciones del prototipo.

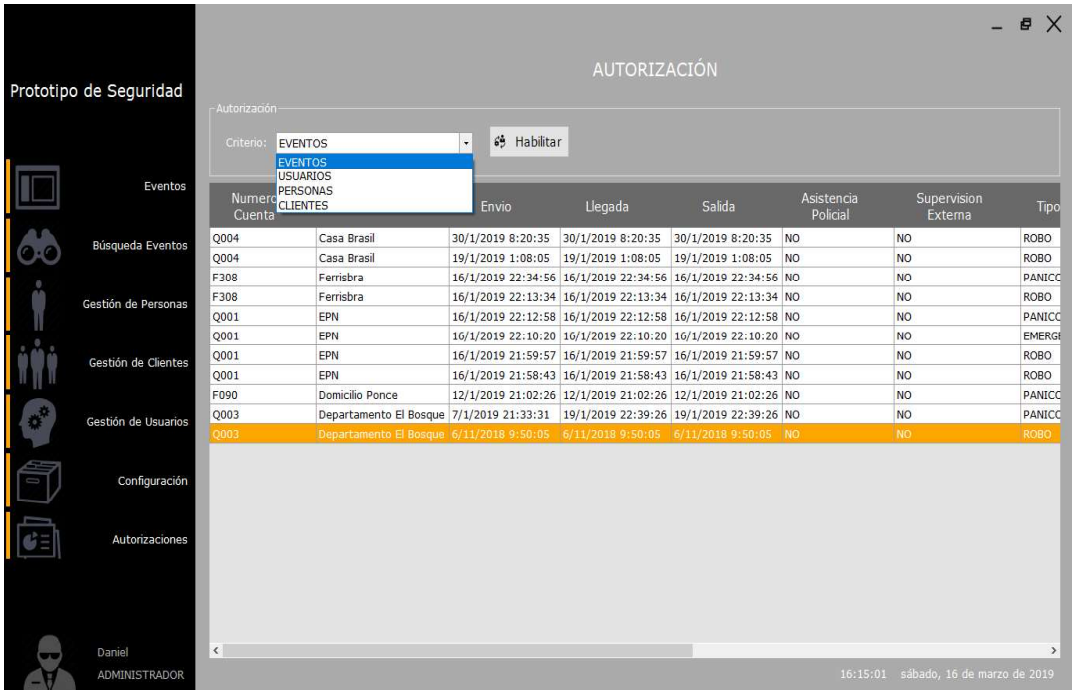


Figura 2.55 Interfaz de usuario para la habilitación de registros

Implementar código para habilitar eventos, usuarios, clientes y personas – aplicación de escritorio: En el Código 2.30 se muestra un fragmento del método para habilitar un evento, en las líneas 402 a 403 se llama al método `Peticion` que recibe: el objeto del usuario autenticado, la URL del método del servicio web, la información del evento seleccionado y el método PUT para indicar al servicio web que se requiere realizar un cambio, como resultado se obtiene un entero para indicar si la habilitación fue correcta.

```
401  //Petición al servidor para habilitar un evento
402  int resultadoPeticion =
403                  conectar.Peticion(user, "DisponibilidadEvento/", variables, "PUT");
```

Código 2.30 Habilitación de un evento

Las habilitaciones de usuarios, clientes y personas, siguen el mismo esquema que la habilitación de eventos, se diferencian en la URL del servicio web y la información seleccionada para habilitar.

3. RESULTADOS Y DISCUSIÓN

En este capítulo se presentan las pruebas de funcionamiento realizadas al prototipo. Se muestran las pruebas siguiendo los *sprints* en los cuales se desarrollaron, pruebas acerca de la ejecución de la base de datos, el servicio web, la aplicación Android y la aplicación de escritorio.

Para las pruebas de validación se aplicó una encuesta a 15 personas de la compañía de seguridad SEDYM CIA LTDA, que validan el funcionamiento del prototipo desde la perspectiva del usuario.

También se incluye una sección de corrección de los errores identificados a lo largo del desarrollo del prototipo y en las pruebas de funcionamiento.

3.1. PRUEBAS DE FUNCIONAMIENTO

Las pruebas de funcionamiento se ejecutaron siguiendo el orden de los *sprints*.

3.1.1 BASE DE DATOS

Para verificar el funcionamiento de la base de datos, se ejecutaron consultas SQL directamente en la base.

En la Figura 3.1 se presentan los resultados de una consulta realizada a la tabla `Usuario`, se observa el primer registro con el id de número uno, el nombre “Daniel”, con apellido “Cadena”, alias “Buho”, su número de teléfono, su cédula, su correo, el hash de su contraseña y su token.

El valor de cero en `primerInicioSesion` indica que ha iniciado sesión al menos una vez, el valor de uno en `disponibilidadUsuario` indica que el usuario está disponible para utilizar las funcionalidades del prototipo y su llave foránea `fkidTipoUsuario` que indica el tipo del usuario, si es uno es un administrador, dos supervisor y tres operador.

```
select * from Usuario
```

100 %

Results | Messages

	idUsuario	nombre...	apellido...	aliasUs...	telefono...	cedula	correo	hashContrase...	token	primerIn...	dispo...	fkidTip...
1	1	Daniel	Cadena	Buho	099117...	1003...	daniel....	03ac674216f...	6a...	0	1	1
2	6	Melanie	Villarreal	alcon	098746...	1003...	melani...	03ac674216f...	d4...	0	1	3
3	8	Hugo	Flores	R1	097278...	1772...	hugo@...	03ac674216f...	f3b...	0	1	2
4	17	Patricio	Gomez	R2	099924...	1772...	patricio...	03ac674216f...	2b...	0	1	2

Figura 3.1 Resultado de la consulta a la tabla `Usuario`

En la Figura 3.2 se presentan los resultados de una consulta realizada a la tabla `TipoUsuario` que identificar los tipos de usuarios, se observa el primer registro con el id

uno y la descripción ADMINISTRADOR, el segundo registro con el id dos y la descripción SUPERVISOR, y el tercer registro con id tres y la descripción OPERADOR.

```
select * from TipoUsuario
```

100 %

Results | Messages

	idTipoUsuario	descripcionTipoUsuario
1	1	ADMINISTRADOR
2	2	SUPERVISOR
3	3	OPERADOR

Figura 3.2 Resultado de la consulta a la tabla `TipoUsuario`

3.1.2 SERVICIO WEB

En la Figura 3.3 se visualiza un navegador con la dirección del sitio web creado para alojar el servicio WCF, se puede observar los archivos que conforman el servicio WCF.

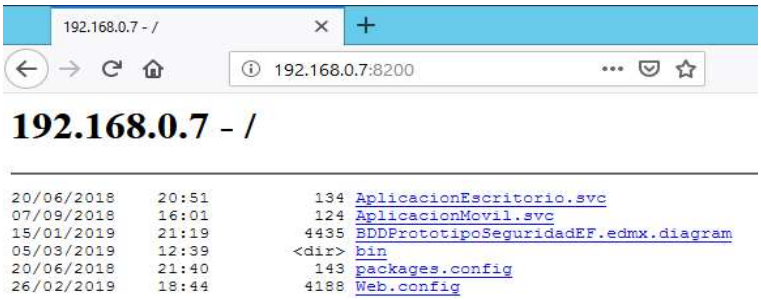


Figura 3.3 Página del sitio web

En la Figura 3.4 se muestra una captura de pantalla del software Postman[32] que permite realizar solicitudes HTTP para probar el funcionamiento del servicio WCF. Se detallan las áreas del software:

1. Establece el método HTTP
2. Establece la URL del método del servicio web.
3. Permite enviar la solicitud.
4. Establece el token en la cabecera necesario para realizar una solicitud al servicio.
5. Establece el Id en la cabecera necesario para realizar una solicitud al servicio.
6. Muestra la respuesta de la solicitud en este caso es la información de los usuarios con disponibilidad `true`.

En consecuencia se puede aseverar que el servicio web permite obtener la información de la base de datos.

[32] **Postman:** Aplicación cliente que permite comprobar un servicio web.

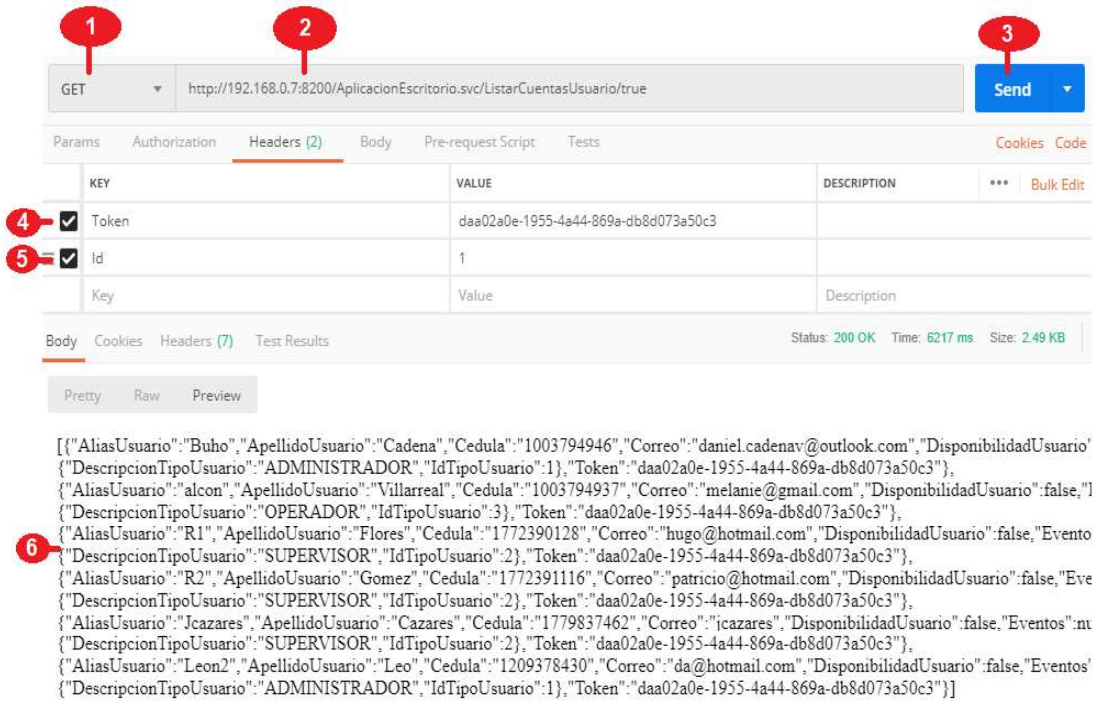


Figura 3.4 Prueba del servicio web WCF

3.1.3 SPRINT 1 – AUTENTICACIONES

Para el *sprint* 1 se prueba la autenticación y el cambio de contraseña tanto para la aplicación Android como para la aplicación de escritorio.

En la Figura 3.5 se muestra la autenticación de un usuario con sus credenciales en la aplicación de escritorio, se puede ver que el acceso fue exitoso, pero debido a que es su primer inicio de sesión, se muestra un mensaje para que el usuario modifique la contraseña.

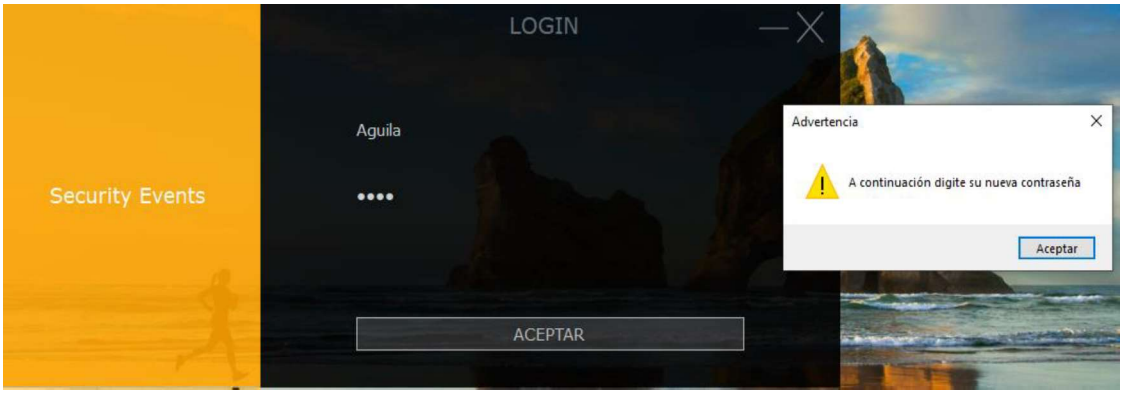


Figura 3.5 Inicio de sesión en la aplicación de escritorio

En la Figura 3.6 se prueba el cambio de contraseña en la aplicación de escritorio, se puede ver que el cambio fue exitoso, debido a que las contraseñas son iguales y además, se muestra un mensaje indicativo.

Figura 3.6 Cambio de contraseña en la aplicación de escritorio

La Figura 3.7 muestra que el usuario autenticado ha ingresado a la aplicación de escritorio debido a que su nombre y tipo de usuario aparecen en la parte inferior de la pantalla.

Figura 3.7 El usuario autenticado luego de cambiar la contraseña en la aplicación de escritorio

En la Figura 3.8 se muestra la autenticación de un usuario con sus credenciales en la aplicación Android.

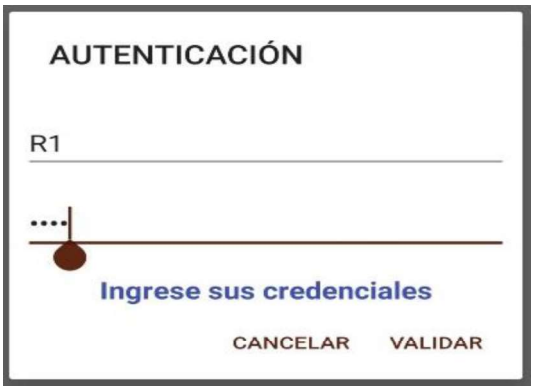


Figura 3.8 Inicio de sesión en la aplicación Android

La Figura 3.9 muestra el cambio de contraseña en la aplicación Android al ser el primer inicio de sesión, si las contraseñas son iguales se almacena la nueva contraseña, caso contrario se mostrará nuevamente la interfaz hasta colocar contraseña iguales o salir de la aplicación.

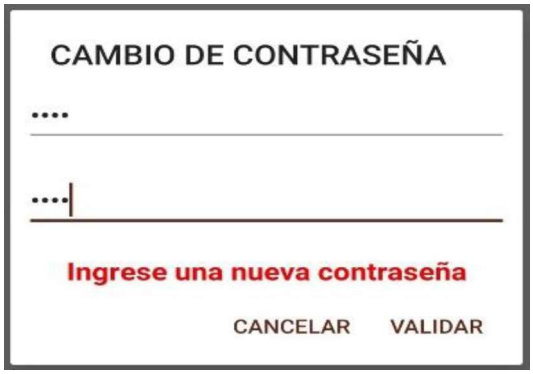


Figura 3.9 Cambio de contraseña en la aplicación Android

En la Figura 3.10 se observa el mensaje de registro exitoso al ingresar la nueva contraseña en la aplicación Android.

Figura 3.10 Mensaje de registro exitoso de la nueva contraseña

En la Figura 3.11 se muestra que el usuario autenticado ha ingresado a la aplicación Android debido a que su nombre, apeliido y alias aparecen en la parte superior de la pantalla.

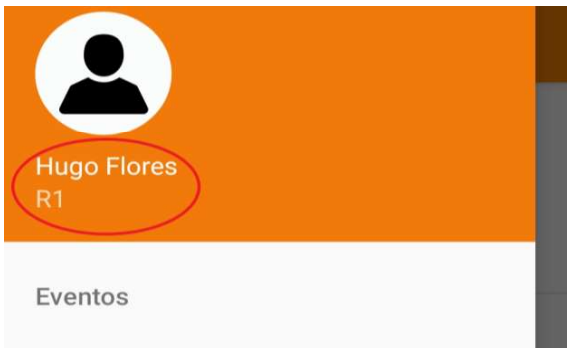


Figura 3.11 El usuario autenticado luego de cambiar la contraseña en la aplicación Android

3.1.4 SPRINT 2 – USUARIOS

El *sprint* 2, realiza la gestión de usuarios para la aplicación de escritorio. Previo a la creación de un usuario en la aplicación de escritorio, se debe validar que la información ingresada en la interfaz sea adecuada, en algunos casos también se muestra un mensaje sobre el campo para alertar con qué información se debe rellenar.

En la Figura 3.12 se muestra la interfaz para ingresar un usuario, donde se observa un mensaje al posicionar el mouse sobre el campo nombre.

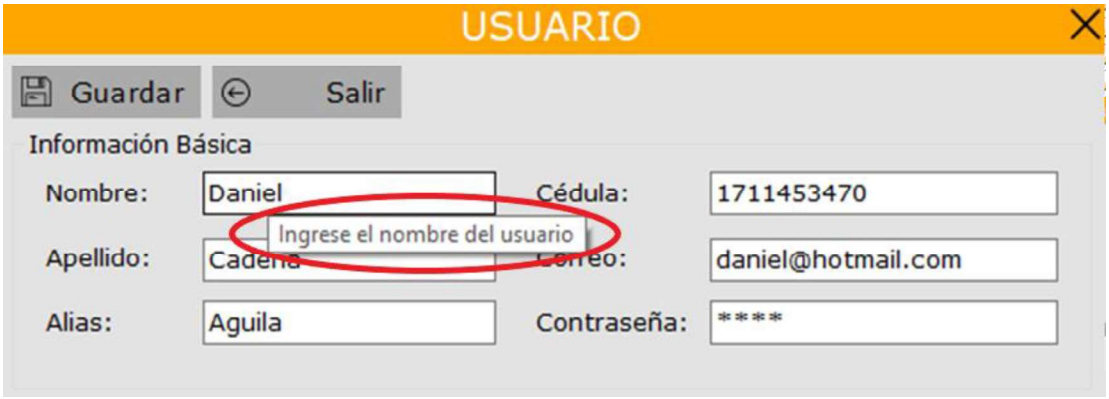


Figura 3.12 Mensaje de aviso sobre la información a llenar

El ejemplo de la Figura 3.13, se produce cuando un usuario ingresa letras en el campo `Teléfono` que permite ingresar solo números, al intentar digitar una letra se muestra un mensaje de advertencia.

Figura 3.13 Validación para ingresar únicamente letras en el campo `Teléfono`

La Figura 3.14 muestra la validación del campo `alias` que permite un único valor registrado en la base de datos, en el ejemplo se trata de cambiar el valor del `alias`, pero al encontrarse registrado por otro usuario, no permite guardar el cambio, mostrándose un mensaje de advertencia.

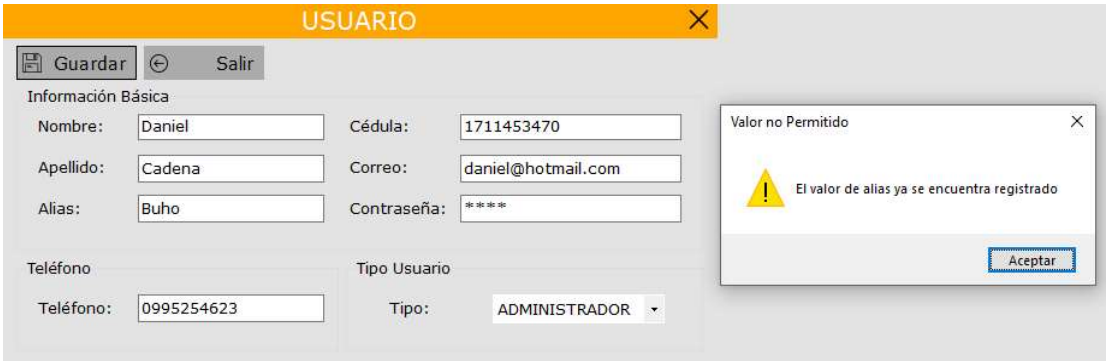


Figura 3.14 Validación para ingresar valores únicos en el campo `alias`

La prueba mostrada en la Figura 3.15 consiste en crear un nuevo usuario, se ingresa todos los campos en la interfaz y como se puede ver en el mensaje el usuario fue registrado con éxito.

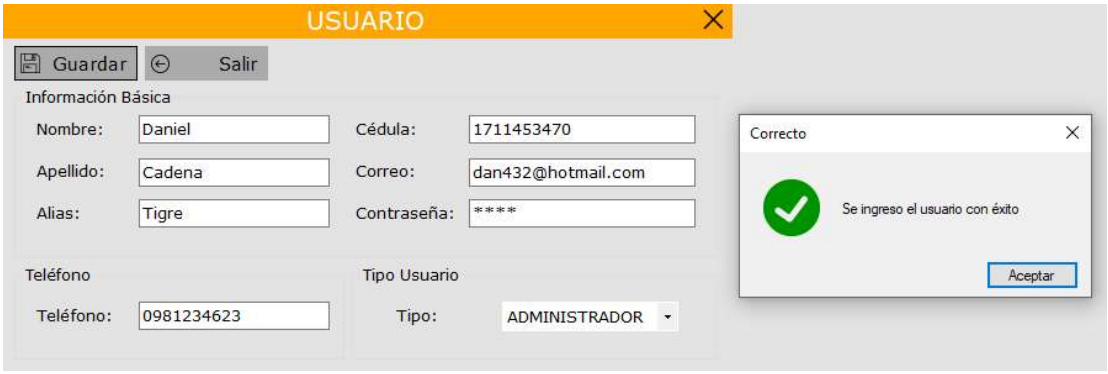


Figura 3.15 Creación de un usuario

En la Figura 3.16 se prueba que en la interfaz para gestionar usuarios se visualice el usuario creado, el registro de color naranja corresponde al usuario previamente creado, por tanto, podemos ver que la aplicación de escritorio está ingresando los usuarios.

Figura 3.16 Interfaz para gestionar usuarios que muestra el usuario creado

En la Figura 3.17 se muestra la consulta del usuario previamente creado a la base de datos; se puede ver que el registro del usuario se está almacenando.

Figura 3.17 Consulta de la base de datos después de ingresar el usuario

La prueba de la Figura 3.18 permite modificar la información de un usuario en la aplicación de escritorio, en la interfaz viene la información precargada con todos los campos exceptuando la contraseña. Para modificar el usuario se ingresan todos los campos, posteriormente se muestra un mensaje de la modificación exitosa.

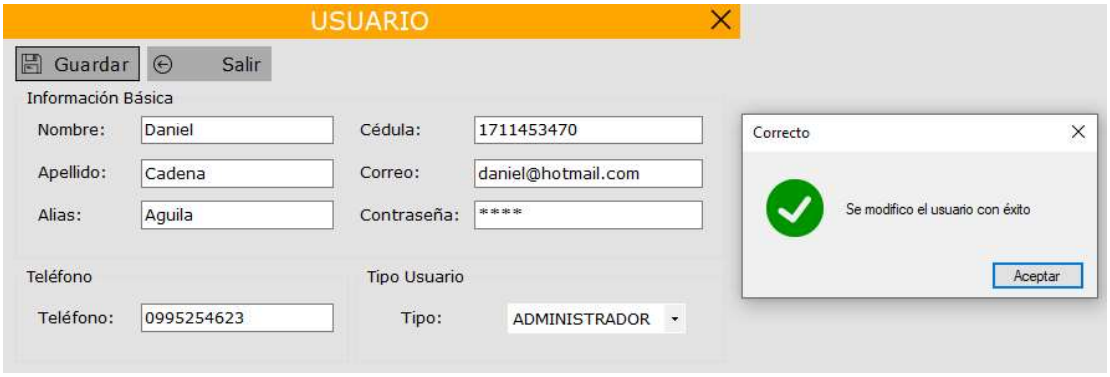


Figura 3.18 Modificación de un usuario

En la Figura 3.19 se muestra la consulta del usuario modificado a la base de datos; se puede ver que el registro se modificó.

Figura 3.19 Consulta de la base de datos después de modificar el usuario

El mensaje de aviso mostrado en la Figura 3.20, se muestra al presionar el botón `Eliminar` de la interfaz para gestionar usuarios, mensaje que permite confirmar la eliminación de un usuario.

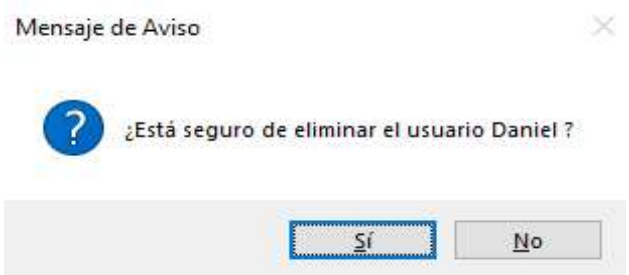


Figura 3.20 Mensaje de aviso de eliminación del usuario

La Figura 3.21 muestra la consulta a la base de datos después de la eliminación del usuario; se puede ver que el registro cambio el campo `disponibilidadUsuario` a 0, por tanto, el usuario no podrá utilizar la funciones del prototipo.

	id...	nombr...	apellid...	alias...	telefonoUsuario	cedula	correo	hashContraseña	token	pri...	disponibilidadUsuario	fki...
1	40	Daniel	Cadena	Aguila	0995254623	1711453470	daniel@hotmail.com	03ac674216f...	NULL	1	0	1

Figura 3.21 Consulta de la base de datos después de eliminar el usuario

3.1.5 SPRINT 3 – CLIENTES

En el *sprint* 3, se realizan los CRUD de clientes y zonas de clientes, los cuales no se muestran debido a la similitud con el CRUD de usuarios; para la aplicación Android se muestran las ubicaciones de los clientes a través de un mapa.

En la Figura 3.22 se prueba la funcionalidad para mostrar las ubicaciones de los clientes en la aplicación Android.

En primer lugar, se carga el mapa, al presionar el botón a lado derecho del título se colocan marcas sobre las direcciones de los clientes. Si se presiona en las marcas se observa el nombre del cliente, existe también la posibilidad de ubicar el dispositivo Android, que en el ejemplo está en la ciudad de Ibarra.

En consecuencia, se puede visualizar según la Figura 3.20 que la interfaz cumple en mostrar las ubicaciones de los clientes, así como la ubicación del dispositivo Android.

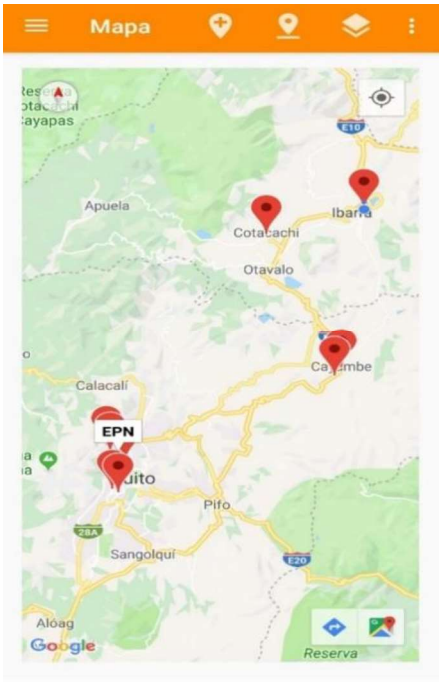


Figura 3.22 Visualización de la ubicación de los clientes

3.1.6 SPRINT 4 – PERSONAS

El *sprint* 4 realiza los CRUD de personas y responsabilidades, no se muestran pruebas de funcionamiento de las personas debido a la similitud con las pruebas del *sprint* 2.

En la Figura 3.23 se prueba el ingreso de una responsabilidad de la persona “Robert” con el cliente “Domicilio Robert Cifuentes”, que se realiza al presionar el botón `Guardar` y se verifica porque se muestra un mensaje del ingreso correcto.

Figura 3.23 Ingreso de una responsabilidad

En la Figura 3.24 se muestra la responsabilidad ingresada en la lista de la parte inferior.

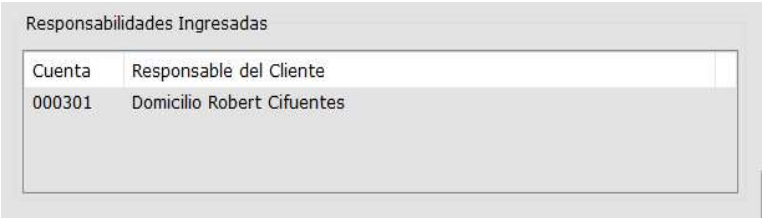


Figura 3.24 Visualización de la responsabilidad creada

3.1.7 SPRINT 5 – EVENTOS

En el *sprint* 5, se realizan el CRUD de eventos, los mecanismos para capturar un código QR y obtener la ubicación del dispositivo Android.

Se realiza una prueba de integración conjunta entre la aplicación Android y la aplicación de escritorio. Consiste en ingresar un evento desde la aplicación escritorio, visualizarlo en la aplicación Android, registrar la información en el evento con la previa validación del código QR y la ubicación, y verificar en la aplicación de escritorio el evento modificado.

En la Figura 3.25 se muestra el evento creado en la aplicación de escritorio para realizar la prueba de integración, el evento de prueba consta del número de cuenta del cliente, el supervisor que se enviará a atender el evento, el tipo del evento, el nombre del cliente, el operador que genera el evento, la fecha y hora de envío, y el estado del evento. Además,

se establece que la zona activada será la número uno, que es la entrada del cliente, en la partición uno del sistema de alarma (permite identificar el grupo al que pertenece una zona).

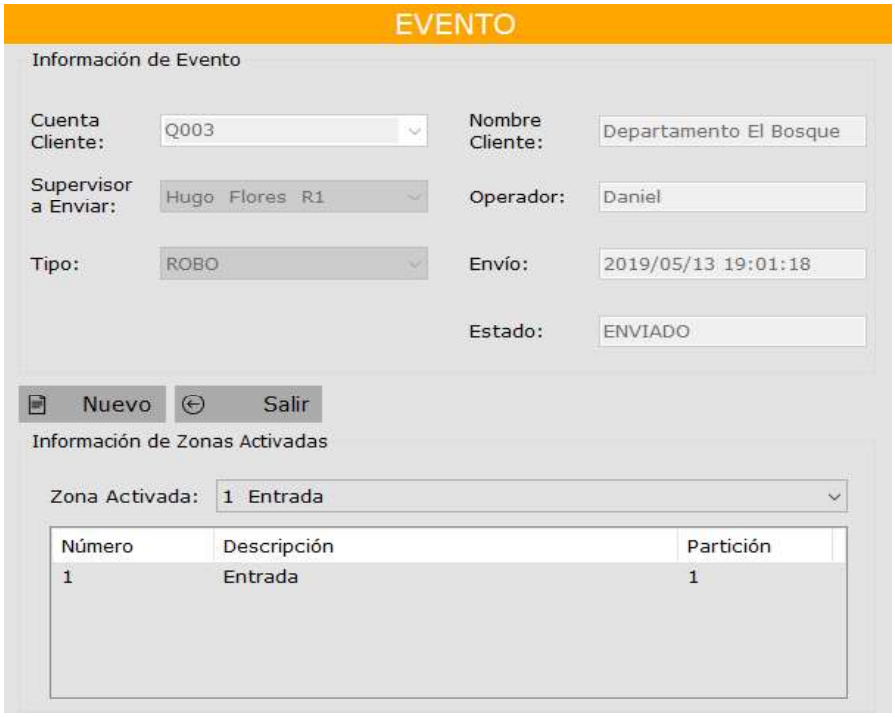


Figura 3.25 Evento de prueba creado en la aplicación de escritorio

La Figura 3.26 presenta una captura de pantalla de la interfaz donde se listan los eventos enviados en la aplicación Android. En el ejemplo se observa el evento de prueba con la información: nombre del cliente, envío con fecha y hora, y tipo de evento.

Figura 3.26 Visualización de un evento enviado en la aplicación Android

Luego de presionar sobre el ítem del evento enviado, se abre una interfaz que inicia la cámara del dispositivo Android, como se observa en la Figura 3.27, la interfaz se encontrará en espera de la lectura de un código QR.

Es importante mencionar que la localización se está capturando constantemente para también corroborar con la localización almacenada del cliente.

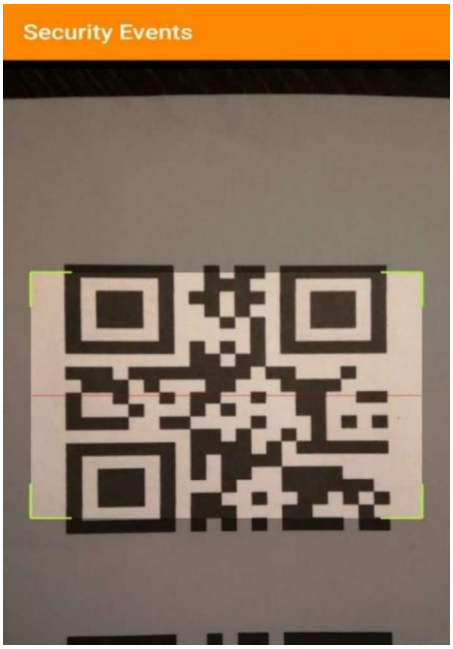


Figura 3.27 Captura de un código QR y validación de la ubicación en la aplicación Android

Una vez se capture un código QR correspondiente al cliente y la validación sobre la ubicación sea correcta, se procede a mostrar los mensajes de validación, como se observa en la Figura 3.28. Al realizar el registro de un evento, se añaden a este la fecha y hora de llegada.

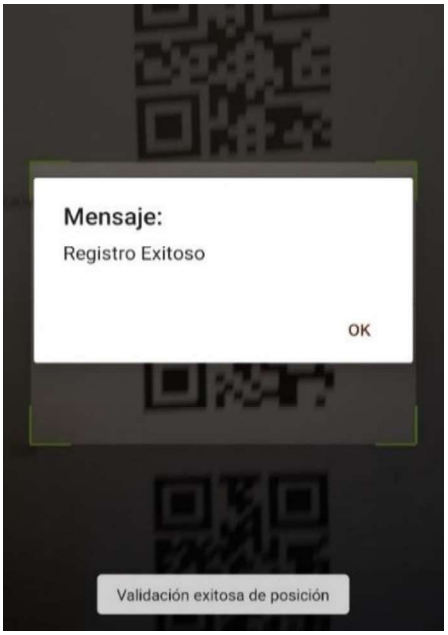


Figura 3.28 Mensaje de registro exitoso al escanear el código del evento enviado en la aplicación Android

La Figura 3.29 presenta una captura de pantalla de la interfaz donde se listan los eventos en desarrollo en la aplicación Android. En el ejemplo se observa el evento creado que cuenta con los valores: nombre del cliente, envío con fecha y hora, llegada con fecha y hora y tipo de evento.

Figura 3.29 Visualización de un evento en desarrollo en la aplicación Android

Luego de presionar sobre el ítem del evento en desarrollo, se abre una interfaz que realiza una similar funcionalidad que la Figura 3.28 de validar el código QR y ubicación. Se distingue porque luego de realizar la validación se muestra una interfaz, como se observa en la Figura 3.30, que permite ingresar la información: causa del evento, observación, asistencia policial y autorizado. Adicionalmente, se agrega la fecha y hora de salida. El evento se modifica al presionar el botón con el visto.

Figura 3.30 Registro de la información de un evento en la aplicación Android

La Figura 3.31 presenta una captura de pantalla de la interfaz donde se muestra el evento finalizado en la aplicación Android.

En el ejemplo se observa únicamente el evento de prueba que cuenta con los valores: nombre del cliente, envío con fecha y hora, llegada con fecha y hora, salida con fecha y hora, nombre de la persona, tipo de evento, causa y observación.

Figura 3.31 Visualización de un evento finalizado en la aplicación Android

En la Figura 3.32 se muestra la interfaz para gestionar eventos de la aplicación de escritorio, donde se encuentra seleccionado de color naranja el evento de prueba con toda la información registrada.

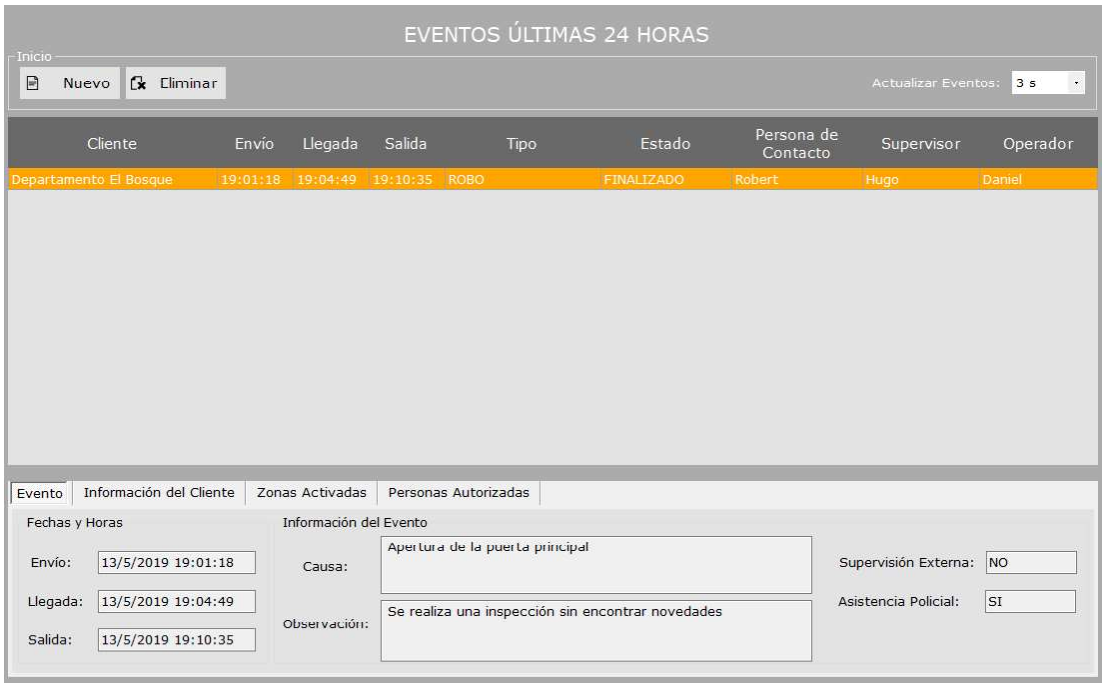


Figura 3.32 Evento de prueba en la aplicación de escritorio

En la Figura 3.33 se muestra el registro de la base de datos con toda la información del evento que incluye la información ingresada por la aplicación de escritorio como por la aplicación Android, por lo tanto, se puede deducir que las aplicaciones registran la información de un evento en la base de datos.

fechaHoraEnvio	fechaHoraLlegada	fechaHoraSalida	causaEvento	observacion	asi...	sup...	disp...	fkidTi...	fkidEs...	fkidCli...	fkidPer...	fkidUsu...	fkidUs...
2019-05-13 19:01:...	2019-05-13 19:04...	2019-05-13 19:10...	Apertura de la...	Se realiza un...	1	0	1	1	3	1011	1011	8	1

Figura 3.33 Registro del evento en la base de datos

3.1.8 SPRINT 6 – CONFIGURACIONES

El *sprint* 6, realiza los CRUD de los parámetros: roles de personas, planes, tipos y ciudades de clientes, tipos de eventos; y las habilitaciones de los eventos, usuarios, clientes y personas.

En la Figura 3.34 se muestra la prueba de habilitación del usuario seleccionado de color naranja, que se produce al presionar el botón `Habilitar`, para verificar la habilitación se muestra un mensaje.

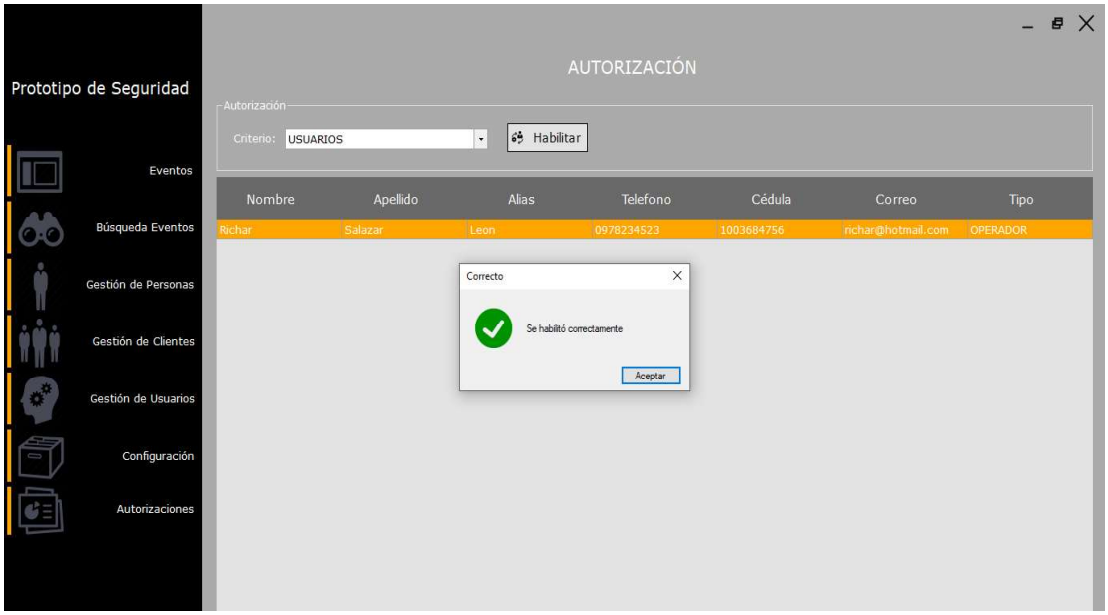


Figura 3.34 Habilitación de un usuario

En la Figura 3.35 se muestra el usuario habilitado de color naranja en la gestión de usuarios, una vez el usuario este habilitado podrá utilizar las funcionalidades de la aplicación.

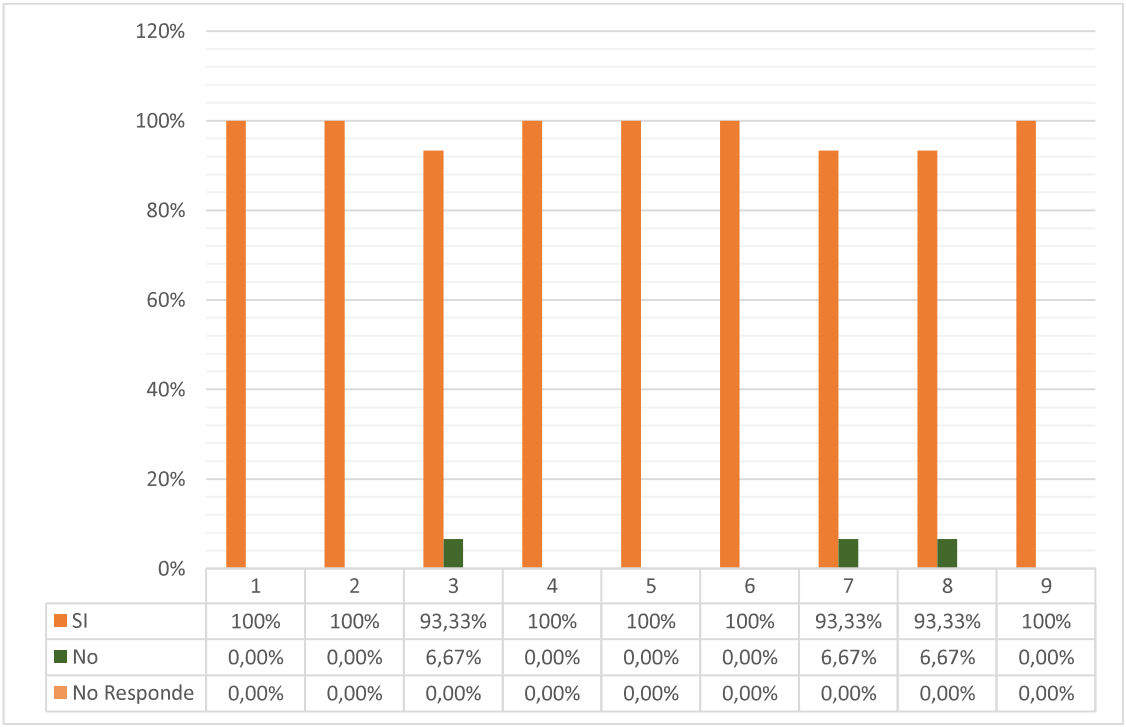
Las habilitaciones de eventos, clientes y personas, siguen el mismo esquema que la habilitación de usuarios.

Figura 3.35 Visualización del usuario habilitado en la gestión de usuarios

3.2. PRUEBAS DE VALIDACIÓN

Para las pruebas de validación del prototipo se desarrolló una encuesta aplicada al personal de la compañía de seguridad SEDYM CIA LTDA, que se encuentra en el Anexo G.

En la Figura 3.36 se observa los resultados de la encuesta, siguiendo un diagrama con los porcentajes según una ponderación: si, no, no responde.



	1	2	3	4	5	6	7	8	9
SI	100%	100%	93,33%	100%	100%	100%	93,33%	93,33%	100%
No	0,00%	0,00%	6,67%	0,00%	0,00%	0,00%	6,67%	6,67%	0,00%
No Responde	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%	0,00%

Figura 3.36 Resultados de las pruebas de validación

La pregunta uno fue ¿Pudo iniciar sesión en las aplicaciones?, para comprobar las autenticación con un resultado de si del 100%.

La pregunta dos fue ¿Pudo ingresar, modificar y eliminar los clientes, personas y usuarios adecuadamente?, para comprobar las gestiones con un resultado de si del 100%.

La pregunta tres fue ¿Pudo ingresar los eventos adecuadamente?, para comprobar exclusivamente el ingreso de eventos con un resultado de si del 93,33% y no del 6,67% que fueron los usuarios con problemas al ingresar un evento.

La pregunta cuatro fue ¿La información de los eventos se actualizaba conforme la información registrada?, para comprobar la actualización de los eventos con un resultado de si del 100%.

La pregunta cinco fue ¿Las búsquedas y exportación a Excel de los eventos se realizan apropiadamente?, con un resultado de si del 100%.

La pregunta seis fue ¿Pudo desplazarse entre las distintas funcionalidades del prototipo?, con un resultado de si del 100%.

La pregunta siete fue ¿La ubicación del dispositivo Android y las ubicaciones de los cliente eran precisas?, con un resultado de si del 93,33% y no del 6,67% que de los encuestados una persona tuvo problemas al visualizar las ubicaciones.

La pregunta ocho fue ¿Tuvo problemas al capturar los códigos QR mediante la cámara? con un resultado de si del 93,33% y no del 6,67% que de los encuestados una persona tuvo problemas escanear un código QR.

La pregunta nueve fue ¿Pudo habilitar los clientes, personas y usuarios? con un resultado de si del 100%.

3.3. CORRECCIÓN DE ERRORES

A lo largo del desarrollo del proyecto se encontraron errores, los cuales se resolvieron en los distintos *sprints*.

En la Tabla 3.1 se encuentran las correcciones del *sprint* 1 - Autenticaciones.

Tabla 3.1 Revisión del *Sprint* 1

Correcciones
Las contraseñas deben tener un símbolo en lugar de los caracteres que se ingresan en la aplicación de escritorio y en la aplicación Android.
Se instaló la versión 6.1.3 de Entity Framework para permitir el acceso a la información de la base de datos.
Se debe cambiar el servicio web de una aplicación de consola a alojarlo en IIS.

En la Tabla 3.2 se encuentran las correcciones del *sprint* 2 - Usuarios.

Tabla 3.2 Revisión del *Sprint* 2

Correcciones
Se deben unificar todos los mensajes de los `MessageBox` con iconos y títulos en la aplicación de escritorio.
No se deben mostrar los identificadores en ninguna interfaz de usuario en la aplicación de escritorio y aplicación Android.
No se deben eliminar usuarios por motivos de auditoria. Se deben poder habilitar o deshabilitar en la aplicación de escritorio.

En la Tabla 3.3 se encuentran las correcciones del *sprint* 3 - Clientes.

Tabla 3.3 Revisión del *Sprint* 3

Correcciones
No se deben eliminar clientes por motivos de auditoria. Se deben poder habilitar o deshabilitar en la aplicación de escritorio.
Poner una descripción de lo que realizan los botones al colocar el mouse sobre estos.

En la Tabla 3.4 se encuentran las correcciones del *sprint* 4 - Personas.

Tabla 3.4 Revisión del *Sprint* 4

Correcciones
No se deben eliminar personas por motivos de auditoria. Se deben poder habilitar o deshabilitar en la aplicación de escritorio.
Se debe ajustar a la izquierda las imágenes de los botones en la aplicación de escritorio.

En la Tabla 3.5 se encuentran las correcciones del *sprint* 5 - Eventos.

Tabla 3.5 Revisión del *Sprint* 5

Correcciones
Se debe cambiar el título del formulario para ver los eventos a “Eventos últimas 24 horas” en la aplicación de escritorio.
Utilizar tres botones para mostrar la información correspondiente de los eventos enviados, en desarrollo y finalizados en la aplicación Android.
Reordenar los controles de las interfaces de los eventos enviados, en desarrollo y finalizados para que sea más legible en la aplicación Android.
Realizar la deserialización de fechas en un formato legible en la aplicación Android.

En la Tabla 3.6 se encuentran las correcciones del *sprint* 6 - Configuraciones.

Tabla 3.6 Revisión del *Sprint* 6

Correcciones
Se debe crear un botón para colocar automáticamente el texto de los códigos QR en la aplicación de escritorio.
Se deben poner con un fondo blanco los `ComboBox` que sean editables en la aplicación de escritorio.
Alinear los controles de las interfaces de usuario en la aplicación de escritorio.
Poner un mismo formato de fechas para los controles `DateTimePicker` en la aplicación de escritorio.

4. CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presentan las conclusiones y recomendaciones obtenidas a lo largo de la realización del prototipo.

4.1. CONCLUSIONES

- Al finalizar el presente Trabajo de Titulación se logró obtener un prototipo que permita la gestión de eventos en el ámbito de la seguridad privada conformado por: una base de datos, un servicio WCF, una aplicación Android y una aplicación de escritorio.

- A través de Entity Framework se logró acceder a la información de la base de datos, y mediante LINQ se pudo consultar la información requerida de la base de datos.

- Para almacenar la contraseña de un usuario se utiliza un algoritmo de hash seguro de tipo SHA-256 que junto con la contraseña genera un valor pseudoaleatorio de 64 dígitos hexadecimales, para que cualquier usuario que acceda a la base de datos, no obtenga la contraseña directamente.

- Los servicios web proveen la funcionalidad de comunicar diferentes aplicaciones independientemente del lenguaje de programación o sistema operativo en el cual se ejecuten, los servicios utilizan un intercambio de mensajes aplicando un formato como JSON. Por tanto, empleando el servicio WCF se permitió el intercambio de información entre la aplicación Android y la aplicación de escritorio.

- Mediante el uso de herramientas como Postman se pudieron generar solicitudes HTTP al servicio WCF. Esto permitió verificar el funcionamiento del servicio, así como, visualizar las solicitudes funcionales y aplicarlas de la misma manera en las aplicaciones.

- Para validar que las solicitudes HTTP al servicio web se realicen por usuarios registrados en la base de datos, se utilizó una funcionalidad que permita comparar el Id y token enviados en la cabecera de las solicitudes, con los valores almacenados en la base de datos.

- La aplicación Android permite al usuario capturar un código QR, además de la posición y a partir de esta información registrar un evento permitiendo conocer si el usuario fue a la propiedad de un cliente. Así mismo, permite al usuario conocer los eventos por estados como: enviado cuando el evento se ingresa por primera vez

desde la aplicación de escritorio, en desarrollo cuando el usuario ha modificado el estado del evento y la fecha y hora de llegada usando la aplicación Android, finalizado cuando el usuario modifica el estado del evento y la información: fecha y hora de salida a través de la aplicación Android.

- Para la aplicación Android se desarrolló una funcionalidad para conseguir la deserialización de fechas y horas de los eventos, ya que utilizando solamente la deserialización la información no era entendible para el usuario.

- En la aplicación Android se utilizaron tareas asíncronas que permitieron la ejecución de solicitudes usando métodos HTTP para realizar: la autenticación, la obtención de los eventos y modificación de la información de los eventos. Logrando así la comunicación entre la aplicación Android y el servicio web WCF.

- La aplicación Android cuenta con la visualización de un mapa que permite al usuario conocer la ubicación de clientes.

- A través de la aplicación Android ya no es necesario ingresar la información de los evento en la hoja de observaciones, sino utilizando las funcionalidades para registrar eventos de la aplicación.

- La aplicación de escritorio permite al usuario crear, modificar, visualizar, deshabilitar y habilitar la información de personas, clientes y usuarios. Además, provee la funcionalidad de realizar búsquedas de los eventos filtrando por: fechas y horas de envío, llegada y salida, si hubo asistencia policial, si la supervisión fue externa, el tipo de evento, el estado del evento, la persona autorizada con la que se tomó contacto, el usuario supervisor que acudió y el usuario operador que creó el evento, de los resultados de las búsquedas es posible exportar a Excel.

- Por motivos de auditoría, en la aplicación Android y la aplicación de escritorio no se permite eliminar la información de eventos, personas, clientes y usuarios.

- Los resultados obtenidos de las pruebas de validación como se puede apreciar en el capítulo tres, denotaron que los usuarios pudieran iniciar sesión en las aplicaciones, ingresar los eventos, realizar búsquedas de los eventos, navegar por las aplicaciones, habilitar clientes, personas y usuarios.

4.2. RECOMENDACIONES

- Se recomienda que se continúe con el desarrollo del prototipo agregando mayores funcionalidades por ejemplo determinando una ruta para dirigirse desde la ubicación del supervisor hacia la propiedad de un cliente.

- Dado que existe un servicio web se podría continuar desarrollando a futuro por ejemplo una aplicación web que permita que los clientes puedan revisar información sobre los eventos suscitados en sus propiedades.

- Se recomienda a futuro agregar al prototipo una funcionalidad para generar registros de supervisión diaria que no son parte de las emergencias, así mismo permitir a los usuarios de la aplicación Android programar las actividades de supervisión diaria

- Para posteriores implementaciones se recomienda utilizar un software para el control de versiones de código como Git que permita manejar los cambios realizados a las aplicaciones.

- Se recomienda utilizar el protocolo HTTPS que aplica el cifrado de datos para realizar las solicitudes al servicio web, debido a que en este Trabajo se utiliza el protocolo HTTP.

- Como una ampliación a las funcionalidades del prototipo se podría realizar un registro de los mantenimientos realizados a los sistemas de alarma.

- A futuro se podría realizar un dashboard con gráficos que permitan identificar cuantos eventos tuvieron asistencia policial, un promedio de tiempos de atención de los eventos desde su envío hasta su finalización.

5. REFERENCIAS BIBLIOGRÁFICAS

[1] O. Cisneros y A. Verónica, “Propuesta para mejorar la comercialización de los servicios de seguridad privada de la empresa Sermanseg Cia. Ltda., ubicada en Guarumos y Seis de Diciembre, sector El Inca, de la ciudad de Quito, Distrito Metropolitano”, 2013.

[2] C. Sánchez García, “Sistema para la geolocalización y recuperación de puntos de interés por medio de terminales Android”, UNIVERSIDAD POLITÉCNICA DE CARTAGENA, 2013.

[3] G. Colouris, J. Dollimore, y T. Kindberg, *Sistemas Distribuidos*, Tercera Edición. PEARSON EDUCACIÓN, S. A., 2001.

[4] *NET application architecture guide*, 2. ed. Redmond, Wash.: Microsoft Press, 2009.

[5] “Leyes del Juego de Rugby: Ley 19: Scrum”. [En línea]. Disponible en: https://laws.worldrugby.org/index.php?law=19.&language=ES. [Accedido: 23-abr-2018].

[6] A. N. Cadavid, J. D. F. Martínez, y J. M. Vélez, “Revisión de metodologías ágiles para el desarrollo de software”, 2013.

[7] A. Pasini, S. Esponda, M. Boracchia, y P. Pesado, “Q-Scrum: una fusión de Scrum y el estándar ISO/IEC PDF”.

[8] A. Orjuela y M. Rojas, “Las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería del Software Educativo”, 2008. [En línea]. Disponible en: http://www.redalyc.org/html/1331/133115027022/. [Accedido: 23-abr-2018].

[9] A. Silberschatz, H. F. Korth, y S. Sudarshan, *Fundamentos de Bases de Datos*, Tercera edición. McGraw-Hill, 1998.

[10] M. Marqués y C. e-libro, *Bases de datos*. Castelló de la Plana: Universitat Jaume I, Servei de Comunicació i Publicacions, 2011.

[11] Microsoft, “Biblioteca de Microsoft SQL Server”. [En línea]. Disponible en: https://msdn.microsoft.com/es-es/library/bb545450.aspx. [Accedido: 24-abr-2018].

[12] J. Gabillaud, *SQL Server 2014 - Administración de una base de datos transaccional con SQL Server Management Studio*. Ediciones ENI, 2015.

[13] J. Ceballos, *VISUAL C#. Interfaces Gráficas y Aplicaciones para Internet con WPF, WCF Y SILVERLIGHT*, Primera edición. RA-MA Editorial, 2013.

[14] "Introducción a los Servicios Web. Invocación de servicios web SOAP." [En línea]. Disponible en: http://www.jtech.ua.es/j2ee/publico/servc-web-2012-13/sesion01-apuntes.html. [Accedido: 25-abr-2018].

[15] P. Castillo y L. Gustavo, "Análisis comparativo de Metro y Axis 2 para el desarrollo de aplicaciones que consuman servicios Web WCF en la ESPOCH", abr. 2014.

[16] C. Pautasso, O. Zimmermann, y F. Leymann, "Restful Web Services vs. 'Big'' Web Services: Making the Right Architectural Decision", en *Proceedings of the 17th International Conference on World Wide Web*, New York, NY, USA, 2008, pp. 805–814.

[17] D. Robledo Fernández, "Desarrollo de aplicaciones para Android II. Ministerio de Educación, Cultura y Deporte SERIE PROGRAMACIÓN COLECCIÓN AULA MENTOR. - PDF". [En línea]. Disponible en: https://docplayer.es/1442475-Desarrollo-de-aplicaciones-para-android-ii-ministerio-de-educacion-cultura-y-deporte-serie-programacion-coleccion-aula-mentor.html. [Accedido: 28-abr-2018].

[18] J. Tomás, *El gran libro de Android*, Sexta Edición. Marcombo, S.A., 2017.

[19] J. A. Tudela, C. C. Vázquez, y E. de, "DESARROLLO DE APLICACIONES PARA DISPOSITIVOS MÓVILES SOBRE LA PLATAFORMA ANDROID DE GOOGLE", p. 199.

[20] "Conoce Android Studio", *Conoce Android Studio*, 2018. [En línea]. Disponible en: https://developer.android.com/studio/intro/?hl=es-419. [Accedido: 29-abr-2018].

[21] J. L. Ordóñez, "Códigos QR", 2012.

[22] "Campaña de turismo de Ecuador con códigos QR: Banana Ambassador", *Marketing Móvil*. [En línea]. Disponible en: http://www.marketing-movil-sms.com/marketing-codigos-qr/campana-de-turismo-de-ecuador-con-codigos-qr-banana-ambassador/. [Accedido: 16-may-2019].

[23] M. Rodero, "¿Qué son los códigos QR?", *inLab FIB*, 08-jun-2012. [En línea]. Disponible en: https://inlab.fib.upc.edu/es/blog/que-son-los-codigos-qr. [Accedido: 21-may-2018].

[24] “LocationListener”, *Android Developers*, 2018. [En línea]. Disponible en: https://developer.android.com/reference/android/location/LocationListener. [Accedido: 23-sep-2018].

[25] G. M. Berríos, A. M. Camacho, y M. J. Castillo, “Desarrollo de aplicaciones de escritorio para los procesos operativos correspondientes al área de estadística y subdirección del Hospital Escuela Oscar Danilo Rosales Argüello (HEODRA).”, 2011.

[26] Microsft Corporation, “Visual Studio Documentation - Visual Studio”. [En línea]. Disponible en: https://docs.microsoft.com/en-us/visualstudio/. [Accedido: 23-oct-2018].

[27] H. Deitel y P. Deitel, *Cómo Programar en C#*, Segunda Edición. 2007.

[28] A. Alvia y L. Eduardo, “Desarrollo de un Sistema Distribuido basado en SOA y MVC para Despliegue de Información Médica de Pacientes”, 2017.

[29] J. Zambrano y C. Andrés, “Desarrollo de un sistema distribuido para el manejo de información de asistencia en tierra a aeronaves para la empresa EMSA CEM”, 2018.

[30] “SQL Server Management Studio (SSMS) - SQL Server”. [En línea]. Disponible en: https://docs.microsoft.com/en-us/sql/ssms/sql-server-management-studio-ssms. [Accedido: 21-may-2019].

[31] J. Huidrobo, “Código QR”, 2009. [En línea]. Disponible en: http://cmapspublic2.ihmc.us/rid=1NS6XZ211-1V8WNZ2-2555/Microcodigos%20qr.pdf. [Accedido: 02-may-2018].

ANEXOS

En esta sección se describe la información de los anexos del Proyecto Integrador.

Los siguientes anexos utilizados son:

ANEXO A. Ficha de Entrevista

ANEXO B. Historias de Usuario

ANEXO C. *Script* de la base de datos

ANEXO D. Código de la aplicación de escritorio

ANEXO E. Código de la aplicación Android

ANEXO F. Código del servicio web

ANEXO G. Ficha de Encuesta de Validación

ANEXO H. Manual de usuario de la aplicación de escritorio

ANEXO I. Manual de usuario de la aplicación Android

Los anexos C, D, E, F, H y I se encuentran en el CD adjunto.

ANEXO A

Ficha de Entrevista

FICHA DE ENTREVISTA La información proporcionada será estrictamente confidencial y será utilizado con fines educativos. La información se usará para el proyecto “Desarrollo De Un Prototipo De Gestión De Eventos Basado En Una Arquitectura Orientada A Servicios Para Seguridad Privada”	
ELABORADOR POR:	Fernando Daniel Cadena Villarreal
INSTITUCIÓN:	Escuela Politécnica Nacional
FECHA:	
PERSONA ENTREVISTADA:	
Número de Pregunta	**PREGUNTA**
1	¿Qué procesos existen actualmente y cómo se actúa frente a ellos en el ámbito de seguridad privada en su compañía?
2	¿Cuáles son los involucrados en los procesos en el ámbito de seguridad privada en su compañía?
3	¿Qué información almacena actualmente de los procesos en el ámbito de seguridad privada?
4	¿Los procesos, involucrados, procedimiento e información que se encuentra en sus registros cambia constantemente o se mantiene con el tiempo?
Se plantea el desarrollo de un sistema distribuido que automatice los eventos y procesos de su compañía, el cual utilizará una base de datos, un servicio web, una aplicación Android y una aplicación de escritorio, conforme a esto se pone a consideración lo siguiente.	
5	¿Considera usted que sería útil automatizar la información de los eventos de la seguridad privada en un sistema?

6	¿Cree usted importante la autenticación de personas para ingresar al sistema?
7	¿Existe información adicional de los eventos de seguridad privada que crea relevante almacenar en el sistema?
8	¿Si se definiera una aplicación que empleará el escaneo de códigos QR para manejar el registro de las incidencias en seguridad privada, le parece de utilidad?
9	¿Si se definiera una aplicación que empleará la localización del dispositivo móvil para realizar un registro de datos comprobable, cree usted le sería de utilidad?
10	¿Le parecería importante generar reportes de los eventos de acuerdo con los clientes registrados, personal a cargo y fechas de incidencias?

ANEXO B

Historias de Usuario para la Aplicación de Escritorio

Código: HU-01	**Actor:** Administrador, Operador
Desarrollador: Fernando Cadena	**Sprint:** Usuarios
Nombre: Autenticación aplicación de escritorio	**Número de Sprint:** 3
Descripción: Como administrador u operador quiero colocar mi alias y contraseña para poder ingresar al sistema.	
Comentarios: Se requiere modificar mi contraseña cuando inicie sesión por primera vez o cuando un administrador restablezca mi contraseña.	

Código: HU-02	**Actor:** Administrador
Desarrollador: Fernando Cadena	**Sprint:** Usuarios
Nombre: Gestión de Cuentas de Usuarios	**Número de Sprint:** 3
Descripción: Como administrador quiero poder gestionar la información de los usuarios como: nombre, apellido, alias, teléfono, cédula, correo y tipo de usuario, para establecer los usuarios que ingresarán al sistema. Además, se requiere que las contraseñas de los usuarios se puedan ingresar y modificar, más no visualizar.	
Comentarios: Se requieren tres tipos de usuarios: administrador, operador y supervisor. Se requiere que al eliminar usuarios la información no se pierda, sino se oculte del sistema. Se requiere exista seguridad en el almacenamiento de las contraseñas. Se requiere exista cierto tipo de seguridad en el retorno de información.	

Código:	**Actor:**
HU-03	Administrador, Operador
Desarrollador:	**Sprint:**
Fernando Cadena	Clientes
Nombre:	**Número de Sprint:**
Gestión de Cuentas de Clientes	4
Descripción:	
Como operador quiero poder ingresar y visualizar la información de clientes como: número de cuenta, nombre, dirección, ciudad, tipo, plan, teléfono principal, teléfono secundario, código QR interno, código QR externo, latitud y longitud, para poder crear eventos de estos clientes. Como administrador además de ingresar y visualizar, quiero poder modificar y eliminar clientes.	
Comentarios:	
Se requiere que el número de cuenta sea único para cada cliente. Se requiere que al eliminar clientes la información no se pierda, sino se oculte del sistema. Se requiere realizar búsquedas de los clientes según: nombre del cliente, número de cuenta y ciudad. Se requiere realizar búsquedas de direcciones de nuevos clientes en un mapa.	

Código:	**Actor:**
HU-04	Administrador, Operador
Desarrollador:	**Sprint:**
Fernando Cadena	Clientes
Nombre:	**Número de Sprint:**
Gestión de Zonas de Clientes	4
Descripción:	
Como administrador u operador quiero poder gestionar la información de las zonas de cada cliente como: número, descripción y partición, para establecer las zonas activadas de un evento.	
Comentarios:	
Se requiere que las zonas ya usadas en eventos de los sistemas de alarma, no se puedan eliminar.	

Código: HU-05	**Actor:** Administrador, Operador
Desarrollador: Fernando Cadena	**Sprint:** Personas
Nombre: Gestión de Cuentas de Personas	**Número de Sprint:** 5
Descripción: Como operador quiero poder ingresar, modificar y visualizar la información de las personas como: nombre, apellido, teléfono principal, teléfono secundario y cargo de la persona, para establecer las personas de contacto. Como administrador además de poder realizar las funciones del operador, quiero poder eliminar personas.	
Comentarios: Se requiere realizar búsquedas de personas según: el nombre de la persona y el número de cuenta del cliente, para obtener las personas responsables de un determinado cliente. Se requiere que al eliminar personas la información no se pierda, sino se oculte del sistema.	

Código: HU-06	**Actor:** Administrador, Operador
Desarrollador: Fernando Cadena	**Sprint:** Personas
Nombre: Gestión de Responsabilidad de las Personas	**Número de Sprint:** 5
Descripción: Como administrador u operador quiero poder gestionar las personas responsables de los clientes para poder establecer en los eventos las personas de contacto.	
Comentarios: Ninguno.	

Código: HU-07	**Actor:** Administrador, Operador
Desarrollador: Fernando Cadena	**Sprint:** Eventos
Nombre: Gestión de Eventos	**Número de Sprint:** 6
Descripción: Como operador quiero poder ingresar y visualizar la información de los eventos como: fecha y hora de envío, tipo del evento, estado del evento, cliente, supervisor, operador y zonas activadas. Además, quiero visualizar: fecha y hora de llegada, fecha y hora de salida, causa, observación, asistencia policial, supervisión y persona responsable, para mantener en un sistema la información de los eventos. Como administrador además de poder realizar las funciones del operador, quiero poder eliminar los eventos.	
Comentarios: Se requiere que la información: causa, observación, asistencia policial, supervisión, tipo del evento, estado del evento, persona de contacto, supervisor y operador se pueda ordenar alfabéticamente. Se requiere que al eliminar eventos la información no se pierda, sino se oculte del sistema. Se requiere que la actualización de la información este acorde a lo que los usuarios ingresen. Se requiere contar con mensajes informativos al gestionar la información. Se requiere contar con un documento como guía de las funcionalidades.	

Código: HU-08	**Actor:** Administrador, Operador
Desarrollador: Fernando Cadena	**Sprint:** Eventos
Nombre: Búsqueda de Eventos y Exportación a Excel	**Número de Sprint:** 6
Descripción: Como administrador u operador quiero poder buscar los eventos por: fecha inicio y fin de envío, llegada y salida, asistencia policial, supervisión, tipo del evento, estado del evento, persona de contacto, supervisor y operador, para obtener reportes y además exportarlos a un archivo en Excel.	

Comentarios:
Se requiere que los campos asistencia policial, supervisión, tipo del evento, estado del evento, persona de contacto, supervisor y operador tengan información precargada.

Código: HU-09	**Actor:** Administrador
Desarrollador: Fernando Cadena	**Sprint:** Configuraciones
Nombre: Configuración de Parámetros	**Número de Sprint:** 7
Descripción: Como administrador quiero poder ingresar y eliminar los parámetros como: roles de las personas; planes, tipos y ciudades de los clientes; y tipos de eventos, para contar con otros parámetros.	
Comentarios: Se requiere que los parámetros ya usados en los eventos de los sistemas de alarma, no se puedan eliminar.	

Código: HU-10	**Actor:** Administrador
Desarrollador: Fernando Cadena	**Sprint:** Configuraciones
Nombre: Autorizaciones	**Número de Sprint:** 7
Descripción: Como administrador quiero poder habilitar los eventos, usuarios, clientes y personas para volver a visualizarlos en el sistema.	
Comentarios: Ninguno.	

- Historias de Usuario para la Aplicación Móvil

Código: HU-11	**Actor:** Administrador, Supervisor
Desarrollador: Fernando Cadena	**Sprint:** Usuarios
Nombre: Autenticación aplicación Android	**Número de Sprint:** 3
Descripción: Como administrador o supervisor quiero colocar mi alias y contraseña para poder ingresar al sistema.	
Comentarios: Se requiere modificar mi contraseña cuando inicie sesión por primera vez o cuando un administrador restablezca mi contraseña.	

Código: HU-12	**Actor:** Supervisor
Desarrollador: Fernando Cadena	**Sprint:** Eventos
Nombre: Registro y Listado de Eventos	**Número de Sprint:** 6
Descripción: Como supervisor quiero registrar la información de los eventos como: fecha de llegada, fecha de salida, causa del evento, observaciones, asistencia policial, supervisión, estado del evento y la persona de contacto, también quiero visualizar la información como: fecha y hora de envío, tipo del evento, cliente, zonas activadas para poder atender los eventos.	
Comentarios: Se requiere que para el registro de los eventos, se utilicen mecanismos de localización y escaneo de códigos para verificar si la supervisión se realizó externa o interna a la propiedad del cliente. Se requiere que los eventos se muestren conforme a los estados: enviado, en desarrollo y finalizado.	

Código: HU-13	**Actor:** Supervisor
Desarrollador: Fernando Cadena	**Sprint:** Eventos
Nombre: Códigos QR	**Número de Sprint:** 6
Descripción: Como administrador quiero que el supervisor utilice un mecanismo de escaneo de códigos para realizar el registro de un evento.	
Comentarios: Se requiere generar dos códigos por cada cliente, el primero se ubicará en la parte exterior de la propiedad del cliente y el segundo se ubicará al interior de la propiedad. Se deben utilizar los mecanismos de códigos y localización en conjunto para realizar el registro de eventos.	

Código: HU-14	**Actor:** Supervisor
Desarrollador: Fernando Cadena	**Sprint:** Eventos
Nombre: Localización	**Número de Sprint:** 6
Descripción: Como administrador quiero que el supervisor utilice un mecanismo de ubicación para realizar el registro de un evento.	
Comentarios: Se deben utilizar los mecanismos de códigos y localización en conjunto para realizar el registro de eventos.	

Código:	**Actor:**
HU-15	Administrador, Supervisor
Desarrollador:	**Sprint:**
Fernando Cadena	Clientes
Nombre:	**Número de Sprint:**
Visulizar Clientes	4
Descripción:	
Como administrador o supervisor quiero observar las ubicaciones de los clientes en un mapa para distinguir las propiedades de los clientes.	
Comentarios:	
Se requiere observar en el mapa: el nombre del cliente y una marca o símbolo.	

ANEXO C

Script de la base de datos

Por su extensión este ANEXO se muestra en el CD adjunto.

ANEXO D

Código de la aplicación de escritorio

Por su extensión este ANEXO se muestra en el CD adjunto.

ANEXO E

Código de la aplicación Android

Por su extensión este ANEXO se muestra en el CD adjunto.

ANEXO F

Código del servicio web

Por su extensión este ANEXO se muestra en el CD adjunto.

ANEXO G

Ficha de encuesta de validación

FICHA DE ENCUESTA DE VALIDACIÓN				
La información proporcionada será estrictamente confidencial y será utilizado con fines educativos. La información se usará para el proyecto "Desarrollo De Un Prototipo De Gestión De Eventos Basado En Una Arquitectura Orientada A Servicios Para Seguridad Privada"				
ELABORADOR POR:	Fernando Daniel Cadena Villarreal			
INSTITUCIÓN:	Escuela Politécnica Nacional			
FECHA:				
PERSONA ENTREVISTADA:				
Número de Pregunta	**PREGUNTA**	**Si**	**No**	**No Responde**
1	¿Pudo iniciar sesión en las aplicaciones?			
2	¿Pudo ingresar, modificar y eliminar los clientes, personas y usuarios adecuadamente?			
3	¿Pudo ingresar los eventos adecuadamente?			
4	¿La información de los eventos se actualizaba conforme la información registrada?			
5	¿Las búsquedas y exportación a Excel de los eventos se realizan apropiadamente?			
6	¿Pudo desplazarse entre las distintas funcionalidades del prototipo?			
7	¿La ubicación del dispositivo Android y las ubicaciones de los cliente eran precisas?			

8	¿Tuvo problemas al capturar los códigos QR mediante la cámara?			
9	¿Pudo habilitar los clientes, personas y usuarios?			

ANEXO H

Manual de usuario de la aplicación de escritorio

Por su extensión este ANEXO se muestra en el CD adjunto.

ANEXO I

Manual de usuario de la aplicación Android

Por su extensión este ANEXO se muestra en el CD adjunto.

ORDEN DE EMPASTADO