ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO DE UNA APLICACIÓN MULTIPLATAFORMA Y DISTRIBUIDA PARA DISPOSITIVOS MÓVILES QUE PERMITA AUTOMATIZAR EL MANEJO DE INFORMACIÓN DEL TRANSPORTE INTERPROVINCIAL DE LA CIUDAD DE QUITO.

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

EMERSON DAMIR CADENA NAVARRETE
emersoncadena@outlook.com

DIRECTOR: ING. MÓNICA VINUEZA RHOR
monica.vinueza@epn.edu.ec

Quito, Julio 2015

DECLARACIÓN

Yo, Emerson Damir Cadena Navarrete, 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 cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

Emerson Damir Cadena Navarrete

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Emerson Damir Cadena Navarrete, bajo mi supervisión.

_____

Ing. Mónica Vinueza Rhor
DIRECTOR DEL PROYECTO

AGRADECIMIENTOS

A Dios, el principio fundamental de mi vida y que me ayuda a ser mejor cada día.

A mi familia, que siempre me han apoyado y han sido el soporte durante mis estudios.

A la directora del presente proyecto, Ing. Mónica Vinueza, por su apoyo y enseñanzas en el desarrollo de este proyecto.

Al personal de la EPMMOP, que colaboró de forma técnica y humana en la realización del proyecto. Sus recomendaciones y opiniones fueron muy importantes en el desarrollo del proyecto.

A mis amigos, que me apoyaron en la realización de esta tesis y en mis estudios durante mi etapa universitaria.

DEDICATORIA

Dedico todo el esfuerzo, la energía y el aprendizaje de la realización de este proyecto a Dios.

CONTENIDO

ÍNDICE DE FIGURAS

CAPÍTULO I

CAPÍTULO II

CAPÍTULO III

ÍNDICE DE TABLAS

CAPÍTULO II

CAPÍTULO III

ÍNDICE DE FRAGMENTOS DE CÓDIGO

CAPÍTULO III

RESUMEN

El presente proyecto consiste en el desarrollo de un prototipo que permita manejar de forma adecuada la información del transporte interprovincial de la ciudad de Quito. Con este desarrollo se propone una solución que facilite a los usuarios obtener información del servicio que prestan los terminales de Quito.

El proyecto se basa en el desarrollo de una aplicación multiplataforma que funcione en distintas versiones de los sistemas operativos *Android* y *Windows Phone*. Se utiliza el lenguaje de programación C# para el desarrollo del cliente y del servidor y una base de datos como repositorio de la información. La parte principal de la aplicación puede utilizarse para aplicaciones en otras plataformas.

En el primer capítulo se detalla los aspectos teóricos que permiten cumplir con el proyecto. Se describe los conceptos para el diseño y el desarrollo de software conjuntamente con las principales características de las herramientas que se utiliza y se incluye conceptos generales del tema.

En el segundo capítulo se diseña la aplicación partiendo de la definición de los requerimientos del sistema. Mediante el uso del lenguaje UML se definen los casos de uso y el diagrama de clases. Posteriormente se define toda la arquitectura del sistema tanto en el cliente como en el servidor detallando la estructura del software y las herramientas a utilizar. Finalmente, se definen los elementos y principales características de diseño para las interfaces de usuario.

En el tercer capítulo se desarrolla la aplicación tomando en cuenta el diseño y funcionamiento que se ha establecido. Se realizan las pruebas de la aplicación sobre las diferentes plataformas y se describe la implementación en cada plataforma así como también de la aplicación del servidor. Se incluyen los costos relacionados al desarrollo del proyecto.

En el cuarto capítulo se presentan las conclusiones y las recomendaciones que son resultado de la investigación y el desarrollo realizados.

PRESENTACIÓN

El progreso tecnológico en los últimos años facilita a la sociedad publicar y acceder a la información de un determinado producto o servicio. La gran variedad de dispositivos como ordenadores, tabletas o teléfonos inteligentes junto con la masificación del Internet plantean nuevos retos en el desarrollo de aplicaciones que aprovechen las capacidades tecnológicas actuales para satisfacer las necesidades de los consumidores.

Utilizando métodos y técnicas adecuadas en el desarrollo de software se puede crear aplicaciones que además de cumplir con su funcionalidad, puedan ser utilizadas por distintas plataformas y dispositivos de manera sencilla. De esta forma, el usuario final no está limitado a un dispositivo específico para hacer uso de la aplicación.

Por tanto, la tendencia actual es que la información, los productos y servicios sean más accesibles al público en general. Es así que se ha visto la necesidad de proponer una solución que facilite a los usuarios del transporte interprovincial de Quito obtener información del servicio de los terminales de Quitumbe y Carcelén.

Para solucionar este problema se plantea el desarrollo de una aplicación que funcione sobre las plataformas *Windows Phone* y *Android* de distintos teléfonos inteligentes que maneje adecuadamente la información del sistema de transporte interprovincial de la ciudad de Quito. La aplicación usa el lenguaje de programación C# y una base de datos para almacenar la información.

La aplicación se denomina STQ-EC acrónimo de Sistema de Terminales Interprovinciales de Quito junto con las iniciales del autor del proyecto.

Este proyecto constituye un ejemplo del desarrollo multiplataforma en dispositivos móviles actuales orientado a resolver un problema específico de la ciudad de Quito, siguiendo la tendencia de nuevas tecnologías e investigando nuevas funcionalidades de sistemas actuales.

CAPÍTULO I

1 FUNDAMENTOS TEÓRICOS

En el primer capítulo se detalla los aspectos teóricos que permiten cumplir con el proyecto. Se describe los conceptos para el diseño y el desarrollo de software, además de las principales características de las herramientas que se utiliza y se incluye conceptos generales del tema.

1.1 METODOLOGÍA DE DESARROLLO [L1]

En el desarrollo de un proyecto de software existen varias actividades que ayudan a cumplir de forma organizada la creación de un producto de software. Existen diferentes procesos para el desarrollo, pero es común a todos ellos las siguientes cuatro actividades: especificación, desarrollo, validación y evolución.

En la etapa de especificación del software, se define la funcionalidad del sistema a través de la comprensión, obtención y definición de los requerimientos. Además, se debe identificar los límites del funcionamiento y desarrollo del sistema.

En la etapa de desarrollo del software, se diseña e implementa un sistema de acuerdo la especificación. Se realizan los procesos de diseño y la programación del software.

En la etapa de validación del software, se asegura que el sistema cumpla con su especificación y las necesidades para las que fue construido. Se realiza diferentes pruebas en distintos niveles que verifiquen y validen el sistema.

En la etapa de evolución del software, el sistema evoluciona para adecuarse a las necesidades o cambios que requiera el cliente.

Las actividades comunes en el desarrollo de software se aplican de distinta

manera, en algunos casos puede ser de manera secuencial, en otros de manera entrelazada, la forma de organizar las actividades depende del tipo de proyecto, el enfoque de desarrollo utilizado o el personal involucrado.

1.2 PROGRAMACIÓN ORIENTADA A OBJETOS (OOP, *OBJECT-ORIENTED PROGRAMMING*)

Es un enfoque de desarrollo de software en el que la estructura de software está basada en objetos que interactúan entre sí para cumplir una tarea determinada. Está conformado por un conjunto de técnicas usadas para desarrollar y mantener aplicaciones complejas.

1.2.1 CARACTERÍSTICAS OOP [L2] [L3]

Existen varios elementos y características fundamentales de la programación orientada a objetos, los cuales son: Objeto, Clase, Abstracción, Encapsulación, Polimorfismo y Herencia.

1.2.1.1 Objeto

Es una estructura de software que incorpora datos y métodos para trabajar con los datos. En un objeto los datos representan el estado y los métodos el comportamiento.

1.2.1.2 Clase

Es una abstracción que describe características comunes de un grupo similar de objetos. Una clase es una plantilla que define la estructura y los métodos para los objetos que se basan en ella.

1.2.1.3 Abstracción

Es el proceso que permite identificar las características importantes de un

escenario o de un objeto para su representación, descartando los detalles irrelevantes.

1.2.1.4 Encapsulación

Es el mecanismo que agrupa el estado y el comportamiento de un objeto de forma lógica. El acceso a los datos está oculto, se tiene que interactuar con el objeto que contiene los datos para acceder a la información.

1.2.1.5 Polimorfismo

Es la capacidad de dos objetos diferentes de responder a la misma petición de un método de forma única según el objeto. También existe la posibilidad de que un objeto responda de diferente manera al mismo método dependiendo del tipo de información que recibe.

1.2.1.6 Herencia

Es la característica que permite que de una clase base o clase padre se pueda crear una clase derivada o clase hija. La clase derivada además de tener las características de la clase base podrá definir su propia funcionalidad. Esto permite agrupar las clases de forma jerárquica.

1.3 PROCESO UNIFICADO DE DESARROLLO [L1] [L4] [W1]

El Proceso Unificado de Desarrollo (RUP) es un proceso de desarrollo de software basado en una arquitectura centralizada y casos de uso, además es iterativo e incremental. RUP agrupa un conjunto de metodologías que se adaptan a distintos proyectos dependiendo de la necesidad y el sistema a desarrollar.

En RUP se definen cuatro fases que seguirá el proceso de desarrollo:

- **Inicio:** la fase de inicio define el alcance y objetivos del sistema, se identifican las entidades y como estas interactúan con el sistema. Además se planifica la forma como se desarrolla el proyecto.
- **Elaboración:** la fase de elaboración permite la comprensión clara del problema y el desarrollo del plan del proyecto conjuntamente con los riesgos que puedan presentarse. Esta fase permite precisar aspectos del modelado del negocio así como los requerimientos y define el comportamiento del sistema de forma estática y dinámica a través del UML.
- **Construcción:** la fase de construcción implica principalmente el diseño, desarrollo e implementación del sistema de acuerdo a las especificaciones. Durante esta fase se realizan las pruebas correspondientes que verifiquen el funcionamiento del sistema.
- **Transición:** la fase de transición corresponde al paso de un ambiente de desarrollo a un ambiente de producción para que el producto sea utilizado por el usuario final. Esta fase también implica las pruebas con los usuarios y el mantenimiento del sistema hasta su correcto funcionamiento.

RUP también define flujos de trabajos, que son actividades que ocurren en varias fases durante el proceso de desarrollo. Existen seis flujos de trabajo del proceso y tres flujos de trabajo de soporte:

- **Modelado del negocio:** describe los procesos del negocio para conocer las necesidades que debe satisfacer el sistema.
- **Requerimientos:** describe la interacción del sistema con los actores, los requerimientos se definen a través de los casos de uso.
- **Análisis y Diseño:** indica como el sistema se implementa en base a los requerimientos y restricciones; varios modelos definen el comportamiento del sistema.
- **Implementación:** define la estructura del sistema, la programación de los componentes y el funcionamiento en conjunto de los elementos.
- **Pruebas:** permite probar el sistema a nivel de componentes individuales y en conjunto para encontrar defectos y corregirlos.

- **Despliegue:** se crea la versión del producto para que sea instalada y usada por los usuarios.
- **Configuración y cambios de gestión:** gestiona los cambios de los elementos del proyecto a lo largo de la duración del mismo.
- **Gestión del proyecto:** gestiona todas las actividades de forma adecuada para cumplir con el proyecto y satisfacer las necesidades de clientes y usuarios.
- **Entorno:** permite el soporte de las herramientas y procesos adecuados para los equipos de desarrollo en las distintas fases del proyecto.

RUP también define seis buenas prácticas para el desarrollo de sistemas:

- Desarrollar el software de forma iterativa.
- Gestionar los requerimientos.
- Utilizar arquitecturas basadas en componentes.
- Modelar visualmente el software.
- Verificar la calidad del software.
- Controlar los cambios en el software.

1.4 REQUERIMIENTOS DEL SISTEMA [L2]

Los requerimientos son la descripción de los servicios, las características y restricciones que debe cumplir un sistema. Existen básicamente dos grupos de requerimientos: requerimientos funcionales y requerimientos técnicos.

Los requerimientos funcionales describen la forma de operación, la funcionalidad y restricciones del sistema desde el punto de vista de una entidad que usa el sistema.

Los requerimientos técnicos describen la estructura interna del sistema, definen con claridad las funciones, servicios y restricciones desde el punto de vista técnico para cumplir los requerimientos funcionales.

1.5 LENGUAJE DE MODELADO UNIFICADO (UML, *UNIFIED MODELING LANGUAGE*) [L2] [L5]

Es un lenguaje estándar que permite modelar el diseño de software orientado a objetos. Se basa en una serie de modelos visuales y gráficos para representar el alcance del sistema, los componentes del sistema, la interacción entre los componentes del sistema y la interacción de los usuarios con el sistema.

Existen gran cantidad de diagramas para representar las distintas perspectivas que puede tener el sistema. Uno de los diagramas más utilizados son los casos de uso, que se modelan a través de los requerimientos funcionales del sistema.

Para modelar el sistema también se toma en cuenta su contexto estático y su contexto dinámico para identificar los componentes y su función por separado.

En el contexto estático identifica la estructura y organización de los elementos que permiten construir el sistema, se describe en especial a través del modelo de diagrama de clases. En el contexto dinámico identifica el comportamiento de los elementos del sistema y la interacción entre ellos, se describe principalmente por medio de escenarios, diagramas de secuencia, diagramas de actividad y diagramas de colaboración.

1.5.1 DIAGRAMAS DE CASOS DE USO [W2] [L5]

Un diagrama de caso de uso es una descripción gráfica o textual de las funciones del sistema desde la perspectiva de los usuarios. Los casos de uso ayudan a definir el alcance y límites del sistema desde el punto de vista funcional.

1.5.1.1 Actor

Representa una entidad externa que interactúa con el sistema, las entidades puede ser usuarios humanos u otros sistemas. Existen diferentes actores dependiendo de su rol en el sistema.

1.5.1.2 Caso de Uso

Representa la interacción entre el actor y el sistema. Las interacciones que se definen a través de los requerimientos funcionales del sistema se modelan como casos de uso.

1.5.1.3 Relaciones [W3]

Entre diferentes casos de uso pueden existir ciertas relaciones: inclusión, extensión y generalización. En la inclusión, un caso de uso base incorpora explícitamente el comportamiento de otro caso de uso; en la extensión un caso de uso base incorpora implícitamente el comportamiento de otro caso de uso; y en la generalización un caso de uso hijo hereda el comportamiento del padre.

1.5.2 DIAGRAMAS DE CLASES [L5]

Es un diagrama que describe la estructura del sistema a través de clases con sus atributos, métodos y las relaciones que existen entre las clases.

Una clase se representa a través de un rectángulo dividido en varias secciones que contienen su estructura. La sección superior tiene el nombre de la clase, la sección intermedia lista los atributos o propiedades de la clase y la sección inferior las operaciones o métodos de la clase.

1.5.2.1 Relaciones entre clases

Una relación existe cuando los objetos interactúan entre sí para realizar una tarea, existen diferentes tipos de relaciones que se modelan en el diagrama de clases, las más comunes son:

1.5.2.1.1 Asociación

Una asociación existe cuando una clase usa otra clase o hace referencia a otra

clase. Se representa a través de una línea entre las clases con el nombre de la asociación sobre la línea.

1.5.2.1.2 Multiplicidad

La instancia de una clase puede asociarse con múltiples instancias de otra clase, la multiplicidad indica la cantidad de objetos de una clase que se relacionan con los objetos de otra clase en la asociación. Se representa a través de un número junto a la clase indicando su multiplicidad.

1.5.2.1.3 Herencia

Representa una clase hija que hereda los atributos y operaciones de la clase padre, la clase hija puede definir sus propios elementos.

1.5.2.1.4 Agregación

Representa una clase que está formada por otras clases. Las clases se consideran componentes o partes de la clase agregada.

1.5.3 ESCENARIOS [L5]

Un escenario es una descripción textual del procesamiento interno que debe ocurrir, desde el inicio al final, para que el sistema lleve a cabo una funcionalidad. Los escenarios definen los pasos necesarios para cumplir los casos de uso.

1.5.4 DIAGRAMAS DE INTERACCIÓN [W2] [L5]

Permiten modelar los elementos que están involucrados en realizar una determinada funcionalidad. Existen dos tipos de diagramas de interacción: los diagramas de secuencia y los diagramas de colaboración.

Los diagramas de colaboración y secuencia son similares ya que presentan en

general la misma información de diferente manera.

Un diagrama de secuencia muestra la interacción de un objeto con otro, mediante mensajes, a través el tiempo en el sistema.

Un diagrama de colaboración muestra la organización de los objetos en el sistema y su comunicación a través del flujo de mensajes.

1.5.5 DIAGRAMAS DE ACTIVIDAD [W2] [L5]

Indican el flujo de actividades realizadas por el sistema en una operación o proceso. Los diagramas de actividad pueden tener distintos niveles de profundidad dependiendo del detalle del proceso del sistema que se requiera representar. Proporcionan una clara idea del funcionamiento del sistema cuando es ejecutado.

1.6 CONCEPTOS DE BASES DE DATOS [L6]

1.6.1 BASE DE DATOS RELACIONAL

Es una colección de datos formada por tablas y columnas que se relacionan entre sí. Las relaciones se basan en datos clave que se almacenan en una columna. Los datos son procesados para obtener información.

1.6.2 SISTEMAS DE GESTIÓN DE BASES DE DATOS (DBMS, *DATABASE MANAGEMENT SYSTEM*)

Es un sistema que almacena datos relacionados entre sí y un conjunto de programas que permiten acceder y manipular los datos. Un DBMS puede contener varias bases de datos. Para manejar la información, un DBMS procesa las solicitudes de los usuarios, recupera los datos y devuelve la respuesta al usuario.

1.6.3 MODELO ENTIDAD-RELACIÓN [W4] [W5]

Es un modelo de datos que permite representar una situación del mundo real a través de entidades y las relaciones entre las entidades.

Una entidad es un objeto del mundo real que puede ser identificable, del cual se necesita almacenar información. Una entidad se representa a través de atributos. Los atributos son las propiedades que describen al objeto.

En el modelo Entidad-Relación una relación es una asociación entre entidades. Pueden existir distintos tipos de relaciones dependiendo del número de entidades que se asocian a otra entidad:

- **Uno a Uno:** una entidad de A se asocia como máximo con una entidad de B, y una entidad de B se asocia como máximo con una entidad de A.
- **Uno a Varios:** una entidad de A se asocia con cualquier número de entidades de B, pero una entidad de B se puede asociar como máximo con una entidad de A.
- **Varios a Varios:** una entidad de A se asocia con cualquier número de entidades de B, y una entidad de B se asocia con cualquier número de entidades de A.

1.6.3.1 Diagrama Entidad-Relación

Es una representación gráfica que permite modelar la estructura de la información que se almacenan en un sistema. Utiliza como elementos entidades y las relaciones existentes entre ellas para representar la información.

1.7 CONCEPTOS GENERALES

1.7.1 RED DE COMPUTADORES

Conjunto de computadores autónomos interconectados entre sí a través de una

subred de comunicaciones.

1.7.2 SISTEMA OPERATIVO [W6]

Es un programa que controla y proporciona servicios a los programas de aplicación, además maneja el hardware del computador.

1.7.3 SERVIDOR WEB [W7]

Es un programa que almacena, procesa y entrega contenido web a los clientes. La comunicación entre el cliente y servidor se realiza mediante el protocolo HTTP (*Hypertext Transfer Protocol*). Entre los servidores más destacados están IIS (*Internet Information Services*) y Apache.

1.7.4 ENTORNO DE DESARROLLO INTEGRADO (IDE, *INTEGRATED DEVELOPMENT ENVIRONMENT*) [W8]

Es un programa compuesto por un conjunto de herramientas para facilitar el desarrollo de aplicaciones de software. Está enfocado a uno o varios lenguajes de programación. Generalmente está compuesto de un editor de código, herramientas de compilación y depuración, herramientas para la creación de interfaces gráficas de usuario, como sus principales elementos.

1.7.5 ARQUITECTURA DE REFERENCIA TCP/IP (TCP/IP, *TRANSMISSION CONTROL PROTOCOL/INTERNET PROTOCOL*) [L7]

Es una arquitectura basada en cuatro niveles que permite conectar múltiples redes. Las capas de TCP/IP son:

- **Capa *Host* – Red:** controla los dispositivos de hardware y los medios de transmisión de la red. Permite que un *host* se conecte a la red mediante un protocolo que le permita enviar paquetes a través de ella. El protocolo no está definido.

- **Capa *Internet*:** permite a los *host* transferir paquetes a la red y que viajen independientemente a su destino. Define el formato del paquete y un protocolo llamado IP. Además, permite determinar la mejor ruta para los paquetes a través de la red.
- **Capa Transporte:** permite la comunicación entre entidades pares en los nodos de origen y destino. Se definen dos protocolos: TCP y UDP (*User Datagram Protocol*).
- **Capa Aplicación:** contiene los protocolos de aplicación que puede utilizar el usuario.

Los principales protocolos de TCP/IP son:

- **Protocolo IP:** es un protocolo no confiable, no orientado a conexión y que corresponde a la capa Internet; se encarga de definir el formato del datagrama IP y ofrecer el servicio de ruteo de datagramas a través de varias redes.
- **Protocolo TCP:** es un protocolo de la capa Transporte orientado a conexión que permite que los datos entre el origen y destino lleguen sin errores y en orden correcto.

1.7.6 PROTOCOLO HTTP (*HYPERTEXT TRANSFER PROTOCOL*)

Es un protocolo de capa aplicación, el más común y utilizado por la web para el intercambio de información entre un cliente y un servidor. Permite transferir distinto tipo de información a través del Internet.

1.8 .NET *FRAMEWORK* [L5] [L8]

Es una plataforma de software desarrollada por Microsoft, introducida en el año 2002, para construir y ejecutar aplicaciones. Las aplicaciones o sistemas desarrollados en esta plataforma están principalmente orientados a sistemas operativos de la familia Windows, pero también se puede construir aplicaciones para sistemas Mac OS X y varias distribuciones Unix/Linux.

El .NET *Framework* está compuesto de los siguientes componentes:

- ***Common Language Runtime* (CLR),** es el componente fundamental de .NET *Framework*, proporciona un entorno de ejecución para el código de las aplicaciones y brinda una capa de abstracción entre el código y el sistema operativo.
- ***Common Type System* (CTS),** es una especificación que describe como los tipos son manipulados por el entorno de ejecución.
- ***Common Language Specification* (CLS),** es una especificación que define un conjunto de características comunes que los lenguajes .NET deben cumplir para su integración.

En la parte superior del CLR se encuentran las bibliotecas de clases base que definen los elementos fundamentales para construir las aplicaciones de software como sitios web, servicios web, aplicaciones de escritorio y otras tecnologías.

1.8.1 CARACTERÍSTICAS DE .NET *FRAMEWORK*

El .NET *Framework* tiene varias características importantes:

1.8.1.1 Soporte de Estándares

Microsoft diseño el .NET *Framework* de forma que estuviera basado en prácticas y estándares de la industria. Existen especificaciones como el CLI (*Common Language Infrastructure*) o el lenguaje C#, que han sido estandarizadas por organizaciones como la ISO (*International Organization for Standardization*), la ECMA (*European Computer Manufacturers Association*) o la IEC (*International Electrotechnical Commission*), permitiendo ampliar el uso de la plataforma en otros sistemas operativos.

1.8.1.2 Soporte de varios lenguajes de programación

Las aplicaciones .NET pueden ser creadas usando diferentes tipos de lenguajes

como C#, *Visual Basic* .NET, etc. Además se permite la integración entre lenguajes, lo cual permite la reutilización de código. Los lenguajes pueden ser desarrollados por Microsoft u otros fabricantes que cumplan con los estándares.

1.8.1.3 Modelo simplificado de desarrollo

Las bibliotecas de .NET no se graban en el registro del sistema, para la implementación de aplicaciones se usa información adicional de los componentes necesarios, la cual es incluida en los bloques de construcción de una aplicación.

1.8.1.4 Jerarquía de clases extensible

La plataforma cuenta con una jerarquía de clases que permite el desarrollo de aplicaciones facilitando el acceso y la extensión de las clases de la forma que sea requerida.

1.8.1.5 Ensamblados

Un ensamblado es el bloque de construcción de las aplicaciones de .NET, está formado por código, recursos y un manifiesto, que provee la información para la ejecución. Un ensamblado se puede guardar en un solo archivo ejecutable, en una sola biblioteca o varios archivos que contengan la información separada en bibliotecas, otros archivos y un archivo de manifiesto.

La función del manifiesto es proveer información que describa al ensamblado, contiene elementos como la identidad ensamblado, descripción de las clases y tipos que expone el ensamblado, referencias a otros ensamblados y detalles de seguridad para ejecutar el ensamblado.

1.8.1.6 Espacios de nombres

Un espacio de nombres es un elemento que proporciona una estructura jerárquica a las clases dentro de la plataforma, permitiendo su organización y reduciendo la

posibilidad de conflictos. Las clases del .NET *Framework* se encuentran dentro del espacio de nombres *System*, el cual se subdivide según la funcionalidad de las clases. Para acceder a un espacio de nombres se debe referenciar al ensamblado que lo contiene.

1.8.1.7 Construcción y Ejecución del Código

El código que produce un lenguaje que se ejecuta y dispone de los servicios del entorno de ejecución de .NET se denomina código administrado, lo cual implica que el código es independiente del procesador.

En el proceso de compilación, el código fuente es convertido a un lenguaje intermedio, mediante un compilador del lenguaje. El lenguaje intermedio se guarda en un ensamblado y representa el código administrado.

Posteriormente, se carga el ensamblado y se convierte el lenguaje intermedio en código binario a través del compilador JIT (*Just-In-Time*) para el procesador en el cual se ejecuta la aplicación. El compilador no convierte todo el código del lenguaje intermedio directamente, sino según se necesite y se almacena en memoria para futuras llamadas.

1.9 LENGUAJE C# [L5]

Es un lenguaje de programación orientado a objetos desarrollado por Microsoft que permite crear y desarrollar aplicaciones para distintos entornos, forma parte del .NET *Framework* y su última versión es la 5.0. C# está estandarizado por la ECMA (ECMA-334) y la ISO (ISO/IEC 23270).

C# tiene varias implementaciones como Microsoft .NET, Mono y DotGNU que se basan además del estándar C# en el estándar del CLI (ECMA-334). La estandarización del lenguaje y el CLI ha permitido que aplicaciones C# puedan funcionar sobre otras plataformas.

El proyecto Mono es una implementación *open source* que permite desarrollar aplicaciones multiplataforma. De este proyecto se deriva *Xamarin* y sus implementaciones que permiten construir aplicaciones para iOS y *Android*. *Xamarin*.iOS y *Xamarin*.*Android* permiten crear aplicaciones nativas para iOS y *Android* basadas en C# y las bibliotecas de clases base.

Existen varios IDE para desarrollar aplicaciones C# como Visual Studio (VS), SharpDevelop, MonoDevelop y *Xamarin Studio*. Visual Studio es desarrollado por Microsoft para el .NET *Framework*, SharpDevelop es una alternativa Visual Studio en entornos Microsoft, MonoDevelop es un entorno multiplataforma para lenguajes .NET y *Xamarin Studio* es una versión basada en MonoDevelop orientada al desarrollo de aplicaciones móviles.

Para el desarrollo de aplicaciones multiplataforma con C# se usa en conjunto las características propias del lenguaje, un entorno de desarrollo e implementaciones del lenguaje adecuadas. Además es necesario utilizar patrones de diseño que permitan un enfoque real y práctico para optimizar tiempos de desarrollo. Monocross es un *framework open source* y multi plataforma basado en el lenguaje C# y el Mono *Framework*, una alternativa real para construir aplicaciones basadas en las características antes mencionadas.

1.9.1 CARACTERÍSTICAS DE C# [L9]

C# es un lenguaje de programación que permite crear aplicaciones para distintos ambientes, es de propósito general, proporciona seguridad de tipos y está orientado a objetos. Sus características principales son:

1.9.1.1 Orientación a Objetos

C# está diseñado bajo el enfoque de la programación orientada a objetos, por lo que incluye características como encapsulación, herencia y polimorfismo. Se distinguen los siguientes aspectos:

1.9.1.1.1 Sistema unificado de tipos

Un tipo es el bloque fundamental de C# que encapsula datos y funciones, y puede ser ejecutado en el entorno de .NET. Todos los tipos en C# comparten al mismo tipo base y las mismas funcionalidades. Los tipos pueden ser propios de la plataforma .NET o creados por el usuario para añadir funcionalidades.

1.9.1.1.2 Interfaces y clases

C# admite diferentes tipos además de clases. Una interfaz es similar a una clase pero solo contienen declaración de sus elementos, no sus implementaciones. Los tipos que usan una interfaz son los encargados de su implementación.

1.9.1.1.3 Propiedades, métodos y eventos

Son elementos que permiten definir la funcionalidad de los objetos. Las propiedades encapsulan el estado de un objeto, los métodos permiten acceder al comportamiento de los objetos, los eventos son un mecanismo de notificación de algún suceso.

1.9.1.2 Seguridad de Tipos

C# está diseñado como un lenguaje con seguridad de tipos, esto significa que el acceso a los tipos de datos siempre se realiza de forma correcta a través de protocolos específicos, lo cual asegura la consistencia del código.

C# permite que un tipo ejecute su comportamiento definido, pero impide que realizar las operaciones que ejecuta otro tipo. La seguridad de tipos se basa en estrictas reglas que se controlan en tiempo de complicación y de ejecución.

1.9.1.3 Administración de memoria

C# realiza de forma automática el manejo de memoria a través del entorno de

ejecución. El CLR a través del recolector de basura supervisa periódicamente los bloques de memoria que no se usan por parte de los objetos y libera la memoria.

1.9.2 VERSIONES C# [W9] [W10]

C# ha sido adoptado como uno de los principales lenguajes para el desarrollo de aplicaciones en el .NET *Framework*, esto se debe al gran soporte por parte de Microsoft con nuevas versiones que añaden nuevas características que facilitan la programación. Las versiones más relevantes son:

1.9.2.1 C# 1.0

Fue lanzada en el año 2002, se relaciona con VS .NET 2002 y .NET 1.0. Su característica principal es el uso del código administrado.

1.9.2.2 C# 2.0

Fue lanzada en el año 2005, se relaciona con VS .NET 2005 y .NET 2.0. Sus características principales son los genéricos, métodos anónimos, iteradores y tipos nulos. En esta versión se vieron cambios considerables para mejorar la eficiencia y manejo de código, lo cual representó una excelente perspectiva para el futuro y evolución del lenguaje.

1.9.2.3 C# 3.0

Fue lanzada en el año 2007, se relaciona con VS 2008 y .NET 3.0 y 3.5. Sus características principales son los tipos implícitos, inicializadores de objetos, propiedades auto implementadas, tipos anónimos, métodos extensores, expresiones lambda, expresiones de consulta y LINQ (*Language-Integrated Query*). Debido a su potencial son las características más conocidas y usadas en las aplicaciones.

1.9.2.4 C# 4.0

Fue lanzada en el año 2010, se relaciona con VS 2010 y .NET 4.0. Sus características principales son los tipos dinámicos, parámetros opcionales, argumentos con nombre, varianza de tipos y mejoras de soporte para COM (*Component Object Model*).

1.9.2.5 C# 5.0

Fue lanzada en el año 2012, se relaciona con VS 2012 y .NET 4.5. Sus características principales son funciones asincrónicas e información de llamada.

1.10 TECNOLOGÍAS MICROSOFT

1.10.1 LINQ (*LANGUAGE-INTEGRATED QUERY*) [W11] [L10]

Es una metodología para realizar consultas y manipular información sobre diferentes fuentes de datos, usando sintaxis de lenguajes como C# o *Visual Basic* .NET. LINQ hace uso de características de lenguaje y del .NET *Framework* para proveer una capa de abstracción que permite trabajar de forma transparente con fuentes de datos. LINQ fue incorporado en C# 3.0 con el *Framework* 3.5.

Los principales proveedores de LINQ son: LINQ a Objetos, LINQ a XML (XML, *Extensible Markup Language*) y LINQ a orígenes de datos de ADO .NET. La sintaxis de consultas es similar al lenguaje SQL (*Structured Query Language*), pero se puede hacer uso de la sintaxis de métodos para obtener los resultados.

1.10.1.1 LINQ a Objetos

Permite realizar consultas directamente con objetos en memoria. El origen de datos son colecciones que implementan la interfaz *IEnumerable* o *IEnumerable<T>.* La colección corresponde a objetos creados por el usuario o propios del .NET *Framework*.

1.10.1.2 LINQ a XML

Es un proveedor para manipular XML mediante una interfaz basada en objetos utilizando lenguajes de programación del .NET *Framework* y las capacidades de LINQ.

1.10.1.3 LINQ a ADO .NET

Es un proveedor que permite realizar consultas sobre objetos enumerables de tipo ADO .NET. Existen tres tipos según el origen de datos: LINQ a *DataSet*, LINQ a SQL y LINQ a Entidades. LINQ a *DataSet* permite hacer consultas con funcionalidades más completas a un elemento *Dataset*, LINQ a SQL permite consultar objetos que representan a elementos que han sido mapeados de una base de datos SQL Server y LINQ a Entidades permite realizar consultas hacia elementos del modelo *Entity Framework*.

1.10.1.4 LINQ a SQL

Es un ORM (*Object-Relational Mapping*) que proporciona un modelo de objetos que representa una base de datos relacional. Permite usar LINQ para realizar consultas y modificaciones sobre los elementos de la base de datos, además permite usar procedimientos almacenados, funciones y realizar transacciones mediante un modelo de objetos.

El API (*Application Programming Interface*) de LINQ a SQL es el encargado de manejar de forma transparente al programador la conversión entre el modelo de objetos y el lenguaje SQL, y viceversa.

Para realizar el mapeo LINQ toma en cuenta los elementos de la base de datos como: tablas, columnas de las tablas, relaciones entre tablas, procedimientos almacenados o funciones y los representa como clases, colecciones o métodos dependiendo del significado de la entidad. Posteriormente, se accede a los elementos mapeados con la sintaxis de objetos.

1.10.1.4.1 Data Context

Es el elemento principal en el que se basa LINQ a SQL ya que permite trabajar directamente en un modelo de objetos debido a que controla toda la comunicación con la base de datos. Contiene la información de los mapeos a los elementos de la base de datos, administra los cambios y actualizaciones al modelo de objetos, mantiene las conexiones con el origen de datos, se encarga de realizar las consultas y guardar modificaciones en la base de datos.

1.10.2 *WINDOWS COMMUNICATION FOUNDATION* (WCF) [L10]

Es un *framework* de Microsoft para la creación de aplicaciones orientadas a servicios usando la plataforma .NET. WCF permite la creación de distintos tipos de servicios basados en estándares como SOAP (*Simple Object Access Protocol*) o REST (*Representational State Transfer*).

Para que los clientes puedan consumir los servicios existen distintas opciones de hospedaje dependiendo de las necesidades.

1.10.2.1 WCF *WebHttp Services* en .NET 4 [W12] [W13]

Consiste la construcción de servicios a través de varias características que permiten controlar el URI (*Uniform Resource Identifier*), el formato y el protocolo de los servicios. Estos servicios pueden o no cumplir con las restricciones de servicios RESTful.

Para acceder a estos servicios se usa el protocolo HTTP y los métodos estándar como *GET, PUT, POST y DELETE.* El acceso a un recurso se lo realiza por medio de un identificador URI. Además, estos servicios permiten distintos formatos para la comunicación entre los clientes y el servidor, por ejemplo XML y JSON (*JavaScript Object Notation*).

1.10.3 ASP .NET [W14]

Es un *framework* de Microsoft que permite construir sitios, aplicaciones y servicios web. ASP .NET es parte de la plataforma .NET, y puede ser usado con los distintos lenguajes de esta plataforma.

1.10.4 *WINDOWS FORMS* [W15]

Es una tecnología que permite crear aplicaciones de interfaz gráfica para Windows basadas en formularios, se utiliza para desarrollos rápidos en los cuales no se requiere una interfaz de usuario compleja.

1.10.5 *WINDOWS PRESENTATION FOUNDATION* (WPF) [W16]

Es una tecnología que permite crear aplicaciones para Windows con un gran conjunto de características de desarrollo que aprovechan los sistemas operativos de última generación. Utiliza el lenguaje XAML (*eXtensible Application Markup Language*) para construir interfaces de usuario. Se incluyó desde las versiones 3.0 y 3.5 de .NET *Framework*.

1.10.6 CONSOLA [W17]

Las aplicaciones de consola son aquellas que se ejecutan en una ventana de línea de comandos y que permiten desarrollar programas sencillos y pruebas de código con una interfaz de usuario muy simple que puede o no requerir de interacción con el usuario.

1.11 REST (*REPRESENTATIONAL STATE TRANSFER*) [L11] [L12] [W18] [W19]

Es una arquitectura para construir servicios en aplicaciones en red mediante el acceso a recursos. Está principalmente enfocado al uso de la web y al protocolo HTTP para realizar peticiones.

1.11.1 RESTRICCIONES REST

Existen varias restricciones que se deben seguir para cumplir con la arquitectura REST:

1.11.1.1 Cliente-Servidor

Permite la división de responsabilidades a través de la arquitectura cliente-servidor. El cliente no se relaciona con las responsabilidades del servidor, como el almacenamiento de datos, y el servidor no se relaciona con las responsabilidades del cliente, como su estado o su interfaz. Esto permite establecer una arquitectura distribuida en el que el cliente realiza peticiones y el servidor puede retornar respuestas. Esta restricción también permite la escalabilidad del servidor y tanto al cliente y servidor desarrollarse independientemente.

1.11.1.2 Comunicación sin estado

Cada petición realizada por el cliente debe contener toda la información necesaria para que el servidor pueda entender su significado sin necesidad de información adicional. El servidor no necesita recordar ningún estado, es el cliente en donde se mantiene el estado de la sesión.

1.11.1.3 Caché

El servidor debe ser capaz de definir las respuestas como almacenable o no en caché, de esta manera los clientes manejan adecuadamente la información de la respuesta y la interacción con el servidor.

1.11.1.4 Sistema en capas

El sistema puede contener muchas capas entre el cliente y el servidor, cada capa interactúa con su capa adyacente. Las capas se pueden agregar, quitar, modificar o reordenar dependiendo de la necesidad de la aplicación.

1.11.1.5 Interfaz Uniforme

Permite simplificar cada interfaz de forma que sea lo más genérica posible, se basa en la identificación de recursos, manipulación de recursos a través de representaciones, mensajes auto descriptivos y el uso de la hipermedia como el motor de estado de la aplicación.

1.11.1.5.1 Identificación de Recursos

Un recurso es cualquier tipo de dato que se puede enviar a un cliente, cada recurso debe ser identificado de forma única. En los servicios REST para la web, para identificar de manera única un recurso en una red se usa un URI.

1.11.1.5.2 Manipulación de recursos a través de sus representaciones

El cliente puede interactuar con los recursos de forma específica, esto es, haciendo uso de los principales métodos HTTP. El método que se usa en una petición hacia un URI en particular indica al servicio lo que se desea hacer.

Los cuatro métodos principales que se usa del estándar HTTP son: *GET, POST, PUT, y DELETE*. *GET* obtiene la representación de un recurso, *POST* crea un nuevo recurso, *PUT* actualiza un recurso existente y *DELETE* elimina un recurso.

1.11.1.5.3 Mensajes auto descritos

Un recurso puede tener múltiples representaciones debido a que no existe una limitación al respecto, las más comunes son XML y JSON. Las peticiones o respuestas deben incluir información de la representación del recurso para analizarlo y manejarlo de forma adecuada. Para especificar el tipo de medio que se representa, se usa la cabecera *Content-Type* de HTTP para indicar el tipo MIME (*Multipurpose Internet Mail Extensions*) que describe el contenido enviado.

1.11.1.5.4 Hipermedia como el motor de estado de la aplicación.

Esta restricción implica que el cliente accede a un servicio a través de un punto de acceso determinado. Posteriormente, cualquier acción futura se basa en la información, como hipermedia o hipertexto, que el servicio haya retornado en los recursos.

1.11.2 FORMATOS PARA INTERCAMBIAR INFORMACIÓN

Para intercambiar información entre distintos sistemas es necesario establecer un formato de comunicación que defina la manera en que los datos son representados. Existen dos alternativas principales que permiten transmitir y compartir información: XML y JSON.

1.11.2.1 XML (*Extensible Markup Language*) [W20]

Es un lenguaje de marcas usado para describir datos en formato de texto simple de manera flexible, está estandarizado a través del W3C (*World Wide Web Consortium*).

XML define un conjunto de reglas para la codificación de documentos a través de etiquetas de manera que sea comprensible tanto para humanos como para máquinas. XML también se utiliza comúnmente para describir archivos de configuración o interfaces de usuario en el desarrollo de aplicaciones.

1.11.2.2 JSON (*JavaScript Object Notation*) [W21]

Es un formato ligero para el intercambio de datos, está estandarizado por las normas RFC 7159 (RFC, *Request for Comments*) y ECMA-404. Se basa en una estructura atributo/valor para transmitir datos. Es comprensible para humanos y es interpretado y generado fácilmente por las máquinas, constituye una alternativa al uso de XML.

1.12 *ANDROID* [L13] [L14]

Es un sistema operativo para dispositivos móviles desarrollado por Google que está basado en Linux, tiene licencia *open source*, además cuenta con numerosas versiones desde su lanzamiento.

Las aplicaciones *Android* se desarrollan principalmente en Java, lenguaje en el cual está la documentación técnica oficial por parte de Google en la página de desarrollo de *Android*. No obstante, se puede desarrollar las aplicaciones en otros lenguajes, un ejemplo es el lenguaje C# a través de las implementaciones de *Xamarin* o dot42.

Android tiene diferentes versiones y una gran cantidad de fabricantes de dispositivos que soportan el sistema operativo. Las nuevas versiones de *Android* incluyen correcciones, mejoras y una nueva API con características no disponibles en versiones anteriores. Existe compatibilidad para aplicaciones desarrolladas en versiones anteriores, que permite su funcionamiento en nuevas versiones de la plataforma además de paquetes de soporte que permiten integrar las nuevas características en las versiones anteriores. En general, la compatibilidad depende de las características del dispositivo y de la aplicación en cuanto a su plataforma y la configuración incluida dentro del archivo de instalación.

Android soporta diferentes arquitecturas de computadora. La principal arquitectura es ARM (*Advanced RISC Machine*), los dispositivos antiguos soportan el conjunto de instrucciones ARMv5TE y los dispositivos recientes el conjunto de instrucciones ARMv7; las aplicaciones diseñadas para ARMv5TE son compatibles con la arquitectura ARMv7, pero no en sentido contrario.

Otra arquitectura importante, sobre todo en el desarrollo de aplicaciones, es la arquitectura x86 ya que permite probar aplicaciones rápidamente en comparación con emuladores basados en ARM. En el archivo de instalación de la aplicación va incluido el código necesario para la arquitectura o arquitecturas correspondientes.

Las aplicaciones *Android* se comprimen en un archivo con extensión APK (*Application PacKage File*), que contiene los componentes necesarios para instalar aplicaciones en los dispositivos. Las aplicaciones pueden ser distribuidas a través de diferentes medios: en un sitio web (mediante un enlace de descarga), a través de email, mediante tiendas de aplicaciones como *Google Play* o *Amazon App Store* para *Android*.

1.12.1 APLICACIÓN *ANDROID* [W22]

Cada aplicación *Android* se ejecuta en un proceso diferente, el cual tiene una instancia de la máquina virtual, lo que establece un ambiente aislado entre aplicaciones. El sistema puede cerrar una aplicación si necesita recursos o si se presenta inestabilidad en el sistema.

1.12.1.1 COMPONENTES

Un componente es un elemento del cual se estructura una aplicación. En *Android* existen cuatro componentes con distinta función y propósito, los cuales son:

- **Actividades:** es una interfaz visual para que el usuario interactúe con el contenido.
- **Servicio:** componente que actúa en segundo plano realizando alguna operación.
- **Proveedores de contenido:** manejan un conjunto de datos, su acceso y manipulación.
- **Receptores de mensajes de difusión:** actúan como receptores de eventos, manejando los anuncios de difusión del sistema.

1.12.1.1.1 Actividad [W23] [W24]

Es el componente principal en las aplicaciones *Android*, las cuales pueden tener en su estructura una o más Actividades. Una actividad será la principal, la cual se presenta al usuario cuando inicia la aplicación. Las actividades se organizan en

una estructura del tipo *Last In*, *First Out*; cuando una actividad inicia, esta se guarda en la pila y la anterior se detiene, al navegar hacia atrás la actividad actual se destruye y la actividad anterior continúa. Una actividad tiene 3 estados:

- ***Active/Resume*:** la actividad está en primer plano, siendo visible al usuario para interactuar a través de la interfaz; tiene la prioridad más alta.
- ***Paused*:** la actividad es visible, pero no puede interactuar directamente con el usuario; tiene el segundo nivel de prioridad.
- ***Stopped*:** la actividad está en segundo plano, no es visible al usuario; tiene el menor nivel de prioridad de los tres estados.

El ciclo de vida de la aplicación es manejado en conjunto por el estado y las transiciones entre ellos. Para controlar las transiciones existen varios métodos que pueden sobre escribirse si es necesario o si se desea modificar el comportamiento original, los cuales se utilizan en fases específicas de la aplicación: iniciar o reanudar: *onCreate*(), *onRestart*(), *onStart*(); pausar: *onPause*(); detener: *onStop*(); y finalizar: *onDestroy*().

1.12.1.1.2 Fragmentos [W25] [W26]

Un fragmento es un componente usado por una actividad que constituye una parte del interfaz o representa un comportamiento. Fue creado para manejar de manera adecuada aplicaciones que funcionan en dispositivos con pantallas de distintos tamaños. Con respecto al sistema operativo, se integró en la versión 3.0 (API 11) de *Android*, pero también se ha añadido librerías de soporte para su funcionamiento en versiones anteriores.

Existen fragmentos que tienen una funcionalidad específica como *ListFragment*, *DialogFragment* o *PreferenceFragment*, que permiten manejar listas de elementos, cuadros de diálogo o configuración de preferencias, respectivamente. Esta clase de elementos tiene su propio comportamiento y elementos de interfaz.

Para administrar fragmentos –que se consideran elementos independientes–,

cada Actividad dispone de la clase *FragmentManager* que le permite añadir, quitar o localizar estos componentes. A través de varios métodos o transacciones se controla los fragmentos en la Actividad.

El ciclo de vida de un fragmento es similar al de una Actividad, y está directamente influenciado por la actividad que lo contiene. Se puede identificar tres estados:

- ***Resumed*:** el fragmento es visible en la actividad, la cual está en primer plano.
- ***Paused*:** la actividad está pausada y el fragmento aún es visible, pero no está en interacción directa con el usuario.
- ***Stop*:** el fragmento no es visible.

Al igual que una actividad, el ciclo de vida de un fragmento es manejado por varios métodos que controlan su transición y su estado. Los métodos pueden sobrescribirse si es necesario alterar o añadir acciones al comportamiento original.

1.13 *WINDOWS PHONE* [L13] [L14] [W27] [W28]

Es un sistema operativo para teléfonos inteligentes desarrollado por Microsoft, basado en sistemas Windows CE (*Embedded Compact*) para *Windows Phone* 7 y Windows NT (*New Technology*) para *Windows Phone* 8, cuenta con dos versiones principales y varias actualizaciones.

Las aplicaciones se desarrollan a través de *Silverligth* para *Windows Phone*, las interfaces son diseñadas con XAML y el lenguaje más común es C#.

Microsoft requiere por parte de los fabricantes de hardware que cumplan características específicas para sus dispositivos con el fin de estandarizar y soportar adecuadamente el uso del sistema operativo.

Las aplicaciones para *Windows Phone* se comprimen en un archivo con extensión XAP y se instalan a través de la tienda de Microsoft; para instalar aplicaciones y probarlas se necesita registrar el dispositivo con una cuenta de desarrollador a fin de desbloquearlo.

Existe compatibilidad para las aplicaciones desarrolladas para *Windows Phone* 7, las cuales son soportadas por *Windows Phone* 8 sin necesidad de hacer cambios en el código o volver a compilarlo. Sin embargo las aplicaciones no aprovecharán las características más recientes del sistema operativo, lo cual puede ser solucionado mediante la migración y adaptación del código a las nuevas características. Por otra parte, los dispositivos *Windows Phone* 7 no pueden actualizarse a *Windows Phone* 8 debido a sus limitaciones de hardware.

1.13.1 APLICACIÓN *WINDOWS PHONE* [L14] [W29]

Se desarrolla mediante Visual Studio, el lenguaje C# y el .NET *Framework*, los principales elementos o archivos de los que se compone son:

- ***Propierties*:** son archivos de configuración del proyecto que contienen información sobre la compilación, los ensamblados, la implementación o depuración de la aplicación, también incluyen el acceso a distintas características o funciones del dispositivo; se editan en el visualizador de propiedades del proyecto.
- ***References*:** contiene los ensamblados requeridos por la aplicación.
- **App.xaml:** es una clase que representa el punto de entrada de la aplicación y maneja los eventos del ciclo de vida de la aplicación.
- **Imágenes:** existen tres imágenes que se agregan al proyecto: *ApplicationIcon.png*, ícono de la aplicación en la lista de aplicaciones del dispositivo; *Background.png*, imagen visualizada cuando la aplicación se fija en la vista rápida de aplicaciones; *SplashScreenImage.jpg*, imagen mostrada cuando la aplicación inicia.
- **MainPage.xaml:** es la vista o página inicial que se muestra cuando la aplicación ha iniciado.

1.13.1.1 XAML (*eXtensible Application Markup Language*)

Es un lenguaje de marcas basado en XML para la creación de la interfaces de usuario, es usado principalmente en *Silverlight* y WPF. XAML ayuda a separar la creación del interfaz y el código usado para su manipulación, que se realiza en otro lenguaje.

1.13.1.2 Navegación [W30]

Una aplicación *Windows Phone* está compuesta por una o más páginas o vistas, el modelo de navegación es similar al usado en los navegadores web. Se puede navegar hacia adelante mediante enlaces hacia otras vistas y retornar hacia atrás mediante el botón Atrás.

Windows Phone utiliza dos controles para manejar la navegación: *Frame* y *Page*. *Frame*, es un control que contiene otros elementos y es el primer nivel de la interfaz de usuario, existe uno por aplicación. *Page*, es un control que muestra el contenido informativo para la visualización del usuario, existe uno o varios por cada aplicación a través de los cuales el usuario realiza la navegación.

Las páginas a las que se navega se almacenan en una pila o historial de navegación. Por una parte, cuando se navega a una nueva página, la página de la cual se accede es el último elemento en la pila. Por otra parte, la última página a la que se accedió no forma parte del historial y si la navegación es hacia atrás –mediante con el botón Atrás del dispositivo o un botón en la barra de una aplicación– la última página es borrada y la página actual pasa a ser el último elemento que fue añadido en el historial.

La navegación se realiza a través de transiciones entre páginas mediante el elemento *NavigationService* de cada página el cual contiene métodos de desplazamiento como *Navigate*(), *GoBack*() además de otros métodos, propiedades y eventos.

Una página expone varios métodos para controlar detalladamente la navegación: *OnNavigatedTo*, *OnNavigatingFrom*, *OnNavigatedFrom* y *OnBackKeyPress*, dependiendo del método se puede controlar los siguientes aspectos: cancelar la navegación, guardar información de la página, inicializar la página y retornar hacia atrás.

1.13.1.3 Ciclo de vida [W31]

El ciclo de vida de una aplicación comprende desde el inicio hasta la finalización de la misma. En *Windows Phone* para controlar el estado de la aplicación se tiene dos niveles: nivel de aplicación y nivel de página.

Windows Phone maneja el ciclo de vida a nivel de aplicación a través de eventos entre las transiciones que ocurran, dependiendo si la aplicación inicia, finaliza, está en primer plano o segundo plano. Los eventos son:

- ***Launching*:** se produce cuando la aplicación inicia.
- ***Activating*:** se produce cuando la aplicación pasa de segundo plano a primer plano.
- ***Deactivating*:** se produce cuando la aplicación pasa de primer plano a segundo plano.
- ***Closing*:** se produce cuando se cierra la aplicación.

Los eventos se manejan a través de los siguientes métodos: *Application_Launching*, *Application_Activated*, *Application_Deactivated* y *Application_Closing* respectivamente, estos métodos se encuentran en el archivo App.xaml.cs y su función principal consiste en controlar el estado de la aplicación. La acción por defecto se ejecuta si no se ha sobrescrito el método.

Windows Phone también controla el ciclo de vida a nivel de cada página a través de eventos, los cuales se manejan a través de los métodos de navegación. La acción por defecto se ejecuta si no se ha sobrescrito el método.

1.14 AMBIENTE DE DESARROLLO

El ambiente de desarrollo está conformado por todos los programas y herramientas necesarios para crear, desarrollar, mantener e implementar el proyecto. A continuación se detalla los elementos para el desarrollo de la aplicación.

1.14.1 WINDOWS 7 [W32]

Es una versión de la familia de sistemas operativos de Windows NT de Microsoft. Está orientada a equipos clientes y cuenta con varias ediciones para arquitecturas de 32 y 64 bits. Tiene una actualización, el *Service Pack* 1.

1.14.2 WINDOWS SERVER 2008 R2 [W33]

Es un sistema operativo de servidor de Microsoft. Corresponde a la familia de sistemas operativos Windows NT, tiene varias ediciones que funcionan únicamente en una arquitectura de 64 bits. Se considera la versión de servidor de Windows 7, y al igual que la versión de cliente tiene una actualización, el *Service Pack* 1.

1.14.3 MICROSOFT VISUAL STUDIO [W34]

Es un IDE desarrollado por Microsoft para el .NET *Framework*. Contiene una colección completa de herramientas y servicios que permiten crear distintas aplicaciones para plataformas Microsoft y otras plataformas, soporta una gran cantidad de lenguajes de programación. Tiene diferentes versiones tanto comerciales como gratuitas.

1.14.4 XAMARIN STUDIO [W35]

Es un IDE desarrollado por *Xamarin*. Contiene características que permiten crear aplicaciones nativas para iOS y *Android*. Está orientado al desarrollo de

aplicaciones móviles mediante la plataforma *Xamarin* y su funcionalidad para el desarrollo móvil es parecida a Visual Studio.

1.14.5 IIS (*INTERNET INFORMATION SERVICES*) [W36] [W37]

Es un servidor web flexible, seguro y administrable que permite hospedar distinto tipo de contenido en la web. IIS permite compartir información con los usuarios en ambientes como una intranet, una extranet o el Internet.

El servidor IIS viene incluido en diferentes sistemas operativos de la familia Microsoft, tanto en línea de servidores como en computadores personales.

1.14.6 MICROSOFT SQL SERVER [W38]

Es un sistema gestor de base de datos desarrollado por Microsoft que se basa en el modelo relacional. Utiliza el lenguaje *Transact-SQL* para las instrucciones que se ejecutan en el servidor. Tiene múltiples versiones y ediciones tanto comerciales como gratuitas.

1.14.7 SQL SERVER MANAGMENT STUDIO [W39]

Es un entorno para acceder, configurar, administrar y desarrollar los componentes de Microsoft SQL Server. Tiene un conjunto de herramientas gráficas y editores de *script* que permiten acceder a los distintos elementos y características de SQL Server.

1.14.8 SQLDBX [W40]

Es un IDE rápido y sencillo con numerosas características para administrar distintas bases de datos de diferentes proveedores. Sus principales componentes son un explorador de objetos de base de datos y un editor SQL. Ofrece una interfaz sencilla y similar entre los distintos DBMS.

1.14.9 FIDDLER [W41]

Es una herramienta que permite la depuración web. Fiddler permite capturar el tráfico HTTP y HTTPS (*Hypertext Transfer Protocol Secure*) para analizar sus elementos. También permite crear solicitudes HTTP y ejecutarlas desde la propia aplicación.

Facilita probar servicios web de manera eficiente a través de solicitudes y respuestas web y muestra el detalle la información del protocolo utilizado.

1.14.10 TELERIK TEST STUDIO

Es una herramienta que permite realizar pruebas a sitios o aplicaciones web para medir el rendimiento bajo diversos parámetros. Permite la configuración de distintas pruebas y presenta los resultados en forma gráfica y numérica para el análisis e interpretación de los datos obtenidos.

1.14.11 PLATAFORMA *ANDROID* [L15]

La plataforma *Android* necesita los siguientes elementos:

1.14.11.1JDK (*Java Development Kit*)

Es un software que permite la creación de programas Java, está compuesto de herramientas como la máquina virtual de Java, el compilador de Java y las utilidades JAR (*Java ARchive*) y documentación de Java.

1.14.11.2*Android* SDK (SDK, *Software Development Kit*)

Es un conjunto de herramientas y bibliotecas necesarias para desarrollar aplicaciones *Android*, permite escoger los paquetes adecuados de acuerdo a la versión de *Android* requerida.

1.14.11.3AVD *Manager* (AVD, *Android Virtual Devices*)

Es una de las herramientas del *Android* SDK que proporciona una interfaz gráfica para la creación de dispositivos virtuales *Android* de acuerdo a las versiones específicas del API instaladas. Los dispositivos virtuales son usados por el emulador de *Android*, incluido en el SDK, para desarrollar y probar las aplicaciones sin un dispositivo físico pero con igual funcionamiento.

Los dispositivos virtuales son genéricos con la versión de *Android* pero también existen implementaciones personalizadas por fabricantes de dispositivos reales que se pueden utilizar para probar aplicaciones. Existen alternativas de emuladores *Android* desarrollados por otras compañías, como *Genymotion*, que mejoran la velocidad de los emuladores originales para probar las aplicaciones.

1.14.11.4*Xamarin.Android* (*Mono for Android*)

Es un conjunto de herramientas que permiten crear aplicaciones nativas para *Android* basadas en el lenguaje C#. Se basan en la implementación open source Mono del .NET *Framework* permitiendo usar las bibliotecas de clases base de .NET y además tiene enlaces para comunicarse con el API de *Android*. Puede ser usado por *Xamarin Studio* o Visual Studio.

1.14.12 PLATAFORMA WINDOWS [L15]

1.14.12.1*Windows Phone* SDK

Es un conjunto de herramientas necesarias para crear aplicaciones para dispositivos *Windows Phone*. Existen diferentes versiones dependiendo del sistema operativo. El SDK incluye emuladores que permiten probar directamente las aplicaciones sin necesidad de un dispositivo físico. Para las versiones de *Windows Phone* 7 se requiere Windows 7 o Windows Vista y para las versiones de *Windows Phone* 8 se requiere Windows 8.

1.15 APLICACIONES MULTIPLATAFORMA EN C#

1.15.1 TÉCNICAS PARA USAR CÓDIGO COMPARTIDO [L13]

C# es un lenguaje que puede ejecutarse sobre varias plataformas, es por ello que se pueden construir aplicaciones con la misma funcionalidad en diferentes entornos. Una ventaja de utilizar C# es que permite compartir código que no se relaciona directamente con la interfaz. Es decir, la lógica del negocio, que por lo general es la parte central y funcional de una aplicación puede ser escrita una vez y utilizarse el mismo código fuente a través de distintas aplicaciones. El código de la interfaz de usuario es difícil de compartir ya que cada plataforma define sus propios elementos visuales de manera única, por tanto se prefiere que la interfaz sea específica de cada plataforma, de esta manera se puede hacer uso también de las características que brinda cada ambiente.

Existen varias técnicas que permiten compartir código entre distintas aplicaciones, estas técnicas están basadas en características del lenguaje, patrones de diseño, características del compilador y funcionalidades del ambiente de desarrollo. A continuación se describen las técnicas más importantes admitidas.

1.15.1.1 Configuración de proyectos [W42]

Los IDE principales como Visual Studio o *Xamarin Studio* utilizan la misma estructura para organizar los elementos de desarrollo. Esta estructura se basa en dos elementos: Soluciones y Proyectos.

Una solución es el nivel superior de la organización, define los elementos necesarios para crear la aplicación. La solución generalmente incluye uno o más proyectos, archivos e información adicional.

Un proyecto se utiliza para organizar, compilar y depurar los elementos que forman parte de la aplicación. El resultado de un proyecto es un ensamblado de tipo biblioteca o un archivo ejecutable para una aplicación.

En el desarrollo de aplicaciones multiplataforma, la solución para cada aplicación generalmente está conformada por un proyecto principal que genera los archivos finales de la aplicación y por uno o varios proyectos que serán referenciados por el proyecto principal y que permiten organizar adecuadamente la lógica la aplicación.

Para compartir código entre diferentes plataformas se utilizan proyectos de bibliotecas de clases que son referenciados por otros proyectos o por el proyecto principal. Los proyectos deben tomar en cuenta la plataforma de la cual forman parte, por lo que en *Windows Phone* se añade un proyecto *Windows Phone Class Library*, en un proyecto para *Android* se añade un proyecto *Android Class Library*, en un proyecto de consola se añade un proyecto *Class Library* estándar, y así sucesivamente.

Los proyectos son definidos dependiendo de la plataforma debido a los límites específicos de cada entorno. Una aplicación para un dispositivo móvil no necesita ni soporta todos los elementos, como espacios de nombres, que puede requerir una aplicación de escritorio. Por tanto, al definir proyectos específicos se asegura que los elementos referenciados estén disponibles en el *framework* y no existan problemas en tiempo de compilación o de ejecución.

1.15.1.2 Vinculación de Archivos (*File Linking*)

Es una estrategia para compartir código que permite crear un archivo de código y compartirlo y utilizarlo por diferentes proyectos. El archivo que es vinculado se mantiene en una ubicación específica pero es utilizado como un archivo normal en el proyecto o proyectos que lo vinculen. Cuando se edita el archivo en un proyecto, el cambio se refleja en los otros proyectos que lo hayan referenciado, de esta manera se mantiene los proyectos actualizados.

1.15.1.3 Estructura de Proyectos y Archivos

La estructura de los proyectos de bibliotecas de clases, en cuanto a carpetas y

archivos y generación del ensamblado, es bastante similar a pesar de la plataforma. Al rediseñar su organización, se puede agrupar los archivos de proyecto en una sola ubicación, lo que implica que hacen referencia automática a las carpetas y archivos en ese lugar. Además las referencias se definen en los archivos de proyecto, por lo cual son independientes entre proyectos de diferentes plataformas. Es de esta manera, que cada proyecto puede tener los mismos elementos y acceder directamente a ellos. La generación del ensamblado dependerá de cada proyecto y será específico a cada plataforma; posteriormente el ensamblado puede ser referenciado por otros proyectos de manera directa dentro de la solución. Este proceso es una alternativa para compartir código, sin utilizar la vinculación de archivos y de forma eficiente al utilizar elementos comunes en un proyecto.

1.15.1.4 Abstracción

La abstracción es una técnica que permite interactuar con una definición en lugar de una implementación del código. La abstracción para reutilización de código usa interfaces, que contienen solo elementos como propiedades y métodos; la implementación se realiza en los objetos que usan la interfaz.

La interfaz forma un contrato o un acuerdo, lo cual permite trabajar directamente con la interfaz y sus definiciones en un objeto que la implemente en lugar del objeto en particular. Esto permite separar código específico de una plataforma de la definición.

1.15.1.5 Patrón *Observer*

Es un patrón de diseño que define una dependencia de uno a muchos entre objetos; un objeto, sujeto, puede publicar actualizaciones cuando cambia de estado, los otros objetos, dependientes, se suscriben al sujeto para recibir notificaciones cuando estas sucedan y actualizarse automáticamente. La implementación de este patrón se basa en eventos C#.

1.15.1.6 Clases Parciales

Es una característica que permite separar la definición del código de una clase en diferentes archivos. Las clases parciales se usan para añadir funcionalidad a una plataforma específica mediante el uso de la API o código adicional que se requiera; de esta manera se puede tener un código común que será compartido por todas las plataformas y código personalizado para cada plataforma.

1.15.1.7 Compilación Condicional

Es una técnica que permite compilar de forma selectiva bloques de código basados en directivas de compilación. A través de esta técnica se puede incluir diferente código que será usado dependiendo de la plataforma o del tipo de configuración de construcción de la aplicación.

1.16 ARQUITECTURA *MONOCROSS* [L15]

La arquitectura multiplataforma permite la portabilidad de código pero no en su totalidad, los principales aspectos de conflicto son la navegación, el flujo de trabajo y en especial la interfaz de usuario. Además las aplicaciones deben aprovechar las capacidades de la interfaz de cada plataforma para mejorar la experiencia del usuario. Por tanto, es necesario separar la aplicación en distintos niveles o capas en el código. Por una parte, el código compartido, que permita incluir la mayor cantidad de código donde se ubica la lógica del negocio y el acceso a los datos. Por otra parte, el código personalizado, que involucra principalmente la interfaz de usuario y aspectos específicos de cada plataforma permitiendo una amplia personalización de la aplicación.

Figura 1.1 Separación del código compartido y código personalizado [W43]

1.16.1 MODELO-VISTA-CONTROLADOR (MVC)

Es un patrón de diseño que se basa en tres componentes, el Modelo, la Vista y el Controlador para separar la lógica de la aplicación de la interfaz de usuario, además controla el comportamiento entre los distintos elementos. El MVC es un patrón compuesto de otros patrones: *Observer, Composite y Strategy.*

1.16.2 *MONOCROSS* MVC

El patrón *Monocross* MVC se basa en el patrón MVC, su principal diferencia radica en la separación de la vista del código de la aplicación lo que posibilita personalizar las diferentes plataformas tomando en cuenta también la navegación y flujo de trabajo.

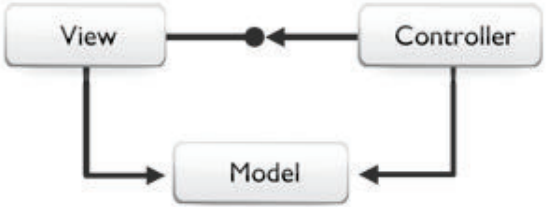

Figura 1.2 Modelo-Vista-Controlador en *Monocross* [W43]

- **Modelo:** representa la información que maneja la lógica de la aplicación. Permite conocer y cambiar su estado, no contiene referencias de la vista ni el controlador.
- **Controlador:** se encarga de obtener y manipular la información del modelo, se relaciona también con el comportamiento de la vista y se comunica con el servidor para procesar información.
- **Vista:** dependerá de la aplicación y es la representación del modelo a través del interfaz de usuario. Tiene acceso al modelo cargado en el controlador para permitir cambiar o mostrar la información.

1.16.3 NAVEGACIÓN

El patrón MVC es la base de la arquitectura *Monocross*, y el mecanismo que permite navegar entre las combinaciones MVC se basa en identificadores URI,

cada controlador se asocia con uno o más identificadores URI para representar de manera unívoca algún estado en el flujo de la aplicación.

La navegación depende de dos elementos: la Aplicación Compartida y el Contenedor de Plataforma.

1.16.3.1 Aplicación Compartida

Permite definir y acoplar los diferentes módulos MVC, especificar el inicio de la aplicación y registrar las etapas del flujo por medio de las combinaciones Controlador/URI. El controlador ejecutará las acciones dependiendo de los parámetros del URI.

1.16.3.2 Contenedor de Plataforma

Para complementar la aplicación se necesita crear un contenedor, el cual depende de cada plataforma y permite controlar la navegación, asociar las vistas con los modelos, inicializar la aplicación compartida y funciones específicas de cada plataforma.

1.16.4 PERSPECTIVAS

Finalmente, un modelo puede tener distintas representaciones de vistas, por tanto el contenedor asocia la vista con una perspectiva, que es una cadena que indica el uso de la vista y que se especifica por el controlador para presentar la vista adecuada al usuario.

CAPÍTULO II

2 DISEÑO DE LA APLICACIÓN

En el segundo capítulo se diseña la aplicación partiendo de la definición de los requerimientos del sistema. Inicialmente, se describe el problema al cual se orienta el proyecto. Mediante el uso del lenguaje UML se definen los casos de uso y el diagrama de clases. Posteriormente se define toda la arquitectura del sistema tanto en el cliente como en el servidor detallando la estructura del software y las herramientas a utilizar. Finalmente, se definen los elementos y principales características de diseño para las interfaces de usuario.

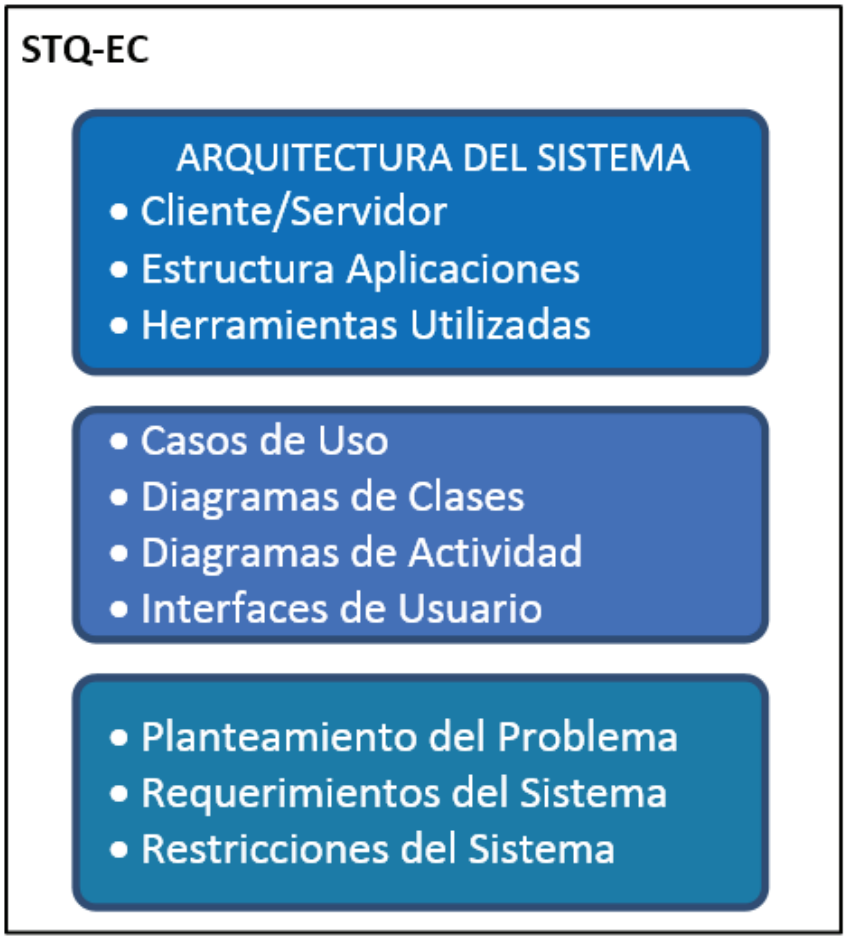

Figura 2.1 Esquema General del Diseño del Sistema

La figura 2.1 muestra el esquema general de la aplicación STQ-EC. Cada aspecto en el diagrama se describe en detalle en el presente capítulo, los cuales son la base para el desarrollo de la aplicación y permiten establecer claramente la estructura y funcionamiento del sistema.

En un principio se inicia con la información del problema para definir los requerimientos y restricciones del sistema. Posteriormente a través del UML se define el comportamiento estático y dinámico del sistema. Finalmente, la arquitectura del sistema define claramente la estructura de la aplicación, sus principales componentes y las herramientas utilizadas en el presente proyecto.

2.1 PLANTEAMIENTO DEL PROBLEMA

La ciudad de Quito cuenta con dos terminales interprovinciales para la movilización de los usuarios hacia las distintas provincias del Ecuador. Los terminales se encuentran ubicados estratégicamente al norte y sur de la ciudad y están integrados al transporte público lo que permite el acceso de la población de manera sencilla.

La Empresa Pública Metropolitana de Movilidad y Obras Públicas (EPMMOP) es la encargada de la administración de los terminales. Existen diferentes cooperativas de transporte que ofrecen sus servicios de transporte de usuarios y de encomiendas. El terminal del sur (Quitumbe) tiene más afluencia de usuarios porque cubre mayor número de provincias que el terminal del norte (Carcelén).

En la actualidad la información de un servicio o producto que se da a la comunidad tiende a estar disponible a través de distintos medios, en especial del Internet, para que su acceso sea más rápido y mediante el uso de la tecnología se optimice el manejo de la información.

Por tanto, se ha visto la necesidad de proponer una solución que facilite a los usuarios del transporte interprovincial obtener información que permita utilizar adecuadamente este servicio.

Existen varios inconvenientes que se pueden solucionar en mayor o menor medida como el tiempo perdido, las filas de espera, la incomodidad, etc., si se utiliza la tecnología para que los usuarios interactúen con la información que se maneja en los terminales.

La solución propuesta es un sistema que permita a los usuarios interactuar con la información de los terminales, las cooperativas, los viajes y demás información relevante del sistema a través de distintos dispositivos móviles.

Esta solución se basa en una aplicación que funcione sobre las plataformas *Android* y *Windows Phone* que usan los teléfonos inteligentes de la actualidad y que se adapte a las distintas necesidades de los usuarios del sistema.

La solución consiste en desarrollar una aplicación multiplataforma mediante un único lenguaje de programación en distintos entornos. El enfoque que se le ha dado para la solución del problema constituye otra parte fundamental de este proyecto por lo cual se toma en cuenta la dificultad de acoplar un sistema a distintas plataformas.

De esta manera también el proyecto constituye una fuente de información para el desarrollo de software sobre distintas plataformas. En este caso, la aplicación se basa en el lenguaje de programación C# y en herramientas relacionadas con este lenguaje, como son el entorno de desarrollo, *frameworks* para la construcción de aplicaciones, el servidor web o la base de datos.

El sistema deberá cumplir con el funcionamiento mencionado y además con varias características como la integración con una base de datos para el manejo de la información, distintos tipos de usuarios de acuerdo a su rol y acoplarse de manera adecuada con las interfaces de usuario para cada plataforma.

El presente proyecto por tanto contempla ofrecer una solución a la problemática planteada y ser un aporte a la sociedad en cuanto al conocimiento que pueda derivar la presente investigación.

2.2 REQUERIMIENTOS DEL SISTEMA

En esta sección se describe los requerimientos técnicos, requerimientos funcionales y ciertas restricciones de la aplicación. Los requerimientos del sistema se basan en la definición del proyecto y en la información sobre el funcionamiento de los terminales por parte del personal de la EPMMOP.

2.2.1 REQUERIMIENTOS TÉCNICOS

- El sistema debe funcionar en las siguientes plataformas de dispositivos móviles: *Android* y *Windows Phone*.
- El sistema debe acoplarse al menos a dos diferentes versiones del sistema operativo *Android* o *Windows Phone*.
- El sistema se diseña y desarrolla en capas para permitir que las aplicaciones sean flexibles y escalables.
- El sistema utiliza el Internet para la comunicación entre cliente y servidor.
- El sistema se desarrolla sobre el lenguaje de programación C#, tanto las aplicaciones cliente y servidor.
- El lenguaje C# es la base fundamental del desarrollo, por tanto se prioriza las herramientas y aplicaciones que permitan usar C# de forma principal; se deja en segundo plano los elementos del sistema que no se relacionen directamente con ese lenguaje de programación.
- La mayor parte del código de la aplicación cliente debe ser compartido por las aplicaciones de distintas plataformas.
- La aplicación cliente se complementa con el uso de otras aplicaciones para presentar información, como por ejemplo Google Maps.
- El sistema cuenta con diferentes niveles de seguridad. Los principales puntos de consideración son: el acceso al sistema depende del tipo de usuario; la comunicación entre aplicaciones cliente y servidor; y, el control de cambios en el sistema.
- El sistema almacena la información en una base de datos.
- El sistema se prueba dentro de un ambiente restringido, si bien su diseño

debería permitir implementarlo en ambientes de producción con las variaciones necesarias.

2.2.2 REQUERIMIENTOS FUNCIONALES

Los usuarios del sistema serán:

- **Administrador de Terminal:** administra en forma general el sistema, en forma particular administra la información de la empresa, los terminales, las cooperativas, las ciudades, las frecuencias, las unidades y los administradores de cooperativa. Existe un usuario especial que es el encargado de gestionar los administradores de terminal.
- **Administrador de Cooperativa:** administra la información de una cooperativa, manejando principalmente los viajes y reservaciones.
- **Usuario Común:** tiene acceso a la información publicada por parte de los administradores, realiza consultas y reservaciones de viajes.
- **Usuario Anónimo:** tiene acceso restringido a la información del sistema, realiza consultas al sistema.

Los requerimientos del sistema son:

- Los usuarios Administrador de Terminal, Administrador de Cooperativa y Usuario Común ingresan al sistema mediante el nombre de usuario y contraseña.
- El administrador del terminal con privilegios gestiona la creación de otros administradores de terminal si es necesario.
- Usuarios anónimos tendrán acceso informativo, accederán a información de terminales, viajes y cooperativas.
- Al acceder al sistema los usuarios mediante un menú acorde a su perfil pueden navegar entre diferentes opciones.
- Un usuario (usuario común) puede registrarse mediante un formulario de registro en el sistema.
- Los usuarios pueden modificar sus datos personales y su contraseña.

Los requerimientos para el usuario Administrador de Terminal son:

- Administrar las ciudades a las cuales se realizan los viajes.
- Administrar la información de los terminales y la empresa.
- Administrar las cooperativas con sus correspondientes opciones: terminales, boleterías, andenes, frecuencias, unidades y administradores de cooperativa.
- Realizar la administración de los administradores de cooperativa.

Los requerimientos para el usuario Administrador de Cooperativa son:

- Verificar la información de la cooperativa que se encuentra en el sistema y cambiar datos informativos opcionales.
- Crear viajes para reservaciones y administrar los viajes y reservaciones asociadas que realizan los usuarios.

Los requerimientos para un Usuario Común son:

- Acceso informativo a datos de la empresa, terminales y cooperativas.
- Consultar viajes dependiendo del destino o la cooperativa de forma informativa.
- Realizar reservaciones dependiendo de viajes disponibles para un determinado origen, destino, fecha y número de asientos. Puede cancelar las reservaciones.
- Mantiene un historial informativo de las reservaciones realizadas en el sistema.

Los requerimientos para un Usuario Anónimo son:

- Acceso informativo a datos de la empresa, terminales y cooperativas.
- Consulta viajes dependiendo del destino o la cooperativa de forma informativa.

En cuanto a los Terminales y Cooperativas se tiene los siguientes requerimientos:

- Se podrá editar la información de los terminales así como sus respectivas boleterías y andenes.
- Las cooperativas se agregan a los terminales disponibles en los cuales pueden operar.
- Las cooperativas se asocian a los terminales y deben tener su correspondiente boletería y andén.
- La cooperativa tendrá datos informativos como nombre, teléfono y opcionalmente twitter y página web. En el caso de twitter se puede acceder a los último *tweets* si la opción está habilitada.
- Las frecuencias de las cooperativas están formadas por el origen (Quitumbe o Carcelén), el destino, la hora de salida, la cooperativa a la cual pertenece y la fecha.
- Los viajes se crean en base a las frecuencias y una determinada fecha.
- La cooperativa tendrá unidades que se asociarán a los viajes en el momento de la creación del viaje o posteriormente en la edición del viaje.

2.2.3 RESTRICCIONES DEL SISTEMA

El sistema es un prototipo que tiene las características de ser multiplataforma y usar un único lenguaje, aparte del funcionamiento que debe cumplir. Por tanto en función de que la aplicación sea favorable a la presentación se debe tomar en cuenta ciertas restricciones en cuanto al diseño e implementación de la aplicación.

La complejidad puede aumentar o disminuir en la implementación de un diseño debido a las características que se pretende establecer. Las consideraciones que se han tomado en cuenta se detallan en las secciones correspondientes y se explica a su vez las alternativas en un entorno más real.

Un ejemplo de lo manifestado constituye la división de las aplicaciones por cada usuario, permitiendo manejar apropiadamente la seguridad y las funcionalidades.

2.2.4 OPCIONES GENERALES DEL SISTEMA

El sistema define las diferentes opciones a las que cada tipo de usuario puede acceder. En la tabla 3.3 se listan las opciones de cada usuario y se describe brevemente cada opción.

Opciones del Sistema		
Usuario	**Opción**	**Descripción**
ADMINISTRADOR DE TERMINAL	Perfil Usuario	Editar información personal del usuario.
	Empresa	Administrar la información de la empresa.
	Terminales	Administrar la información de los terminales.
	Cooperativas	Administrar la información de las cooperativas.
	Ciudades	Administrar la información de las ciudades.
	Reportes	Visualizar los reportes del sistema.
	Recuperar Contraseña	Reestablece la contraseña del usuario.
ADMINISTRADOR DE COOPERATIVA	Perfil Usuario	Editar información personal del usuario.
	Cooperativa	Visualizar y editar opciones informativas de la cooperativa.
	Unidades	Visualizar las unidades de la cooperativa.
	Frecuencias	Visualizar las frecuencias de la cooperativa.
	Viajes	Administrar los viajes de una cooperativa.
	Empresa	Información de la empresa.
	Terminales	Información de los terminales interprovinciales.
	Recuperar Contraseña	Reestablece la contraseña del usuario.
USUARIO COMÚN	Registro	Crear un nuevo usuario en el sistema.
	Perfil Usuario	Editar información personal del usuario.
	Reservaciones	Reservaciones del usuario a los viajes.
	Historial	Historial de reservaciones del usuario.
	Viajes	Información de viajes por destino o cooperativa.
	Empresa	Información de la empresa.
	Terminales	Información de los terminales interprovinciales.
	Recuperar Contraseña	Reestablece la contraseña del usuario.
USUARIO ANÓNIMO	Empresa	Información de la empresa.
	Terminales	Información de los terminales interprovinciales.
	Viajes	Información de viajes por destino o cooperativa.

Tabla 2.1 Opciones del Sistema

2.2.5 CASOS DE USO

En esta sección se describen los casos de uso del sistema STQEC. Se presenta una descripción y el gráfico correspondiente de las interacciones que ocurren entre los distintos usuarios y el sistema.

2.2.5.1 Acceso al Sistema

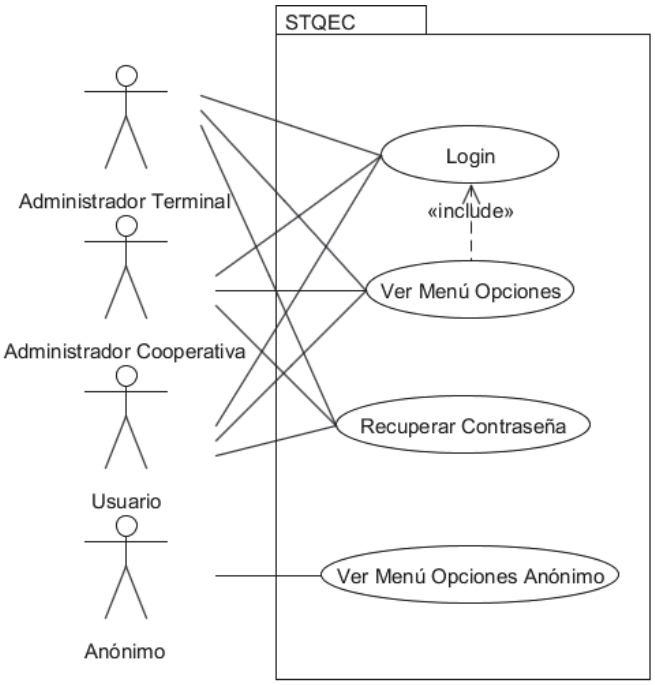

Figura 2.2 Caso de Uso Acceso al Sistema

Descripción: El usuario accede a la pantalla de login en la cual puede digitar el nombre de usuario y contraseña para ingresar al sistema. Si el login es correcto accede a un menú con opciones de acuerdo a su perfil, caso contrario se muestra un mensaje de error. Tiene una opción para recuperar la contraseña. Puede ingresar de forma anónima y tener acceso a las opciones del menú anónimo.

Precondiciones: Ninguna.

Post condiciones: El usuario puede acceder a cualquier opción de su perfil o cerrar la sesión.

2.2.5.2 Opciones Informativas

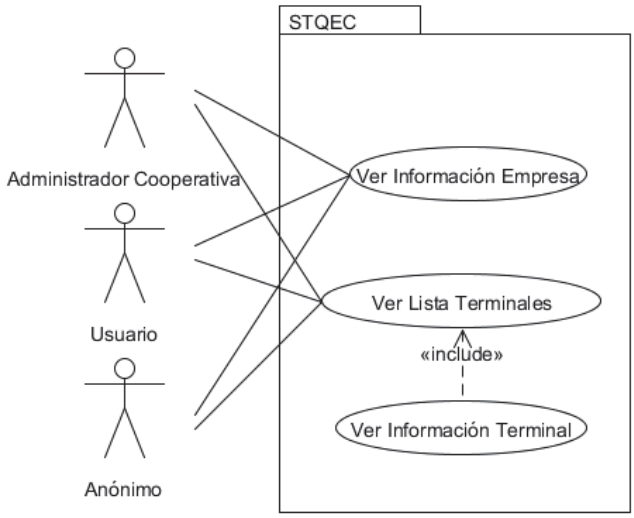

Figura 2.3 Caso de Uso Opciones Informativas

Descripción: El usuario visualiza la información de la empresa. El usuario visualiza la lista de los terminales, luego puede visualizar la información de un terminal específico.

Precondiciones: El administrador cooperativa y el usuario común deben acceder al sistema con sus credenciales, usuario anónimo directamente desde el menú de opciones.

Post condiciones: El usuario puede interactuar con elementos como página web, mapas o twitter y acceder a información más detallada.

2.2.5.3 Viajes Usuario Anónimo

Descripción: El usuario anónimo accede a un menú con distintas opciones para ver viajes. Se puede acceder a los viajes de las ciudades y a los viajes y la información de las cooperativas.

Precondiciones: Acceso desde el menú de opciones.

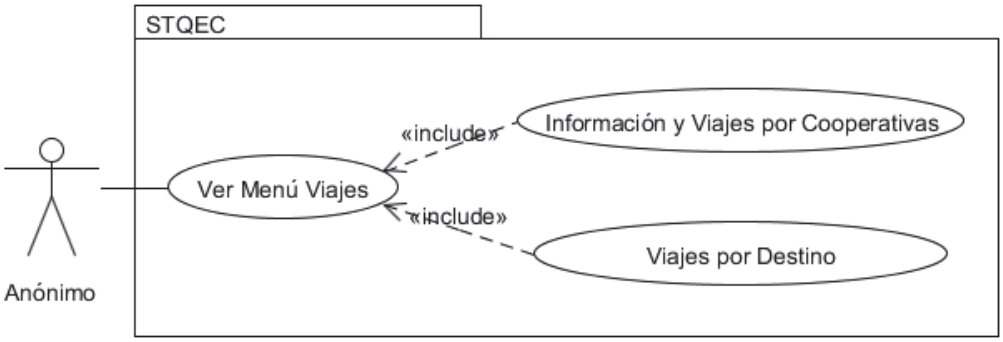

Figura 2.4 Caso de Uso Viajes Usuario Anónimo

Post condiciones: El acceso es solo informativo.

2.2.5.4 Perfil de Usuario

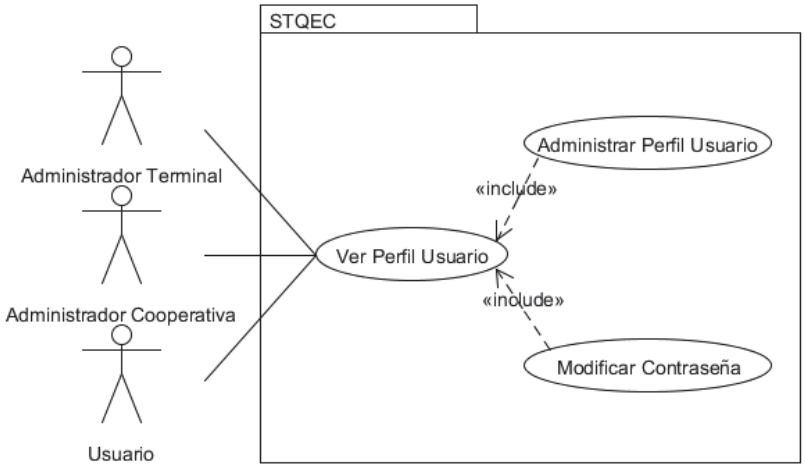

Figura 2.5 Caso de Uso Perfil de Usuario

Descripción: El usuario visualiza su información personal en el sistema. El usuario puede modificar sus datos personales a excepción del nombre de usuario. El usuario puede cambiar la contraseña.

Precondiciones: El usuario accede al sistema con sus credenciales y escoge la opción en el menú.

Post condiciones: El usuario puede visualizar la nueva información o ingresar al sistema con la nueva contraseña al iniciar nuevamente la aplicación.

2.2.5.5 Viajes y Reservaciones Usuario

Descripción: El usuario ingresa a la lista de reservaciones. Para crear una nueva reservación accede a la pantalla de búsqueda con filtros para origen, destino, asientos y fecha. Se muestran los viajes disponibles de acuerdo a los criterios de búsqueda. Escoge el viaje que quiere reservar. Visualiza la lista de reservaciones.

Precondiciones: Acceso desde el menú de opciones.

Post condiciones: La reservación es válida para un viaje, el usuario puede cancelar la reservación.

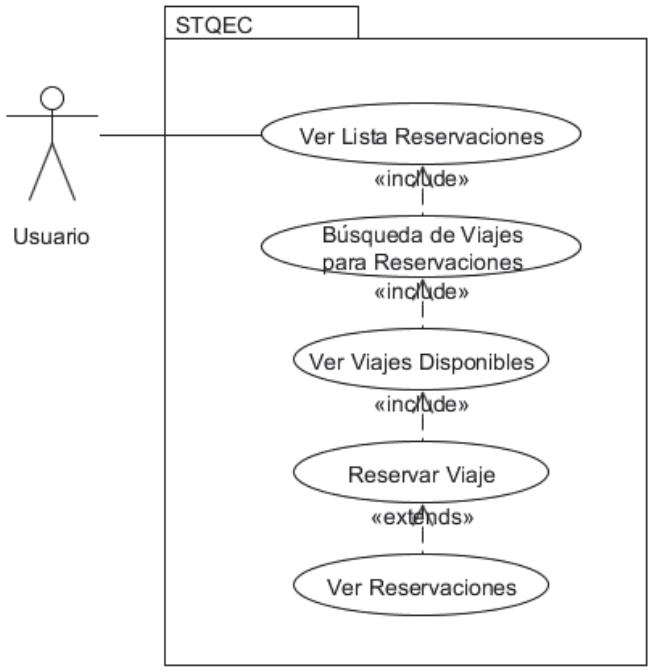

Figura 2.6 Caso de Uso Viajes y Reservaciones Usuario

2.2.5.6 Reservaciones Usuario

Descripción: El usuario visualiza las reservaciones activas que tiene. Accede a la información de la reservación. Puede cancelar la reservación y visualizarla en el historial. El historial de reservaciones muestra las reservaciones finalizadas, canceladas o anuladas.

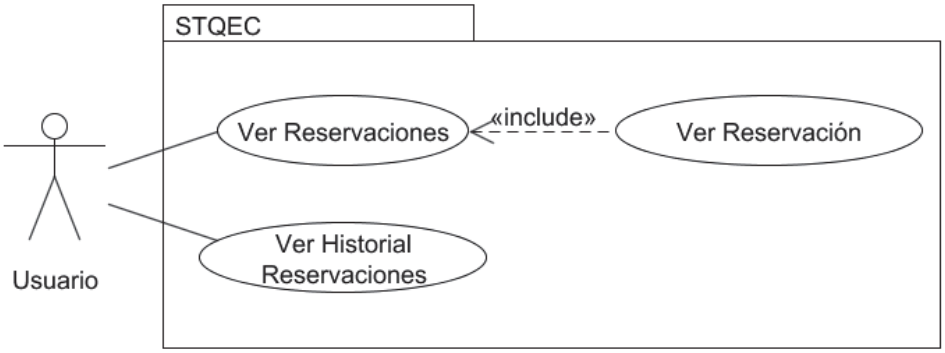

Figura 2.7 Caso de Uso Reservaciones Usuario

Precondiciones: Acceso desde el menú de opciones.

Post condiciones: Ninguna.

2.2.5.7 Información Cooperativa

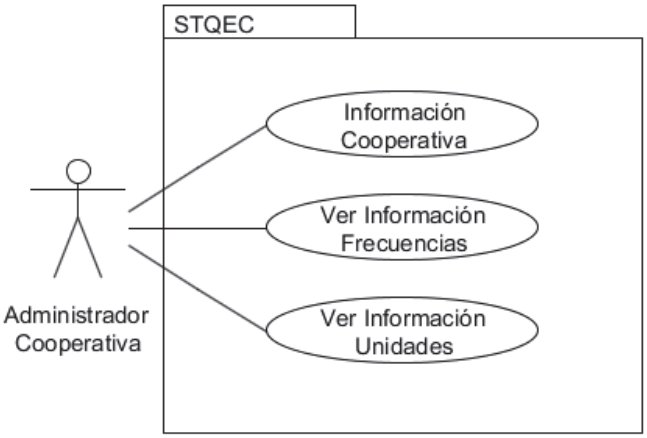

Figura 2.8 Caso de Uso Información Cooperativa

Descripción: El usuario visualiza y edita información limitada de la cooperativa. El usuario visualiza las frecuencias y las unidades de la cooperativa.

Precondiciones: Acceso desde el menú de opciones.

Post condiciones: Al editar algunas opciones de la cooperativa la información será visible a otros usuarios.

2.2.5.8 Administración Viajes y Reservaciones

Descripción: El usuario visualiza la lista de viajes para una determinada fecha, puede cambiar la fecha para visualizar los viajes correspondientes. Puede crear un viaje o crear varios viajes seleccionándolos de una lista. Puede editar la información del viaje como capacidad, la unidad o el estado. El usuario puede acceder a las reservaciones de un viaje para visualizar los clientes o cambiar su estado.

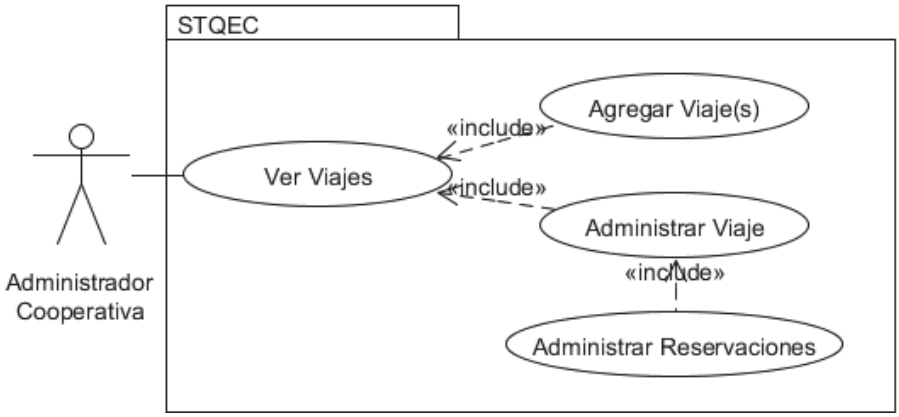

Figura 2.9 Caso de Uso Administración Viajes y Reservaciones

Precondiciones: Acceso desde el menú de opciones.

Post condiciones: Se puede visualizar los cambios realizados en los viajes como en las reservaciones.

2.2.5.9 Administrar Empresa

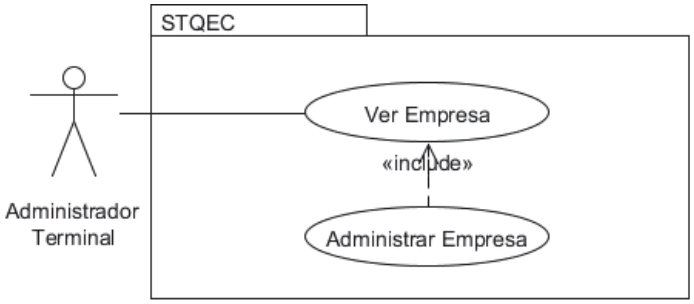

Figura 2.10 Caso de Uso Administrar Empresa

Descripción: El usuario visualiza la información de la empresa. El usuario puede modificar la información de la empresa.

Precondiciones: Acceso desde el menú de opciones.

Post condiciones: El usuario puede interactuar con elementos como página web, twitter y acceder a información más detallada.

2.2.5.10 Administrar Terminales

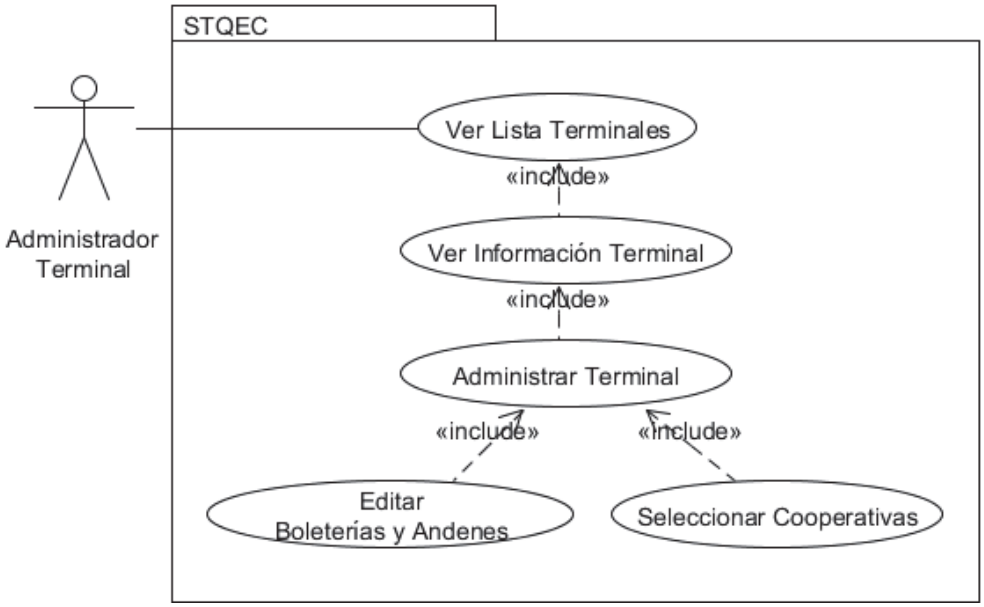

Figura 2.11 Caso de Uso Administrar Terminales

Descripción: El usuario visualiza la lista de terminales existentes, luego visualiza la información de un terminal específico. Puede modificar la información de terminales existentes, así como la capacidad de boleterías y andenes, además escoger las cooperativas que pertenecen a ese terminal.

Precondiciones: Acceso desde el menú de opciones.

Post condiciones: Otros usuarios que tienen acceso a la información del terminal pueden ver la información que ha sido modificada, interactuar con la dirección

(mapa), además la información de boleterías o andenes está disponible para configurar la ubicación de las cooperativas.

2.2.5.11 Administrar Cooperativas

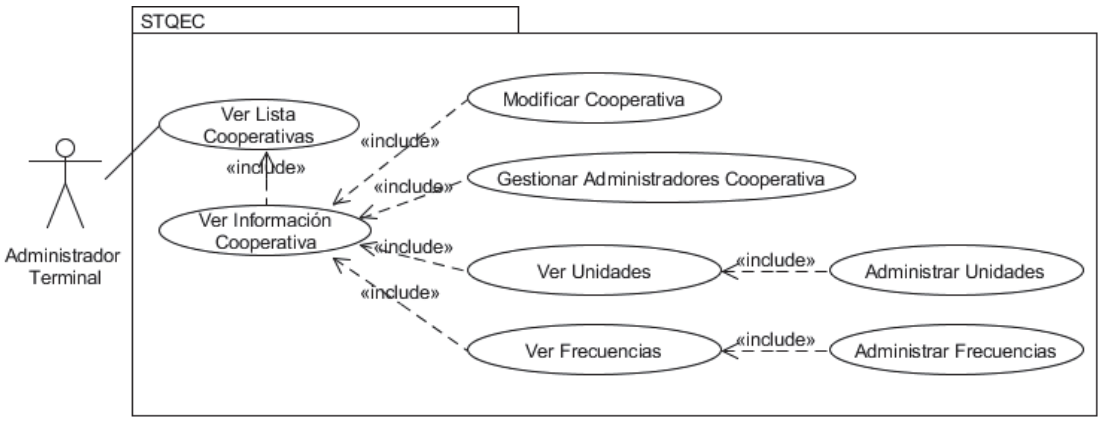

Figura 2.12 Caso de Uso Administrar Cooperativas

Descripción: El usuario visualiza las cooperativas existentes en una lista que permite filtrar información debido a la cantidad de registros, luego puede visualizar la información de una cooperativa específica. Puede crear cooperativas y modificar la información limitada de cooperativas existentes. Puede acceder a la lista de administradores de cooperativa y crear usuarios, la contraseña de estos usuarios se enviará por correo electrónico. También puede ver la lista de unidades, crear y modificar los registros. El usuario puede visualizar, crear o modificar frecuencias y asignar los días disponibles seleccionándolas en una lista.

Precondiciones: Acceso desde el menú de opciones.

Post condiciones: La información creada o modificada es accesible a otros usuarios que la usan en el sistema de forma informativa.

2.2.5.12 Administrar Ciudades

Descripción: El usuario visualiza las ciudades existentes en una lista que permite filtrar información debido a la cantidad de registros, luego puede visualizar la

información de una ciudad específica. El usuario puede crear ciudades y modificar la información de ciudades existentes.

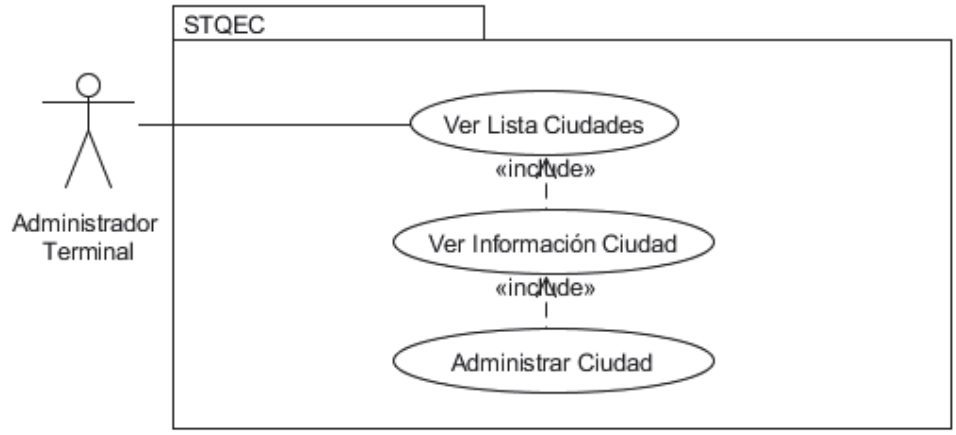

Figura 2.13 Caso de Uso Administrar Ciudades

Precondiciones: Acceso desde el menú de opciones.

Post condiciones: Las ciudades se usan principalmente para la creación de frecuencias y búsqueda de viajes por otros usuarios, una ciudad representa el origen y otra ciudad el destino.

2.2.5.13 Administradores de Terminal

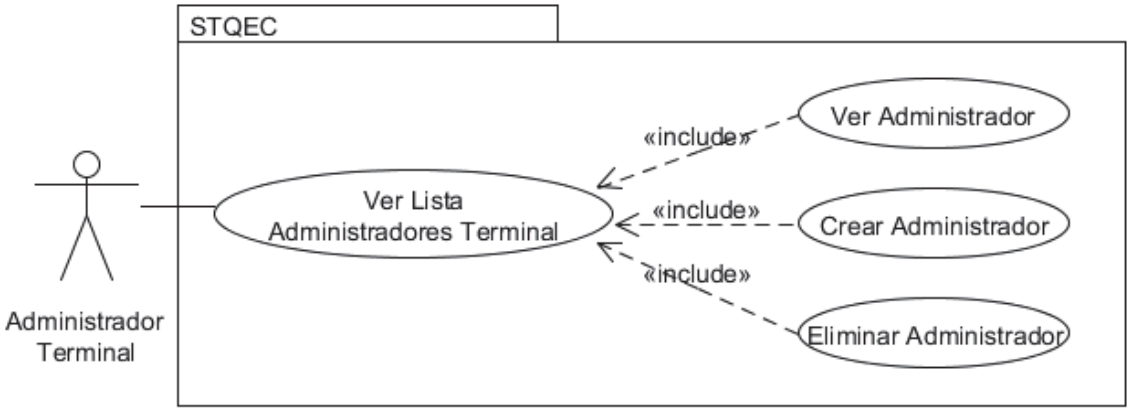

Figura 2.14 Caso de Uso Administradores de Terminal

Descripción: El usuario visualiza los usuarios de tipo administrador terminal, este acceso es restringido. El usuario puede crear administradores de terminal, la contraseña de estos usuarios se enviará por correo electrónico. También puede visualizar sus detalles o eliminarlos.

Precondiciones: Acceso desde el menú de opciones.

Post condiciones: Con la contraseña enviada por correo el usuario creado puede acceder al sistema como administrador del terminal.

2.2.5.14 Reportes

Figura 2.15 Caso de Uso Reportes

Descripción: El usuario visualiza la lista de reportes existentes, luego puede seleccionar un ítem de la lista. Ingresa los filtros si es necesario. Se visualiza el reporte y se puede actualizar la vista.

Precondiciones: Acceso desde el menú de opciones.

Post condiciones: Ninguna.

2.2.6 DIAGRAMA DE CLASES

El diagrama de clases de la figura 2.16 muestra en forma general la organización del sistema. Es decir, representa los elementos más importantes y sus respectivas propiedades en cuanto a la estructura.

Por tanto, en el diagrama no consta información detallada que se incluye ya en diagramas más específicos del sistema y además es información que se considera que siempre se incluye en una implementación.

En el capítulo posterior se muestra el detalle de las entidades y su representación correspondiente en el diagrama de la base de datos y en el diseñador del ORM.

Las relaciones más importantes en el funcionamiento general son las que ocurren entre un terminal y una cooperativa, entre la cooperativa con sus componentes, entre un viaje con la frecuencia y la unidad y finalmente entre la reservación, el viaje y el usuario.

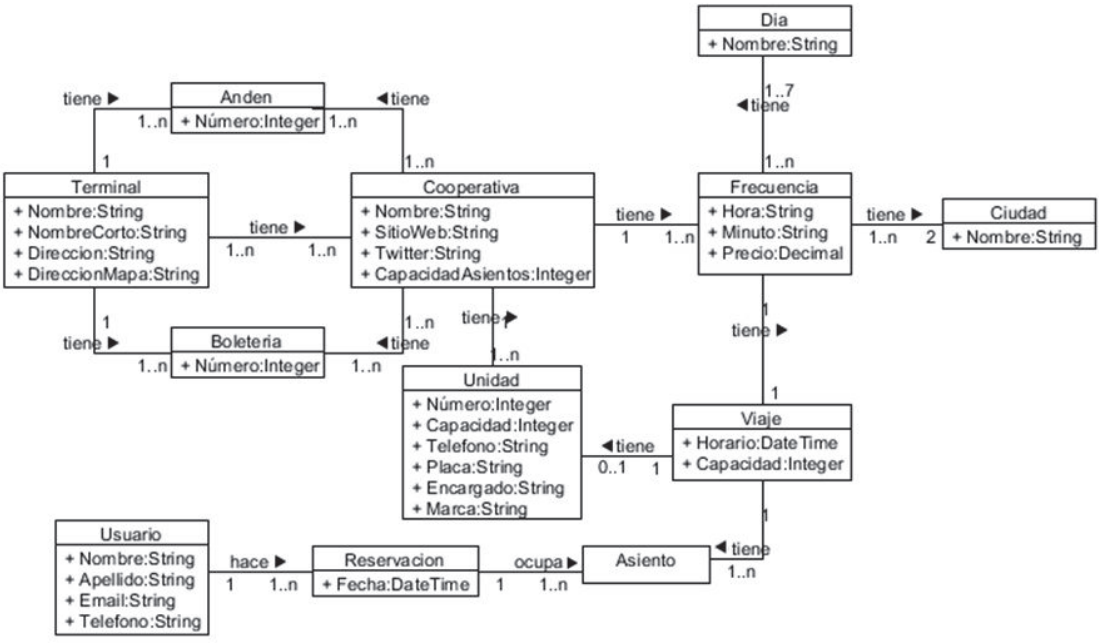

Figura 2.16 Diagrama de Clases

2.2.7 DIAGRAMAS DE ACTIVIDAD

Los diagramas de actividad muestran los flujos que suceden en la aplicación entre las distintas funcionalidades. En la representación de los diagramas se observa las distintas acciones que se pueden originar en una funcionalidad y las vistas a las cuales se puede acceder.

2.2.7.1 Acceso al Sistema

El diagrama describe las funcionalidades a las que puede acceder el usuario al momento de ingresar a la aplicación como son: registro, ingreso mediante clave y contraseña, ingreso anónimo y la recuperación de la clave de usuario.

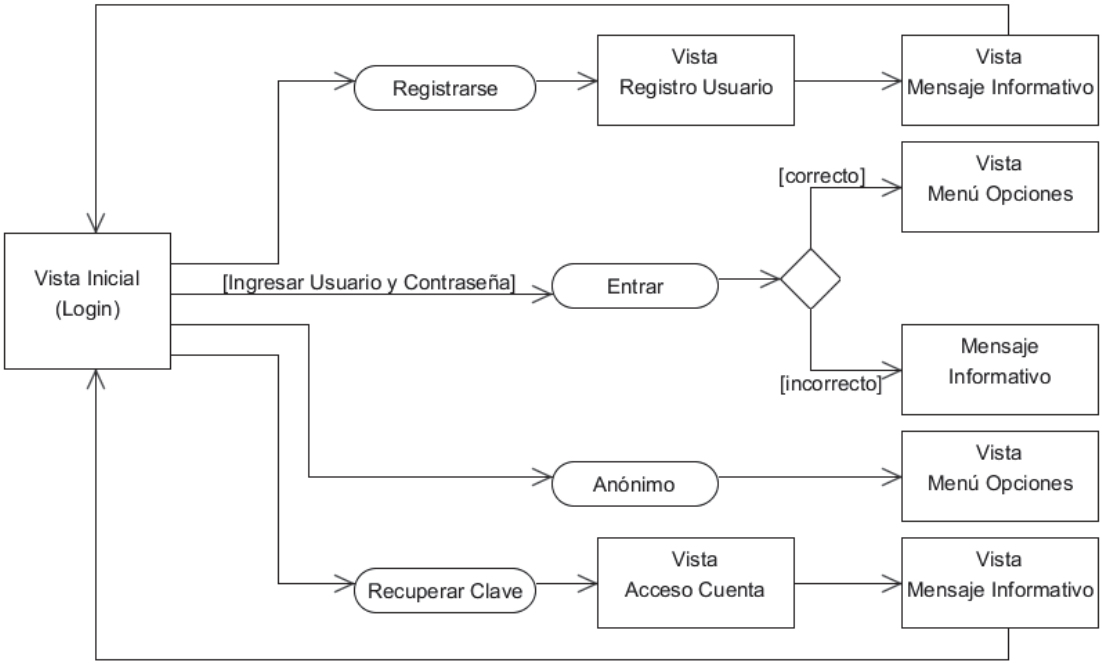

Figura 2.17 Diagrama de Actividad Acceso al Sistema

2.2.7.2 Opciones Informativas

El diagrama describe las funcionalidades que presentan información sobre la empresa y los terminales. También la interacción con otras aplicaciones para mejor visualización de la información.

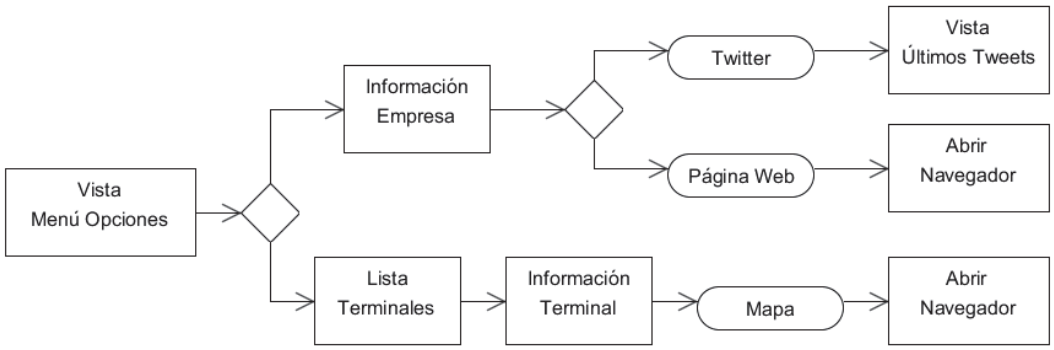

Figura 2.18 Diagrama de Actividad Opciones Informativas

2.2.7.3 Información de Viajes

El diagrama describe las funcionalidades que presentan información sobre los viajes de los destinos o los viajes y la información de una cooperativa.

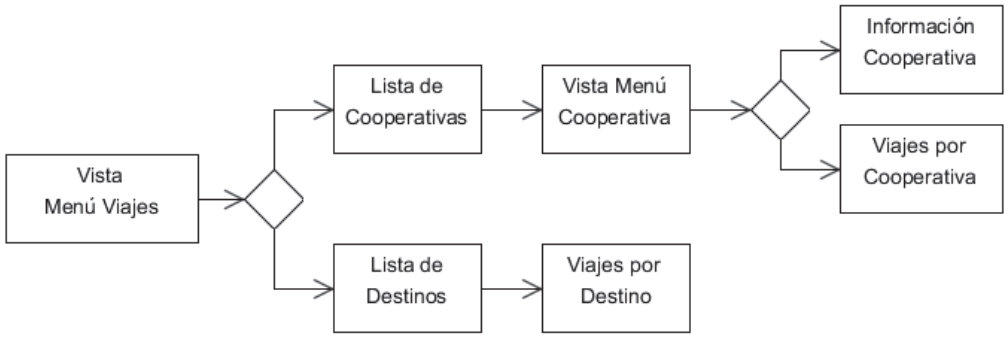

Figura 2.19 Diagrama de Actividad Información de Viajes

2.2.7.4 Información del Usuario

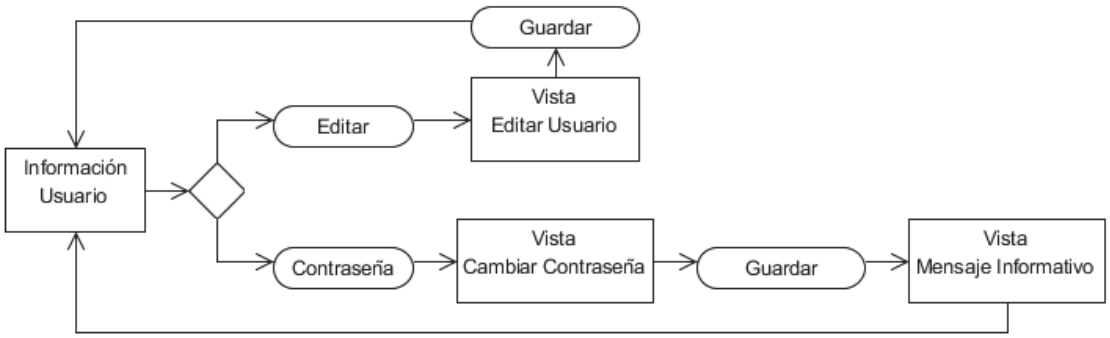

Figura 2.20 Diagrama de Información del Usuario

El diagrama describe las funcionalidades de edición y cambio de contraseña a las que accede un usuario registrado en el sistema.

2.2.7.5 Reservaciones

El diagrama describe las funcionalidades de creación de una reservación mediante la búsqueda de viajes y la cancelación de una reservación.

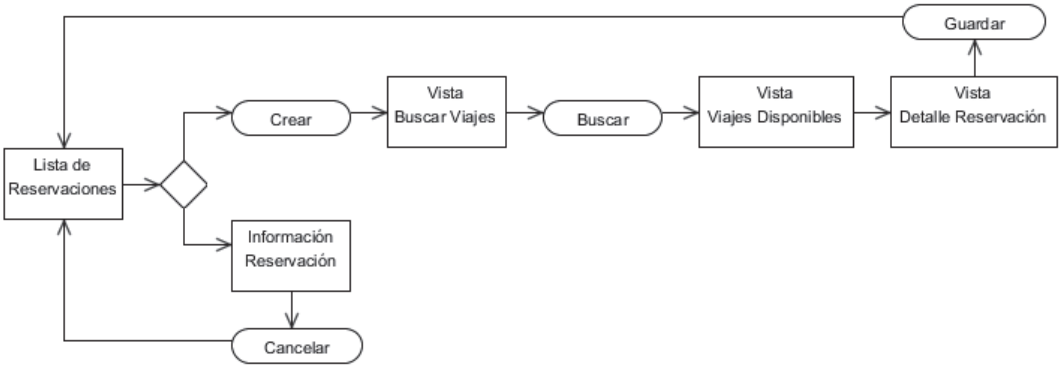

Figura 2.21 Diagrama de Actividad Reservaciones

2.2.7.6 Información Cooperativa

El diagrama describe las funcionalidades de edición de opciones informativas de la cooperativa y la información de las frecuencias y unidades de la cooperativa.

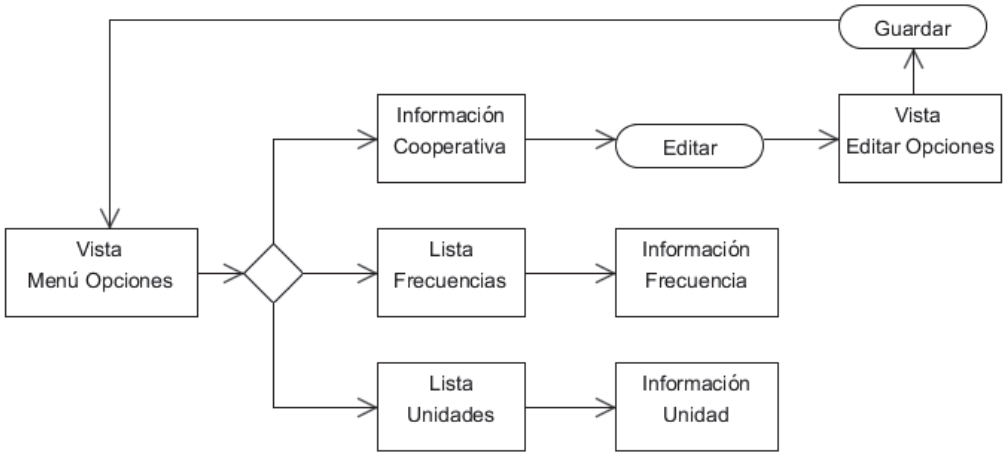

Figura 2.22 Diagrama de Actividad Información Cooperativa

2.2.7.7 Administradores Terminal

El diagrama describe las funcionalidades de creación y eliminación de usuarios administradores de terminal.

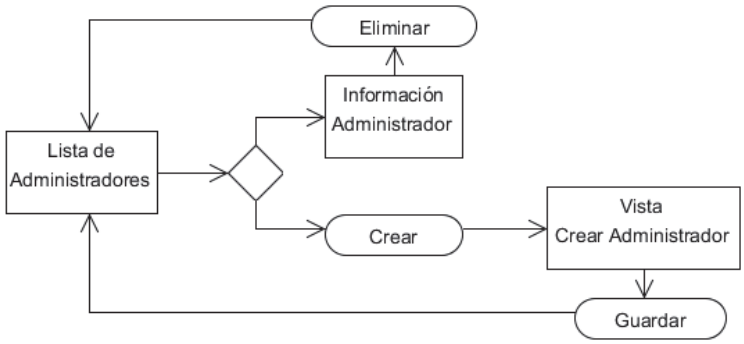

Figura 2.23 Diagrama de Actividad Administradores Terminal

2.2.7.8 Viajes

El diagrama describe las funcionalidades de creación y edición de viajes, además el manejo de las reservaciones y el estado de un viaje.

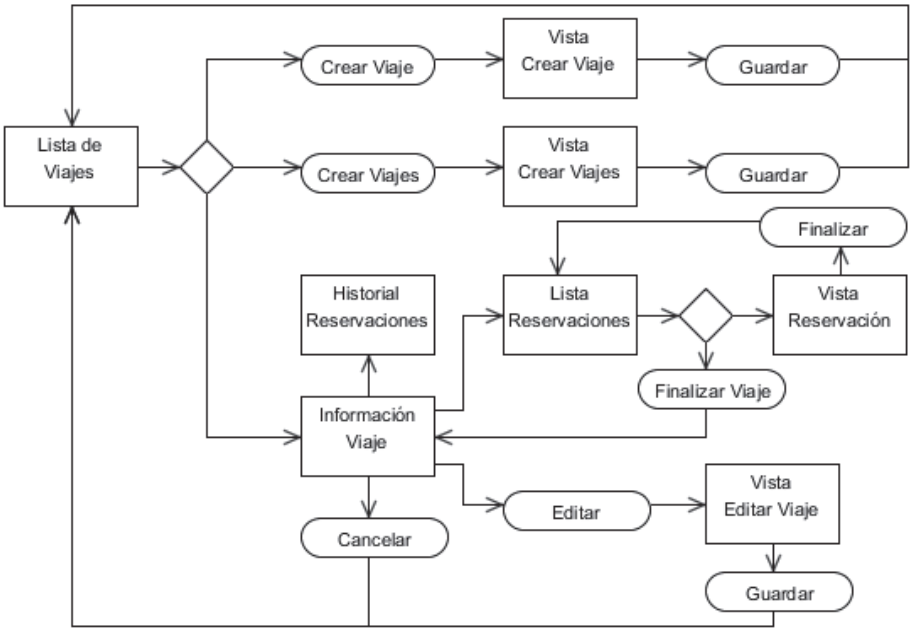

Figura 2.24 Diagrama de Actividad Viajes

2.2.7.9 Ciudades

El diagrama describe las funcionalidades de creación, eliminación y modificación de ciudades en el sistema.

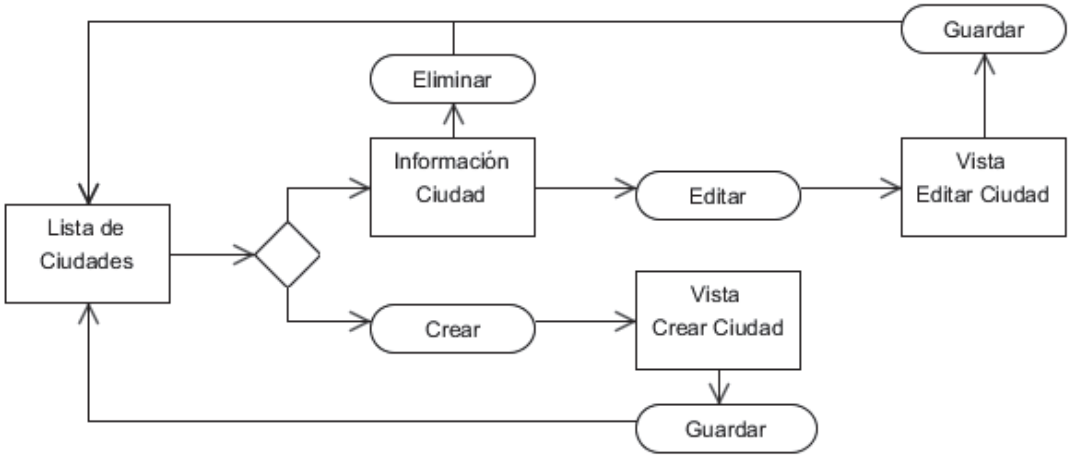

Figura 2.25 Diagrama de Actividad Ciudades

2.2.7.10 Empresa

El diagrama describe las funcionalidades de edición y el acceso a información de la empresa en el sistema.

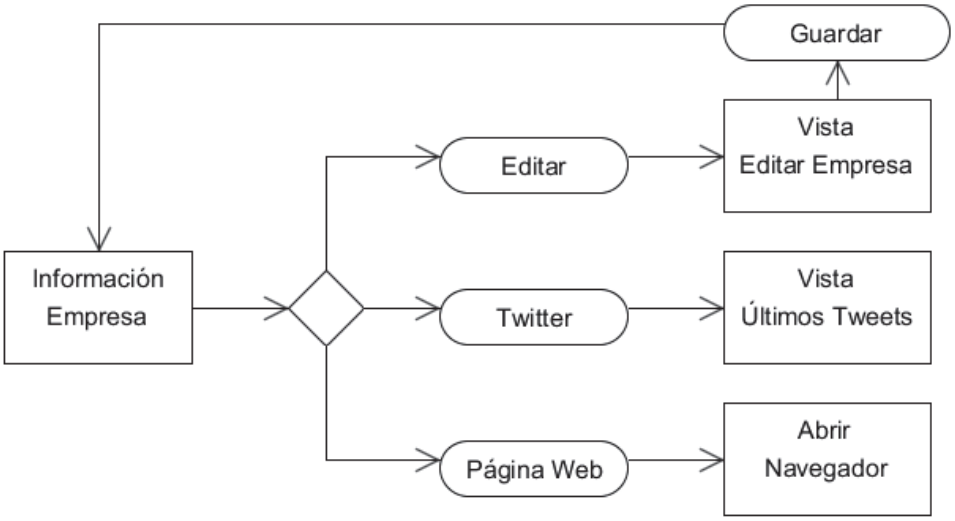

Figura 2.26 Diagrama de Actividad Empresa

2.2.7.11 Terminales

Describe las funcionalidades de creación y edición de los terminales en el sistema, además incluye las funcionalidades que permiten editar boleterías y andenes y la asociación de cooperativas con su respectivo terminal.

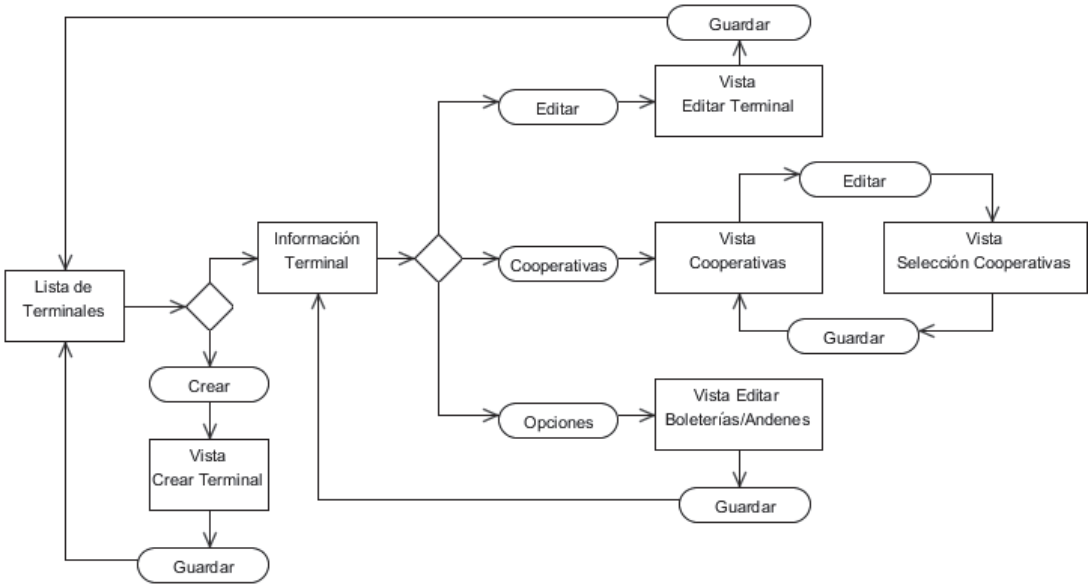

Figura 2.27 Diagrama de Actividad Terminales

2.2.7.12 Reportes

Describe la funcionalidad de los reportes en el sistema.

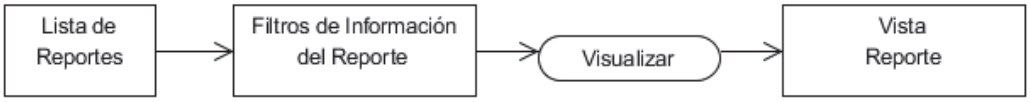

Figura 2.28 Diagrama de Actividad Reportes

2.2.7.13 Cooperativas

Describe las funcionalidades de creación, edición y eliminación de las cooperativas en el sistema, además incluye las funcionalidades que permiten editar los administradores, las frecuencias y las unidades de la cooperativa.

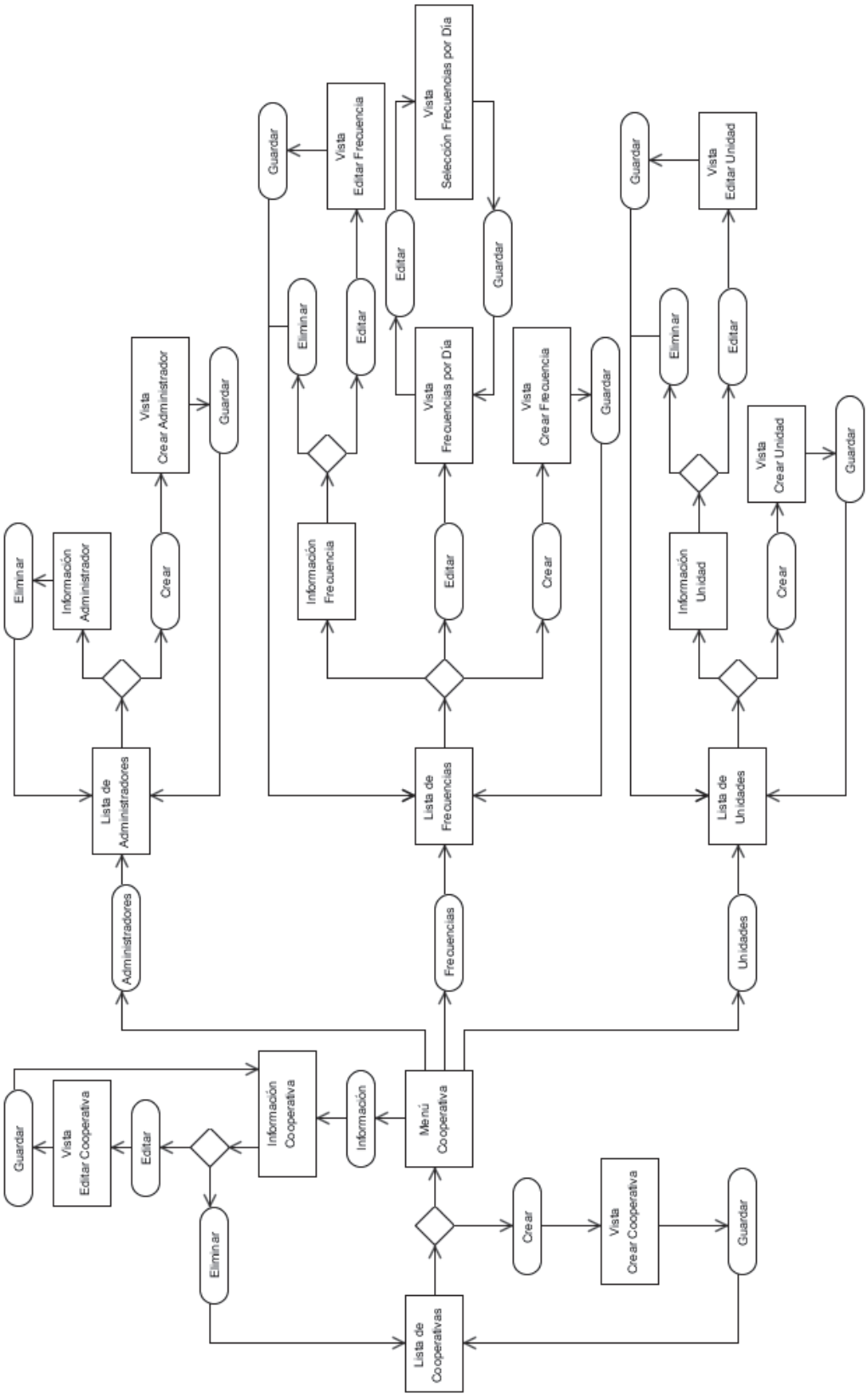

Figura 2.28 Diagrama de Actividad Cooperativas

2.3 DISEÑO DE LAS INTERFACES DE USUARIO. [W44] [W45]

La interfaz de la aplicación es el componente que permite al usuario interactuar con el sistema, está compuesto por distintos elementos visuales para que el usuario acceda y manipule la información.

Las plataformas *Android* y *Windows Phone* proporcionan su propio estilo para construir aplicaciones. La aplicación debe presentar una interfaz y comportamiento similar entre las distintas plataformas. Por tanto, se utiliza elementos similares para que la interfaz visual sea equivalente y no exista gran variación entre plataformas.

Los dispositivos tienen pantallas táctiles, lo cual hace más intuitivo y fácil la navegación al usuario. Por ello, dependiendo del tipo de vista existen elementos que se presionan y permiten navegar hacia otras opciones y otros elementos que serán solo informativos. Además existen las opciones de menú que indican acciones específicas del usuario sobre una entidad o un conjunto de elementos.

En la aplicación existen múltiples vistas, a continuación se resume la descripción y características de vistas comunes que existen para implementar las distintas funcionalidades:

2.3.1 VISTA DE INGRESO

Es la vista inicial del sistema, se conforma por elementos para ingreso de información del usuario: nombre de usuario y contraseña, además cuenta con elementos que permiten navegar a otras opciones de la aplicación como recuperación de contraseña y el ingreso anónimo. Sus características son:

- Muestra un mensaje informativo si existe algún inconveniente en el proceso de ingreso.
- No tiene opciones de menú.

Figura 2.30 Vistas de Ingreso

2.3.2 VISTAS DE SELECCIÓN (LISTAS)

Permiten desplegar un conjunto de elementos con información básica que los identifique. Cada ítem puede representar un punto de acceso a la información detallada de un elemento. Se utilizan para mostrar los elementos de un menú o elementos comunes a una entidad. Sus características son:

- Cada ítem tendrá texto informativo que lo identifique. El texto puede dividirse en: el texto principal, de mayor tamaño que otros elementos y muestra la información principal que identifica al elemento; el texto secundario, de menor tamaño que el texto principal y muestra información adicional o complementaria.
- Cada ítem puede contener elementos de selección, imágenes u otros componentes dependiendo de la necesidad visual que se requiera para llevar a cabo una funcionalidad.
- Un ítem puede permitir el acceso a la información detallada del elemento.
- Dependiendo de la cantidad de ítems que puede contener la lista, se incluyen elementos que permitan filtrar la información al usuario o que permitan desplegar más elementos. Esto ayuda a limitar el uso de recursos para optimizar la aplicación.
- Permite la actualización de los ítems de la lista, esta opción por lo general se aplica a vistas que tienen filtros y paginación.

- Puede tener opciones de menú a través de elementos como la barra de aplicaciones, la barra de acción o paneles de menú dependiendo de la plataforma. Los elementos de menú por lo general son para añadir un nuevo ítem o actualizar la lista.

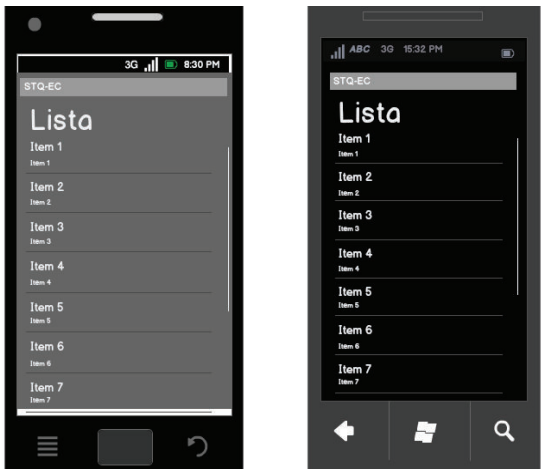

Figura 2.31 Vistas de Selección

2.3.3 VISTAS DE INFORMACIÓN

Presentan información detallada de alguna entidad. Por lo general se accede a estos elementos desde una vista de selección que presenta información básica del objeto. Sus características son:

- Cada atributo que se visualice del objeto se representa por medio de etiquetas.
- En el caso más común existe dos etiquetas: la de menor tamaño va en la parte superior indicando el nombre del atributo; la de mayor tamaño indica el valor de la información del atributo.
- Una etiqueta puede ser el enlace a otra vista o aplicación dependiendo de su información, como puede ser un número telefónico o una dirección.
- Puede tener opciones de menú a través de elementos como la barra de aplicaciones, la barra de acción o paneles de menú dependiendo de la plataforma. Los elementos de menú por lo general son para editar el objeto o eliminar la entidad.

Figura 2.32 Vistas de Información

2.3.4 VISTAS DE EDICIÓN

Permiten editar las propiedades de una entidad. Generalmente se accede a ellas desde vistas informativas del elemento a editar, o cuando se necesita crear un nuevo ítem de a través de vistas de selección. Sus características son:

- Los atributos que se visualice de la entidad se representan por medio de etiquetas y algún tipo de control que permita el ingreso o modificación de datos.
- En el caso de un nuevo registro, los elementos de edición no tienen ningún valor o asumen el valor por defecto, las etiquetas se ubican en la parte superior del elemento indicando el nombre del atributo al cual corresponde.
- En el caso de la edición de un registro, los elementos de edición tienen el valor actual del registro almacenado por el sistema, las etiquetas se ubican en la parte superior del elemento indicando el nombre del atributo al cual corresponde.
- Tiene opciones de menú a través de elementos como la barra de aplicaciones, la barra de acción o paneles de menú dependiendo de la plataforma. Los elementos de menú por lo general son para guardar el objeto que ha sido creado o modificado.

Figura 2.33 Vistas de Edición

2.3.5 CARACTERÍSTICAS COMUNES DE LAS VISTAS

- Existe una barra de progreso que indica que se está realizando la acción ejecutada por el usuario. Este elemento estará en casi todas las vistas y aparece en la mayoría de acciones realizadas por el usuario. De esta forma, el usuario tiene claro que se está procesando su solicitud en la aplicación.
- En el caso de un error en la solicitud por parte del usuario, el sistema mostrará un mensaje en la vista de lo ocurrido y el usuario procede a realizar la acción más conveniente.
- El tema que se aplique a una aplicación se configura para que sea similar en todas las vistas de manera que se presente un aspecto uniforme. Dependiendo de la plataforma el tema se puede configurar en las opciones del sistema y se replica a la aplicación como en *Windows Phone*, o el tema se configura en la aplicación como es el caso de *Android*.
- En las listas que no contengan elementos, el sistema puede mostrar un mensaje indicativo o la lista vacía dependiendo de la plataforma.
- En las opciones de menú, su visualización puede ser diferente dependiendo de la plataforma o de la versión específica de una plataforma. Por tanto las opciones de menú están formadas por un ícono y un texto que describan a la opción, en una opción específica estarán presentes los dos elementos o solo uno de ellos.

2.4 ARQUITECTURA DEL SISTEMA

El sistema se basa en la arquitectura cliente-servidor, a continuación se presentan las características de su diseño.

2.4.1 SERVIDOR

El servidor tiene las siguientes características y componentes:

2.4.1.1 Diseño en Capas

El servidor está constituido por un conjunto de capas, cada capa tiene su función específica dentro de la aplicación. Las capas en las que se divide el servidor son:

2.4.1.1.1 Capa de Servicios

Define y expone los distintos servicios para que sean accesibles a los clientes. Esta capa usa los métodos que le proporciona la capa de información para realizar las operaciones del servicio. También se encarga de definir el formato de los servicios, el identificador para acceder a un servicio y el acceso de los distintos usuarios.

2.4.1.1.2 Capa de Negocio

Realiza las operaciones para procesar la información de la capa de servicios y de la capa de acceso a datos. La información que envía la capa de servicios se cambia al formato adecuado para almacenarse en la base de datos. Obtiene la información requerida en la capa de servicios desde la capa de acceso a datos.

2.4.1.1.3 Capa de Acceso a Datos

Permite acceder y guardar la información necesaria en la aplicación. Utiliza un ORM que se comunica con la base de datos y con la capa de negocio.

2.4.1.2 Servicios REST [W46]

El servidor web expone servicios REST, por tanto para consumir estos servicios se usa el protocolo HTTP. Las aplicaciones cliente pueden consumir los servicios a través del Internet.

Características:

- El dispositivo en el cual se instala la aplicación cliente solo necesita tener una conexión a Internet para interactuar con el sistema.
- Permiten intercambiar la información en distintos formatos como XML y JSON. Esto se aplica tanto en peticiones como en respuestas entre cliente y servidor.
- Una de las tendencias actuales de las compañías es construir API REST que permitan exponer servicios para que sean consumidos de forma sencilla por los clientes.
- Es compatible con diferentes versiones de la plataforma .NET y con cualquier cliente que requiera utilizar los servicios conociendo claramente la definición del servicio.

Otra de las alternativas para construir servicios web es SOAP. Se escoge REST además de las características descritas debido a que en REST el identificador para acceder a un servicio (recurso) es más simple y familiar que SOAP, REST no necesita la cabecera que se define en el protocolo SOAP, los clientes REST necesitan por lo general una biblioteca HTTP para consumir servicios en tanto que SOAP necesita herramientas para consumir servicios.

2.4.1.3 Hospedaje IIS [L11]

La aplicación servidor se aloja en un servidor IIS, de esta forma se utiliza sus características para simplificar la configuración. Su entorno gráfico ayuda a depurar la aplicación mediante Visual Studio. La implementación de la aplicación se realiza mediante el asistente de implementación de aplicaciones.

Características:

- Se utiliza el servidor IIS permitiendo que maneje la comunicación con el cliente, se puede hacer uso de distintas características de un servidor con gran soporte y ampliamente utilizado.
- Permite la incorporación de herramientas que facilitan las tareas de implementación o de monitorización de la aplicación servidor.

Self-Hosting es otra alternativa que existe para hospedar servicios, se refiere a crear algún proceso del CLR que permita exponer los servicios mediante su configuración, inicialización y finalización mediante código.

IIS es la mejor opción debido a que administra la aplicación encargándose de su funcionamiento y de la configuración a través de su entorno gráfico o mediante su API, en *Self-Hosting* se debe crear el código necesario que cumpla con la administración de la aplicación. Por tanto, IIS simplifica el desarrollo y además permite incorporar características propias a la aplicación.

2.4.1.4 Lenguaje y Plataformas de Desarrollo

Se utiliza el lenguaje de programación C#, cumpliendo una de las características principales del proyecto. Se utiliza como plataforma el .NET *Framework*, debido a sus características y además es la principal plataforma para desarrollar sistemas para el lenguaje C# en entornos Microsoft.

2.4.1.5 Tecnologías Utilizadas

2.4.1.5.1 WCF

Para la creación de servicios se utiliza WCF, los cuales permiten cumplir con las restricciones de la arquitectura REST. WCF tiene una configuración bastante simple para el hospedaje de la aplicación en un servidor y el manejo del código de la aplicación por su compatibilidad con ASP .NET.

2.4.1.5.2 LINQ a SQL

LINQ a SQL se utiliza para acceder a la base de datos. Este ORM permite que la capa de acceso se enfoque en las operaciones a través de LINQ, de esta manera se trabaja enteramente con características del lenguaje C# y de la programación orientada a objetos.

2.4.1.6 Herramientas de Desarrollo

2.4.1.6.1 Windows 7 Ultimate

El sistema operativo escogido para el desarrollo es Windows 7 *Ultimate* debido a sus características como la interfaz de usuario, estabilidad, rapidez, etc. Esta edición del sistema operativo soporta los programas y herramientas para el desarrollo de la aplicación, deberá estar actualizado con el *Service Pack* 1. Además algunas herramientas para el desarrollo como IIS vienen incluidas cuando se instala el sistema operativo y solo se necesita habilitarlas.

2.4.1.6.2 SQL Server (Microsoft SQL Server 2008 R2 RTM - Express con Advanced Services) [W47]

Este DBMS permitirá desarrollar el servidor enfocado en C# para el acceso a datos y tiene correspondencia con el sistema de la empresa. Además se considera esta edición, ya que incluye un servidor de informes (SSRS, *SQL Server Reporting Services*) que también se usa para la aplicación.

2.4.1.6.3 IIS 7.5

Es el encargado de exponer los servicios para que puedan ser consumidos por los clientes, permitirá la implementación y depuración de la aplicación servidor. Viene incluido en Windows 7 y permite la inclusión de diferentes herramientas para configuración y administración del servidor y las aplicaciones que hospede.

2.4.1.6.4 Visual Studio 2010 Professional

Permite el desarrollo de la aplicación del servidor, es el principal IDE para desarrollar aplicaciones de la plataforma .NET. Cuenta con los elementos necesarios para desarrollar el servidor y permite añadir nuevos componentes a través de descarga de paquetes, extensiones y plantillas si fueran necesarios.

2.4.1.6.5 Clientes de Servicios

Para ejecutar y comprobar el correcto funcionamiento de los servicios se puede utilizar diversas alternativas:

- Un navegador, permite realizar las peticiones de métodos GET y recibir la respuesta en un determinado formato, es una prueba simple.
- Herramientas que permiten generar solicitudes con los diferentes métodos HTTP y visualizar el detalle de la información que se transmite en el protocolo en las solicitudes y respuestas. Fiddler es una herramienta completa y ampliamente utilizada para estos propósitos.

2.4.2 CLIENTE

El cliente tiene las siguientes características y componentes:

2.4.2.1 Patrón *Monocross*

Las aplicaciones cliente de *Windows Phone* y *Android* se basan en el patrón *Monocross* MVC. Por tanto, la mayoría del código será compartido por ellas y su principal diferencia son las interfaces del usuario que tiene que construirse por separado, aunque deben ser lo más parecido posible. *Monocross* es la mejor opción para el desarrollo de la aplicación debido a las siguientes razones:

- Cumple las restricciones del sistema en cuanto al lenguaje y al soporte de distintas plataformas.

- Está basado en el lenguaje C# y en el *framework* Mono y .Net.
- Tiene un diseño en capas y se basa en un patrón de diseño como MVC.
- Hace uso de varias técnicas para utilizar código compartido.
- Es *open source*, por tanto el código fuente está disponible para realizar cualquier cambio.

No existe otra opción de desarrollo para C# que reúna estas características y a la vez está disponible para su uso sin restricciones.

2.4.2.2 Lenguaje y Plataformas de Desarrollo

Se utiliza el lenguaje de programación C#, cumpliendo una de las características principales del proyecto. Las plataformas de desarrollo son: .NET *Framework*, para el desarrollo de la aplicación de *Windows Phone* y Mono *Framework* para el desarrollo de la aplicación *Android*.

2.4.2.3 Plataformas de la Aplicación

Se utiliza las plataformas *Android* y *Windows Phone* con distintas versiones del sistema operativo para comprobar la validez de la aplicación y mencionar los cambios que se deben tener en cuenta para su correcto funcionamiento.

2.4.2.4 Herramientas Utilizadas

2.4.2.4.1 Visual Studio 2010 Professional

Permite el desarrollo de la aplicación mediante la plataforma .NET para *Windows Phone* y a través de extensiones incluye los componentes de la plataforma *Xamarin* para *Android*.

Xamarin Studio tiene características similares a Visual Studio pero no tiene todas sus funcionalidades, elementos y soporte por lo que será un elemento secundario que ayude a tener otra perspectiva de algunos elementos propios de *Xamarin*.

2.4.2.4.2 Android

Las herramientas para la plataforma que se necesitan son el Java JDK, *Android* SDK, AVD *Manager* y *Xamarin.Android*. Estas herramientas se pueden instalar manualmente, o mediante el instalador de *Xamarin* que proporciona automáticamente la instalación de las diferentes herramientas.

2.4.2.4.3 Windows Phone

Los componentes para el desarrollo están incluidos en la instalación del *Windows Phone* SDK. Por tanto no se necesita una configuración adicional.

2.4.2.4.4 Emuladores Android

Se tiene una amplia gama de emuladores *Android* debido a las numerosas versiones del sistema operativo, también se toma en cuenta la arquitectura del sistema que puede ser ARM o x86. Para ambas arquitecturas se configurar los emuladores mediante el AVD *Manager* y se definen las características necesarias.

Existe también la opción de usar la arquitectura x86 y utilizar las imágenes del sistema operativo y mediante máquinas virtuales emular *Android* sin las restricciones que implica el AVD *Manager*.

2.4.2.4.5 Emuladores Windows Phone

Los emuladores se incluyen cuando se instala el *kit* de desarrollo. Para *Windows Phone* 7 existe un emulador estándar para probar las aplicaciones, por tanto no se necesita configuración especial para definir sus características.

CAPÍTULO III

3 DESARROLLO, PRUEBAS, IMPLEMENTACIÓN Y COSTOS DE LA APLICACIÓN.

En el tercer capítulo se desarrolla la aplicación tomando en cuenta el diseño y funcionamiento que se ha establecido. Se realizan las pruebas de la aplicación sobre las diferentes plataformas y se describe la implementación en cada plataforma así como también de la aplicación del servidor. Finalmente se incluyen los costos relacionados al desarrollo del proyecto.

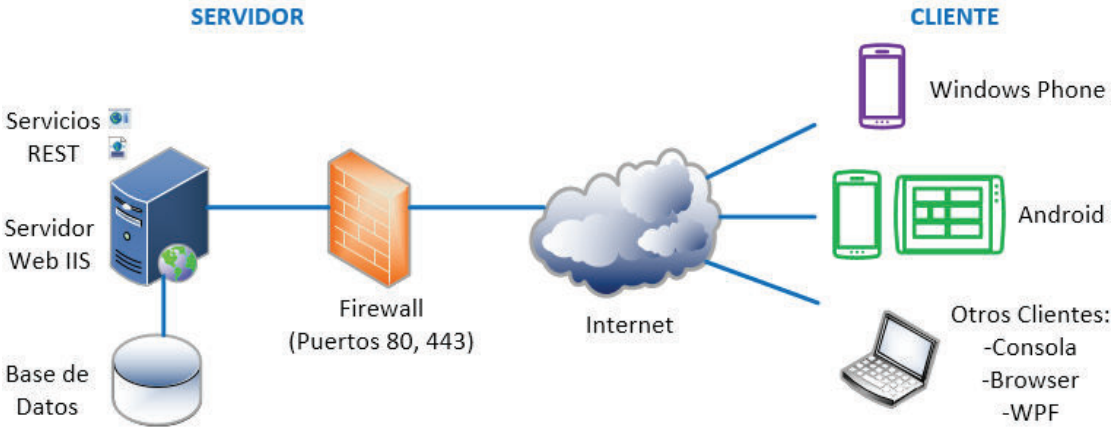

Figura 3.1 Esquema General del Sistema

La figura 3.1 muestra el esquema general de la arquitectura del sistema. En el servidor los aspectos más importantes son la publicación de servicios REST y la base de datos como repositorio de información. En el cliente el aspecto más importante es la aplicación multiplataforma a través de la arquitectura *Monocross*.

Tanto el cliente como el servidor se desarrollan bajo el lenguaje de programación C#. En las siguientes secciones se describe en detalle la estructura de la aplicación cliente y la aplicación servidor y la forma en que se relacionan los distintos componentes que conforman cada aplicación.

3.1 APLICACIÓN SERVIDOR

Para cumplir las especificaciones de diseño el servidor está conformado por varios niveles. Cada nivel se encarga de funciones específicas como el almacenamiento, los reportes y los servicios. Un nivel puede dividirse para obtener una mejor estructura de la aplicación dependiendo de la circunstancia.

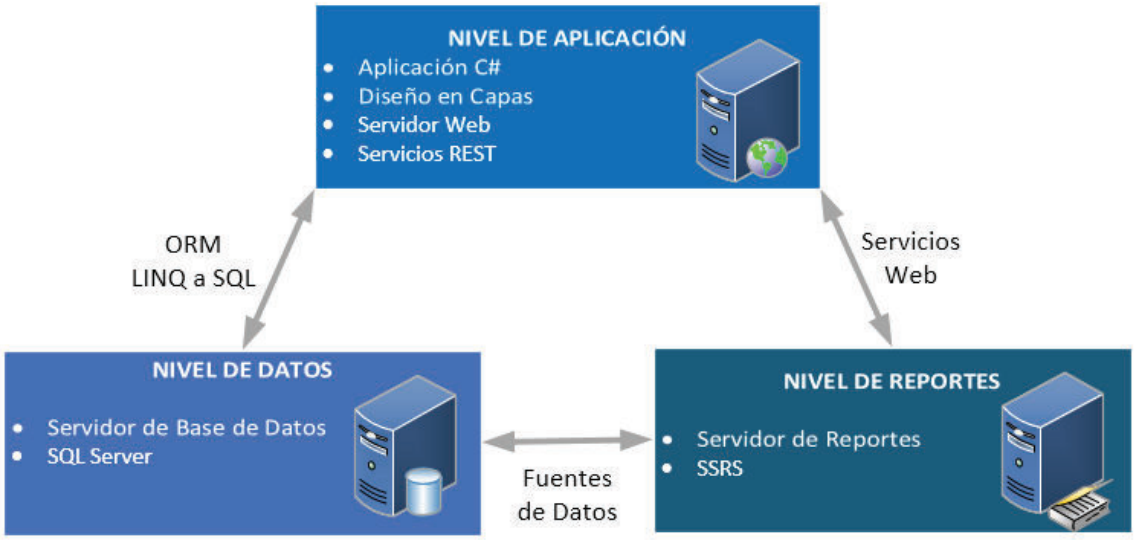

Figura 3.2 Esquema de los niveles de la Aplicación Servidor

La figura 3.2 muestra los niveles del servidor. Cada nivel es un elemento importante que permiten manipular la información y representarla en el formato adecuado para los otros niveles de la aplicación. La comunicación depende de tipo de niveles que interactúen y puede ser mediante el uso de LINQ a SQL, servicios web o mediante fuentes de datos.

3.1.1 NIVEL DE DATOS

En este nivel se encuentra la base de datos que está constituida por un conjunto de tablas que representan las entidades o modelos del negocio. Estas tablas se mapean a las entidades LINQ a SQL para tener un modelo de objetos con el cual trabajar de forma más eficiente. Algunas características importantes de la base de datos son:

- La cantidad de tablas es pequeña, por tanto no es necesario dividir a la base de datos en esquemas, se utiliza un solo esquema.
- Para manejar el control de concurrencia se utiliza en la mayoría de entidades el campo Version, el cual establece si un registro al modificarse ha tenido alguna alteración respecto a los valores obtenidos.
- La clave primaria en la mayoría de entidades es de tipo entero (INT) establecido de forma IDENTITY, de esta manera se generan las claves automáticamente por medio de la base de datos y se tiene uniformidad en el diseño. En el caso de otras entidades, la clave principal se define mediante códigos, éstas tablas por lo general representan elementos fijos que son de configuración en el sistema y no existe crecimiento del número de registros.
- La eliminación de datos se realiza de una manera lógica, es decir un campo en las tablas permite identificar si un registro ha sido eliminado y por tanto no se incluye en las consultas posteriores. En tablas donde la información no es importante se realiza la eliminación del registro.
- Para el control de cambios se manejan dos tablas específicas *AuditRecords* y *AuditRecordsFields*. *AuditRecords* guarda la información del tipo de acción –Creación, Modificación, Eliminación– que realiza el usuario en el sistema, la entidad a la cual pertenece, el identificador del registro modificado, la fecha y usuario que realizó la acción. *AuditRecordsFields* guarda la información del campo modificado, el valor antiguo y el nuevo valor de un registro relacionado a la tabla *AuditRecords*.

3.1.1.1 Diagramas de Base de Datos

Los diagramas permiten tener una clara idea de la estructura de la base de datos. Para una mejor comprensión se ha dividido en varios diagramas el conjunto total de la base y se presenta la información más relevante. En el Anexo A se encuentran todos los diagramas con los elementos de cada entidad.

Los diagramas a continuación presentan la organización de los Terminales, las Cooperativas, los Viajes, las Reservaciones y el control de cambios en el sistema.

Para el manejo de terminales se define la entidad Terminales. Cada terminal tiene sus andenes y boleterías correspondientes. Para relacionar terminales y cooperativas, se tiene una estructura que permite asociar la cooperativa a uno o más terminales. La cooperativa también define andenes y boleterías correspondientes a cada terminal en la cual está funcionando.

La figura 3.3 muestra la organización de las cooperativas y terminales en la estructura de la base de datos para el sistema.

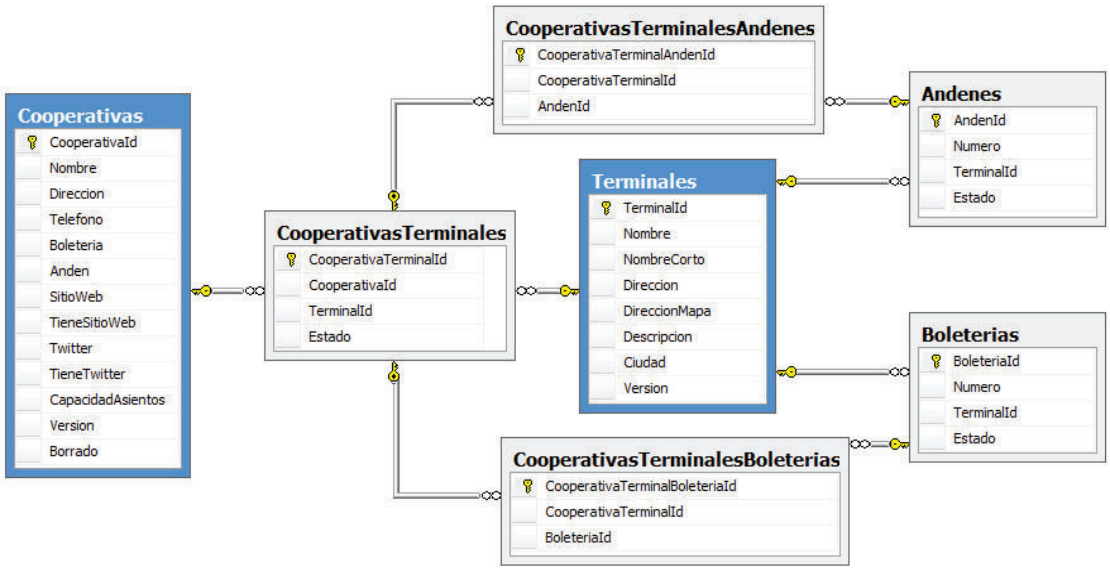

Figura 3.3 Diagrama de Base de Datos Primera Parte

Para manejar la información de una cooperativa se cuenta con la entidad Cooperativas. La tabla Cooperativas se relaciona con varias tablas que permiten organizar la información de elementos como unidades, frecuencias y destinos.

La figura 3.4 muestra la organización de las cooperativas en la estructura de la base de datos para el sistema.

En el caso de la información de los viajes, se conforma por su respectivo estado, la frecuencia, la cual a su vez pertenece a una cooperativa, además se puede asignar una unidad. Las reservaciones y viajes se relacionan por medio de los asientos que puede reservar un usuario, la reservación maneja su propio estado.

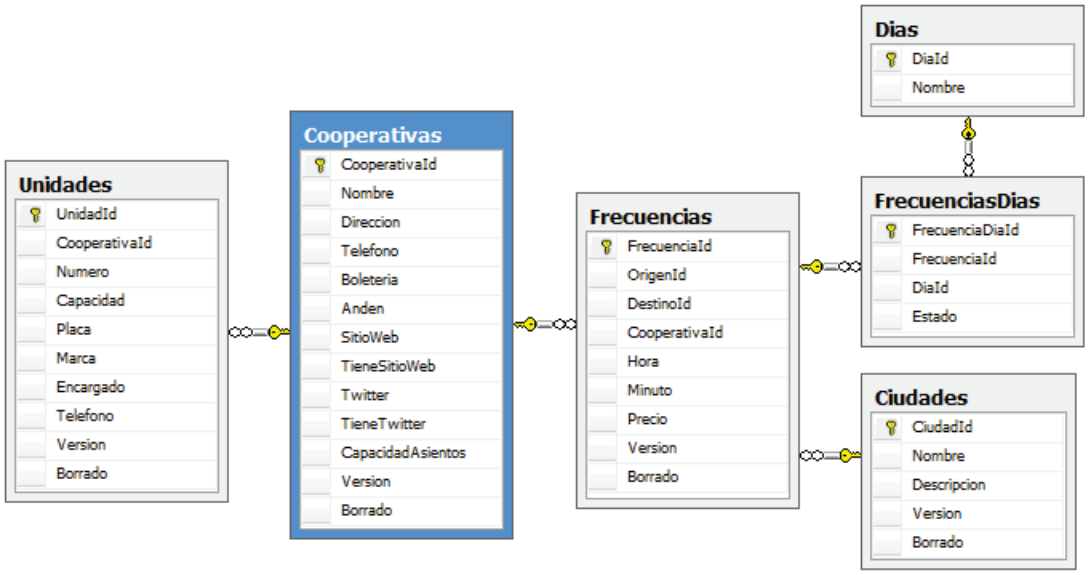

Figura 3.4 Diagrama de Base de Datos Segunda Parte

La figura 3.5 muestra la organización de los viajes y las reservaciones en la estructura de la base de datos para el sistema.

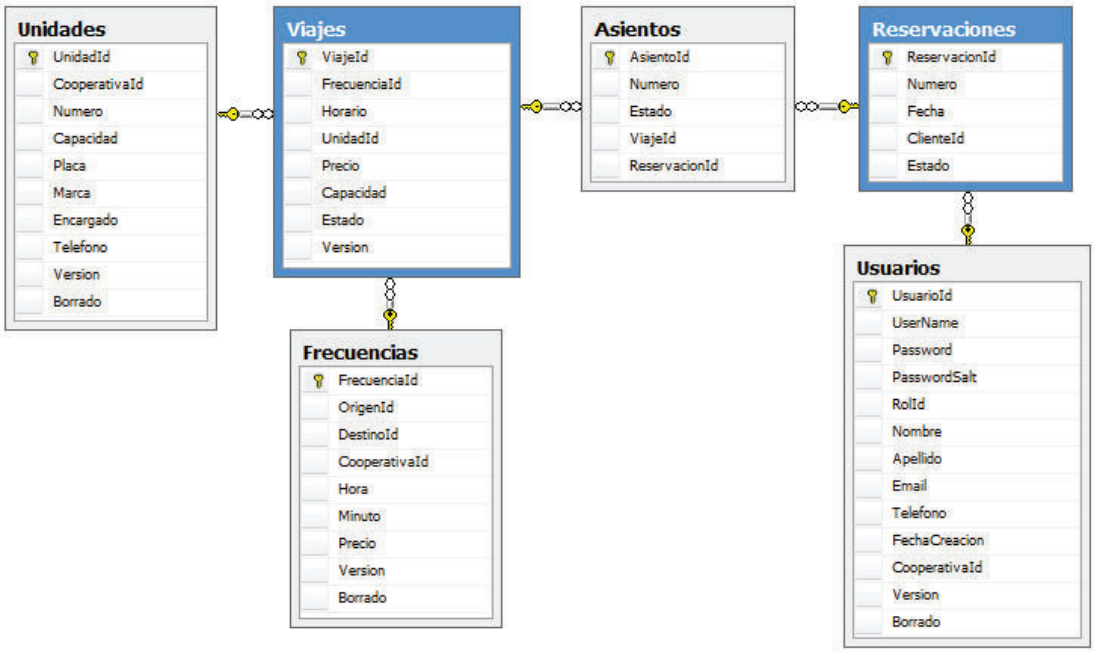

Figura 3.5 Diagrama de Base de Datos Tercera Parte

El manejo de cambios en el sistema se controla mediante dos tablas relacionadas entre sí, para manejar el registro que se cambia y el detalle de los cambios.

La figura 3.6 muestra la estructura de las tablas para los cambios en el sistema.

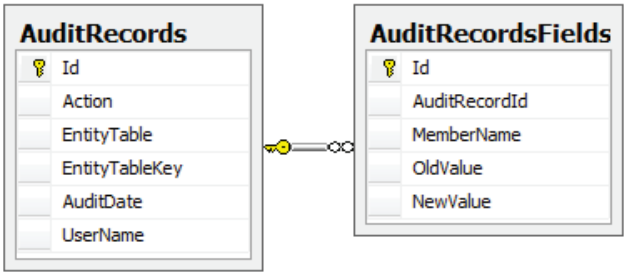

Figura 3.6 Diagrama de Base de Datos Cuarta Parte

3.1.2 NIVEL DE REPORTES

Está constituido por el servidor de reportes (SSRS) y los reportes diseñados mediante *Report Builder*. El acceso al servidor se lo hace mediante servicios web en el nivel de aplicación, de esta forma se especifica el formato requerido.

La figura 3.7 muestra la página principal de la administración del servidor de reportes, se aprecia tres reportes que han sido creados para la utilización en el sistema. También se observa un origen de datos, el cual se usa para establecer la conexión con la base de datos del sistema.

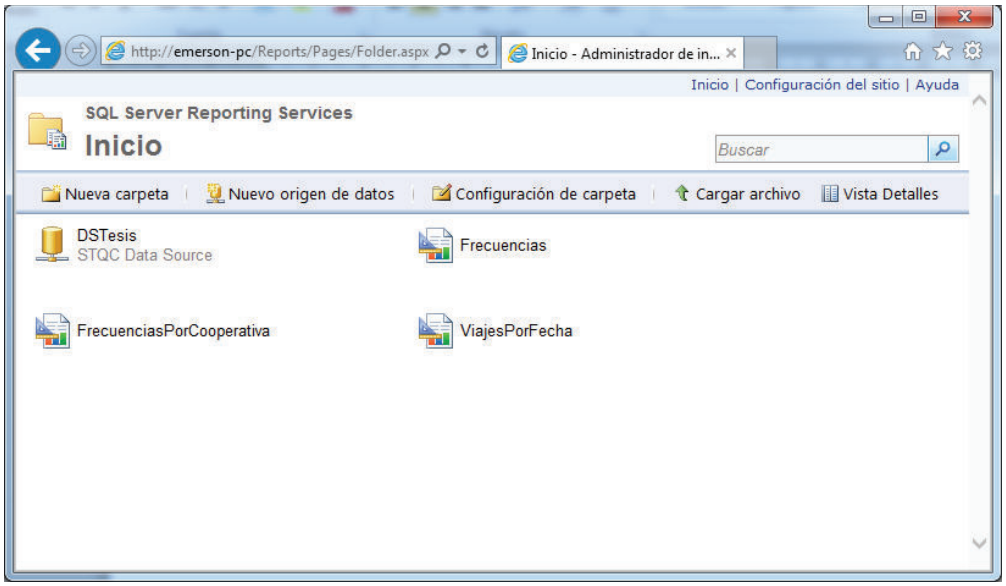

Figura 3.7 Página de Administración del Servidor de Reportes

Los reportes creados obtienen la información de la base de datos del sistema y mediante el diseñador se configuran las distintas consultas realizadas a la base de datos, los parámetros si son necesarios y el aspecto del reporte.

El servidor puede generar los reportes en distintos formatos. Por tanto la definición del formato la realiza el cliente cuando consume el servicio. Luego de diseñar y probar, los reportes se implementan en el servidor y se configuran para el acceso a los usuarios.

3.1.3 NIVEL DE APLICACIÓN

Está constituido por la aplicación del servidor desarrollada en C#. Está conformada por una solución y varios proyectos en Visual Studio. La aplicación se implementa en el servidor para publicar los servicios.

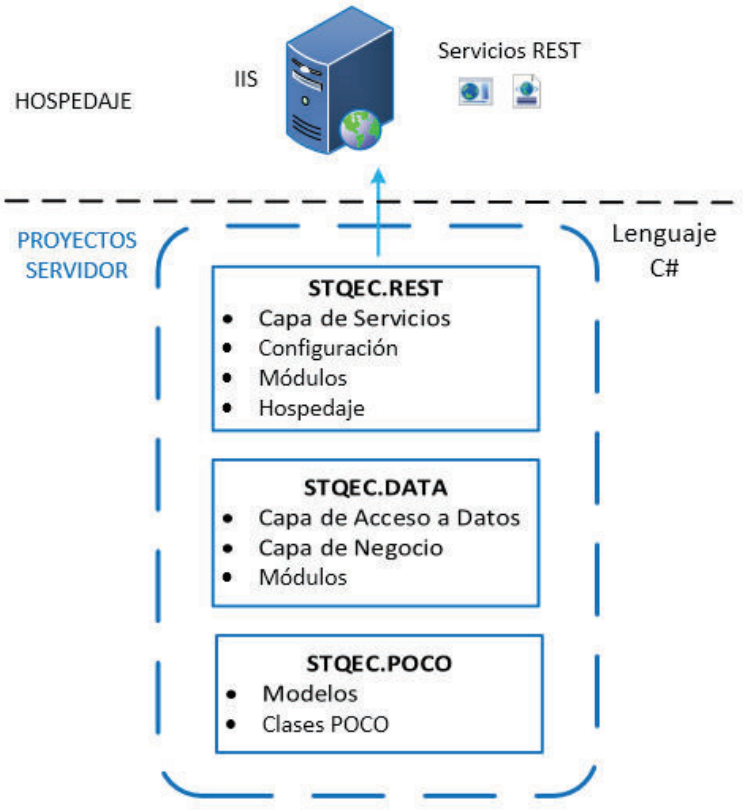

Figura 3.8 Esquema de la Aplicación Servidor

La figura 3.8 muestra el esquema de la aplicación servidor. Existen tres proyectos que conforman la aplicación y cada proyecto tiene elementos y funciones específicas para la construcción de la aplicación.

3.1.3.1 Solución STQEC.*Server*

Es la solución global de la aplicación, que permite agrupar los proyectos de la aplicación de forma estructurada.

3.1.3.2 Proyecto STQEC.POCO

Es un proyecto formado por clases que son utilizadas por otros proyectos y permiten manejar adecuadamente las entidades del sistema.

3.1.3.2.1 Clases POCO (Plain Old CLR Object)

Las clases POCO hacen referencia a clases simples, es decir clases que contienen elementos básicos –por lo general propiedades– y no elementos especializados que generalmente se usan en *frameworks* como en un ORM. Estas clases son importantes y necesarias debido a las siguientes razones:

- Permiten establecer entre las aplicaciones cliente y servidor el mismo modelo de negocios, de esta forma se facilita la comunicación de los mensajes entre cliente y servidor.
- Facilitan la serialización reduciendo la complejidad permitiendo el envío de la información necesaria entre cliente y servidor.
- No dependen de ningún *framework* específico por lo que son compatibles con las distintas versiones de .NET *Framework*, en el caso de trabajar con esta tecnología.

Existen clases POCO muy similares a la representación que realiza el ORM de las entidades en cuanto a los atributos, como la clase ciudad. El fragmento de código 3.1 muestra la definición de una clase POCO conjuntamente con sus atributos.

```
public partial class Ciudad
{
    public int CiudadId { get; set; }
    public string Nombre { get; set; }
    public string Descripcion { get; set; }
    public int Version { get; set; }
}
```

Fragmento de código 3.1 Clase Ciudad

Otras clases no corresponden directamente con las entidades del ORM, en su lugar sirven para manejar información de forma optimizada debido a que transmiten la información necesaria al cliente, generalmente ocurre en la creación o edición de una entidad que tiene dependencias de otras.

```
public partial class Frecuencia
{
    public int FrecuenciaId { get; set; }
    public int OrigenId { get; set; }
    public int DestinoId { get; set; }
    public int CooperativaId { get; set; }
    public string Hora { get; set; }
    public string Minuto { get; set; }
    public decimal Precio { get; set; }
    public Ciudad Origen { get; set; }
    public Ciudad Destino { get; set; }
    public string Dias { get; set; }
    public int Version { get; set; }
}


public class FrecuenciaInformacion
{
    public List<Ciudad> Ciudades { get; set; }
    public List<Ciudad> Terminales { get; set; }
    public List<Unidad> Unidades { get; set; }
}
```

Fragmento de código 3.2 Clases Frecuencia y FrecuenciaInformacion

3.1.3.3 Proyecto STQEC.DATA

Este proyecto está formado por dos capas: la Capa de Acceso a Datos y la Capa de Negocio, además de módulos complementarios que contienen clases y funciones específicas que permiten realizar otras tareas.

3.1.3.3.1 Capa de Acceso a Datos

La capa de acceso a datos se conforma por el ORM a través de LINQ a SQL. Los objetos mapeados a través de la base de datos se representan mediante un diseñador del modelo de clases de LINQ a SQL. Estas clases se crean específicamente para el ORM, por tanto son clases especiales que contienen elementos exclusivos para LINQ a SQL. Estas clases se las denomina entidades para diferenciar de las clases POCO que hace uso el sistema.

En la figura se observa una parte de la representación del modelo de clases para LINQ a SQL a través del diseñador. En el Anexo A se encuentran los diagramas y la información de las entidades generadas por el diseñador.

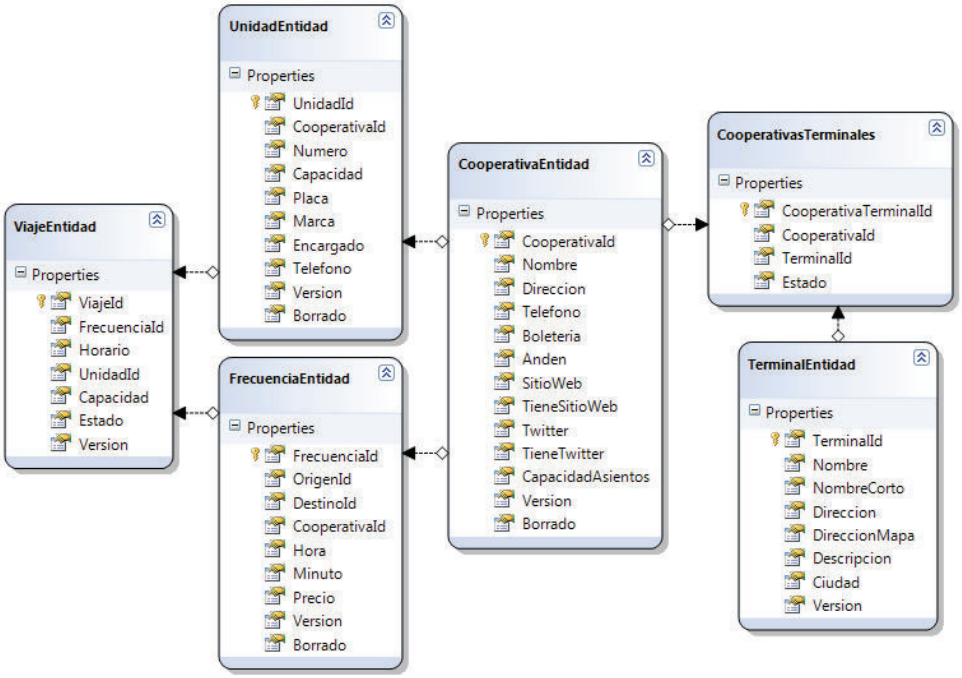

Figura 3.9 Diagrama de Entidades de LINQ a SQL

3.1.3.3.2 Capa de Negocio

La capa de negocio hace uso de las entidades mapeadas por el ORM mediante la instancia de un elemento *DataContext* para realizar las operaciones CRUD (Create, Read, Update, Delete) sobre la base de datos.

En esta capa se definen los distintos métodos de las operaciones que se necesita efectuar por los distintos modelos de negocio. Esta capa también hace uso de los distintos módulos provistos en este nivel.

El siguiente fragmento de código muestra la definición de un método en la Clase BLLCiudad, el cual retorna las ciudades almacenadas en la base de datos.

```
public partial class BLLCiudad
{
    public static List<Ciudad> GetCiudades()
    {
        using (STQECDataContext stqec = new STQECDataContext())
        {
            return (from c in stqec.Ciudades
                    orderby c.Nombre
                    select new Ciudad
                    {
                        CiudadId = c.CiudadId,
                        Nombre = c.Nombre,
                        Descripcion = c.Descripcion
                    }).ToList();
        }
    }
...
}
```

Fragmento de código 3.3 Método GetCiudades

3.1.3.3.3 Módulo de Reportes

Permite la comunicación con el servidor de reportes a través de servicios web que se incluyen en el proyecto mediante referencias web. Por medio de este módulo se establece la conexión con el servidor para la información del reporte y el formato adecuado del reporte que requiere la capa de servicios.

3.1.3.3.4 Módulo de Control de Cambios

Permite almacenar la información de las operaciones de creación, actualización o eliminación de registros en la base de datos. En las clases se definen las entidades y sus propiedades de las cuales se va registrar los cambios. La información se registra en base a la fecha y el usuario que realiza la acción.

3.1.3.3.5 Módulo de Utilidades

Está conformado por distintas clases para realizar las siguientes funciones:

- **Correo:** enviar correos a los usuarios, puede ser el caso de usuarios que se han añadido al sistema o usuarios que necesitan recuperar la información de acceso al sistema.
- **Cifrado:** permite cifrar las claves de los usuarios del sistema, también permite generar claves diferentes para otros usuarios y cifrarlas. Además provee el mecanismo para comprobar la validez de una clave cuando un usuario realiza el ingreso al sistema o el cambio de clave.

3.1.3.4 Proyecto STQEC.REST

Este proyecto permite la publicación de las operaciones del servicio, está formado por la capa de servicios y módulos que permiten realizar otras tareas.

3.1.3.4.1 Capa de Servicios

La capa de servicios incluye todos los servicios o recursos que expone el servidor. Los servicios se dividen de forma lógica por el tipo de modelo de negocio. Además existe una división para diferenciar los servicios de usuarios anónimos de los usuarios que necesitan acceder al sistema mediante la autenticación.

```
public class Global : HttpApplication
{
    void Application_Start(object sender, EventArgs e)
    {
        RegisterRoutes();
    }
    private void RegisterRoutes()
    {
        RouteTable.Routes.Add(new ServiceRoute("", new WebServiceHostFactory(),
        typeof(Service)));
    }
...
}
```

Fragmento de código 3.4 Definición del servicio

El archivo principal en el cual se registran los servicios que se exponen en el servidor es Global.asax y a la vez determina el inicio de la aplicación. El fragmento de código 3.5 muestra el código para registrar un servicio.

Para definir un servicio es necesario añadir una clase con el nombre del servicio y a su vez incorporar los atributos requeridos. En la clase se definen los métodos del servicio, los formatos de requerimiento o respuesta y el formato del URI.

```
[ServiceContract]
[AspNetCompatibilityRequirements(RequirementsMode =
AspNetCompatibilityRequirementsMode.Allowed)]
[ServiceBehavior(InstanceContextMode = InstanceContextMode.PerCall)]
// NOTE: If the service is renamed, remember to update the global.asax.cs file
public partial class Service
{
    [WebGet(UriTemplate = "", ResponseFormat = WebMessageFormat.Json)]
    public string ServiceStatus()
    {
        return "Servicio activo: " + DateTime.Now.ToString();
    }
}
```

Fragmento de código 3.5 Definición de una operación del servicio

Para las operaciones de servicio de cada modelo se definen los métodos específicos que se exponen y que son los encargados de utilizar las operaciones CRUD a través de la capa de negocio. Se utilizan clase parciales para definir las operaciones de cada entidad y facilitar el manejo del código en la aplicación.

```
// GET Methods
[XmlSerializerFormat]
[WebGet(UriTemplate = "ciudades.xml")]
public List<Ciudad> GetCiudades()
{
    return BLLCiudad.GetCiudades();
}
[XmlSerializerFormat]
[WebGet(UriTemplate = "ciudades/{ciudad}.xml")]
public Ciudad GetCiudad(string ciudad)
{
    return BLLCiudad.GetCiudad(Int32.Parse(ciudad));
}
...
```

Fragmento de código 3.6 Operaciones de servicio de Ciudad

El servicio proporciona una página de ayuda si se configura para que sea visible, la cual muestra el URI del servicio, el método y una descripción que es un valor configurable. La página se puede visualizar a través del cualquier navegador.

Figura 3.10 Página de ayuda con varias operaciones de servicio

3.1.3.4.2 Archivo de Configuración

El archivo Web.config es un archivo XML que permite realizar configuraciones sobre el servicio que se publica. Puede definir variables que serán utilizadas por la aplicación, algunas configuraciones del hospedaje que se usan en el IIS y la definición de los módulos que usa la aplicación. Es un archivo que se puede configurar después del despliegue de la aplicación en el servidor.

3.1.3.4.3 Módulo de Autenticación

Es el encargado de realizar la autenticación a los usuarios mediante la verificación del nombre de usuario y contraseña. Este módulo es el primero que se ejecuta y permitirá al cliente obtener la información necesaria para su acceso a los diferentes servicios del sistema.

3.1.3.4.4 Operaciones de Servicio

Una vez publicado adecuadamente el servicio en el servidor web, se puede consumir una determinada operación, como es el caso de peticiones *GET* mediante un navegador web.

Figura 3.11 Resultado de una operación del servicio

Otra forma de consumir una operación es haciendo uso de Fiddler. Esta herramienta permite construir peticiones web para los distintos métodos existentes. También permite analizar la estructura de cada una de las peticiones y respuestas.

La figura 3.12 muestra una petición *GET* y el resultado obtenido. El formato de respuesta se ha establecido en XML.

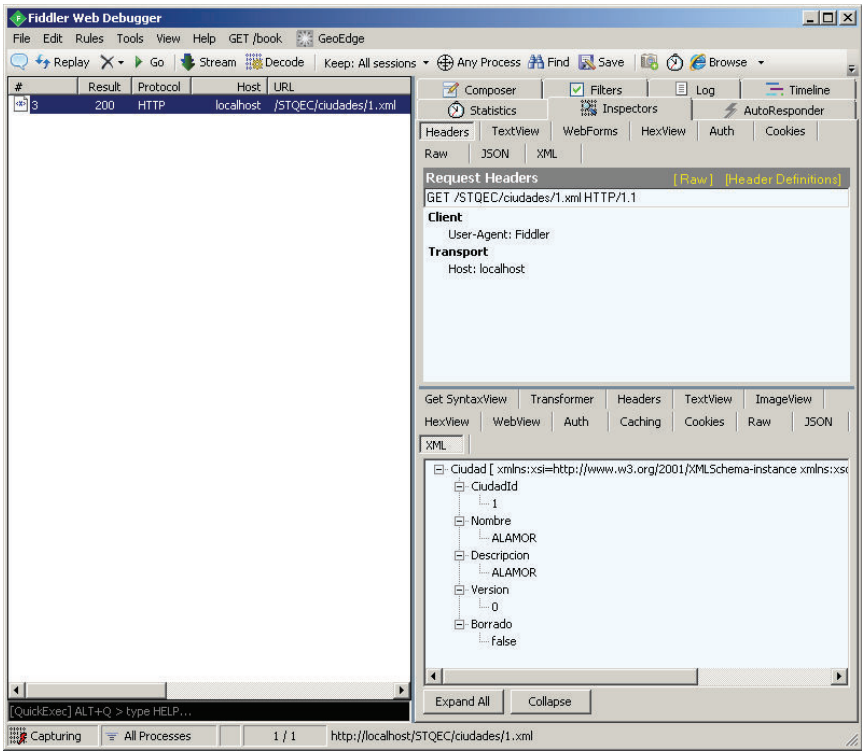

Figura 3.12 Petición de una operación en Fiddler

3.2 APLICACIONES CLIENTE

Para cumplir las especificaciones de diseño, las aplicaciones de cliente están estructuradas por diferentes proyectos. Cada proyecto se encarga de funciones específicas, existen proyectos comunes a las aplicaciones y proyectos específicos que requiere cada plataforma.

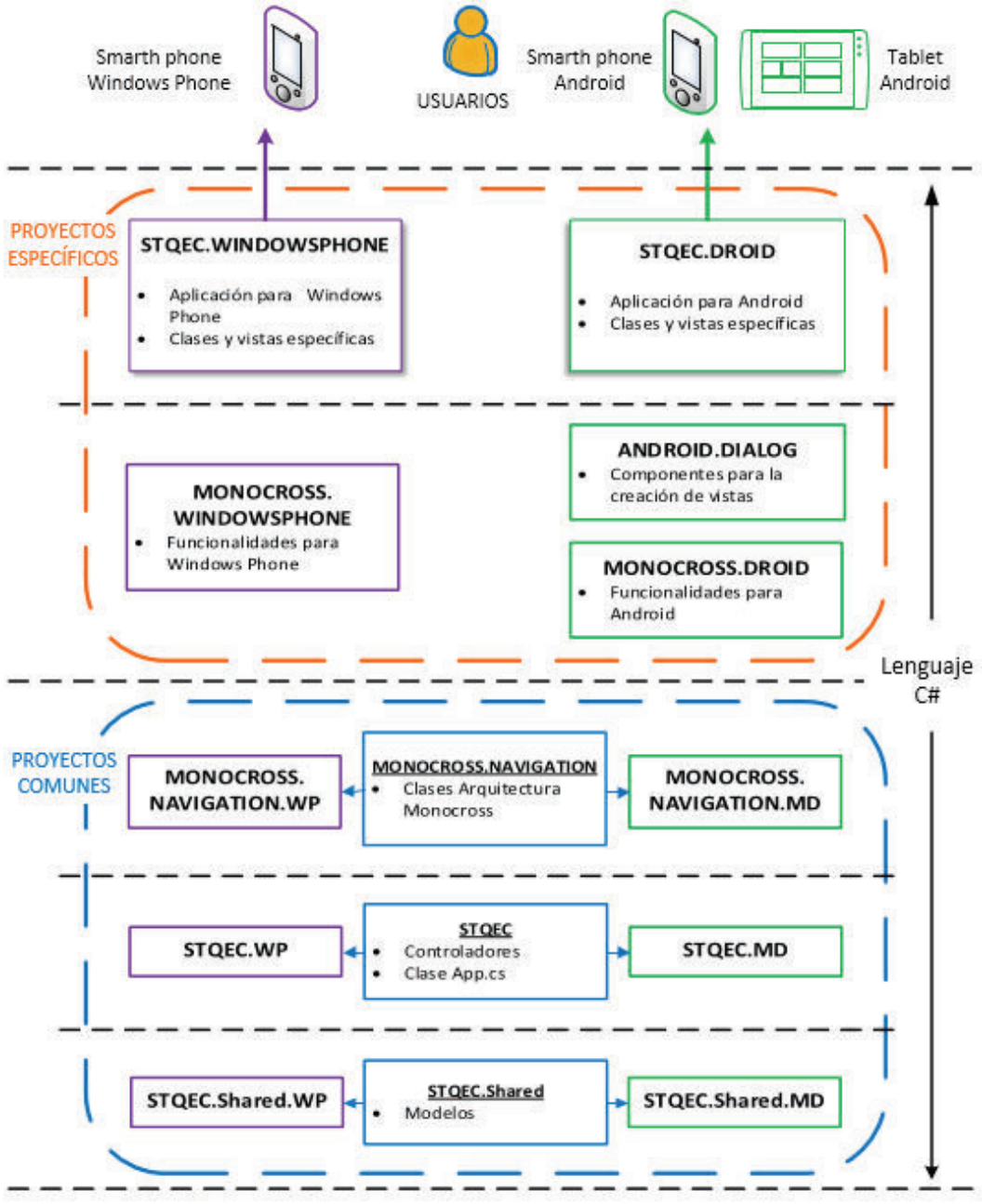

Figura 3.13 Esquema de las Aplicaciones Cliente

La figura 3.13 muestra el esquema de las aplicaciones cliente. Existen tres proyectos comunes, los cuales utilizan las mismas clases en su estructura, pero generan el ensamblado correspondiente para una plataforma específica. Los proyectos específicos son exclusivos para cada plataforma, los cuales definen los elementos necesarios para generar la aplicación final que es utilizada por los usuarios a través de distintos dispositivos.

En la siguiente sección se describen las características y los elementos de cada proyecto utilizados para cada aplicación.

3.2.1 PROYECTOS COMUNES

Los proyectos comunes a las aplicaciones están conformados por los mismos archivos, ya que corresponde al código compartido utilizado por las aplicaciones, diferenciándose por el tipo de proyecto para generar el ensamblado para la plataforma específica. Los proyectos de este tipo son: STQEC.*Shared*.*, STQEC.* y *MonoCross*.*Navigation*.*.

3.2.1.1 Proyectos STQEC.*Shared*.WP y STQEC.*Shared*.MD

Estos proyectos contienen los Modelos dentro en la estructura MVC de cada aplicación. Las clases de este proyecto corresponden los distintos modelos del negocio que usa el sistema, contienen la información que se transmite entre cliente y servidor en el formato que se establezca para la comunicación.

Las clases son POCO, por tanto deberán tener exactamente los mismos componentes las clases del cliente y el servidor, o pueden existir diferencias en las clases ya que pueden existir elementos adicionales que se requieran para interactuar con la interfaz de usuario adecuadamente.

El fragmento de código 3.7 muestra la clase Frecuencia. Existen elementos adicionales con respecto a la clase Frecuencia definida en el fragmento de código 3.2 del servidor. Estos elementos se visualizan en las pantallas de la aplicación.

```
public class Frecuencia
{
    public int FrecuenciaId { get; set; }
    public int OrigenId { get; set; }
    public int DestinoId { get; set; }
    public string CooperativaId { get; set; }
    public string Minuto { get; set; }
    public decimal Precio { get; set; }
    public Ciudad Origen { get; set; }
    public Ciudad Destino { get; set; }
    public string Dias { get; set; }
    public string Hora { get; set; }

    public string Horario { get { return String.Format("{0} : {1}", Hora, Minuto);}}
    public string Ruta { get { return String.Format("{0} - {1}", Origen.Nombre,
       Destino.Nombre); }}
}
```

Fragmento de código 3.7 Clase Frecuencia

3.2.1.2 Proyectos STQEC.WP y STQEC.MD

Estos proyectos contienen los Controladores dentro de la estructura MVC de cada aplicación. Las clases de este proyecto manipulan los modelos de negocio según su tipo de modelo y permiten definir las operaciones del modelo.

La aplicación se configura a través del archivo App.cs. En esta clase se define el título de la aplicación, se registra los controladores y sus identificadores de navegación y el controlador de inicial al momento de cargar la aplicación.

```
public class App : MXApplication
{
    public override void OnAppLoad()
    {
        // Application title
        Title = "STQ-EC";
        // administradores cooperativa
        AdministradorCooperativaController administradorCooperativaController = new
        AdministradorCooperativaController();
        NavigationMap.Add("AdministradorCooperativa/{UsuarioId}",
        administradorCooperativaController);
        NavigationMap.Add("AdministradorCooperativa/{UsuarioId}/{Action}",
        administradorCooperativaController);
...
        NavigateOnLoad = "Login/GET";
    }
}
```

Fragmento de código 3.8 Clase App

El fragmento de código 3.8 muestra parte del código de la clase App.cs en la cual se observa la definición de los controladores de tipo AdministradorCooperativa y como se relacionan con los identificadores para su utilización.

En el controlador de ciudad, CiudadController, se tiene como parámetro un objeto de tipo Ciudad, en el método *Load* se realiza la operación que corresponda a los parámetros que se envían, como el identificador de la ciudad o la acción.

```
public class CiudadController : MXController<Ciudad>
{
    public override string Load(Dictionary<string, string> parameters, int
    numeracion)
    {
        string perspective = ViewPerspective.Default;

        string ciudadId;
        parameters.TryGetValue("CiudadId", out ciudadId);

        string action;
        if (!parameters.TryGetValue("Action", out action))
        {
            action = "GET";
        }

        switch (action)
        {
            case "EDIT":
            case "GET":
                    ...
                    Model = new Ciudad();
                    perspective = ViewPerspective.Update;
                    ...
                break;

            case "DELETE":
                    ...
                    if (DeleteCiudad(ciudadId))
                    ...

            case "CREATE":
                    ...

            case "UPDATE":
                    ...
        }

        return perspective;
    }
...
}
```

Fragmento de código 3.9 Clase CiudadController

El controlador también define los métodos que se ejecutan en cada acción. Los métodos son los que consumen los servicios mediante peticiones HTTP y devuelven los resultados según la acción ejecutada.

En el fragmento de código 3.10 se muestra dos de los cuatro métodos que utiliza el controlador de la ciudad para consumir servicios. Cada petición consume la operación de servicio según el identificador correspondiente.

En el método GetCiudad se obtiene del servicio la respuesta en formato XML y se devuelve la ciudad como resultado del método. En el método UpdateCiudad se envía una ciudad a la operación del servicio para ser actualizada y se retorna un valor verdadero si la petición se realizó sin inconvenientes.

```
public static Ciudad GetCiudad(string ciudadId)
{
    string urlCiudades = string.Format("http://server/STQEC/ciudades/{0}.xml",
    ciudadId).GetServidor();

    HttpWebRequest request = (HttpWebRequest)HttpWebRequest.Create(urlCiudades);

    using (StreamReader reader = new
    StreamReader(request.GetResponse().GetResponseStream(), true))
    {
        XmlSerializer serializer = new XmlSerializer(typeof(Ciudad));
        return (Ciudad)serializer.Deserialize(reader);
    }
}

public static bool UpdateCiudad(Ciudad ciudad)
{
    string urlCiudades = "http://server/STQEC/ciudades/ciudad.xml";

    HttpWebRequest request =
    (HttpWebRequest)HttpWebRequest.Create(urlCiudades.GetServidor());
    request.Method = "PUT";
    request.ContentType = "application/xml";

    using (Stream dataStream = request.GetRequestStream())
    {
        XmlSerializer serializer = new XmlSerializer(typeof(Ciudad));
        serializer.Serialize(dataStream, ciudad);
    }

    request.GetResponse();
    return true;
}
```

Fragmento de código 3.10 Métodos de la clase CiudadController

3.2.1.3 Proyectos *MonoCross.Navigation*.WP y *MonoCross.Navigation*.MD

Estos proyectos contienen los elementos generales del *framework* que se usan por otros proyectos en alguna implementación específica. Entre ellos están las definiciones de la aplicación, los controladores, las vistas, los contenedores, la navegación y las perspectivas.

Las clases que aquí se manejan permiten la implementación del modelo *Monocross* MVC y brindan al desarrollador la abstracción necesaria para facilitar la construcción de la aplicación.

3.2.2 PROYECTOS ESPECÍFICOS

Los proyectos específicos de cada plataforma son implementaciones de elementos genéricos o también proyectos que generan el archivo de instalación de cada plataforma. Contienen código que no puede ser compartido. Los proyectos de este tipo son: STQEC.*, *MonoCross*.* y *Android*.*Dialog*.

3.2.2.1 Proyectos *MonoCross.Droid* y *MonoCross.WindowsPhone*

Estos proyectos contienen implementaciones específicas para cada plataforma requerida por el *framework*. Por tanto, no contienen los mismos elementos y el código fuente es diferente. Los proyectos se basan en las clases definidas en los proyectos *MonoCross.Navigation.Droid* y *MonoCross.Navigation*.WP e implementan las funcionalidades de acuerdo a la plataforma.

3.2.2.2 Proyecto *Android*.Dialog

Es un proyecto específico de la plataforma *Android* que define elementos visuales que serán utilizados para la interfaz gráfica del usuario. El modelo *Monocross* hace uso de estos elementos como forma principal para construir las vistas de la aplicación.

3.2.2.3 Proyecto STQEC.*WindowsPhone*

Proyecto específico de *Windows Phone*. Es el encargado de generar la aplicación final para esta plataforma. Está estructurado por varios elementos que permiten la configuración de la aplicación, las vistas de la aplicación, íconos e imágenes.

La clase principal es del proyecto es la clase App. Esta clase permite inicializar la aplicación, añade al contenedor las diferentes vistas creadas y las asocia al modelo específico y define también la perspectiva relacionada, maneja las excepciones generadas por la aplicación a nivel de controladores, define el punto inicial de la aplicación y realiza otras funcionalidades requeridas para la aplicación.

El fragmento de código 3.11 muestra parte del código de la clase App en la cual se definen los principales elementos para el funcionamiento de la aplicación.

```
public partial class App : Application
{
   ...
   public App()
   {
       ...
       // initialize app
       MXPhoneContainer.Initialize(new STQEC.App(), RootFrame);

       // administrador cooperativa
       MXPhoneContainer.AddView<AdministradoresPorCooperativa>(typeof(
       AdministradorCooperativaListView), ViewPerspective.Default);
       MXPhoneContainer.AddView<AdministradorCooperativa>(typeof(
       AdministradorCooperativaView), ViewPerspective.Default);
       MXPhoneContainer.AddView<AdministradorCooperativa>(typeof(
       AdministradorCooperativaEditView), ViewPerspective.Update);
       ...
       // excepciones
       new    MXPhoneContainer.ControllerExcepcionEventHandler(
       MXPhoneContainer_ExcepcionController);
       ...
   }
   ...
   private void Application_Launching(object sender, LaunchingEventArgs e)
   {
       MXPhoneContainer.Navigate(null, MXContainer.Instance.App.NavigateOnLoad);
   }
   ...
}
```

Fragmento de código 3.11 Clase App

Las interfaces de la aplicación se definen en este proyecto y se crean en el diseñador como se muestra en la figura 3.14, que indica el interfaz y su código.

Figura 3.14 Visualización del código e interfaz de una vista en *Windows Phone*

3.2.2.4 Proyecto STQEC.*Droid*

Proyecto específico de *Android*. Es el encargado de generar la aplicación final para esta plataforma. Está estructurado por varios elementos que permiten la configuración de la aplicación, y también por las vistas de la aplicación, íconos, la definición de cada menú para las vistas e imágenes.

La clase principal es del proyecto es *MainActivity*. Esta clase maneja los fragmentos de la aplicación y la configuración básica para la navegación entre fragmentos, permite inicializar la aplicación, añade al contenedor las diferentes vistas creadas y las asocia al modelo específico y define también la perspectiva relacionada, maneja las excepciones generadas por la aplicación a nivel de controladores, define el punto inicial de la aplicación y realiza otras funcionalidades requeridas para la aplicación.

El fragmento de código 3.12 muestra parte del código de la clase *MainActivity* en la cual se definen los principales elementos del funcionamiento de la aplicación.

```
public class MainActivity : FragmentActivity
{
    protected override void OnCreate(Bundle savedInstanceState)
    {
        base.OnCreate(savedInstanceState);
        SetContentView(Resource.Layout.Main);
        ...
        // initialize app
        MXDroidContainer.Initialize(new App(), ApplicationContext);
        MXDroidContainer.NavigationHandler = FragmentHandler;

        // administrador cooperativa
        MXContainer.AddView<AdministradoresPorCooperativa>(typeof(
        AdministradorCooperativaListView), ViewPerspective.Default);
        MXContainer.AddView<AdministradorCooperativa>(typeof(
        AdministradorCooperativaView), ViewPerspective.Default);
        MXContainer.AddView<AdministradorCooperativa>(typeof(
        AdministradorCooperativaEditView), ViewPerspective.Update);
        ...
        // excepciones
        MXDroidContainer.ExcepcionController += new
        MXDroidContainer.ControllerExcepcionEventHandler(
        MXDroidContainer_ExcepcionController);
        ...
    }
}
```

Fragmento de código 3.12 Clase *MainActivity*

Las interfaces de la aplicación se definen en este proyecto y se crean en forma de recursos mediante el diseñador como se muestra en la figura 3.15, en la cual se representa el interfaz y su código relacionado.

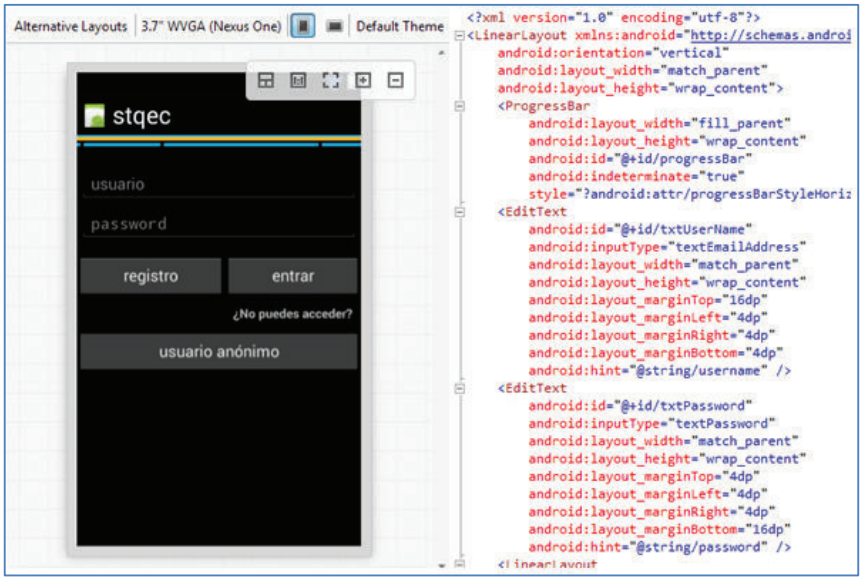

Figura 3.15 Visualización del código e interfaz de una vista en *Android*

3.3 PRUEBAS

Las pruebas verifican el funcionamiento de la aplicación. Permiten comprobar que las diferentes funcionalidades se ejecuten en los dispositivos correctamente. Se dividen en pruebas de la aplicación y en pruebas de carga.

Se analiza distintos casos de prueba, los cuales detallan las condiciones, la ejecución y los resultados obtenidos en una prueba específica. Posteriormente se indica los aspectos más relevantes de la prueba mediante imágenes.

3.3.1 PRUEBAS DE LA APLICACIÓN

Son pruebas generales que verifican el funcionamiento tanto del cliente y el servidor, los casos de prueba reflejan la interacción entre cliente y servidor.

En el caso del cliente, se muestra mediante capturas de pantalla de la aplicación en los dispositivos. En el caso del servidor, se muestra los cambios que han tenido efecto en la base de datos o los resultados de un determinado evento.

3.3.1.1 Visualizar Aplicación

Caso de Prueba	
Nombre	Visualizar aplicación
Descripción	Visualización de la aplicación en diferentes dispositivos.
Condiciones	La aplicación se ha instalado en el sistema.
Ejecución	• Ingresar a la ubicación de las aplicaciones en los dispositivos *Android* y *Windows Phone*.
Resultado Esperado	• Visualizar el logo de la aplicación entre las demás aplicaciones de cada plataforma, acoplado según el sistema.
Conclusión	Cumple con el resultado, se visualiza el logo en diferentes dispositivos y acoplado de acuerdo a la plataforma.

Tabla 3.1 Caso de Prueba Visualizar Aplicación

La siguiente figura muestra la visualización de la aplicación en dispositivos *Android*, en la opción de las aplicaciones.

Figura 3.16 Caso de Prueba Visualizar Aplicación Primera Parte

La siguiente figura muestra la visualización de la aplicación en dispositivos *Windows Phone* en los diferentes menús de aplicaciones. El ícono se acopla al tema establecido por el usuario.

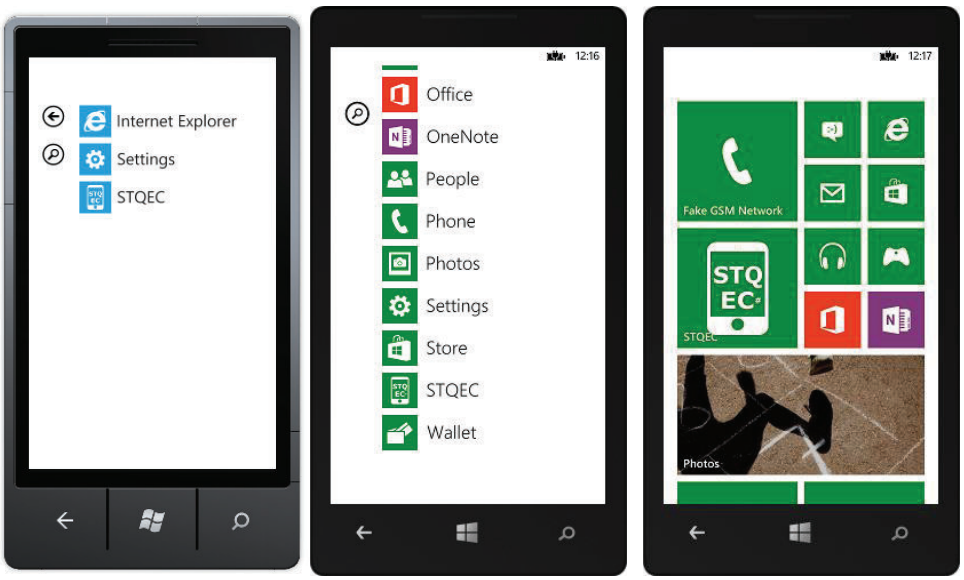

Figura 3.17 Caso de Prueba Visualizar Aplicación Segunda Parte

3.3.1.2 Iniciar Sesión

Caso de Prueba	
Nombre	Iniciar Sesión
Descripción	Acceso al sistema por parte de los diferentes usuarios al sistema.
Condiciones	Debe existir usuarios de tipo Administrador Terminal, Administrador Cooperativa y Usuario Normal registrados en el sistema.
Ejecución	• Ingresar a la aplicación desde la ubicación de las aplicaciones dependiendo del dispositivo móvil. • En caso de usuarios de tipo Administrador Terminal, Administrador Cooperativa y Usuario Normal se debe ingresar la clave y contraseña. • El usuario presiona la opción Entrar para el acceso. Usuario Anónimo ingresa desde la pantalla inicial.
Resultado Esperado	• Visualizar las opciones del menú correspondiente al tipo usuario con el que se ingresó en el sistema. • Si se ingresa incorrectamente los datos de usuario y contraseña, visualizar un mensaje indicativo.
Conclusión	Cumple con el resultado, los usuarios ingresaron desde distintos terminales al sistema.

Tabla 3.2 Caso de Prueba Iniciar Sesión

Se visualiza la pantalla inicial de la aplicación en la plataforma *Android*.

Figura 3.18 Caso de Prueba Iniciar Sesión Primera Parte

Se visualiza la pantalla inicial de la aplicación en la plataforma *Windows Phone*.

Figura 3.19 Caso de Prueba Iniciar Sesión Segunda Parte

Se visualiza el menú de usuario para usuarios de tipo Administrador de Terminal y Administrador de Cooperativa.

Figura 3.20 Caso de Prueba Iniciar Sesión Tercera Parte

Se visualiza el menú de usuario para usuarios de tipo Anónimo y Usuario Normal.

Figura 3.21 Caso de Prueba Iniciar Sesión Cuarta Parte

3.3.1.3 Opciones Informativas

Caso de Prueba	
Nombre	Opciones Informativas
Descripción	Visualización de elementos informativos que son comunes a usuarios al sistema.
Condiciones	Los usuarios de tipo Administrador Terminal, Administrador Cooperativa y Usuario Normal han ingresado en el sistema. Usuario Anónimo directamente desde las opciones del menú.
Ejecución	• Seleccionar Empresa en el menú de opciones para visualizar información de la empresa. • Seleccionar el twitter, visualizar los últimos tweets de la empresa. Seleccionar la página web, redirección para visualización en un navegador. • Seleccionar Terminales en el menú de opciones. • Visualizar la lista de Terminales, seleccionar un Terminal para ver la información específica. • Interactuar con elementos como la dirección para visualizar el mapa en otra aplicación.
Resultado Esperado	• Visualizar datos informativos de la empresa y los terminales. • Interacción con información de twitter o google maps.
Conclusión	Cumple con el resultado, se visualiza la información y se redirige a aplicaciones externas dependiendo del caso.

Tabla 3.3 Caso de Prueba Opciones Informativas

La siguiente figura muestra el acceso a la información de la empresa, la información del twitter y la página web.

Figura 3.22 Caso de Prueba Opciones Informativas Primera Parte

La siguiente figura muestra el acceso información de los terminales y al mapa de la dirección en el navegador.

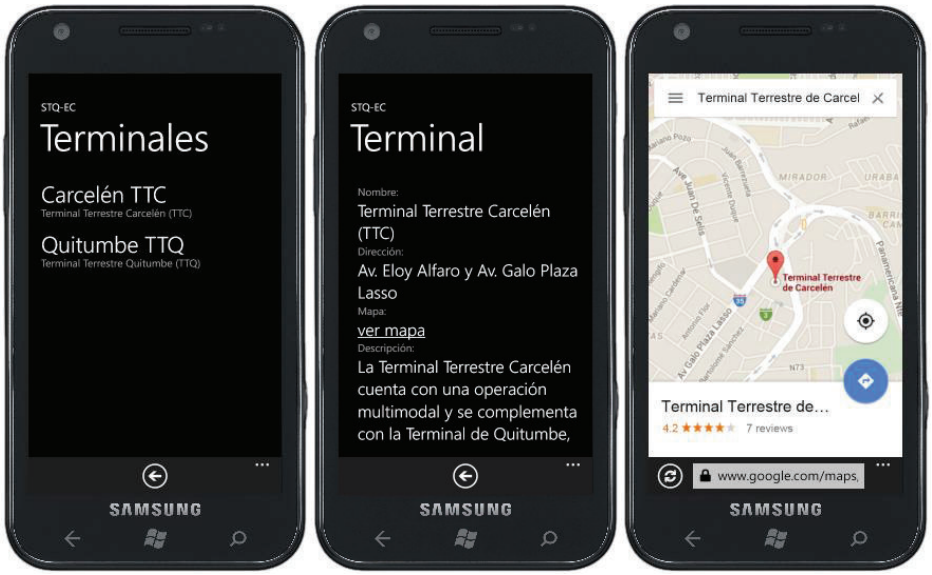

Figura 3.23 Caso de Prueba Opciones Informativas Segunda Parte

3.3.1.4 Cambiar Contraseña

Caso de Prueba	
Nombre	Cambiar Contraseña
Descripción	Cambiar la contraseña de un usuario al sistema.
Condiciones	Los usuarios de tipo Administrador Terminal, Administrador Cooperativa y Usuario Normal han ingresado en el sistema.
Ejecución	• Seleccionar Perfil Usuario en el menú de opciones para visualizar la información del usuario. • Seleccionar la opción de contraseña, proceder a editar los campos necesarios. • Guardar los cambios. • Acceder nuevamente al sistema.
Resultado Esperado	• Mensaje informativo, si el cambio fue exitoso o si no se pudo realizar el cambio. • Ingreso del usuario al sistema con la nueva contraseña.
Conclusión	Cumple con el resultado, se muestran mensajes informativos del resultado del procedimiento. El usuario está habilitado para ingresar con una nueva contraseña.

Tabla 3.4 Caso de Prueba Cambiar Contraseña

La siguiente figura muestra el cambio de contraseña del usuario.

Figura 3.24 Caso de Prueba Cambiar Contraseña Primera Parte

La siguiente figura indica los cambios en la base de datos.

	Nombre de Usuario	Nombre	Apellido	Telefono	Contraseña
1	admint1	Admint1	Adm1	0987654321	cPsEvUGrZ9SrQe...

	Nombre de Usuario	Nombre	Apellido	Telefono	Contraseña
1	admint1	Admint1	Adm1	0987654321	Gcr6gES6uhiNBo...

Figura 3.25 Caso de Prueba Cambiar Contraseña Segunda Parte

3.3.1.5 Editar Usuario

Caso de Prueba	
Nombre	Editar Usuario
Descripción	Edición de los datos personales de un usuario en la aplicación.
Condiciones	Los usuarios de tipo Administrador Terminal, Administrador Cooperativa y Usuario Normal han ingresado en el sistema.
Ejecución	• Seleccionar Perfil Usuario en el menú de opciones para visualizar la información del usuario. • Seleccionar la opción de edición, proceder a editar los campos necesarios. • Guardar los cambios.
Resultado Esperado	• Datos actualizados de la información de un usuario. • Visualización de la nueva información de un usuario.
Conclusión	Cumple con el resultado, se visualiza el cambio de la información de un usuario en el sistema.

Tabla 3.5 Caso de Prueba Editar Usuario

La siguiente figura indica los cambios en la base de datos.

	Nombre de Usuario	Nombre	Apellido	Telefono	Correo
1	admint1	Admint1	Adm1	0987654321	stqemail+admint1@gmail.com

	Nombre de Usuario	Nombre	Apellido	Telefono	Correo
1	admint1	Admint12	Adm12	0987654321	stqemail+admint12@gmail.com

Figura 3.26 Caso de Prueba Editar Usuario Primera Parte

La siguiente figura muestra los cambios en los datos del usuario.

Figura 3.27 Caso de Prueba Editar Usuario Segunda Parte

3.3.1.6 Crear Usuario

Caso de Prueba	
Nombre	Crear Usuario
Descripción	Creación de usuarios de tipo Usuario Normal en el sistema.
Condiciones	Acceder a la aplicación.
Ejecución	• Seleccionar la opción registrarse en la pantalla inicial de la aplicación. • Completar correctamente los campos solicitados para el registro y seleccionar la opción guardar. • Ingresar al sistema con el nuevo usuario creado.
Resultado Esperado	• Creación de un nuevo usuario y acceso al sistema. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, una persona puede crear un Usuario Normal y acceder posteriormente al sistema.

Tabla 3.6 Caso de Prueba Crear Usuario

La siguiente figura indica el nuevo usuario en la base de datos.

	Nombre de Usuario	Contraseña	Nombre	Apellido	Email
1	usuario1	cJVSiopfgspz3Sp...	Usuario1	Usr1	stqemail+usuario1@gmail.com

Figura 3.28 Caso de Prueba Crear Usuario Primera Parte

La siguiente figura indica el registro de un usuario en el sistema.

Figura 3.29 Caso de Prueba Crear Usuario Segunda Parte

3.3.1.7 Recuperar Contraseña

Caso de Prueba	
Nombre	Recuperar Contraseña
Descripción	Restablecimiento de la contraseña de un usuario.
Condiciones	Acceso a la aplicación por parte de un usuario registrado en el sistema.
Ejecución	• Seleccionar la opción para recuperar el acceso en la pantalla inicial de la aplicación. • Seleccionar el nombre de usuario o correo electrónico y seleccionar la opción aceptar. • Acceder en el correo electrónico al enlace de recuperación de contraseña, válido por un determinado tiempo. • Verificar la nueva contraseña enviada al correo electrónico y acceder al sistema.
Resultado Esperado	• Envío de correo electrónico al usuario para recuperar la contraseña. • Acceso con los nuevos datos del usuario.
Conclusión	Cumple con el resultado, un usuario puede recuperar el acceso al sistema a través del correo electrónico.

Tabla 3.7 Caso de Prueba Recuperar Contraseña

La siguiente figura indica la recuperación de la información de la contraseña.

Figura 3.30 Caso de Prueba Recuperar Contraseña Primera Parte

La siguiente figura indica los datos del usuario en la base de datos.

	Nombre de Usuario	Nombre	Apellido	Contraseña	Correo electrónico
1	admint2	Admint2	Adm2	U++Blkqx6Aha0...	stqemail+admint2@gmail.com

Figura 3.31 Caso de Prueba Recuperar Contraseña Segunda Parte

La siguiente figura indica el correo enviado por parte del sistema al usuario.

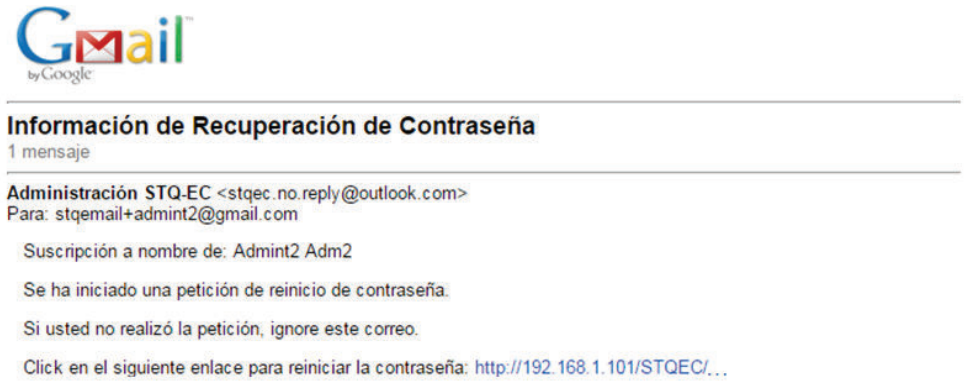

Figura 3.32 Caso de Prueba Recuperar Contraseña Tercera Parte

La siguiente figura indica el correo con la nueva contraseña del usuario.

Figura 3.33 Caso de Prueba Recuperar Contraseña Cuarta Parte

La siguiente figura indica la contraseña cambiada en la base de datos.

	Nombre de Usuario	Nombre	Apellido	Contraseña	Correo electrónico
1	admint2	Admint2	Adm2	wS/mBxty5jeSm...	stqemail+admint2@gmail.com

Figura 3.34 Caso de Prueba Recuperar Contraseña Quinta Parte

El usuario puede acceder nuevamente al sistema con la nueva contraseña.

3.3.1.8 Administrar Empresa

Caso de Prueba	
Nombre	Administrar Empresa
Descripción	Administración de la información de la empresa por medio del Administrador del Terminal.
Condiciones	Un usuario de tipo Administrador Terminal ha ingresado en el sistema.
Ejecución	• Seleccionar Empresa en el menú de opciones para visualizar la información de la Empresa. • Editar la información de la empresa, visualizar los cambios realizados en la entidad. • Acceder al twitter y la página web.
Resultado Esperado	• Visualización de la información de la empresa. • Visualización de los cambios realizados sobre los registros. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador del terminal accede a la información de la empresa y edita los datos.

Tabla 3.8 Caso de Prueba Administrar Empresa

La siguiente figura indica la información y edición de datos de la empresa.

Figura 3.35 Caso de Prueba Administrar Empresa Primera Parte

La siguiente figura indica los cambios en la base de datos.

	Nombre	DireccionWeb	NombreTwitter
1	Empresa Pública Metropolitana de Movilidad y Obras Públicas (EPMMOP)	http://www.epmmop.gob.ec/epmmop/	obrasquito

	Nombre	DireccionWeb	NombreTwitter
1	Prueba Empresa Pública Metropolitana de Movilidad y Obras Públicas (EPMMOP)	http://www.epmmop.gob.ec/epmmop/	obrasquitoprueba

Figura 3.36 Caso de Prueba Administrar Empresa Segunda Parte

3.3.1.9 Administrar Terminales

Caso de Prueba	
Nombre	Administrar Terminales
Descripción	Administración de los terminales por medio del Administrador del Terminal.
Condiciones	Un usuario de tipo Administrador Terminal ha ingresado en el sistema.
Ejecución	• Seleccionar Terminales en el menú de opciones para visualizar la lista de los terminales. • En la lista de terminales, seleccionar un Terminal para acceder a la información, edición o eliminación de la entidad.

Caso de Prueba	
Ejecución	• En la lista de terminales, seleccionar la opción crear para añadir una nueva entidad. Ingresar los valores correspondientes a la entidad, guardar el nuevo registro.
Resultado Esperado	• Visualización de los terminales y la información detallada. • Visualización de los cambios realizados sobre los registros. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador de terminal accede a los terminales y manipula los registros visualizando los cambios.

Tabla 3.9 Caso de Prueba Administrar Terminales

La siguiente figura indica la lista de terminales y la información y edición de datos de un terminal específico.

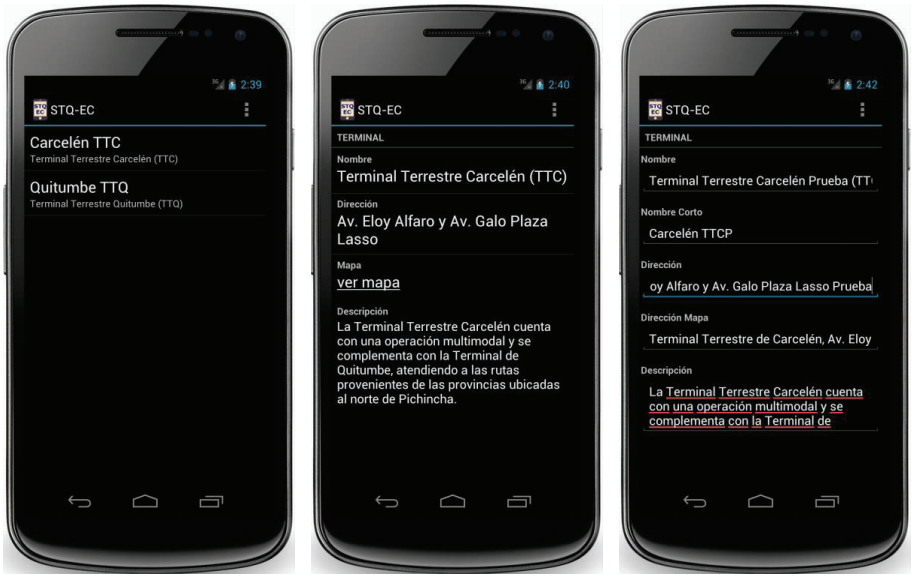

Figura 3.37 Caso de Prueba Administrar Terminales Primera Parte

La siguiente figura indica los cambios realizados en una entidad terminal en la base de datos.

	Nombre	Nombre Corto	Direccion	Mapa	Descripcion
1	Terminal Terrestre Carcelén (TTC)	Carcelén TTC	Av. Eloy Alfaro y Av. Galo Plaza Lasso	Terminal Ter...	La Terminal Terrestre C...

	Nombre	Nombre Corto	Direccion	Mapa	Descripcion
1	Terminal Terrestre Carcelén Prueba (TTC)	Carcelén TTCP	Av. Eloy Alfaro y Av. Galo Plaza Lasso Prueba	Terminal Terrestre...	La Terminal Terrestr...

Figura 3.38 Caso de Prueba Administrar Terminales Segunda Parte

3.3.1.10 Opciones de Terminal

Caso de Prueba	
Nombre	Opciones de Terminal
Descripción	Edición de boleterías y andenes de los terminales y selección de las cooperativas por medio del Administrador del Terminal.
Condiciones	Un usuario de tipo Administrador Terminal ha ingresado en el sistema.
Ejecución	• Seleccionar un Terminal para acceder a la vista de información. • Seleccionar opciones para editar las boleterías y andenes del terminal, guardar los cambios. • Seleccionar la opción cooperativas en la vista de información para ver la lista de cooperativas del terminal, seleccionar la opción de edición para modificar las cooperativas y guardar los cambios.
Resultado Esperado	• Visualización de cambios realizados en la lista de cooperativas del terminal o la capacidad de boleterías y andenes. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador del terminal accede a las opciones avanzadas del terminal y manipula los registros visualizando los cambios.

Tabla 3.10 Caso de Prueba Opciones de Terminal

La siguiente figura indica la información y edición de las opciones de un terminal.

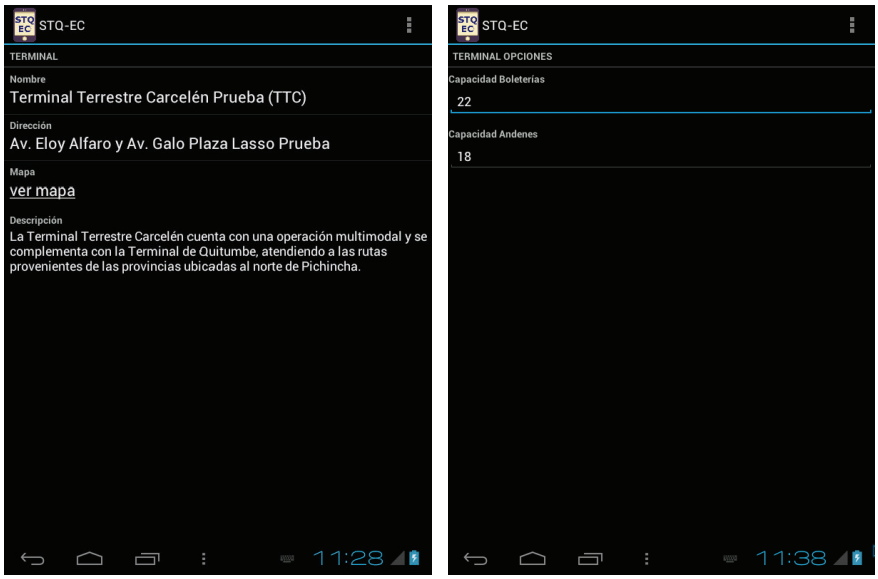

Figura 3.39 Caso de Prueba Opciones de Terminal Primera Parte

La siguiente figura indica los cambios en la base de datos.

	Nombre	Andenes	Boleterias
1	Terminal Terrestre Carcelén (TTC)	16	20

	Nombre	Andenes	Boleterias
1	Terminal Terrestre Carcelén (TTC)	18	22

Figura 3.40 Caso de Prueba Opciones de Terminal Segunda Parte

3.3.1.11 Administrar Cooperativas

Caso de Prueba	
Nombre	Administrar Cooperativas
Descripción	Administración de las Cooperativas por medio del Administrador del Terminal.
Condiciones	Un usuario de tipo Administrador Terminal ha ingresado en el sistema.
Ejecución	• Seleccionar Cooperativas en el menú de opciones para visualizar la lista de ciudades. • Filtrar la información para buscar una Cooperativa específica, cargar los registros mediante paginación y actualizar la lista. • En la lista de cooperativas, seleccionar una Cooperativa para acceder a la información, edición o eliminación de la entidad. • En la lista de cooperativas, seleccionar la opción crear para añadir una nueva entidad. Ingresar los valores correspondientes a la entidad, guardar el nuevo registro.
Resultado Esperado	• Visualización de las cooperativas, filtrado y paginación de registros. • Visualización de los cambios realizados sobre los registros. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador del terminal accede a las cooperativas y manipula los registros visualizando los cambios.

Tabla 3.11 Caso de Prueba Administrar Cooperativas

La siguiente figura indica la nueva cooperativa en la base de datos.

	Nombre	Direccion	Telefono
1	Prueba	Direccion	0987654321

Figura 3.41 Caso de Prueba Administrar Cooperativas Primera Parte

La siguiente figura indica la lista de las cooperativas en el sistema y el filtrado de información por nombre.

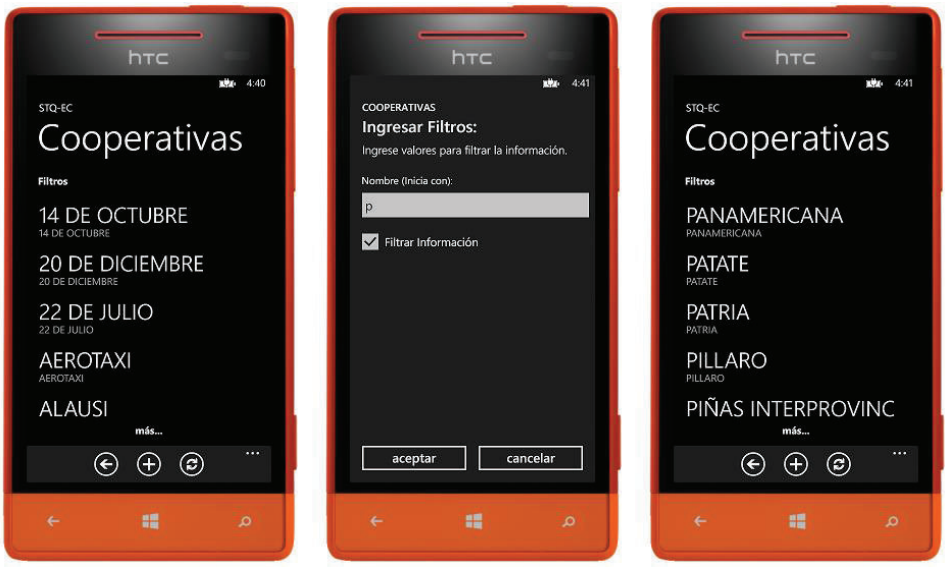

Figura 3.42 Caso de Prueba Administrar Cooperativas Segunda Parte

La siguiente figura indica la creación de una cooperativa y su posterior visualización en la aplicación.

Figura 3.43 Caso de Prueba Administrar Cooperativas Tercera Parte

3.3.1.12 Administrar Administradores de Cooperativa

Caso de Prueba	
Nombre	Administrar Administradores de Cooperativa
Descripción	Creación y eliminación de usuarios de tipo Administradores de Cooperativa por medio de un Administrador Terminal.
Condiciones	Un usuario de tipo Administrador Terminal ha ingresado en el sistema, y ha seleccionado la opción Cooperativas y ha seleccionado una cooperativa de la lista.
Ejecución	• Seleccionar Administradores en las opciones que presenta una determinada Cooperativa. • Seleccionar la opción crear en la lista de Administradores, ingresar los datos del nuevo Administrador y guardar los cambios. • Seleccionar un Administrador en la lista de Administradores para visualizar un Administrador, eliminar el Administrador. • Ingresar al sistema con el nuevo usuario creado y la contraseña enviada al correo electrónico.
Resultado Esperado	• Visualización de los Administradores de Cooperativa creados. • Envío de correo electrónico al nuevo usuario con la contraseña. • Ingreso al sistema con el nuevo usuario. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador de terminal ha creado nuevos usuario de tipo administradores de cooperativa y pueden acceder al sistema de acuerdo a su información personal.

Tabla 3.12 Caso de Prueba Administrar Administradores de Cooperativa

La siguiente figura indica el correo con la información de acceso al sistema para el nuevo administrador de cooperativa.

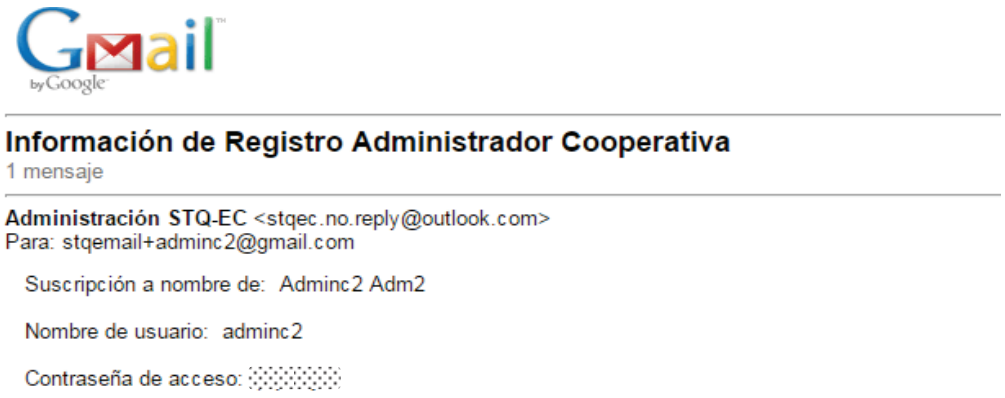

Figura 3.44 Caso de Prueba Administrar Administradores de Cooperativas Primera Parte

La siguiente figura indica la lista de administradores para una cooperativa y la creación de un nuevo administrador de cooperativa.

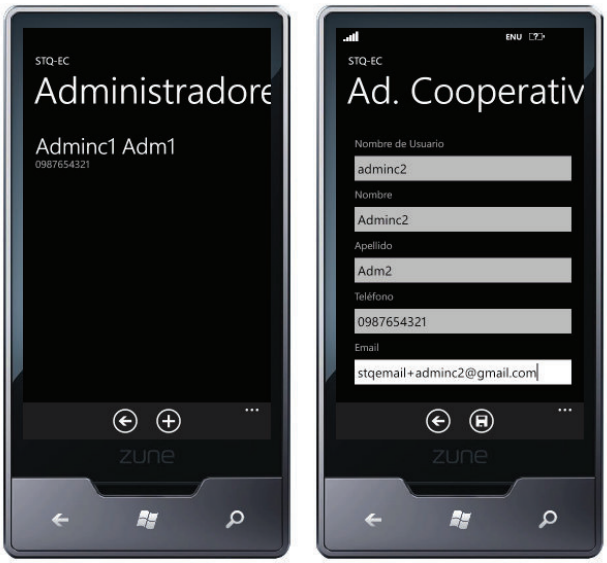

Figura 3.45 Caso de Prueba Administrar Administradores de Cooperativas Segunda Parte

La siguiente figura indica la lista de administradores para una cooperativa y la información del nuevo administrador de cooperativa creado en el sistema.

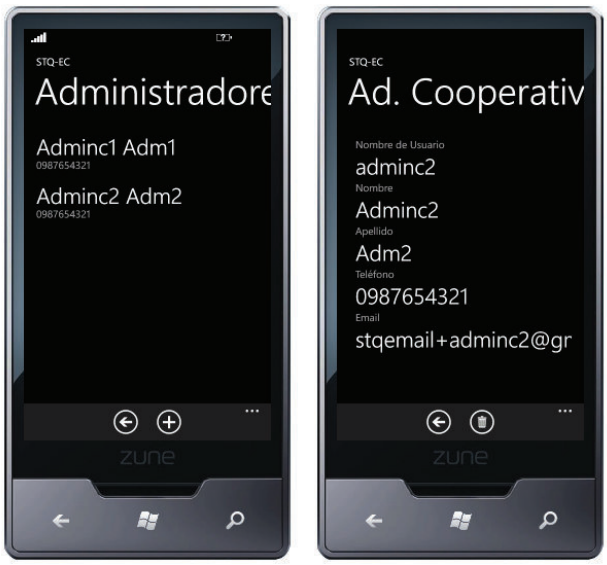

Figura 3.46 Caso de Prueba Administrar Administradores de Cooperativas Tercera Parte

La siguiente figura indica el nuevo administrador de cooperativa en la base de datos.

	Nombre de Usuario	Contraseña	Nombre	Apellido	Email
1	adminc2	6ULKDgVOwUpiVu...	Adminc2	Adm2	stqemail+adminc2@gmail.com

Figura 3.47 Caso de Prueba Administrar Administradores de Cooperativas Cuarta Parte

3.3.1.13 Administrar Frecuencias

Caso de Prueba	
Nombre	Administrar Frecuencias
Descripción	Administración de Frecuencias de una Cooperativa por medio de un Administrador Terminal.
Condiciones	Un usuario de tipo Administrador Terminal ha ingresado en el sistema, y ha seleccionado la opción Cooperativas y ha seleccionado una cooperativa de la lista.
Ejecución	• Seleccionar Frecuencias en las opciones de una determinada Cooperativa para acceder a la lista de Frecuencias. • Seleccionar la opción editar en la lista de Frecuencias. • Seleccionar el día correspondiente para visualizar la lista frecuencias que están asociadas. • Seleccionar la opción para editar la lista de frecuencias para un determinado día y guardar los cambios. • Seleccionar la opción crear en la lista de Frecuencias, ingresar los datos de la nueva Frecuencia y guardar los cambios. • Seleccionar una Frecuencia en la lista de Frecuencias para visualizar la información detallada. • Seleccionar la opción editar en la vista de información para modificar la Frecuencia y guardar los cambios. • Seleccionar la opción días en la vista de información para asociar los días de operación la Frecuencia y guardar los cambios. • Seleccionar la opción eliminar en la vista de información para borrar la Frecuencia.
Resultado Esperado	• Visualización de las Frecuencias creadas. • Visualización de los cambios realizados sobre los registros. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador de terminal ha creado nuevas frecuencias para una cooperativa, se puede editar o eliminar las frecuencias y asociar con los días correspondientes.

Tabla 3.13 Caso de Prueba Administrar Frecuencias

La siguiente figura indica la lista de frecuencias de una cooperativa, las frecuencias correspondientes a un día y la asignación de las frecuencias correspondientes para un día determinado.

Figura 3.48 Caso de Prueba Administrar Frecuencias Primera Parte

La siguiente figura indica las frecuencias deshabilitadas en la base de datos.

	Origen	Destino	Hora	Minuto	Estado
1	TT QUITUMBE	ATACAMES	23	20	1
2	TT QUITUMBE	ESMERALDAS	08	00	1
3	TT QUITUMBE	ESMERALDAS	10	15	1
4	TT QUITUMBE	ESMERALDAS	12	45	1
5	TT QUITUMBE	ESMERALDAS	15	10	1

	Origen	Destino	Hora	Minuto	Estado
1	TT QUITUMBE	ATACAMES	23	20	0
2	TT QUITUMBE	ESMERALDAS	08	00	0
3	TT QUITUMBE	ESMERALDAS	10	15	1
4	TT QUITUMBE	ESMERALDAS	12	45	1
5	TT QUITUMBE	ESMERALDAS	15	10	1

Figura 3.49 Caso de Prueba Administrar Frecuencias Segunda Parte

La siguiente figura indica la información y edición de una frecuencia.

Figura 3.50 Caso de Prueba Administrar Frecuencias Tercera Parte

La siguiente figura indica los cambios en la base de datos.

	Origen	Destino	Precio	Hora	Minuto
1	TT QUITUMBE	ESMERALDAS	6,15	08	00

	Origen	Destino	Precio	Hora	Minuto
1	TT QUITUMBE	ESMERALDAS	6,2	8	05

Figura 3.51 Caso de Prueba Administrar Frecuencias Cuarta Parte

3.3.1.14 Administrar Terminales de Cooperativa

Caso de Prueba	
Nombre	Administrar Terminales de Cooperativa
Descripción	Administración de Terminales de una Cooperativa por medio de un Administrador Terminal.
Condiciones	Un usuario de tipo Administrador Terminal ha ingresado en el sistema, y ha seleccionado la opción Cooperativas y ha seleccionado una cooperativa de la lista.
Ejecución	• Seleccionar Terminales en las opciones de una determinada Cooperativa para acceder a la lista de Terminales. • Seleccionar la opción para editar la lista de Terminales, escoger los terminales en los que opera la Cooperativa y guardar los cambios

Caso de Prueba	
Ejecución	• Seleccionar un Terminal en la lista de Terminales para visualizar o editar las boleterías y andenes asociadas a la cooperativa y guardar los cambios.
Resultado Esperado	• Visualización de los Terminales de la Cooperativa. • Visualización de los cambios realizados sobre los registros. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador de terminal ha asociado los terminales con una cooperativa y configurado los andenes y boleterías correspondientes.

Tabla 3.14 Caso de Prueba Administrar Terminales de Cooperativa

La siguiente figura indica la asignación de terminales a una cooperativa.

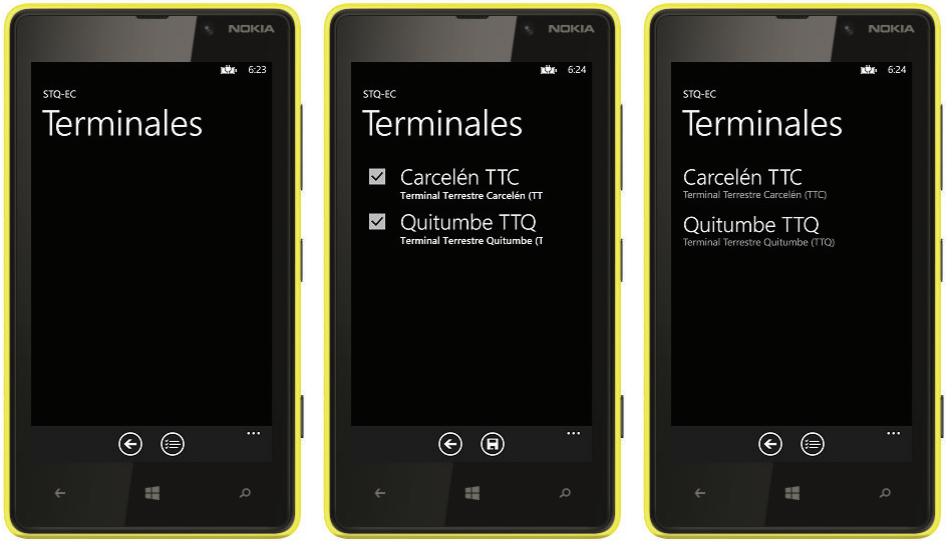

Figura 3.52 Caso de Prueba Administrar Terminales de Cooperativa Primera Parte

La siguiente figura indica los cambios en la base de datos.

	Cooperativa	Terminal	Estado
1	Prueba	Terminal Terrestre Carcelén (TTC)	1
2	Prueba	Terminal Terrestre Quitumbe (TTQ)	1

Figura 3.53 Caso de Prueba Administrar Terminales de Cooperativa Segunda Parte

La siguiente figura indica la asignación de boleterías y andenes en una cooperativa.

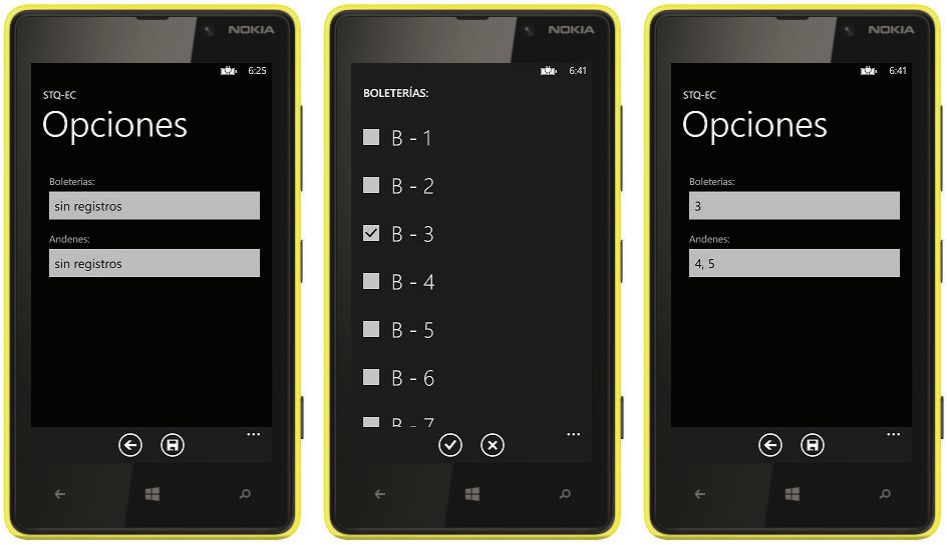

Figura 3.54 Caso de Prueba Administrar Terminales de Cooperativa Tercera Parte

La siguiente figura indica los cambios en la base de datos.

	Cooperativa	Terminal	Andenes
1	Prueba	Terminal Terrestre Quitumbe (TTQ)	4
2	Prueba	Terminal Terrestre Quitumbe (TTQ)	5

	Cooperativa	Terminal	Boleterias
1	Prueba	Terminal Terrestre Quitumbe (TTQ)	3

Figura 3.55 Caso de Prueba Administrar Terminales de Cooperativa Cuarta Parte

3.3.1.15 Administrar Unidades

Caso de Prueba	
Nombre	Administrar Unidades
Descripción	Administración de Unidades de una Cooperativa por medio de un Administrador Terminal.
Condiciones	Un usuario de tipo Administrador Terminal ha ingresado en el sistema, y ha seleccionado la opción Cooperativas y ha seleccionado una cooperativa de la lista.
Ejecución	• Seleccionar Unidades en las opciones que presenta una determinada Cooperativa para acceder a la lista de Unidades. • Seleccionar la opción crear en la lista de Unidades, ingresar los datos de la nueva Unidad y guardar los cambios.

Caso de Prueba	
Ejecución	• Seleccionar una Unidad en la lista de Unidades para visualizar la información detallada, editar o eliminar la Unidad.
Resultado Esperado	• Visualización de las Unidades creadas. • Visualización de los cambios realizados sobre los registros. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador de terminal ha creado nuevas unidades con los detalles correspondientes para una cooperativa, las unidades se pueden editar o eliminar.

Tabla 3.15 Caso de Prueba Administrar Unidades

La siguiente figura indica la lista de unidades y la creación de una unidad de una cooperativa.

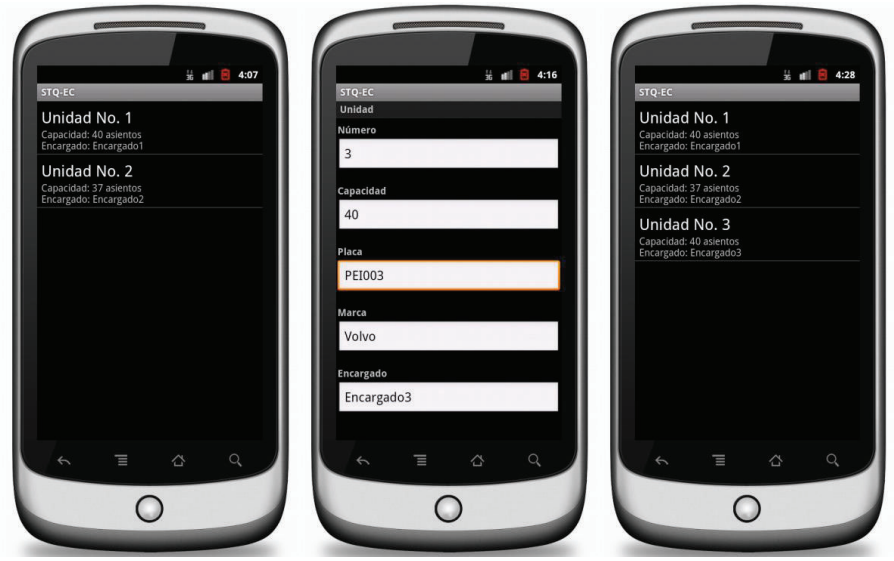

Figura 3.56 Caso de Prueba Administrar Unidades Primera Parte

La siguiente figura indica los cambios en la base de datos.

	Numero	Capacidad	Placa	Marca	Encargado	Telefono
1	1	40	PEI007	Hino	Encargado1	0987654321
2	2	37	PEI002	Volvo	Encargado2	0987654321

	Numero	Capacidad	Placa	Marca	Encargado	Telefono
1	1	40	PEI007	Hino	Encargado1	0987654321
2	2	37	PEI002	Volvo	Encargado2	0987654321
3	3	40	PEI003	Volvo	Encargado3	0987654321

Figura 3.57 Caso de Prueba Administrar Unidades Segunda Parte

La siguiente figura indica la información y edición de datos de una unidad.

Figura 3.58 Caso de Prueba Administrar Unidades Tercera Parte

La siguiente figura indica los cambios en la base de datos.

	Numero	Capacidad	Placa	Marca	Encargado	Telefono
1	3	40	PEI003	Volvo	Encargado3	0987654321

	Numero	Capacidad	Placa	Marca	Encargado	Telefono
1	3	40	PEI004	Scania	Encargado4	0987654321

Figura 3.59 Caso de Prueba Administrar Unidades Cuarta Parte

3.3.1.16 Administrar Ciudades

Caso de Prueba	
Nombre	Administrar Ciudades
Descripción	Administración de las ciudades por medio del Administrador del Terminal.
Condiciones	Un usuario de tipo Administrador Terminal ha ingresado en el sistema.
Ejecución	• Seleccionar Ciudades en el menú de opciones para visualizar la lista de ciudades.

Caso de Prueba	
Ejecución	• Filtrar la información para buscar una ciudad específica, cargar los registros mediante paginación y actualizar la lista. • En la lista de ciudades, seleccionar una Ciudad para acceder a la información, edición o eliminación de la entidad. • En la lista de ciudades, seleccionar la opción crear para añadir una nueva entidad. Ingresar los valores correspondientes a la entidad y guardar el nuevo registro.
Resultado Esperado	• Visualización de las ciudades, filtrado y paginación de registros. • Visualización de los cambios realizados sobre los registros. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador del terminal accede a las ciudades y manipula los registros visualizando los cambios.

Tabla 3.16 Caso de Prueba Administrar Ciudades

La siguiente figura indica las ciudades y el filtrado correspondiente.

Figura 3.60 Caso de Prueba Administrar Ciudades Primera Parte

La siguiente figura indica la nueva ciudad en la base de datos.

	Ciudad	Descripcion
1	Prueba	Descripcion

Figura 3.61 Caso de Prueba Administrar Ciudades Segunda Parte

La siguiente figura indica la creación de una ciudad.

Figura 3.62 Caso de Prueba Administrar Ciudades Tercera Parte

3.3.1.17 Administración de Administradores Terminal

Caso de Prueba	
Nombre	Administración de Administradores Terminal
Descripción	Creación de usuarios de tipo Administradores de Terminal por medio de un Administrador Terminal con permisos.
Condiciones	Un usuario de tipo Administrador Terminal con permisos ha ingresado en el sistema.
Ejecución	• Seleccionar Administradores en el menú de opciones para visualizar la información de los Administradores de Terminal que existen actualmente en el sistema. • Seleccionar la opción crear en la lista de Administradores, ingresar datos del nuevo Administrador y guardar los cambios. • Seleccionar un Administrador en la lista de Administradores para visualizar un Administrador, eliminar el Administrador. • Ingresar al sistema con el nuevo usuario creado y la contraseña enviada al correo electrónico.
Resultado Esperado	• Visualización de los Administradores de Terminal creados. • Envío de correo electrónico al nuevo usuario con la contraseña. • Ingreso al sistema con el nuevo usuario. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador terminal con permisos ha creado nuevos usuario de tipo administradores de terminal y pueden acceder al sistema de acuerdo a su información personal.

Tabla 3.17 Caso de Prueba Administración de Administradores Terminal

La siguiente figura indica la lista de administradores de terminal y la creación de un nuevo administrador de terminal.

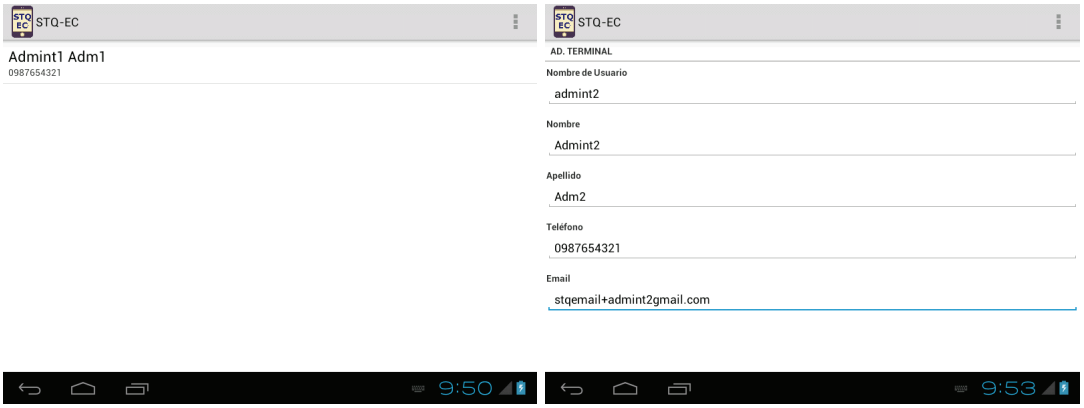

Figura 3.63 Caso de Prueba Administración de Administradores Terminal Primera Parte

La siguiente figura indica la lista de administradores de terminal y la información del nuevo administrador de terminal.

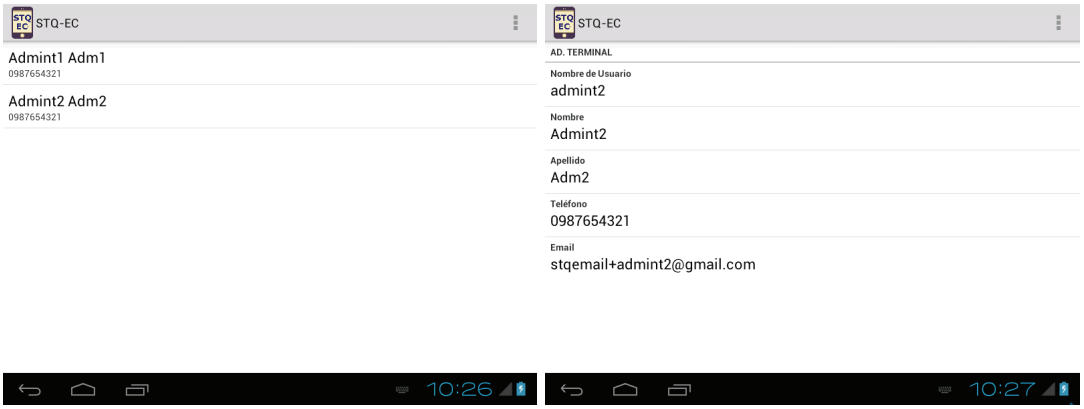

Figura 3.64 Caso de Prueba Administración de Administradores Terminal Segunda Parte

La siguiente figura indica el nuevo administrador de terminal en la base de datos.

	Nombre de Usuario	Contraseña	Nombre	Apellido	Email
1	admint2	U++Blkqx6Aha0JM...	Admint2	Adm2	stqemail+admint2@gmail.com

Figura 3.65 Caso de Prueba Administración de Administradores Terminal Tercera Parte

La siguiente figura indica el correo con la información de acceso al sistema para el nuevo administrador de terminal.

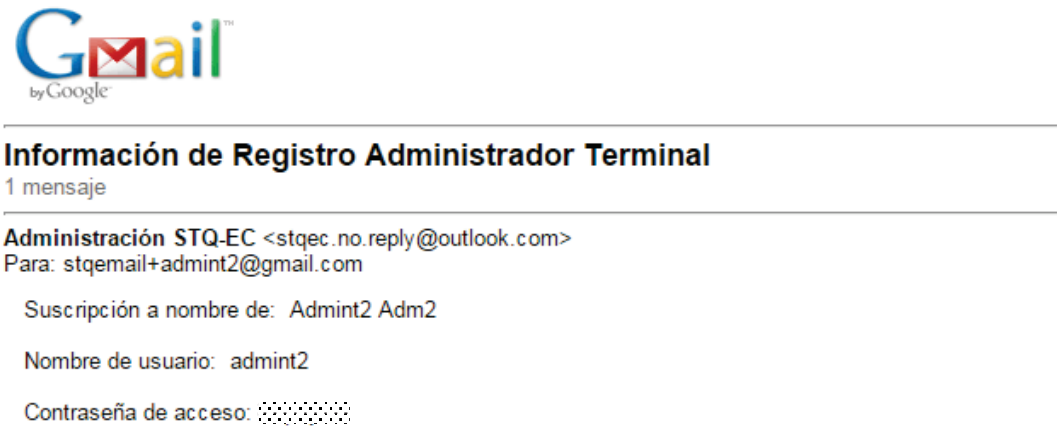

Figura 3.66 Caso de Prueba Administración de Administradores Terminal Cuarta Parte

3.3.1.18 Reportes

Caso de Prueba	
Nombre	Reportes
Descripción	Visualización de reportes por medio del Administrador del Terminal.
Condiciones	Un usuario de tipo Administrador Terminal ha ingresado en el sistema.
Ejecución	• Seleccionar Reportes en el menú de reportes para visualizar la lista de reportes a los cuales se puede acceder. • Seleccionar la opción Frecuencias del menú de reportes, ingresar el nombre de la Cooperativa que se desea visualizar o marcar la opción para visualizar todas las Cooperativas. • Seleccionar la opción Viajes del menú, seleccionar la fecha en la cual se desea visualizar los Viajes. • Seleccionar la opción reporte para visualizar el reporte con los resultados de los filtros.
Resultado Esperado	• Visualización de los reportes generados de acuerdo a los parámetros.
Conclusión	Cumple con el resultado, un administrador del terminal visualiza la información de frecuencias y viajes del sistema.

Tabla 3.18 Caso de Prueba Reportes

La siguiente figura indica la generación del reporte de frecuencias en un dispositivo *Android*.

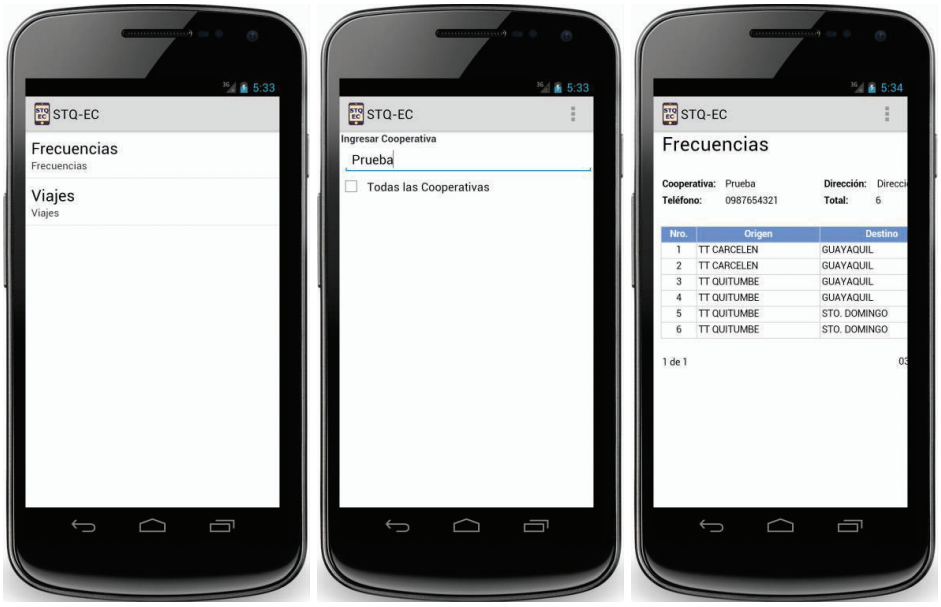

Figura 3.67 Caso de Prueba Reportes Primera Parte

La siguiente figura indica la generación del reporte de viajes en un dispositivo *Windows Phone*.

Figura 3.68 Caso de Prueba Reportes Segunda Parte

3.3.1.19 Opciones de Cooperativa

Caso de Prueba	
Nombre	Opciones de Cooperativa
Descripción	Edición de información adicional de la cooperativa por medio de un Administrador de Cooperativa.
Condiciones	Un usuario de tipo Administrador Cooperativa ha ingresado en el sistema.
Ejecución	• Seleccionar Cooperativa en el menú de opciones para acceder a la vista informativa de la Cooperativa. • Seleccionar opciones y editar la información de twitter y página web de la Cooperativa si existen, editar la capacidad máxima de reservaciones de las unidades y guardar los cambios.
Resultado Esperado	• Visualización de los cambios realizados, permitiendo visualizar, ocultar o cambiar la información adicional de la cooperativa. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador de cooperativa accede a las opciones adicionales de la cooperativa y realiza los cambios.

Tabla 3.19 Caso de Prueba Opciones de Cooperativa

La siguiente figura indica la edición de varios datos de la cooperativa.

Figura 3.69 Caso de Prueba Opciones de Cooperativa Primera Parte

La siguiente figura indica los cambios en la base de datos.

	Nombre	Direccion	Telefono	Capacidad	Twitter	SitioWeb
1	Prueba	Direccion	0987654321	10	stqec	http://www.prueba.com

Figura 3.70 Caso de Prueba Opciones de Cooperativa Segunda Parte

3.3.1.20 Opciones Informativas Cooperativa

Caso de Prueba	
Nombre	Opciones Informativas Cooperativa
Descripción	Visualización de elementos informativos de la Cooperativa por medio de un Administrador Cooperativa.
Condiciones	Un usuario de tipo Administrador Cooperativa ha ingresado en el sistema.
Ejecución	• Seleccionar Unidades en el menú de opciones para visualizar la lista de unidades de la Cooperativa. • Seleccionar Frecuencias en el menú de opciones para visualizar la lista de frecuencias de la Cooperativa.
Resultado Esperado	• Visualizar datos informativos de las unidades y las frecuencias de una Cooperativa en particular.
Conclusión	Cumple con el resultado, se visualiza la información y permite al administrador verificar si la información es correcta.

Tabla 3.20 Caso de Prueba Opciones Informativas Cooperativa

La siguiente figura indica la información de unidades y frecuencias.

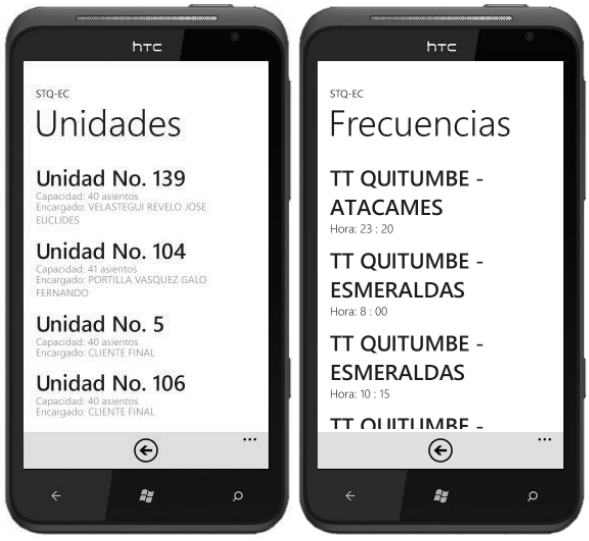

Figura 3.71 Caso de Prueba Opciones Informativas Cooperativa

3.3.1.21 Administrar Viajes

Caso de Prueba	
Nombre	Administrar Viajes
Descripción	Creación de viajes de una cooperativa por medio del Administrador de la Cooperativa.
Condiciones	Un usuario de tipo Administrador Cooperativa ha ingresado en el sistema.
Ejecución	• Seleccionar Viajes en el menú de opciones para visualizar la lista de viajes de la cooperativa en una determinada fecha. • Seleccionar la opción crear viaje en la lista de Viajes, ingresar los datos del nuevo Viaje y guardar los cambios. • Seleccionar la opción crear viajes en la lista de Viajes, seleccionar los viajes con valores por defecto, seleccionar un viaje para modificar los valores iniciales y guardar los cambios. • Seleccionar un viaje en la lista de Viajes, modificar los datos del viaje y guardar los cambios.
Resultado Esperado	• Visualización de los viajes creados. • Visualización de los cambios realizados sobre los registros. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador de cooperativa crea nuevos viajes para la cooperativa y puede modificar la información.

Tabla 3.21 Caso de Prueba Administrar Viajes

La siguiente figura indica la creación de un viaje para una fecha en *Android*.

Figura 3.72 Caso de Prueba Administrar Viajes Primera Parte

La siguiente figura indica la visualización y edición de la información de un viaje en *Android*.

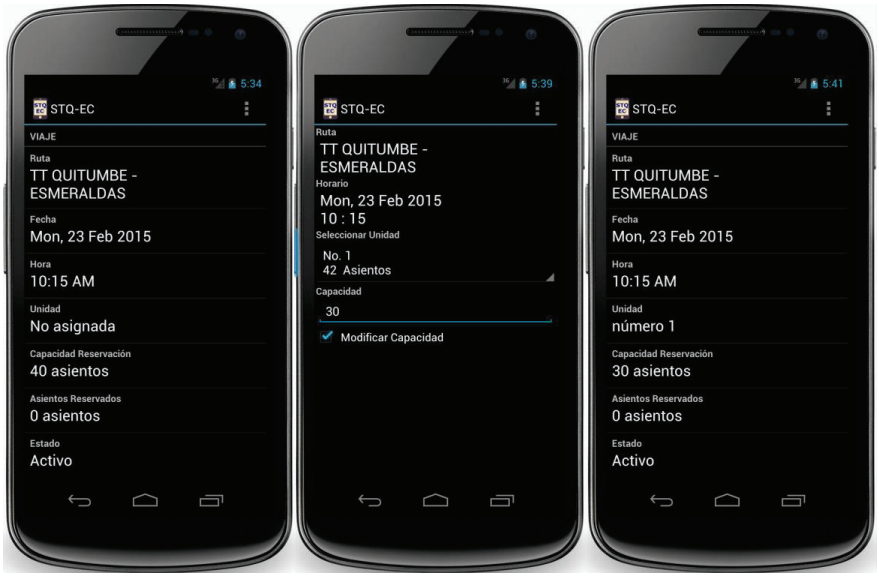

Figura 3.73 Caso de Prueba Administrar Viajes Segunda Parte

La siguiente figura indica la creación de un viaje para una fecha en *Windows Phone*.

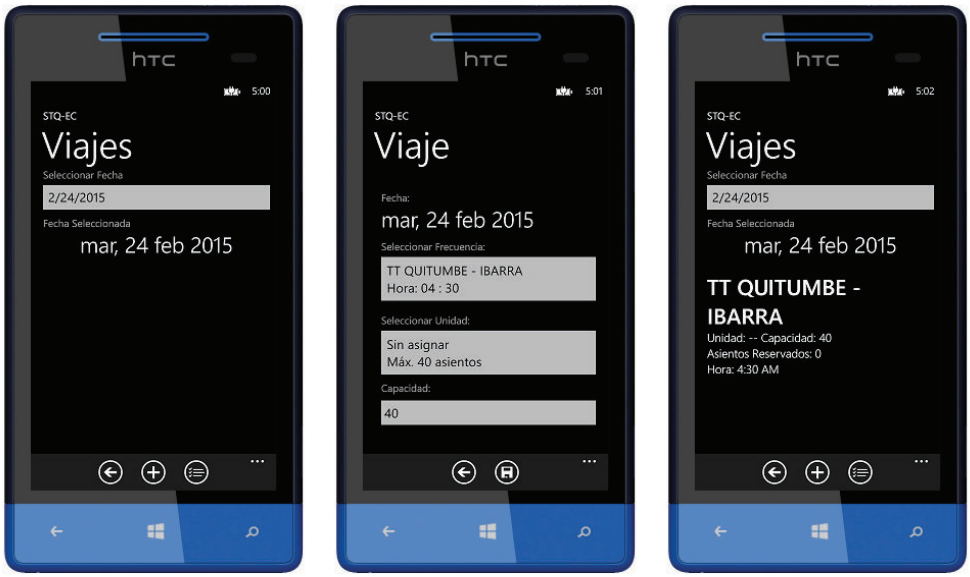

Figura 3.74 Caso de Prueba Administrar Viajes Tercera Parte

La siguiente figura indica la visualización y edición de un viaje en *Windows Phone*.

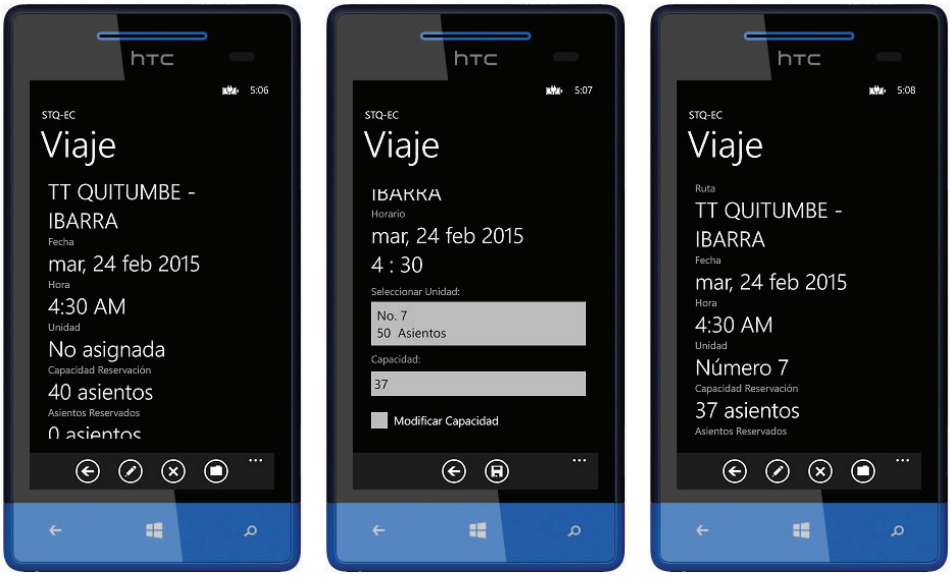

Figura 3.75 Caso de Prueba Administrar Viaje Cuarta Parte

La siguiente figura indica los cambios en la base de datos de las acciones realizadas en *Android*.

	Origen	Destino	Horario	Capacidad	Unidad	Estado	Asientos	Asientos Reservados
1	TT QUITUMBE	ESMERALDAS	23/02/2015 10:15:00	40	NULL	Activo	40	0

	Origen	Destino	Horario	Capacidad	Unidad	Estado	Asientos	Asientos Reservados
1	TT QUITUMBE	ESMERALDAS	23/02/2015 10:15:00	30	1	Activo	30	0

Figura 3.76 Caso de Prueba Administrar Viajes Quinta Parte

La siguiente figura indica los cambios en la base de datos de las acciones realizadas en *Windows Phone*.

	Origen	Destino	Horario	Capacidad	Estado	Unidad	Asientos	Asientos Reservados
1	TT QUITUMBE	IBARRA	24/02/2015 4:30:00	40	Activo	NULL	40	0

	Origen	Destino	Horario	Capacidad	Estado	Unidad	Asientos	Asientos Reservados
1	TT QUITUMBE	IBARRA	24/02/2015 4:30:00	37	Activo	7	37	0

Figura 3.77 Caso de Prueba Administrar Viajes Sexta Parte

La siguiente figura indica la creación de múltiples viajes en *Android* en una fecha determinada por el usuario.

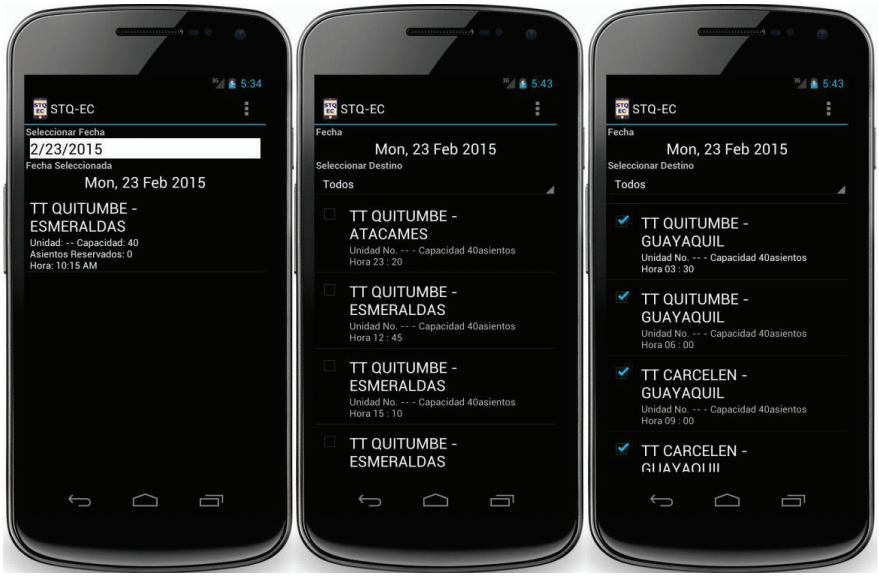

Figura 3.78 Caso de Prueba Administrar Viajes Séptima Parte

La siguiente figura indica la creación y modificación de valores por defecto de viajes en *Android* en una fecha determinada por el usuario.

Figura 3.79 Caso de Prueba Administrar Viajes Octava Parte

La siguiente figura indica la creación de múltiples viajes en *Windows Phone* en una fecha determinada por el usuario.

Figura 3.80 Caso de Prueba Administrar Viajes Novena Parte

La siguiente figura indica la creación y modificación de valores por defecto de viajes en *Windows Phone* en una fecha determinada por el usuario.

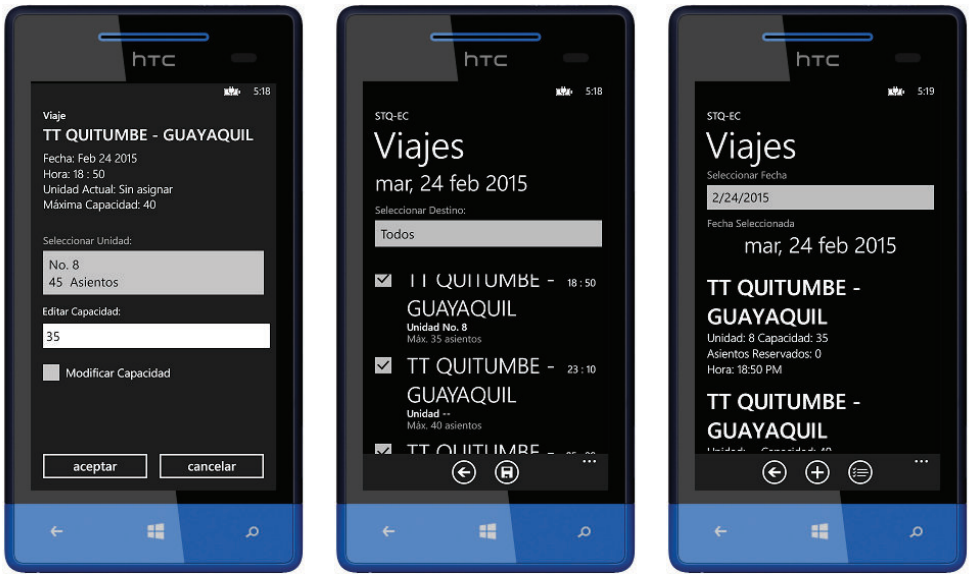

Figura 3.81 Caso de Prueba Administrar Viajes Décima Parte

La siguiente figura indica los cambios en la base de datos para *Android*.

	Origen	Destino	Horario	Capacidad	Estado	Unidad	Asientos	Asientos Reservados
1	TT QUITUMBE	ESMERALDAS	23/02/2015 10:15:00	30	Activo	1	30	0
2	TT QUITUMBE	GUAYAQUIL	23/02/2015 3:30:00	20	Activo	5	20	0
3	TT QUITUMBE	GUAYAQUIL	23/02/2015 6:00:00	40	Activo	NULL	40	0
4	TT CARCELEN	GUAYAQUIL	23/02/2015 9:00:00	40	Activo	NULL	40	0
5	TT CARCELEN	GUAYAQUIL	23/02/2015 11:20:00	40	Activo	NULL	40	0

Figura 3.82 Caso de Prueba Administrar Viajes Undécima Parte

La siguiente figura indica los cambios en la base de datos para *Windows Phone*.

	Origen	Destino	Horario	Capacidad	Estado	Unidad	Asientos	Asientos Reservados
1	TT QUITUMBE	IBARRA	24/02/2015 4:30:00	37	Activo	7	37	0
2	TT QUITUMBE	GUAYAQUIL	24/02/2015 18:50:00	35	Activo	8	35	0
3	TT QUITUMBE	GUAYAQUIL	24/02/2015 23:10:00	40	Activo	NULL	40	0
4	TT QUITUMBE	IBARRA	24/02/2015 5:00:00	40	Activo	NULL	40	0

Figura 3.83 Caso de Prueba Administrar Viajes Duodécima Parte

3.3.1.22 Administrar Viajes Reservaciones

Caso de Prueba	
Nombre	Administrar Viajes Reservaciones
Descripción	Administración de las reservaciones y el estado de un viaje de una cooperativa por medio del Administrador de la Cooperativa.
Condiciones	Un usuario de tipo Administrador Cooperativa ha ingresado en el sistema.
Ejecución	• Seleccionar Viajes en el menú de opciones para visualizar la lista de viajes de la cooperativa en una determinada fecha. • Seleccionar un viaje en la lista de Viajes para acceder a la vista informativa del viaje y las distintas opciones. • Para cancelar un Viaje, seleccionar la opción correspondiente del menú y confirme la acción a realizar. • Para ver las reservaciones del viaje o el historial de reservaciones, seleccionar las opciones correspondientes. • En la vista de reservaciones, seleccionar una reservación para confirmar que se ha realizado o seleccionar la opción finalizar viaje para terminar el proceso y actualizar las reservaciones.
Resultado Esperado	• Visualización de reservaciones y usuarios del viaje. • Visualización de los cambios sobre el viaje y las reservaciones. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, un administrador de cooperativa maneja las reservaciones y el estado del viaje o las reservaciones.

Tabla 3.22 Caso de Prueba Administrar Viajes Reservaciones

La siguiente figura indica la lista de viajes en una fecha determinada, la información de un viaje y las reservaciones que se han realizado en *Android*.

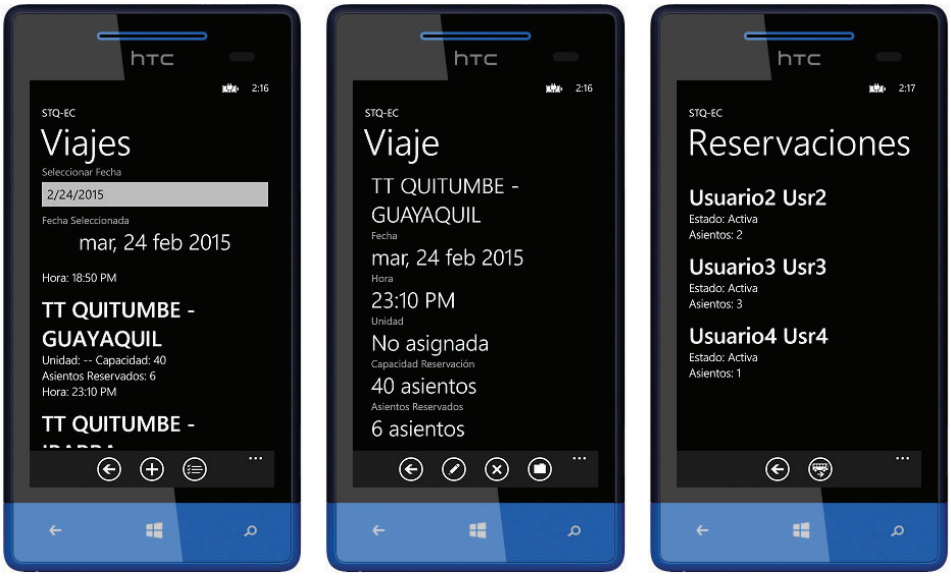

Figura 3.84 Caso de Prueba Administrar Viajes Reservaciones Primera Parte

La siguiente figura indica la información y la finalización de una reservación en *Windows Phone*.

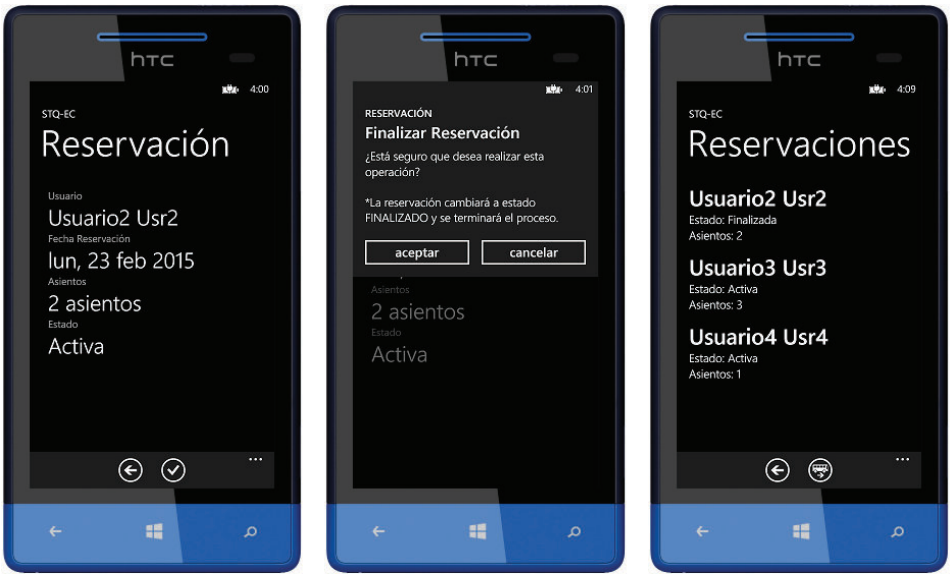

Figura 3.85 Caso de Prueba Administrar Viajes Reservaciones Segunda Parte

La siguiente figura indica la finalización de un viaje a través de un mensaje que se despliega para la confirmación del usuario y el detalle del viaje finalizado en *Windows Phone*.

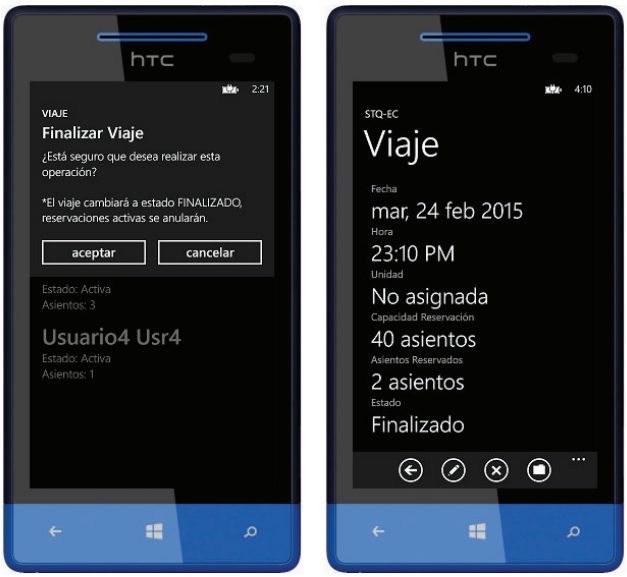

Figura 3.86 Caso de Prueba Administrar Viajes Reservaciones Tercera Parte

La siguiente figura indica los cambios en la base de datos realizados en *Windows Phone*. La secuencia muestra el estado inicial de las reservaciones, luego la finalización de una reservación y la finalización del viaje.

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario2 Usr2	Activo	TT QUITUMBE - GUAYAQUIL	AEROTAXI	24/02/2015 23:10:00	Activa	2
2	Usuario3 Usr3	Activo	TT QUITUMBE - GUAYAQUIL	AEROTAXI	24/02/2015 23:10:00	Activa	3
3	Usuario4 Usr4	Activo	TT QUITUMBE - GUAYAQUIL	AEROTAXI	24/02/2015 23:10:00	Activa	1

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario2 Usr2	Activo	TT QUITUMBE - GUAYAQUIL	AEROTAXI	24/02/2015 23:10:00	Finalizada	2
2	Usuario3 Usr3	Activo	TT QUITUMBE - GUAYAQUIL	AEROTAXI	24/02/2015 23:10:00	Activa	3
3	Usuario4 Usr4	Activo	TT QUITUMBE - GUAYAQUIL	AEROTAXI	24/02/2015 23:10:00	Activa	1

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario2 Usr2	Finalizado	TT QUITUMBE - GUAYAQUIL	AEROTAXI	24/02/2015 23:10:00	Finalizada	2
2	Usuario3 Usr3	Finalizado	TT QUITUMBE - GUAYAQUIL	AEROTAXI	24/02/2015 23:10:00	Anulada	3
3	Usuario4 Usr4	Finalizado	TT QUITUMBE - GUAYAQUIL	AEROTAXI	24/02/2015 23:10:00	Anulada	1

Figura 3.87 Caso de Prueba Administrar Viajes Reservaciones Cuarta Parte

La siguiente figura indica la lista de viajes en una fecha determinada, la información de un viaje y las reservaciones que se han realizado en *Android*.

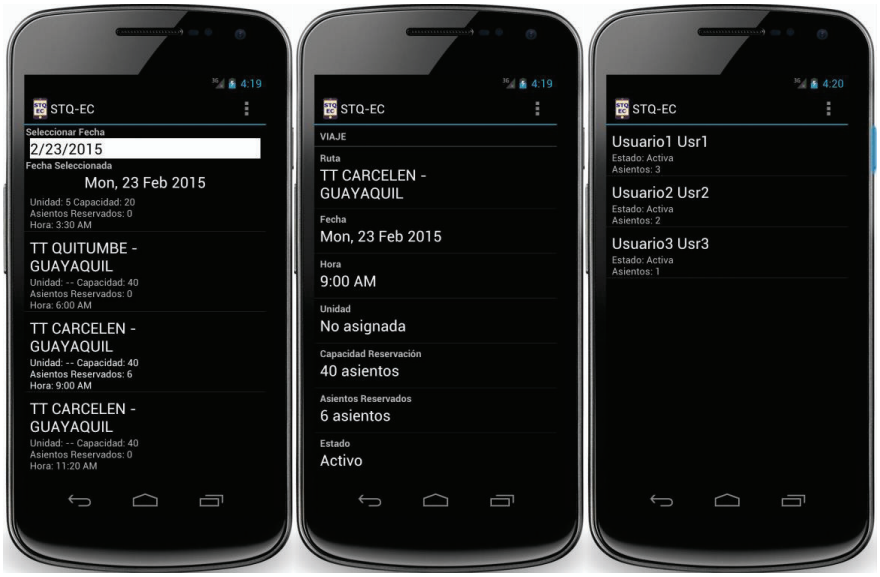

Figura 3.88 Caso de Prueba Administrar Viajes Reservaciones Quinta Parte

La siguiente figura indica la información y la finalización de una reservación en *Android*.

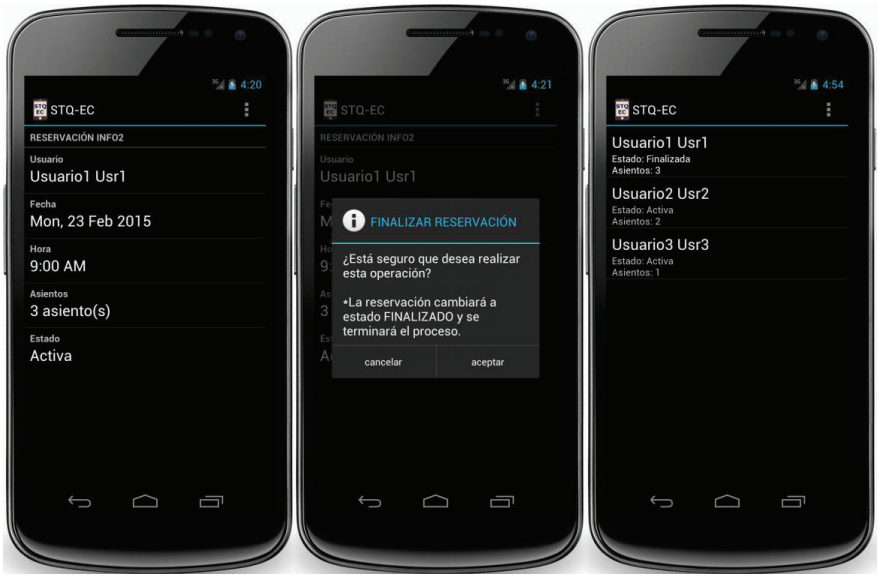

Figura 3.89 Caso de Prueba Administrar Viajes Reservaciones Sexta Parte

La siguiente figura indica la finalización de un viaje a través de un mensaje que se despliega para la confirmación del usuario y el detalle del viaje finalizado en la plataforma *Android*.

Figura 3.90 Caso de Prueba Administrar Viajes Reservaciones Séptima Parte

La siguiente figura indica los cambios en la base de datos realizados en *Android*. La secuencia muestra el estado inicial de las reservaciones, luego la finalización de una reservación y la finalización del viaje.

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario1 Usr1	Activo	TT CARCELEN - GUAYAQUIL	AEROTAXI	23/02/2015 9:00:00	Activa	3
2	Usuario2 Usr2	Activo	TT CARCELEN - GUAYAQUIL	AEROTAXI	23/02/2015 9:00:00	Activa	2
3	Usuario3 Usr3	Activo	TT CARCELEN - GUAYAQUIL	AEROTAXI	23/02/2015 9:00:00	Activa	1

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario1 Usr1	Activo	TT CARCELEN - GUAYAQUIL	AEROTAXI	23/02/2015 9:00:00	Finalizada	3
2	Usuario2 Usr2	Activo	TT CARCELEN - GUAYAQUIL	AEROTAXI	23/02/2015 9:00:00	Activa	2
3	Usuario3 Usr3	Activo	TT CARCELEN - GUAYAQUIL	AEROTAXI	23/02/2015 9:00:00	Activa	1

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario1 Usr1	Finalizado	TT CARCELEN - GUAYAQUIL	AEROTAXI	23/02/2015 9:00:00	Finalizada	3
2	Usuario2 Usr2	Finalizado	TT CARCELEN - GUAYAQUIL	AEROTAXI	23/02/2015 9:00:00	Anulada	2
3	Usuario3 Usr3	Finalizado	TT CARCELEN - GUAYAQUIL	AEROTAXI	23/02/2015 9:00:00	Anulada	1

Figura 3.91 Caso de Prueba Administrar Viajes Reservaciones Octava Parte

La siguiente figura indica la información de un viaje y las reservaciones de los usuarios en *Windows Phone*.

Figura 3.92 Caso de Prueba Administrar Viajes Reservaciones Novena Parte

La siguiente figura indica la cancelación de un viaje y el detalle del viaje cancelado en *Windows Phone*.

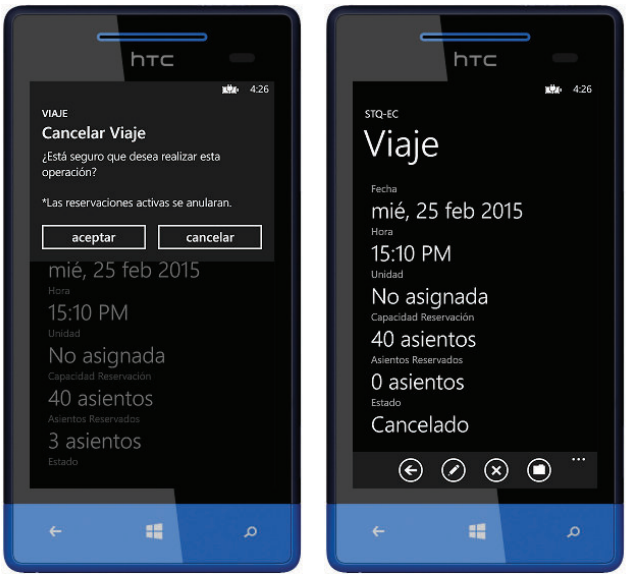

Figura 3.93 Caso de Prueba Administrar Viajes Reservaciones Décima Parte

La siguiente figura indica la información de un viaje y las reservaciones de los usuarios en *Android*.

Figura 3.94 Caso de Prueba Administrar Viajes Reservaciones Undécima Parte

La siguiente figura indica la cancelación de un viaje y el detalle del viaje cancelado en *Android*.

Figura 3.95 Caso de Prueba Administrar Viajes Reservaciones Duodécima Parte

La siguiente figura indica los cambios en la base de datos en *Windows Phone*.

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario2 Usr2	Activo	TT QUITUMBE - ESMERALDAS	AEROTAXI	25/02/2015 10:15:00	Activa	3
2	Usuario4 Usr4	Activo	TT QUITUMBE - ESMERALDAS	AEROTAXI	25/02/2015 10:15:00	Activa	1

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario2 Usr2	Cancelado	TT QUITUMBE - ESMERALDAS	AEROTAXI	25/02/2015 10:15:00	Anulada	3
2	Usuario4 Usr4	Cancelado	TT QUITUMBE - ESMERALDAS	AEROTAXI	25/02/2015 10:15:00	Anulada	1

Figura 3.96 Caso de Prueba Administrar Viajes Reservaciones Decimotercera Parte

La siguiente figura indica los cambios en la base de datos en *Android*.

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario1 Usr1	Activo	TT QUITUMBE - ESMERALDAS	AEROTAXI	25/02/2015 15:10:00	Activa	2
2	Usuario2 Usr2	Activo	TT QUITUMBE - ESMERALDAS	AEROTAXI	25/02/2015 15:10:00	Activa	1

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario1 Usr1	Cancelado	TT QUITUMBE - ESMERALDAS	AEROTAXI	25/02/2015 15:10:00	Anulada	2
2	Usuario2 Usr2	Cancelado	TT QUITUMBE - ESMERALDAS	AEROTAXI	25/02/2015 15:10:00	Anulada	1

Figura 3.97 Caso de Prueba Administrar Viajes Reservaciones Decimocuarta Parte

3.3.1.23 Consultar Viajes

Caso de Prueba	
Nombre	Consultar Viajes
Descripción	Consulta de Viajes dependiendo del Destino o la Cooperativa por medio de un Usuario Normal.
Condiciones	Un Usuario Normal ha ingresado en el sistema.
Ejecución	• Seleccionar Viajes en el menú de opciones para visualizar las opciones informativas de viajes en el sistema. • Seleccionar la opción Destinos para consultar los viajes disponibles para un determinado destino. • Seleccionar la opción Cooperativas para consultar la información y los viajes disponibles de una determinada Cooperativa.
Resultado Esperado	• Visualización de las cooperativas y de los viajes disponibles para una determinada Cooperativa. • Visualización de los destinos y de los viajes disponibles para un determinado Destino.
Conclusión	Cumple con el resultado, un usuario normal accede a la información de próximos viajes.

Tabla 3.23 Caso de Prueba Consultar Viajes

La siguiente figura indica las opciones del menú, los destinos y los viajes hacia un destino.

Figura 3.98 Caso de Prueba Consultar Viajes Primera Parte

La siguiente figura indica las cooperativas, el menú con opciones de una cooperativa y los viajes de una cooperativa.

Figura 3.99 Caso de Prueba Consultar Viajes Segunda Parte

3.3.1.24 Manejar Reservaciones

Caso de Prueba	
Nombre	Manejar Reservaciones
Descripción	Consulta de Viajes y realización de una Reservación por parte de un Usuario Normal.
Condiciones	Un Usuario Normal ha ingresado en el sistema.
Ejecución	• Seleccionar Reservaciones en el menú de opciones para visualizar la lista de reservaciones del usuario. • Seleccionar la opción crear para acceder a los filtros de búsqueda de viajes. • Completar los campos con los valores de Origen, Destino, Número de asientos y Fecha, realizar la búsqueda. • Seleccionar un viaje de la lista de viajes encontrados. • Comprobar los datos de la reservación y guardar el resultado para realizar la reservación. • Seleccionar una reservación en la lista de reservaciones para acceder al detalle informativo.
Resultado Esperado	• Creación de reservaciones mediante parámetros del usuario. • Visualización del detalle de la reservación. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, el usuario realiza la reservación y tiene acceso al detalle del viaje.

Tabla 3.24 Caso de Prueba Manejar Reservaciones

La siguiente figura indica la búsqueda de viajes con filtros en *Android*.

Figura 3.100 Caso de Prueba Manejar Reservaciones Primera Parte

La siguiente figura indica la selección y reservación de un viaje y su información en *Android.*

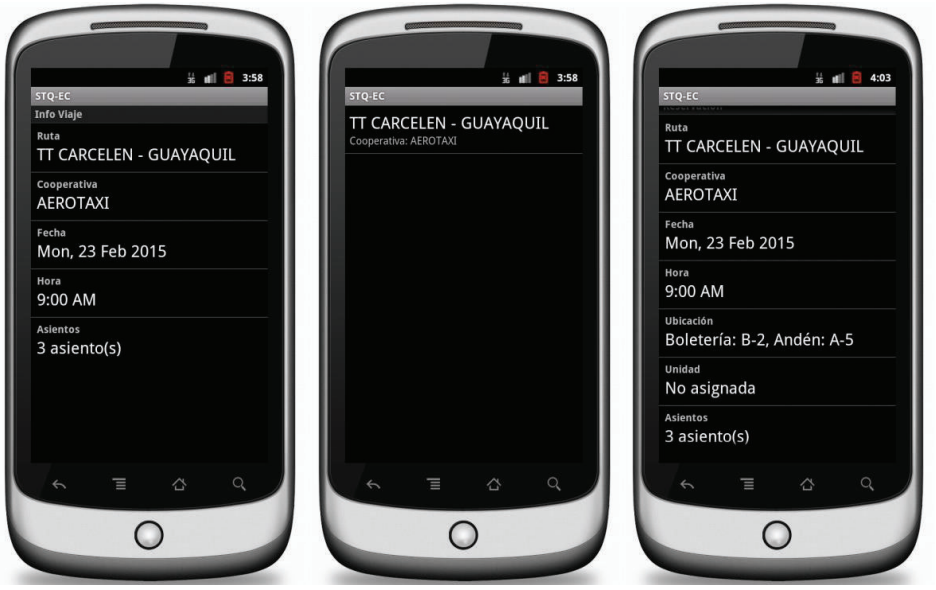

Figura 3.101 Caso de Prueba Manejar Reservaciones Segunda Parte

La siguiente figura indica la búsqueda de un viaje mediante filtros de información en *Windows Phone*.

Figura 3.102 Caso de Prueba Manejar Reservaciones Tercera Parte

La siguiente figura indica la selección y reservación de un viaje y su información en *Windows Phone*.

Figura 3.103 Caso de Prueba Manejar Reservaciones Cuarta Parte

La siguiente figura indica los cambios en la base de datos realizados mediante *Android*.

	Cliente	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario1 Usr1	TT CARCELEN - GUAYAQUIL	AEROTAXI	23/02/2015 9:00:00	Activa	3

Figura 3.104 Caso de Prueba Manejar Reservaciones Quinta Parte

La siguiente figura indica los cambios en la base de datos realizados mediante *Windows Phone*.

	Cliente	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario2 Usr2	TT QUITUMBE - GUAYAQUIL	AEROTAXI	24/02/2015 23:10:00	Activa	2

Figura 3.105 Caso de Prueba Manejar Reservaciones Sexta Parte

3.3.1.25 Cancelación e Historial Reservaciones

Caso de Prueba	
Nombre	Cancelación e Historial Reservaciones
Descripción	Cancelación de una Reservación y consulta del Historial de Reservaciones por parte de un Usuario Normal.
Condiciones	Un Usuario Normal ha ingresado en el sistema.
Ejecución	• Seleccionar Reservaciones en el menú de opciones para visualizar la lista de reservaciones del usuario. • Seleccionar una reservación en la lista de reservaciones para acceder al detalle informativo, seleccionar la opción Cancelar y confirmar la acción a realizar. • Seleccionar Historial en el menú de opciones para visualizar el historial de reservaciones del usuario.
Resultado Esperado	• Cancelación de una reservación. • Visualización del historial de reservaciones del usuario. • Base de datos actualizada según las operaciones.
Conclusión	Cumple con el resultado, el usuario cancela la reservación y visualiza el historial de reservaciones.

Tabla 3.25 Caso de Prueba Cancelación e Historial Reservaciones

La siguiente figura indica la cancelación de una reservación.

Figura 3.106 Caso de Prueba Cancelación e Historial Reservaciones Primera Parte

La siguiente figura indica los cambios en la base de datos.

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario3 Usr3	Activo	TT QUITUMBE - GUAYAQUIL	AEROTAXI	26/02/2015 6:00:00	Activa	4

	Cliente	Viaje	Ruta	Cooperativa	Horario	Estado	Asientos
1	Usuario3 Usr3	Activo	TT QUITUMBE - GUAYAQUIL	AEROTAXI	26/02/2015 6:00:00	Cancelada	4

Figura 3.107 Caso de Prueba Cancelación e Historial Reservaciones Segunda Parte

La siguiente figura indica el historial de reservaciones de un usuario.

Figura 3.108 Caso de Prueba Cancelación e Historial Reservaciones Tercera Parte

3.3.2 PRUEBAS DE CARGA

Son pruebas que verifican el funcionamiento del servidor al configurar distintos parámetros, de esta forma se refleja el comportamiento bajo diferentes circunstancias como número de usuarios, respuestas ejecutadas o tiempo.

El servidor forma parte del ambiente de pruebas lo cual implica que tiene capacidades limitadas en cuanto a sus características y funcionamiento. Sin embargo, las pruebas reflejan el comportamiento del prototipo de la aplicación en pequeña escala.

3.3.2.1 Prueba de Carga A

Caso de Prueba	
Nombre	Prueba de Carga A
Descripción	Realizar una prueba de carga con un número de usuarios simultáneos en un tiempo determinado.
Condiciones	Servidor activo.
Ejecución	• Configurar los parámetros de prueba con un número específico de usuarios, un tiempo determinado y varias peticiones. • Ejecutar la prueba de carga al servidor de la aplicación.
Resultado Esperado	• Bajo número de errores. • Tiempos de respuesta de la aplicación menores a cinco segundos.
Conclusión	Cumple con el resultado, el servidor responde con tiempos menores a cinco segundos y sin errores.

Tabla 3.26 Prueba de Carga A

La siguiente figura indica los parámetros bajo los cuales se realiza el test. Los perfiles de usuario corresponden a peticiones HTTP que los clientes realizan, además se configuran diferentes porcentajes de carga para peticiones que son más comunes que otras. La configuración de la prueba en el panel derecho indica cincuenta usuarios de forma constante en dos minutos.

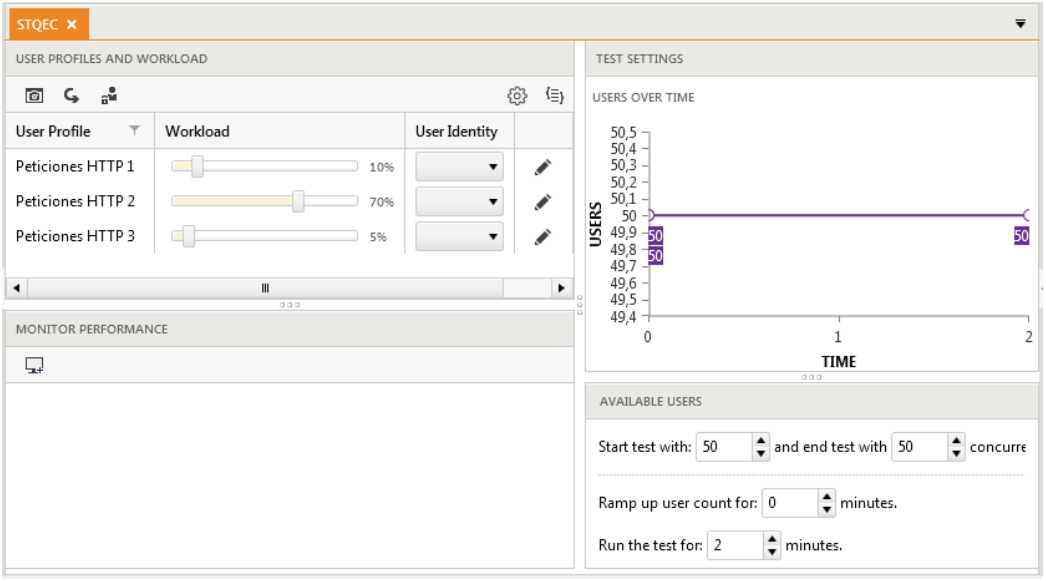

Figura 3.109 Prueba de Carga A Primera Parte

La siguiente figura indica los resultados generales de la prueba:

- Máximo tiempo de respuesta de una petición: 1,8 segundos.
- Máximo número de usuarios activos: 50 usuarios.
- No se registran errores durante la prueba.
- Máximo número de respuestas recibidas por segundo: 51 respuestas.

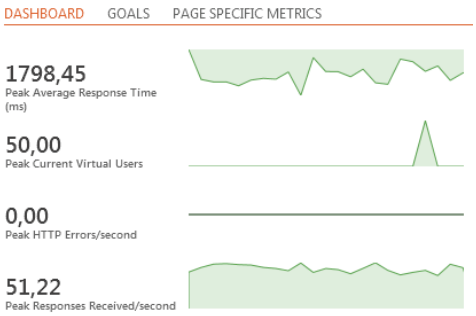

Figura 3.110 Prueba de Carga A Segunda Parte

La siguiente figura indica los resultados a través del tiempo de la prueba. El tiempo de respuesta promedio es 1,5 segundos, el promedio de respuestas por segundo es 45 y el número de usuarios activos promedio es 48. El comportamiento es constante con una carga fija de usuarios utilizando el sistema.

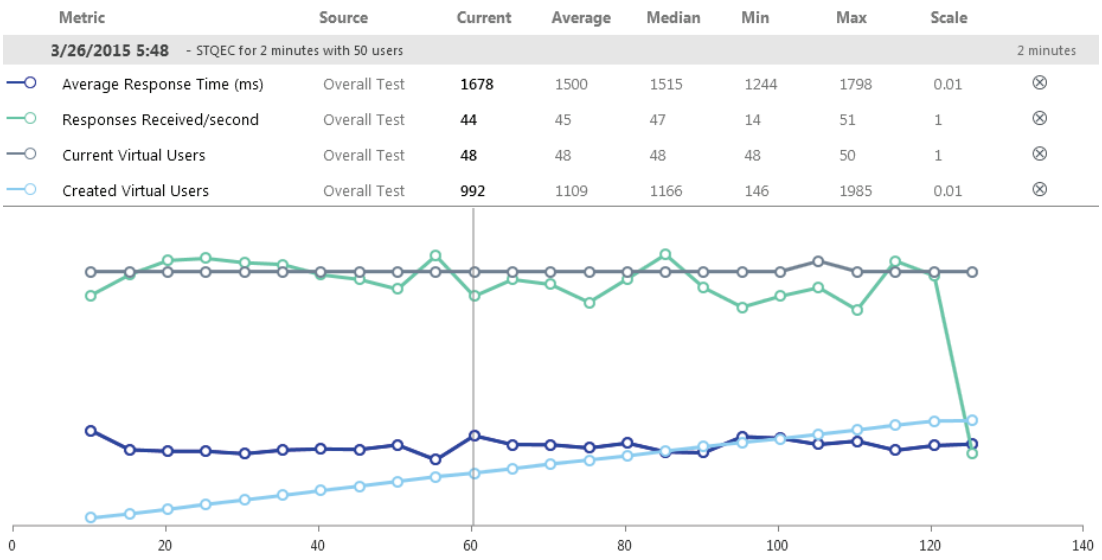

Figura 3.111 Prueba de Carga A Tercera Parte

3.3.2.2 Prueba de Carga B

Caso de Prueba	
Nombre	Prueba de Carga B
Descripción	Realizar una prueba de carga con incremento del número de usuarios en un tiempo determinado.
Condiciones	Servidor activo.
Ejecución	• Configurar los parámetros de la prueba con incremento del número de usuarios, un tiempo determinado y varias peticiones. • Ejecutar la prueba de carga al servidor de la aplicación.
Resultado Esperado	• Bajo número de errores. • Tiempos de respuesta de la aplicación menores a cinco segundos.
Conclusión	Cumple con el resultado, el servidor responde con tiempos menores a cinco segundos y sin errores.

Tabla 3.27 Prueba de Carga B

La siguiente figura indica los parámetros bajo los cuales se realiza el test. Los perfiles de usuario corresponden a peticiones HTTP que los clientes realizan, además se configuran diferentes porcentajes de carga para peticiones que son más comunes que otras. La configuración de la prueba en el panel derecho indica un incremento de diez a noventa usuarios de forma progresiva en dos minutos.

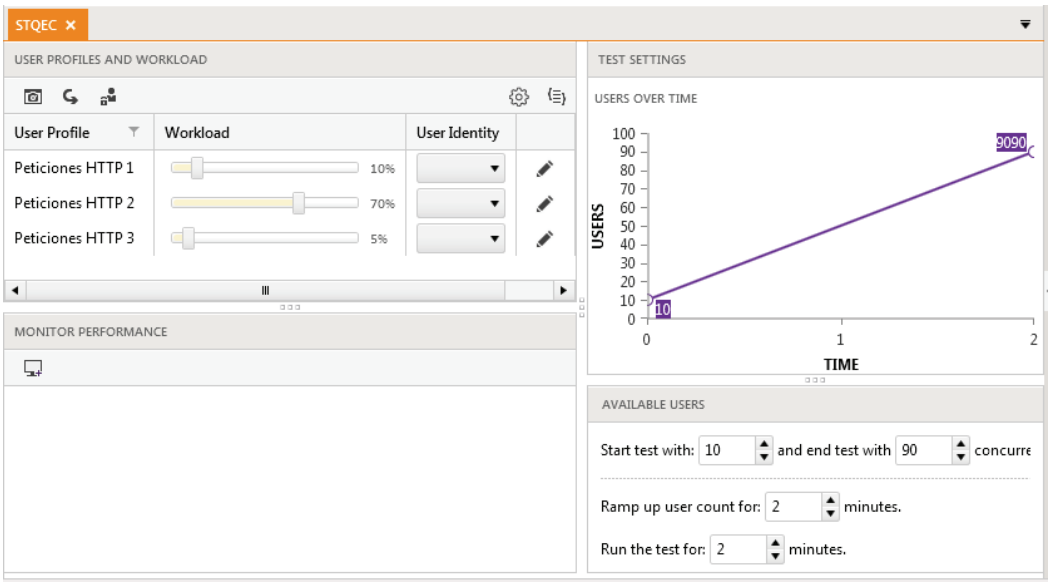

Figura 3.112 Prueba de Carga B Primera Parte

La siguiente figura indica los resultados generales de la prueba:

- Máximo tiempo de respuesta de una petición: 3,6 segundos.
- Máximo número de usuarios activos: 88 usuarios.
- No se registran errores durante la prueba.
- Máximo número de respuestas recibidas por segundo: 61 respuestas.

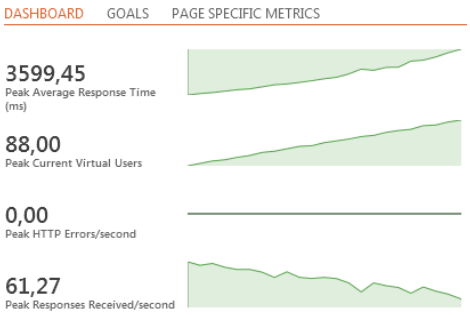

Figura 3.113 Prueba de Carga B Segunda Parte

La siguiente figura indica los resultados a través del tiempo de la prueba. El tiempo de respuesta promedio es 1,7 segundos, el promedio de respuestas por segundo es 46 y el número de usuarios activos promedio es 46. El comportamiento es variable, con el aumento del número de usuarios el tiempo de respuesta aumenta disminuyendo las respuestas recibidas por segundo.

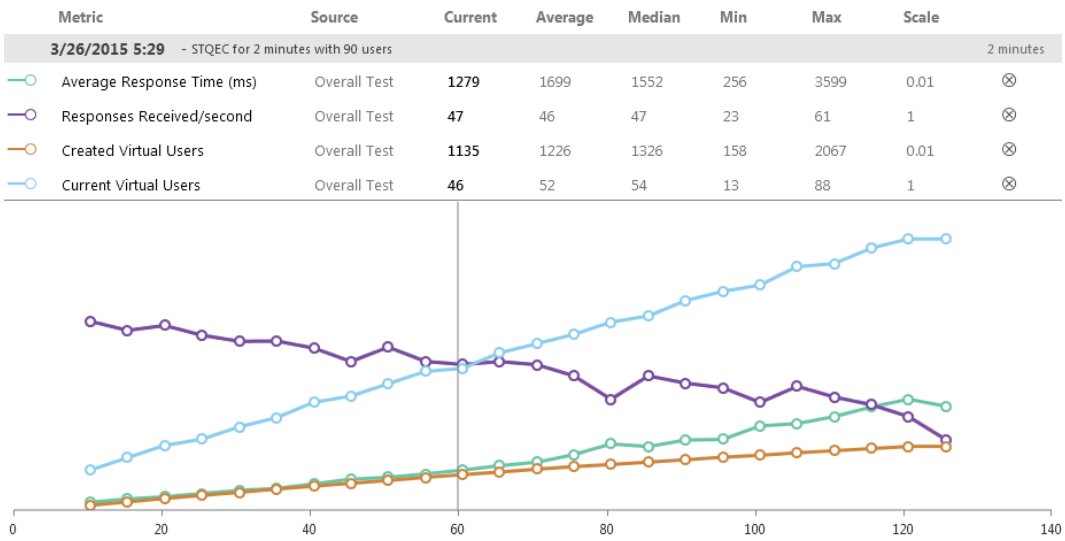

Figura 3.114 Prueba de Carga B Tercera Parte

3.4 ASPECTOS GENERALES DE LA APLICACIÓN

En esta sección se analiza varios aspectos relacionados a la aplicación desde el punto de vista de las diferentes plataformas y el servidor. Se toma en cuenta la compatibilidad entre dos versiones de cada una de las plataformas, la seguridad a través del protocolo HTTPS y finalmente las implementaciones de las aplicaciones cliente y servidor.

3.4.1 COMPATIBILIDAD

La compatibilidad es un aspecto importante en el desarrollo de aplicaciones. Los fabricantes generan nuevas versiones de sistemas operativos y las aplicaciones deben adaptarse o modificarse de acuerdo a las nuevas características para funcionar adecuadamente.

Es por ello que se examinan las consideraciones a tomar en cuenta al momento de crear las aplicaciones para diferentes versiones en las plataformas *Android* y *Windows Phone*.

3.4.1.1 *Android* [W48]

La compatibilidad se analiza entre la versión de *Android* 2.3.3 (API *Level* 10) y la versión de *Android* 4.0.3 (API *Level* 15). Se toma en cuenta estas versiones ya que existieron importantes incorporaciones para el desarrollo de aplicaciones a partir del nivel 11 del API. En la figura 3.115 se muestra la pantalla inicial de las dos versiones, las cuales difieren principalmente en su aspecto y en los controles que permiten la navegación.

Los cambios de una versión pueden incluir aspectos como actualizaciones, el entorno de ejecución o la parte visual del sistema. El código fuente relacionado a la configuración de la aplicación, así como del interfaz de usuario pueden requerir modificaciones.

Figura 3.115 Pantallas iniciales de diferentes dispositivos *Android*

3.4.1.1.1 Versión de Android

La versión de *Android* de la aplicación y la plataforma sobre la cual funciona se definen en las propiedades del proyecto. Al configurar la versión mínima del API, la aplicación no funciona en plataformas anteriores. Existe compatibilidad a las versiones superiores del sistema operativo.

3.4.1.1.2 Componentes [W49]

Los nuevos elementos de un API se configuran mediante paquetes de soporte. En el caso de los fragmentos existe el componente *Android Support Library v4*. Las aplicaciones del API 4 al API 10 son capaces de usar fragmentos al añadir el componente y se puede cambiar la versión mínima de *Android*.

Otro paquete importante es *Android Support Library v7 AppCompat*. Este elemento incorpora la funcionalidad de la barra de acción introducida en las nuevas versiones de *Android* y permite manejar los menús de manera similar.

La figura 3.116 muestra dos componentes que pueden ser usados para incluir la compatibilidad con versiones anteriores de *Android*.

Figura 3.116 Bibliotecas de soporte de versiones de *Android*

3.4.1.1.3 Visualización

Los elementos en diferentes versiones pueden cambiar en su aspecto. Entre las versiones analizadas las principales diferencias son:

- Los controles tienen un diseño diferente en cuanto a su aspecto visual entre las aplicaciones.
- La información presentada en diferentes controles puede visualizarse incompleta o difícil de apreciar al usuario.
- La estructura del menú de una vista se realiza mediante el botón de menú, en las nuevas versiones se puede realizar a través de la barra de la aplicación.

La figura 3.117 muestra las diferencias de visualización en la pantalla inicial.

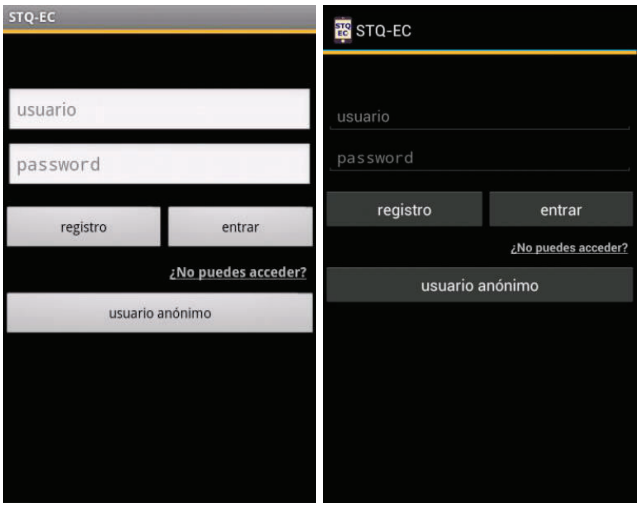

Figura 3.117 Pantallas iniciales en diferentes versiones de *Android*

La figura 3.115 muestra las diferencias de visualización en el menú de opciones entre las diferentes versiones. En la imagen de la izquierda el menú se encuentra en la parte inferior de la vista, cada opción muestra una imagen conjuntamente con el texto descriptivo de la acción. En la imagen de la derecha, el menú se encuentra en la parte superior de la vista, cada opción muestra únicamente el texto descriptivo de la acción.

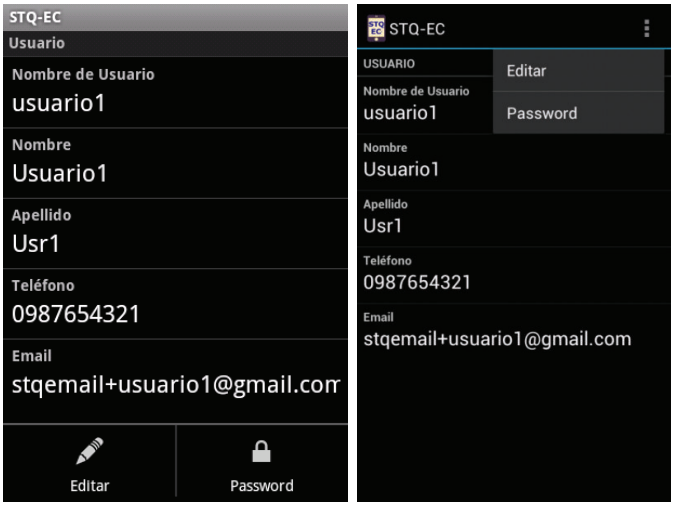

Figura 3.118 Visualización del menú en diferentes versiones de *Android*

3.4.1.1.4 Cambios del código

Las diferencias entre versiones de *Android* tienen impacto sobre el código fuente y la interfaz visual. Por lo tanto, se toma en consideración los siguientes aspectos para su solución:

- La compilación de código fuente puede generar errores si las características de una plataforma no se encuentran en la otra plataforma. El código incompatible debe ser cambiado o eliminado adecuadamente.
- En el interfaz de usuario en las diferentes vistas se debe verificar que la información se visualiza correctamente y se procede con los cambios si es necesario.
- La compilación condicional permite ejecutar un código que funcione correctamente dependiendo de la versión.

3.4.1.2 *Windows Phone* [W50]

La compatibilidad se analiza entre la versión *Windows Phone* 7 y *Windows Phone* 8. Se toma en cuenta estas versiones ya que el núcleo del sistema operativo de los dispositivos es diferente.

Figura 3.119 Pantallas iniciales de diferentes dispositivos *Windows Phone*

3.4.1.2.1 Versiones

El hardware y el software del equipo son diferentes entre las versiones. Una aplicación compilada para *Windows Phone* 7 es compatible con *Windows Phone* 8, pero no en sentido contrario, ya que una aplicación compilada para *Windows Phone* 8 no funcionará en dispositivos *Windows Phone* 7.

3.4.1.2.2 Ejecución

Para proveer la compatibilidad a aplicaciones desarrolladas en versiones anteriores, el sistema operativo de la versión 8 se encarga de emular el comportamiento de la versión 7. En este caso la aplicación ha sido compilada para la versión 7 y se ejecuta en la versión 8.

En el caso de una aplicación recompilada para *Windows Phone* 8, la aplicación funciona directamente en el nuevo entorno de ejecución. Puede presentarse impacto tanto en aspectos visuales como en la ejecución del código.

3.4.1.2.3 Implementación [W51]

Para implementar la aplicación y que sea funcional se usan diferentes enfoques:

- Usar la compilación de la versión 7, funciona en las dos plataformas pero no aprovecha las capacidades incluidas de la versión 8.
- Creación de versiones diferentes para las plataformas, se vuelve complicado la actualización entre los proyectos pero cada versión funcionará correctamente en su plataforma específica.
- Crear bibliotecas o código fuente compartido, permite definir las características del código dependiendo de la versión.

En *Windows Phone* el impacto que puede tener una versión de instalación depende a su vez de las funcionalidades y los elementos con los cuales se desarrolla la aplicación.

En el código la aplicación desarrollada, la aplicación puede compilarse con la versión 8 sin problemas y no requiere cambios, por lo tanto se puede utilizar la primera opción en la implementación de la aplicación.

En el aspecto visual, los elementos y la visualización de los mismos son similares debido al enfoque de la implementación. En general si se comparan las pantallas de la aplicación no se aprecia diferencias en los elementos principales de la pantalla ni en la barra que contiene los elementos del menú.

En la figura 3.120 se muestra las vistas de la aplicación en diferentes versiones. En la imagen izquierda la aplicación en *Windows Phone* 7 y en la imagen derecha la aplicación en *Windows Phone* 8.

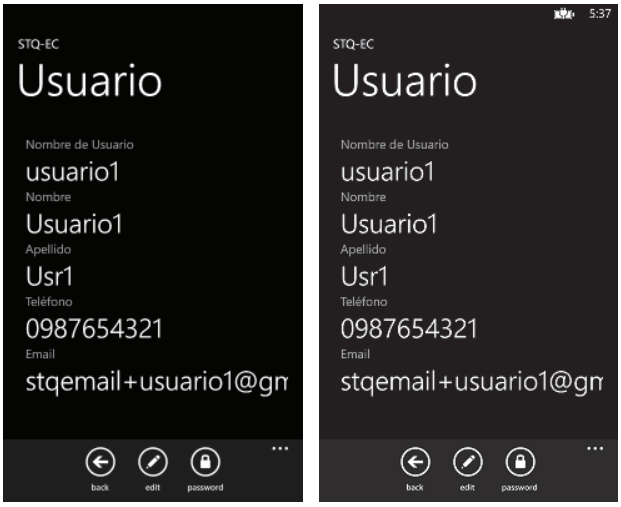

Figura 3.120 Pantallas de la aplicación de diferentes versiones *Windows Phone*

3.4.2 INFORMACIÓN Y ERRORES

La aplicación controla de varias maneras la información y los errores que pueden surgir al utilizar el sistema. De esta manera el usuario conoce el estado de la aplicación y las acciones que puede realizar.

En el caso de los datos que un usuario ingresa en un campo, la aplicación se acopla al tipo de información que recibe por parte del usuario adecuando el teclado para la entrada de datos. El menú de opciones en una vista se activa si el usuario ha ingresado los datos de forma correcta.

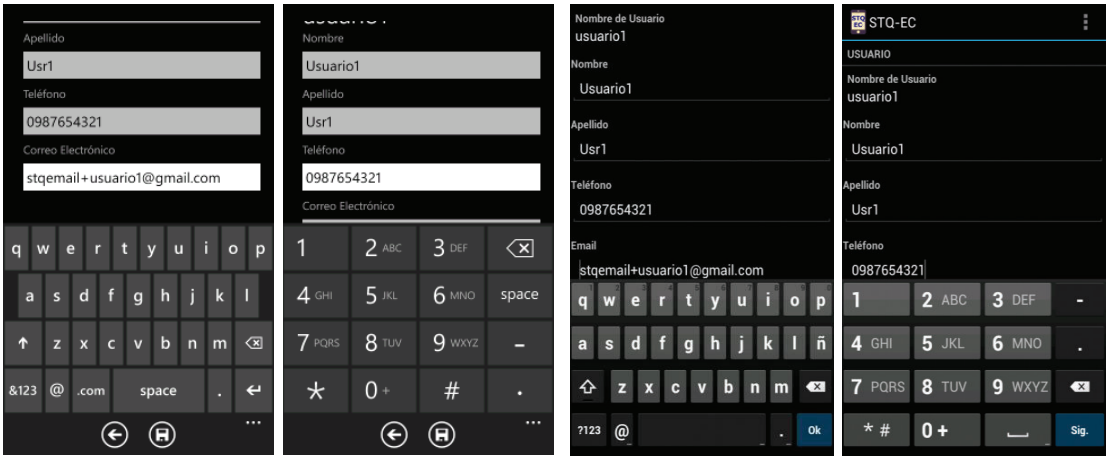

Figura 3.121 Pantallas de la aplicación con teclados diferentes

La aplicación cuenta con una barra de progreso que indica al usuario que una acción se está ejecutando y desaparece cuando termina su ejecución. En la vista inicial se presenta en la parte superior y en las otras vistas en la parte inferior.

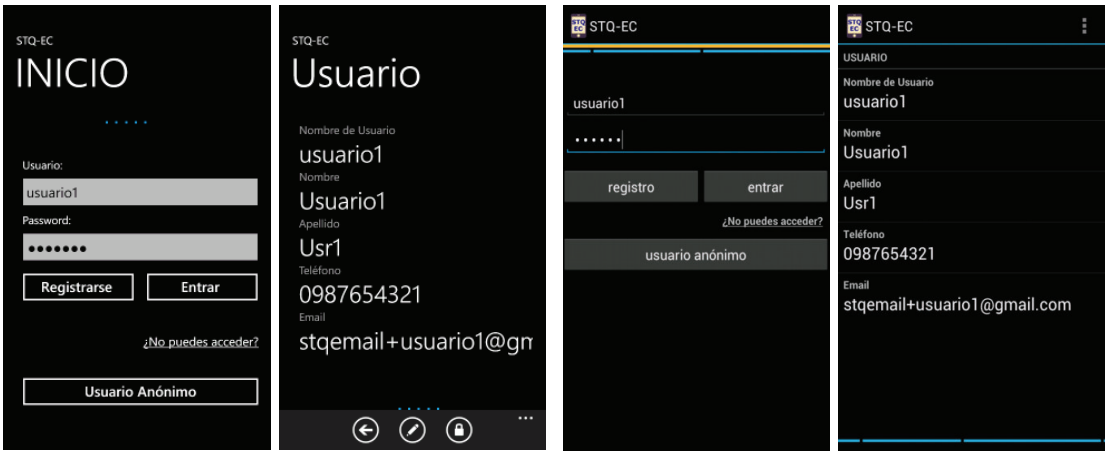

Figura 3.122 Pantallas de la aplicación con la barra de progreso

Para el manejo de errores, debido a causas como la conexión a Internet o problemas con el servidor, la aplicación informa al usuario de diferentes formas el problema. En la vista inicial, mediante una etiqueta de texto en la parte superior, en otras vistas se informa al usuario mediante un mensaje que se despliega en la pantalla, la barra de progreso también se detiene. Cuando se requiere un mensaje de confirmación si la acción ha sido correcta o incorrecta, se puede redirigir al usuario a una pantalla informativa con el resultado.

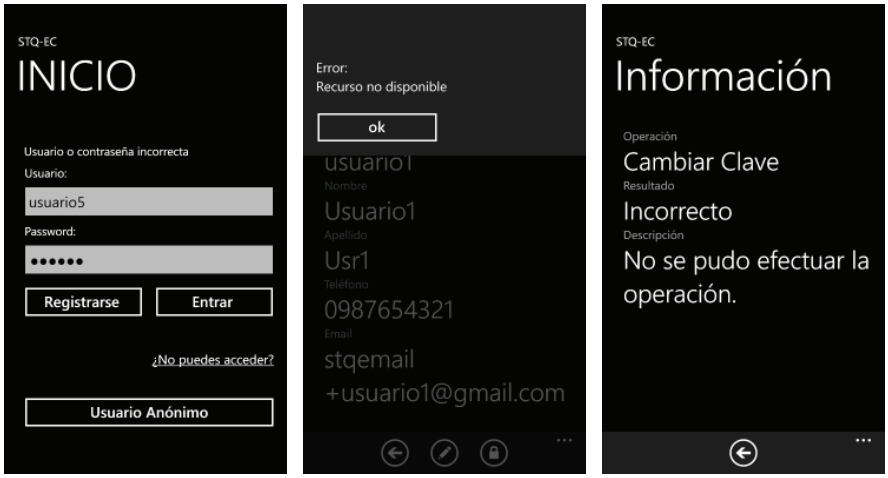

Figura 3.123 Pantallas con mensajes informativos al usuario en *Windows Phone*

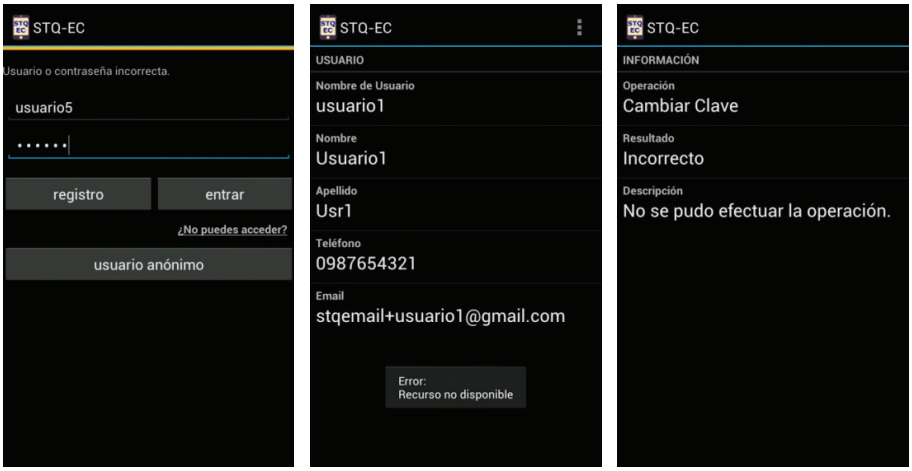

Figura 3.124 Pantallas con mensajes informativos al usuario en *Android*

3.4.3 HTTPS

La seguridad es un aspecto importante en la comunicación entre cliente y servidor. Los datos transmitidos pueden contener información sensible, la cual debe protegerse de forma adecuada. Una de las formas en que los servicios REST se protegen es mediante el uso de un certificado SSL (*Secure Sockets Layer*) y el protocolo HTTPS.

Para implementar esta seguridad se requiere configurar el servidor de forma adecuada. Por un lado, para la publicación de la aplicación se necesita un dominio en Internet. Por otra parte, se necesita un certificado SSL generado por una autoridad certificada para el dominio correspondiente.

Una vez solicitado el certificado, se procede a instalarlo en el servidor IIS. Posteriormente, se configura los enlaces para que el servidor acepte las conexiones HTTPS. El cliente debe configurarse para utilizar los URI correspondientes del servicio si el servidor acepta únicamente peticiones HTTPS.

La figura 3.125 muestra el acceso al servicio mediante un dominio. En la imagen de la izquierda se aprecia el acceso mediante http y en la imagen de la derecha el acceso mediante https. La configuración para permitir el acceso al servicio por medio de http o https se define en el servidor.

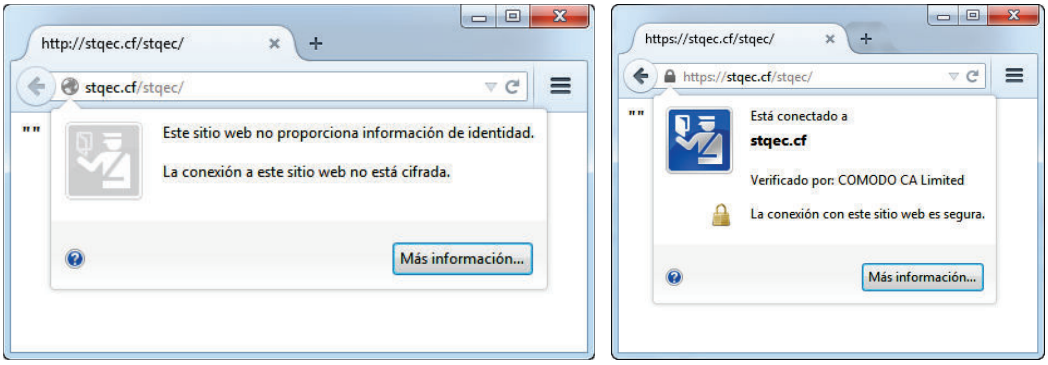

Figura 3.125 Acceso a una operación mediante HTTP o HTTPS

3.4.4 IMPLEMENTACIONES

La implementación de las aplicaciones consiste en realizar una publicación de la versión final que ha sido desarrollada y probada e implementarla en el destino correspondiente. En el caso de la aplicación de servidor se implementa en el servidor IIS, en el caso de las aplicaciones cliente se implementan en los dispositivos móviles de cada plataforma como teléfonos inteligentes o tablets.

En este proyecto se realiza la implementación de las aplicaciones cliente sobre los dispositivos directamente, es decir, se generan los archivos correspondientes y dependiendo de la plataforma se instalan en el dispositivo. La publicación de la aplicación en las tiendas de aplicaciones particulares para cada aplicación no forma parte de este proyecto.

3.4.4.1 Aplicación Servidor [W52]

La implementación de la aplicación web requiere la generación del archivo de publicación y su posterior importación en el servidor web correspondiente.

Las conexiones a la base de datos y al servidor de reportes se establecen mediante los archivos de configuración en los proyectos correspondientes. Se debe verificar que las conexiones sean las correctas y la base de datos sea accesible a la aplicación en el servidor.

3.4.4.1.1 Generación del paquete de implementación

El paquete de implementación se genera a través de Visual Studio. La opción *Package/Publish Web* en las propiedades del proyecto permite definir las características y ubicación del archivo generado.

La compilación se establece en *Release* y se configura la ubicación donde se genera el paquete así como el nombre de la aplicación en el servidor.

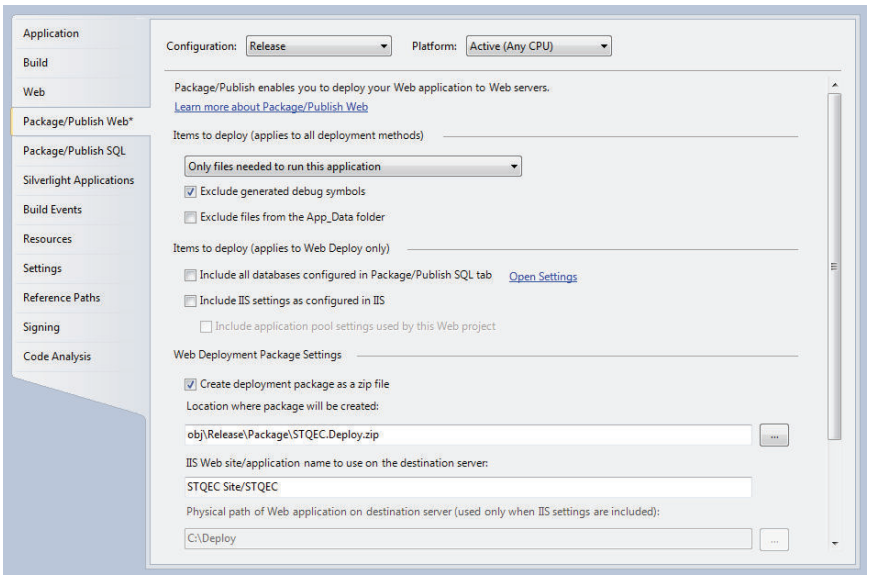

Figura 3.126 Configuración de publicación de la aplicación servidor

Posteriormente, se genera el paquete a través de la opción *Project/Build Deployment Package*. Esta acción genera varios archivos, el archivo en formato ZIP es el que se utiliza en este caso para su implementación en el servidor.

Nombre	Tipo	Tamaño
PackageTmp	Carpeta de archivos	
STQEC.Deploy	Archivo WinRAR ZIP	2.606 KB
STQEC.Deploy.deploy-readme	Documento de texto	4 KB
STQEC.Deploy.SetParameters	Documento XML	1 KB
STQEC.Deploy.SourceManifest	Documento XML	1 KB
STQEC.Deploy.deploy	Script de comandos de Windows	13 KB

Figura 3.127 Archivos de implementación del servidor

3.4.4.1.2 Importación de la aplicación en el servidor

En el servidor, se abre la consola de administración de IIS, se ubica en el sitio en el cual se va a implementar la aplicación y se procede a importar el paquete que se generó anteriormente.

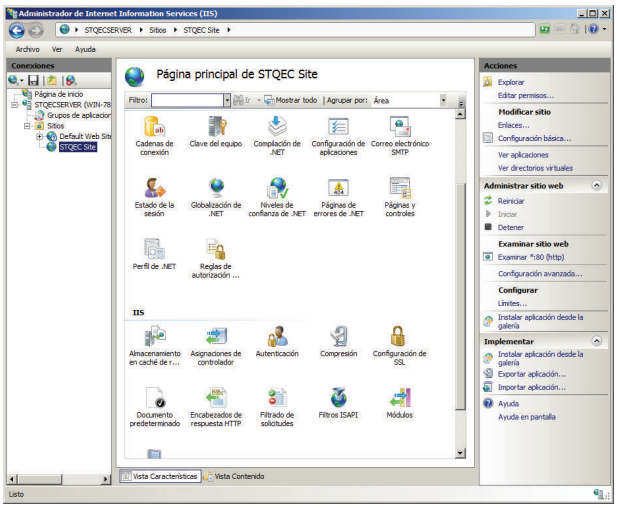

Figura 3.128 Consola de Administración de IIS en el servidor

Se configura mediante el asistente los datos de la implementación y se procede a finalizar la acción. Una vez concluido la aplicación web es accesible a los clientes.

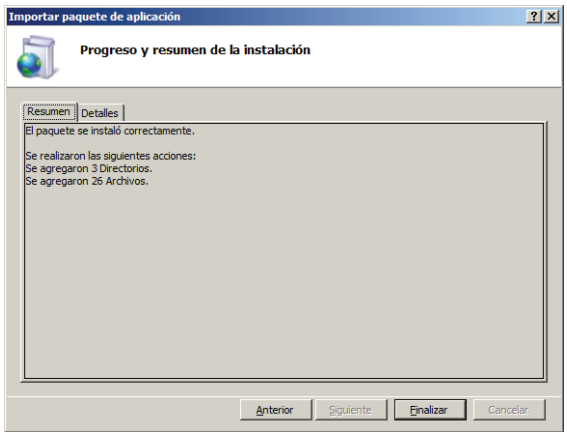

Figura 3.129 Asistente de importación de la aplicación en el servidor

La figura 3.130 muestra la aplicación implementada en el servidor.

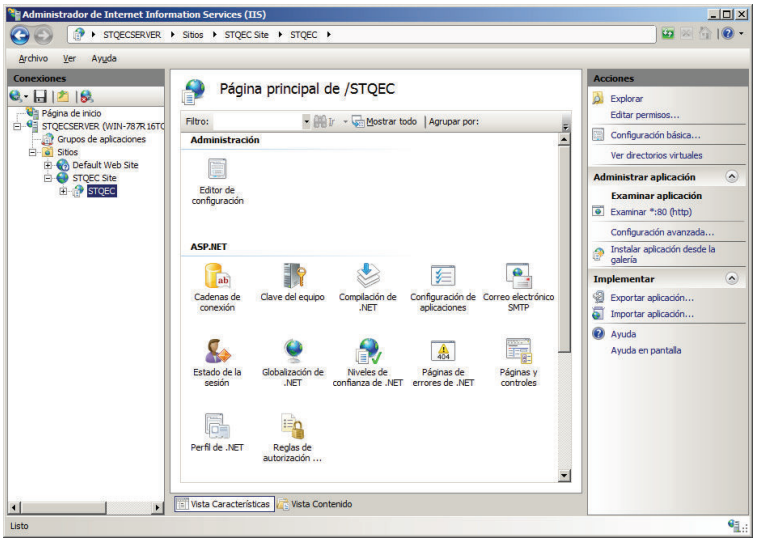

Figura 3.130 Aplicación importada en el servidor IIS

3.4.4.2 *Windows Phone* [W53]

La implementación en *Windows Phone* requiere que la aplicación sea publicada y posteriormente dependiendo del destino se toma en cuenta las configuraciones necesarias.

La implementación de una aplicación en un dispositivo *Windows Phone* requiere un conjunto de requerimientos para habilitar el dispositivo para la instalación. En primer lugar, el dispositivo debe registrarse para desarrollo para lo cual es necesario lo siguiente:

- Las herramientas de desarrollo, que viene incluidas en el SDK de la versión correspondiente al sistema del dispositivo.
- Un dispositivo *Windows Phone* con la una de las versiones 7.1, 8.0 ó 8.1.
- Una cuenta Microsoft, lo cual permite probar la aplicación sobre un dispositivo, o una cuenta del centro de desarrollo que permite probar la aplicación hasta en tres dispositivos y brinda características como la publicación de aplicaciones en la tienda de aplicaciones.
- Configurar el dispositivo adecuadamente para el registro, *Windows Phone* 7.1 requiere además del software *Zune*.

Una vez cumplidos los requisitos se procede al registro del dispositivo a través de la herramienta *Windows Phone Developer Registration*.

Figura 3.131 Herramienta de registro de *Windows Phone*

Posteriormente el dispositivo está listo tanto para probar, depurar e implementar aplicaciones. En el caso de la implementación se usa la herramienta de implementación de aplicaciones, incluida en el SDK, en la cual se escoge el tipo de destino en el cual se implementa, el archivo XAP correspondiente a la aplicación y se selecciona implementar, mostrándose el estado del procedimiento realizado.

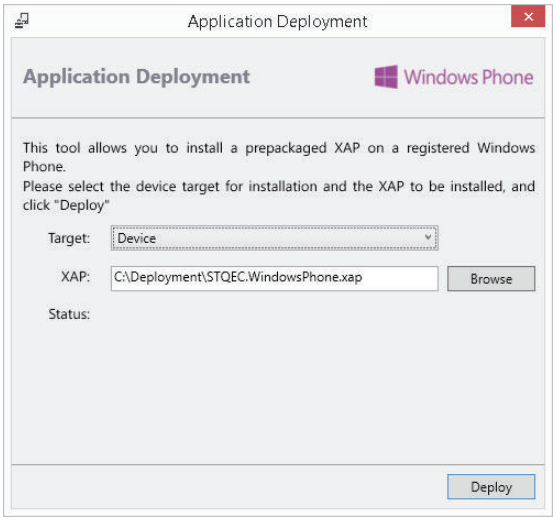

Figura 3.132 Herramienta de implementación de aplicaciones de *Windows Phone*

3.4.4.3 *Android* [W54]

La implementación en *Android* requiere que la aplicación sea publicada y posteriormente distribuida a los clientes para su instalación. La publicación sigue los siguientes pasos:

3.4.4.3.1 Compilación de la aplicación para Release

Una aplicación en modo *release* no necesita toda la sobrecarga que se incluye en modo *debug*. Por tanto, debe configurarse varios atributos en la compilación para obtener el archivo final, se tiene en consideración lo siguiente:

- **Deshabilitar la depuración:** esta opción impide que otros usuarios puedan hacer depuración al código, brindando seguridad a la aplicación.
- **Icono de la aplicación:** el icono es requerido para mostrar una imagen de la aplicación, se configura en las propiedades del proyecto, en las opciones de manifiesto.
- **Versión de la aplicación:** se utiliza como dato informativo y para actualizaciones, al igual que el icono se configura en las opciones de manifiesto.
- **Configuración del *Linker*:** permite reducir el tamaño de la aplicación eliminando archivos que no se utilizan.
- **Compilación en *Release*:** es la acción de compilación del proyecto en general, que permite la generación del archivo APK optimizado para su implementación.

En la figura 3.133 se muestra la configuración del archivo manifiesto a través de la interfaz gráfica, existen varios campos para definir las características de la aplicación.

En la configuración también se definen los diferentes permisos que requiere la aplicación, en este caso para la comunicación con el servidor se necesita marcar la opción INTERNET.

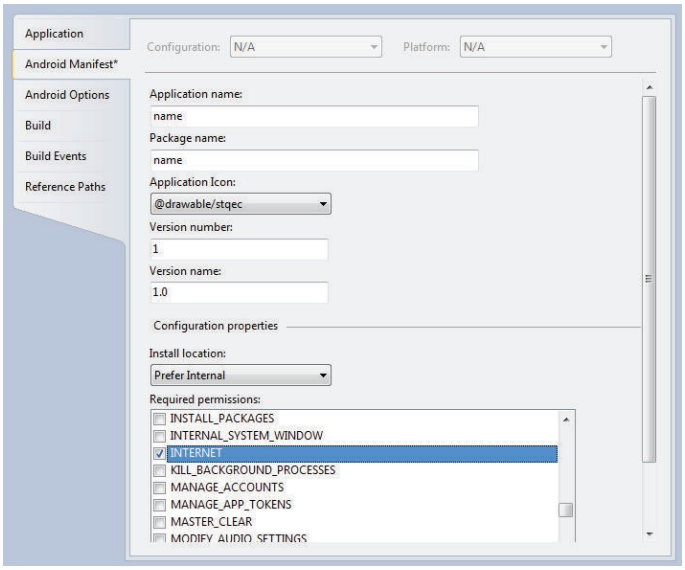

Figura 3.133 Configuración del archivo Manifiesto

La figura 3.134 indica la configuración para la acción de compilación por defecto, *Release*, y la configuración del *Linker* para el archivo final.

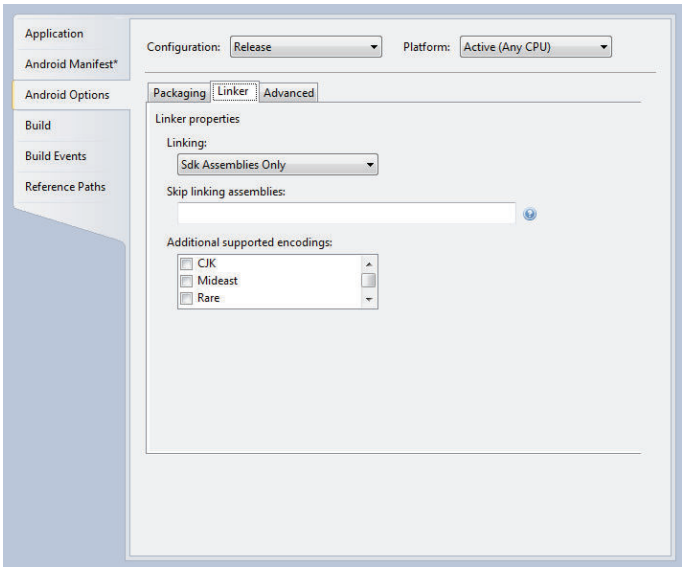

Figura 3.134 Configuración de compilación en *Android*

3.4.4.3.2 Creación de una Llave Privada

Android necesita una llave privada para firmar digitalmente la aplicación. Las aplicaciones que no son firmadas no funcionan sobre el sistema.

Para la creación de la llave privada se utiliza la herramienta de línea de comandos *keytool* del Java SDK, la cual solicita información privada y permite la configuración de ciertas características para la creación de la llave.

3.4.4.3.3 Generación y firma del archivo APK

Para la generación del archivo APK se verifica que la acción de compilación sea *Release*. Visual Studio permite la generación mediante un entorno gráfico, lo cual resulta mucho más fácil y rápido que realizar el proceso por medio de línea de comandos.

Se requiere tener en cuenta la llave generada previamente, así como las credenciales configuradas en la creación de la llave.

Para iniciar el proceso se dirige a la opción *Tools/Android/Publish Android App* o su equivalente dependiendo de la versión de *Xamarin*. A continuación, aparece un cuadro de diálogo como se muestra en la figura 3.135. Se completan los campos solicitados y se selecciona siguiente.

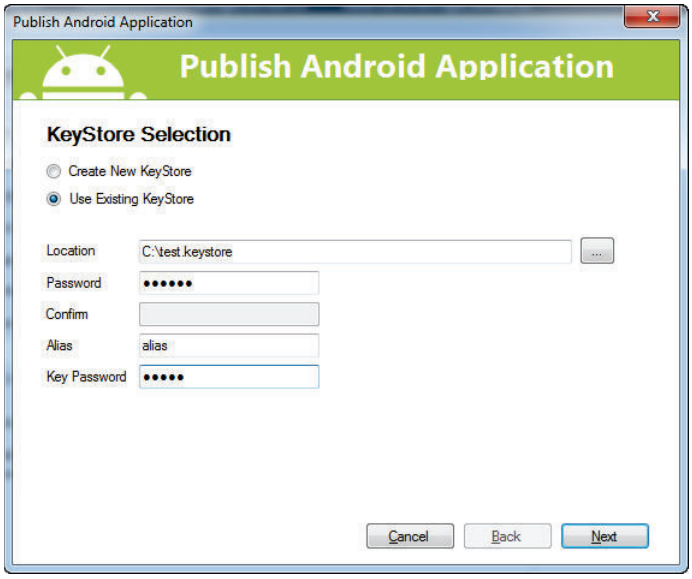

Figura 3.135 Configuración de la publicación de la aplicación *Android*

Posteriormente, se selecciona la ubicación y el nombre del archivo APK como se muestra en la figura 3.136 y se selecciona Publicar.

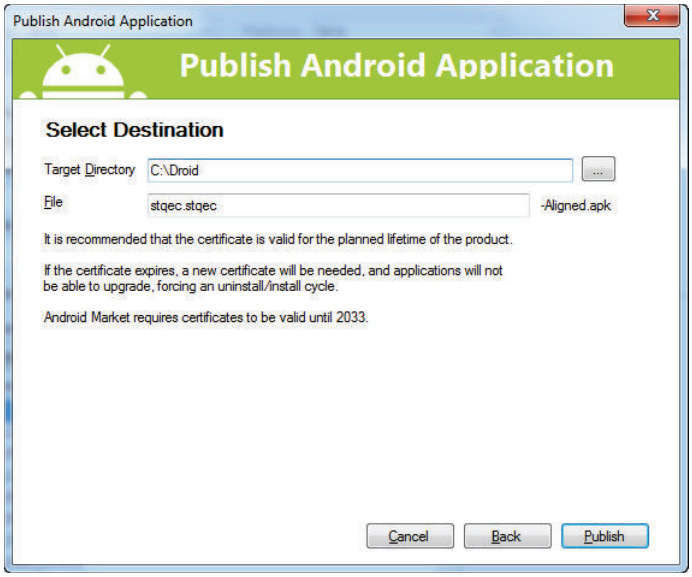

Figura 3.136 Generación de la aplicación en *Android*

Finalmente se ha generado el archivo en el directorio especificado, como se muestra en la figura 3.137, y está listo para la distribución e instalación.

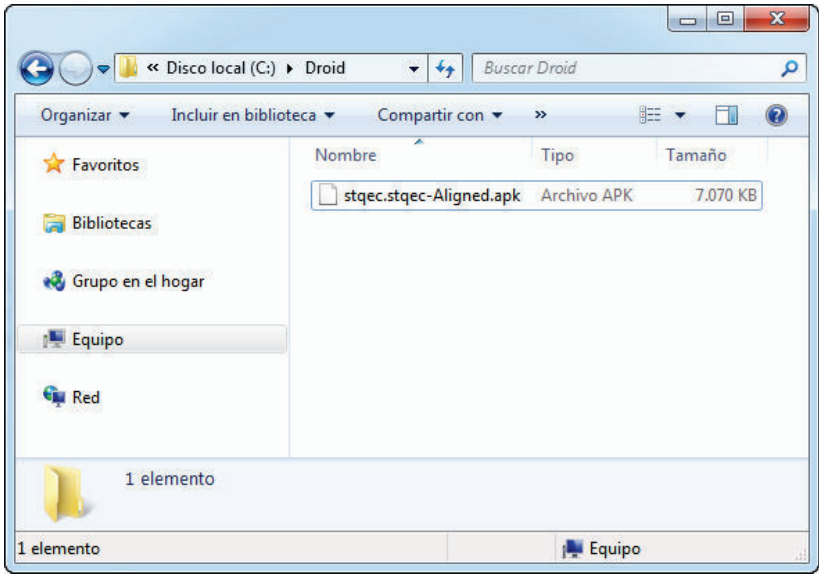

Figura 3.137 Archivo de aplicación generado en *Android*

3.5 CONSIDERACIONES FINALES

En esta sección se hace referencia a determinados aspectos de la aplicación y su diferencia con una implementación en un entorno real. Estas consideraciones se toman en cuenta debido al tipo de proyecto y las restricciones que existen al desarrollar una aplicación multiplataforma bajo un solo lenguaje.

Existe una aplicación por plataforma con el acceso de distintos usuarios a través de una misma pantalla. En la práctica para manejar adecuadamente los usuarios, cada usuario debería tener acceso a una aplicación independiente de los otros. Por otra parte, se debe restringir el acceso al servidor, dependiendo del usuario la conexión puede realizarse en la LAN de la empresa o a un servidor por separado.

Para manejar los usuarios a nivel de base de datos existe una sola tabla, y usando la herencia se identifica el usuario correspondiente, esto agrega un nivel extra de dificultad si bien para un prototipo es aceptable por la presentación y utilidad que se requiere. Para una mejor adaptación de la base de datos, tanto en las relaciones de la tabla y el manejo de los distintos tipos de usuario se debe crear tablas independientes y controlar la información por separado.

En el caso del manejo de terminales, en Quito existe el terminal de Carcelén y de Quitumbe y una sola empresa que realiza la administración. Con el propósito de presentar claramente la aplicación se puede crear entidades de tipo Terminal y realizar los procesos de administración y asociación con otras entidades.

En general, la mayoría de aplicaciones de software tienen parámetros que permiten su funcionamiento y que se configuran únicamente en *backend*. El sistema requiere de ciertos parámetros que se configuran en la aplicación servidor y en la base de datos para manejar conexiones, estados de entidades, etc.

Como resultado de algunas consideraciones la base de datos en algunas tablas, las relaciones pueden cambiar en un rediseño para la segmentación de los tipos de usuarios o en la definición de entidades específicas.

3.6 COSTOS

Para definir los costos del proyecto se hace un análisis por separado del hardware, software y los recursos para el desarrollo de la aplicación.

En el Anexo E se especifica los valores de los costos para el software y hardware y los respectivos proveedores en el mercado local. También se hace referencia al costo de los recursos y el valor promedio que se maneja actualmente como remuneración en empresas de desarrollo de software.

3.6.1 COSTOS DE HARDWARE

El principal elemento en cuanto al hardware es un computador de escritorio. El equipo debe cumplir con características necesarias para emular adecuadamente las máquinas virtuales requeridas para el desarrollo de *Android* y *Windows Phone*. Las principales características son:

- Procesador: Intel Core i3.
- Memoria RAM: 4 GB.
- Disco Duro: 500 GB.
- Unidad de DVD.

Un equipo de estas características permite desarrollar y probar rápidamente las aplicaciones y en distintos emuladores o plataformas a la vez. Una máquina de menores características implica mayor tiempo en el desarrollo e incluso puede limitar la funcionalidad del software requerido.

Los otros elementos importantes son los dispositivos móviles como teléfonos inteligentes o tabletas que permiten realizar las pruebas de implementación sobre un equipo real. El único requisito es que dispongan del sistema operativo para el cual se desarrolla la aplicación, es decir *Windows Phone* o *Android* para teléfonos inteligentes y *Android* para tabletas.

Ítems	Costo USD
Equipo de Desarrollo	$ 535
Dispositivos Móviles	$ 320
Total	**$ 855**

Tabla 3.28 Costos del Hardware

3.6.2 COSTOS DE SOFTWARE

La mayoría de elementos del presente proyecto son versiones gratuitas de los programas, lo cual muestra que se puede desarrollar a través del lenguaje C# a través de distintas alternativas como son las versiones de evaluación, versiones *express* o software libre sin ningún tipo de inconveniente.

En cuanto al desarrollo realizado en este proyecto, se necesita licencia para el sistema operativo *Windows 7 Ultimate*. En el caso de *Visual Studio* no se necesita licencia debido a la versión *Community* de *Visual Studio* 2013, que incluye todas las funciones de la versión *Professional* de forma gratuita tomado en cuenta la utilización del producto. Para la publicación de una aplicación en un entorno real también es necesaria la licencia de *Xamarin*.

Ítems	Costo USD
Windows 7 Ultimate	$ 55
Xamarin suscripción *Indie*	$ 25
Visual Studio Community 2013	-
Total	**$ 80**

Tabla 3.29 Costos del Hardware

3.6.3 COSTOS DE LOS RECURSOS

Los costos de la aplicación involucran el tiempo empleado en el desarrollo conjuntamente con elementos como materiales de oficina, Internet, etc.

Ítems	Horas	Valor Hora	Costo USD
Desarrollo	800	$ 5	$ 4000
Documentación	200	$ 5	$ 1000
Materiales de Oficina			$ 200
Internet			$ 200
Otros			$ 400
Total			**$ 5800**

Tabla 3.30 Costos de los Recursos

3.6.4 COSTO TOTAL DEL PROYECTO

El costo total del proyecto se basa en los costos de hardware, costos de software y costos de recursos.

Ítems	Costo USD
Costo del Hardware	$ 855
Costo del Software	$ 80
Costo de los Recursos	$ 5800
Total	**$ 6735**

Tabla 3.31 Costo Total del Proyecto

CAPÍTULO IV

4 CONCLUSIONES Y RECOMENDACIONES

En el cuarto capítulo se presentan las conclusiones y las recomendaciones que son resultado de la investigación y el desarrollo realizados en el presente proyecto.

4.1 CONCLUSIONES

- El prototipo desarrollado permite manejar de forma adecuada mediante las aplicaciones cliente y servidor la información del transporte interprovincial de la ciudad de Quito para diferentes tipos de usuarios, los cuales interactúan con el sistema y obtienen la información de acuerdo a su rol para consultar, modificar o crear los registros de la base de datos del sistema.

- La aplicación funciona correctamente en dos versiones de las plataformas *Android* y *Windows Phone* y hace uso de otras aplicaciones como son sitios web, *Twitter* o *Google Maps* para mostrar información con mayor claridad a los usuarios finales.

- El presente proyecto constituye una propuesta para facilitar el acceso al servicio que prestan los terminales interprovinciales mediante la aplicación para los sistemas *Windows Phone* y *Android*, la arquitectura *Monocross* está diseñada de forma que permite extender el desarrollo multiplataforma a otros ambientes como puede ser para el sistema iOS, para un sitio web o aplicaciones de escritorio, entre otros.

- El uso de una metodología que incorpore elementos descriptivos y visuales del funcionamiento de la aplicación como son los diagramas casos de uso, diagramas de clases, diagramas de actividad ayuda a diseñar y desarrollar el sistema de mejor manera.

- El desarrollo del interfaz de usuario es un aspecto complejo debido a que cada plataforma define sus propios elementos y la forma como se relaciona con el código. Una interfaz sencilla que utiliza elementos visuales que se puedan acoplar a diferentes pantallas y que sean comunes entre las diferentes vistas simplifica el desarrollo y facilita la utilización de la aplicación a los usuarios.

- La segmentación del sistema en diferentes niveles tanto en el cliente como en el servidor permite que la estructura de la aplicación se pueda modificar sin que exista gran impacto en las desarrollo o en el mantenimiento del sistema y que cada nivel se desarrolle con cierta independencia de los otros niveles.

- El lenguaje C# se puede utilizar para desarrollar diferentes tipos de aplicaciones para distintos tipos de plataformas, esto se debe a la estandarización del lenguaje así como los componentes principales del .NET *Framework* a través de organizaciones internacionales.

- La optimización de la aplicación en la comunicación del cliente y servidor permite enviar la información adecuada y que la transferencia sea más rápida. A su vez, el limitar la cantidad de datos y enviar la información paginada permite el ahorro de recursos. Esto es apreciable ya en una implementación en Internet o con gran cantidad de usuarios.

- El disponer de las herramientas adecuadas de software y hardware disminuye los tiempos de desarrollo. Es necesario contar con un equipo de desarrollo que cuente con las características de emulación de los dispositivos reales debido a que las pruebas de aplicaciones en arquitecturas como ARM se demoran minutos, en tantos que las mismas pruebas en la arquitectura x86 se pueden realizar en segundos.

- La utilización de una base de datos conjuntamente con un ORM facilita el desarrollo de la aplicación debido a que el desarrollador no tiene que ocuparse de aspectos complejos de la comunicación con la base de datos y en su lugar puede ocuparse del desarrollo con el enfoque orientado a objetos.

- La aplicación desarrollada se basa en la arquitectura *Monocross*, la cual utiliza código abierto. Esta característica facilita la personalización del código final principalmente en la navegación, control de errores y las interfaces de usuario y de esta forma se puede adaptar la aplicación según lo que se requiera implementar.

- La aplicación necesita establecer aspectos de seguridad en distintos niveles. De esta forma se logra prevenir los riesgos en un entorno como puede ser el Internet. Sin embargo, la seguridad es un proceso por lo que continuamente se necesita analizar los posibles fallos y tomar las medidas adecuadas.

- La implementación de la seguridad añade complejidad al diseño, desarrollo y funcionamiento del sistema. Sin embargo, es necesario incorporar características que brinden privacidad, integridad y disponibilidad para mantener protegido el sistema y su información.

4.2 RECOMENDACIONES

- Es importante que las metodologías y técnicas utilizadas se adapten al tipo de proyecto y sus características para permitir un desarrollo más rápido y de mejor calidad del producto.

- Existen diferentes herramientas para la implementación de un determinado producto de software, por tanto se recomienda definir claramente en un proyecto que tipo de tecnología utilizar para evitar pérdida de tiempo y recursos.

- En un desarrollo multiplataforma como el realizado se recomienda utilizar el sistema original para el cual una tecnología está diseñada, en este caso C# con *Windows Phone* y luego extender la aplicación a otras plataformas como *Android* o incluso iOS.

- Las herramientas tecnológicas continuamente se actualizan, se recomienda mantener en lo posible actualizado el software para desarrollo debido a que las actualizaciones corrigen errores, presentan mejor rendimiento e incorporan características que facilitan su uso.

- Existen nuevos productos de software de Microsoft como APIs o *Frameworks* que permiten el desarrollo de servicios REST o la utilización de ORMs con distintas bases de datos. Es recomendable al actualizar la aplicación o al hacer nuevos desarrollos hacer uso de estas nuevas tecnologías conjuntamente con el potencial y las ventajas con las cuales se han construido.

- En un proyecto de software el personal debe tener tareas definidas y ocuparse de distintos aspectos que al final se complementen en un producto final. Es complejo que una persona maneje aspectos tanto de programación como del diseño de interfaces de usuario avanzadas, por tanto es recomendable conformar un equipo en el que cada integrante aporte su conocimiento en el área en la que mejor se desenvuelve y tiene mayor experiencia.

REFERENCIAS BIBLIOGRÁFICAS

LIBROS

[L1] Sommerville, I., *Ingeniería del Software*, 7th ed.: Pearson, 2005.

[L2] Barker, J. y Palmer, G., *Beginning C# 2008 Objects: From Concept to Code*. New York: Apress, 2008.

[L3] Moreno, F., *Introducción a la OOP*, 1st ed.: Grupo Eidos, 1999.

[L4] Pressman, R., *INGENIERIA DEL SOFTWARE*, 6th ed., 2005.

[L5] Clark, D., *Beginning C# Object-Oriented Programming*, 1st ed. New York: Apress, 2011.

[L6] Silberschatz, A., Korth, H., y Sudarshan, S., *FUNDAMENTOS DE BASES DE DATOS*, 4th ed. Madrid: McGRAW-HILL, 2002.

[L7] Stallings, W., *Comunicaciones y Redes de Computadores*, Séptima ed. Madrid: Pearson, 2004.

[L8] Troelsen, A., *Pro C# 5.0 and the.NET 4.5 Framework*, 6th ed. New York: Apress, 2012.

[L9] Albahari , J. y Joseph, B., *C# 5.0 IN A NUTSHELL*, 5th ed. California: O'Reilly Media, 2012.

[L10] Firman, M. y Natale, L., *Visual Studio.NET Framework 3.5 para profesionales*, 1st ed. México: Alfaomega, 2010.

[L11] Flanders, J., *RESTful.NET*, 1st ed., O' Reilly, Ed. California, 2009.

[L12] Lakshmiraghavan, B., *Pro ASP.NET Web API Security*, 1st ed. New York: Apress, 2013.

[L13] Shackles, G., *Mobile Development with C#*, 1st ed. California: O'Reilly Media, 2012.

[L14] Zhou Zhinan, Z., *WINDOWS PHONE 7 PROGRAMMING FOR ANDROID AND iOS DEVELOPERS*. Indianapolis: John Wiley & Sons, 2011.

[L15] Olson, S., Hunter, J., Horgen, B., y Kenny, G., Professional Cross-Platform Mobile Development in C#, 1st ed. Indianapolis: John Wiley & Sons, 2012.

PÁGINAS WEB

[W1] Rational,. (2014, Abril) Rational Unified Process. https://www.ibm.com/developerworks/rational/library/content/03July/1000/1251/1251_bestpractices_TP026B.pdf

[W2] Tutorials Point. (2014, Junio) UML Standard Diagrams. http://www.tutorialspoint.com/uml/uml_standard_diagrams.htm

[W3] Megino, J. (2015, Abril) Tipos de relaciones en diagramas de casos de uso. UML. http://www.seas.es/blog/informatica/tipos-de-relaciones-en-diagramas-de-casos-de-uso-uml/

[W4] Tutorials Point. (2015, Enero) DBMS Quick Guide. http://www.tutorialspoint.com/dbms/dbms_quick_guide.htm

[W5] Tutorials Point. (2014, Mayo) ER Model : Basic Concepts. http://www.tutorialspoint.com/dbms/er_model_basic_concepts.htm

[W6] Rouse, M. (2014, Junio) Operating system. http://whatis.techtarget.com/definition/operating-system-OS

[W7] Lingan, J. (2014, Junio) Web server. http://whatis.techtarget.com/definition/Web-server

[W8] Furey, A. y Pottjewijd, A. (2015, Febrero) Integrated Development Environment. http://searchsoftwarequality.techtarget.com/definition/integrated-development-environment

[W9] Skeet, J. (2014, Mayo) Untangling the Versions. http://csharpindepth.com/articles/chapter1/versions.aspx

[W10] Kunal-Chowdhury. (2014, Septiembre) Evolution of C#. http://www.kunal-chowdhury.com/2012/07/evolution-of-c-10-50-what-are-new.html

[W11] Horton, A. (2014, Julio) La evolución de LINQ y su impacto en el diseño de C#. http://msdn.microsoft.com/es-es/magazine/cc163400.aspx

[W12] NET Connected Framework Team. (2014, Mayo) Introducing WCF WebHttp Services in.NET 4. http://blogs.msdn.com/b/endpoint/archive/2010/01/06/introducing-wcf-webhttp-services-in-net-4.aspx

[W13] Microsoft Developer Network. (2014, Mayo) WCF Web HTTP Programming Model Overview. http://msdn.microsoft.com/en-us/library/bb412172(v=vs.100).aspx

[W14] Microsoft Developer Network. (2014, Junio) ASP.NET y Visual Studio para Web. https://msdn.microsoft.com/es-ec/library/dd566231.aspx

[W15] Microsoft Developer Network. (2014, Noviembre) Crear aplicaciones de Windows Forms (Visual C#). http://msdn.microsoft.com/es-es/library/hk4ts42s(v=vs.90).aspx

[W16] Microsoft Developer Network. (2014, Noviembre) Introducción (WPF). http://msdn.microsoft.com/es-es/library/ms742119(v=vs.110).aspx

[W17] Microsoft Developer Network. (2014, Julio) Crear aplicaciones de consola (Visual C#). http://msdn.microsoft.com/es-es/library/452fz12a(v=vs.90).aspx

[W18] Arcitura. (2014, Agosto) REST Constraints. http://whatisrest.com/rest_constraints/index

[W19] Manisha Patil. (2014, Septiembre) REST Architectural Elements and Constraints. http://mrbool.com/rest-architectural-elements-and-constraints/29339

[W20] Rouse, M. (2014, Julio) Extensible Markup Language. http://searchsoa.techtarget.com/definition/XML

[W21] JSON. (2015, Enero) Introducing JSON. http://json.org/

[W22] Android. (2014, Agosto) Application Fundamentals. http://developer.android.com/guide/components/fundamentals.html

[W23] Android. (2014, Agosto) Activities. http://developer.android.com/guide/components/activities.html

[W24] Xamarin. (2014, Abril) Activity Lifecycle. http://developer.xamarin.com/guides/android/application_fundamentals/activity_lifecycle/

[W25] Android. (2014, Agosto) Fragments. http://developer.android.com/guide/components/fragments.html

[W26] Xamarin. (2014, Mayo) Fragments.

http://developer.xamarin.com/guides/android/platform_features/fragments

[W27] Yeray, J. y Landa, I. Windows Phone 7.5 "Mango". https://es.scribd.com/doc/66693271/Windows-Phone-7-5-Mango-Krasis-Press

[W28] Yeray, J., Serna, R., y Landa, I. Desarrollo en Windows 8 y Windows Phone 8 con XAML y C#. https://es.scribd.com/doc/119660895/Desarrollo-en-Windows-8-y-Windows-Phone-8-con-XAML-y-C-VVAA-Krasis-Press

[W29] Boschin, A. (2014, Junio) Windows Phone 7 - Part #2: Your First Application. http://www.silverlightshow.net/items/Windows-Phone-7-Your-first-application.aspx

[W30] Boschin, A. (2014, Junio) Windows Phone 7 - Part #3: Understanding navigation. http://www.silverlightshow.net/items/Windows-Phone-7-Part-3-Understanding-navigation.aspx

[W31] Boschin, A. (2014, Junio) Windows Phone 7 - Part #4: The application lifecycle. http://www.silverlightshow.net/items/Windows-Phone-7-Part-4-The-application-lifecycle.aspx

[W32] Alegsa, L. (2014, Mayo) Definición de Windows 7. http://www.alegsa.com.ar/Dic/windows%207.php

[W33] Smyth, N. (2015, Febrero) Windows Server 2008 R2 Editions and System Requirements. http://www.techotopia.com/index.php/Windows_Server_2008_R2_Editions_and_System_Requirements

[W34] Microsoft. (2014, Junio) Visual Studio. http://www.visualstudio.com/

[W35] Xamarin. (2014, Abril) Introducing Xamarin Studio. http://developer.xamarin.com/guides/cross-platform/getting_started/introducing_xamarin_studio/

[W36] Microsoft TechNet. (2014, Junio) Información acerca de IIS 7. http://technet.microsoft.com/es-es/library/cc753734(v=ws.10).aspx

[W37] Microsoft. (2014, Mayo) IIS. http://www.iis.net/

[W38] Microsoft Developer Network. (2014, Agosto) Microsoft SQL Server. https://msdn.microsoft.com/en-us/library/bb545450.aspx

[W39] Microsoft Developer Network. (2014, Junio) Use SQL Server Management Studio. http://msdn.microsoft.com/en-us/library/ms174173.aspx

[W40] SqlDbx. (2015, Febrero) SqlDbx. http://www.sqldbx.com/

[W41] Telerik. (2014, Julio) Fiddler. http://www.telerik.com/fiddler

[W42] Microsoft Developer Network. (2014, Junio) Soluciones y proyectos. http://msdn.microsoft.com/es-es/library/b142f8e7.aspx

[W43] Monocross. (2015, Febrero) Monocross Model. http://monocross.net/portfolio/monocross-model

[W44] UNITiD. (2014, Septiembre) Android Patterns. http://unitid.nl/androidpatterns/

[W45] UNITiD. (2014, Septiembre) Windows Phone Patterns. http://unitid.nl/windowsphonepatterns/

[W46] Lublinsky, B. (2014, Octubre) Is REST the future for SOA? http://www.infoq.com/articles/RESTSOAFuture/

[W47] Microsoft. (2014, Octubre) Microsoft SQL Server2008 R2 RTM - Express with Advanced Services https://www.microsoft.com/en-us/download/details.aspx?id=25174

[W48] Android. (2014, Septiembre) Device Compatibility. http://developer.android.com/guide/practices/compatibility.html

[W49] Xamarin. (2014, Junio) Part 4 - Providing Backwards Compatibility with the Android Support Package. http://developer.xamarin.com/guides/android/platform_features/fragments/part_4_-_providing_backwards_compatibility_with_the_android_support_package/

[W50] Microsoft Developer Network. (2015, Febrero) App platform compatibility for Windows Phone 8. https://msdn.microsoft.com/en-us/library/windows/apps/jj206947(v=vs.105).aspx

[W51] Microsoft Developer Network. (2015, Febrero) How to target multiple versions with your app for Windows Phone 8. https://msdn.microsoft.com/en-us/library/windows/apps/jj206997(v=vs.105).aspx

[W52] Microsoft Developer Network. (2015, Enero) Deploying a Web Application Project Using a Web Deployment Package. Deploying a Web Application Project Using a Web Deployment Package

[W53] Microsoft Developer Network. (2015, Febrero) How to deploy and run an app for Windows Phone 8. https://msdn.microsoft.com/en-us/library/windows/apps/ff402565%28v=vs.105%29.aspx

[W54] Xamarin. (2015, Enero) Publishing an Application. http://developer.xamarin.com/guides/android/deployment,_testing,_and_metrics/publishing_an_application/

ANEXOS

Los anexos se conforman por cinco secciones que corresponden a varios temas que complementan el documento principal del presente proyecto. Los anexos se adjuntan mediante un disco compacto al presente documento.

ANEXO A: DIAGRAMAS DE BASE DE DATOS Y DEL ORM

Contiene la estructura del sistema a través de los diagramas de base de datos y de los diagramas generados por el diseñador del ORM para las entidades utilizadas en la aplicación.

ANEXO B: MANUALES DE USUARIO

Contiene los manuales de usuario del sistema para los usuarios Administrador Terminal, Administrador Cooperativa y Usuario Común. Los manuales tienen su versión para las plataformas *Windows Phone* y *Android*.

ANEXO C: CÓDIGO FUENTE

Contiene el código fuente y otros archivos que forman parte del sistema.

ANEXO D: APLICACIÓN EN OTRAS PLATAFORMAS

Contiene vistas de la aplicación en otros entornos. Se muestra la aplicación para Consola, Escritorio y entorno Web.

ANEXO E: COSTOS REFERENCIALES

Contiene los costos del proyecto para el hardware, el software y los recursos para el desarrollo de la aplicación.