UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

CARRERA DE INGENIERÍA ELECTRÓNICA Y TELECOMUNICACIONES

"Trabajo de grado previo a la obtención del Título de Ingeniero Electrónica y Telecomunicaciones"

TRABAJO DE GRADUACIÓN

Título del Proyecto:

"IMPLEMENTAR EL SISTEMA DE TRANSMISIÓN DE ALERTA DE EMERGENCIA (EWBS) EN LA PLATAFORMA VILLAGEFLOW PARA ACTIVAR LAS APLICACIONES TDT DE ALERTAS TEMPRANAS"

AUTOR:

DARWIN JAVIER NAVARRETE PINTO

DIRECTOR:

ING. ANÍBAL LLANGA

Riobamba – Ecuador

AÑO 2016

Los miembros del Tribunal de Graduación del proyecto de investigación de título:

IMPLEMENTAR EL SISTEMA DE TRANSMISIÓN DE ALERTA DE EMERGENCIA (EWBS) EN LA PLATAFORMA VILLAGEFLOW PARA ACTIVAR LAS APLICACIONES TVDT DE ALERTAS TEMPRANAS presentado por: Darwin Javier Navarrete Pinto y dirigida por: Ing. Aníbal Llanga

Una vez escuchada la defensa oral y revisado el informe final del proyecto de investigación con fines de graduación escrito en la cual se ha constatado el cumplimiento de las observaciones realizadas, remite la presente para uso y custodia en la biblioteca de la Facultad de Ingeniería de la UNACH.

Para constancia de lo expuesto firman:

Ing. Paulina Vélez -

Presidente del Tribunal

Firma

Ing. Aníbal Llanga

Director del Proyecto

Firma

Ing. José Jinez

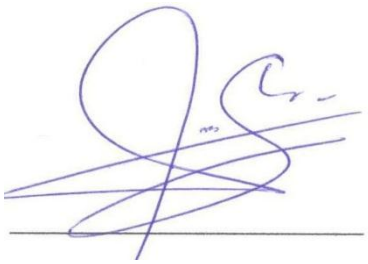
Miembro de Tribunal

Firma

CERTIFICACIÓN DEL TUTOR

Certifico que el presente trabajo de investigación previo a la obtención del grado de Ingeniero en ELECTRÓNICA Y TELECOMUNICACIONES. Con el tema: **"IMPLEMENTAR EL SISTEMA DE TRANSMISIÓN DE ALERTA DE EMERGENCIA (EWBS) EN LA PLATAFORMA VILLAGEFLOW PARA ACTIVAR LAS APLICACIONES TVDT DE ALERTAS TEMPRANAS"** ha sido elaborado por el estudiante **Darwin Navarrete,** el mismo que ha sido revisado y analizado en un cien por ciento con el asesoramiento permanente de mi persona en calidad de Tutor por lo que se encuentran aptos para su presentación y defensa respectiva.

Es todo cuanto puedo informar en honor de la verdad.

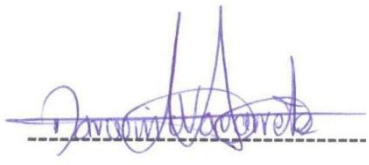


Ing. Aníbal Llanga

C.I. 060293332-7

AUTORÍA DE LA INVESTIGACIÓN

"La responsabilidad del contenido de este Proyecto de Graduación, nos corresponde exclusivamente a: Darwin Javier Navarrete Pinto, Ing. Aníbal Llanga y el patrimonio intelectual de la misma a la Universidad Nacional de Chimborazo."



Darwin Javier Navarrete Pinto

C.I. 131379124-4

AGRADECIMIENTO

A mis padres Darwin y Mercede cuyo apoyo incondicional me han llevado hasta donde estoy ahora, son ejemplo de trabajo y esfuerzo, durante toda mi formación académica han sido un pilas fundamental jamás dudaron de mis capacidades, a mi tutor Ing. Aníbal Llanga quien me guio apropiadamente con su experiencia y conocimiento para la culminación de este trabajo y como final agradecemos a todas las amistades sinceras que se han formado en toda nuestra vida académica.

DEDICATORIA

Este presente trabajo está dedicado a mi amada familia, y en especial a mis padres por ser el pilar fundamental en todo lo que soy, en toda mi educación, tanto académica, como de la vida, por su incondicional apoyo perfectamente mantenido a través del tiempo, a mi hermanos que siempre han estado juntos a mi brindándome su apoyo y a veces haciendo el papel de padres y como olvidarme a mis amigos ya que ellos fueron un pilar fundamentan en todos mis años de estudio formando una gran amistad.

ÍNDICE GENERAL

CERTIFICACIÓN DEL TUTOR iii
AUTORÍA DE LA INVESTIGACIÓN iv
RESUMEN xiii
SUMMARY xiv
INTRODUCCIÓN 16
CAPITULO 1 18
1 FUNDAMENTACIÓN TEÓRICA 18
1.1 Desastre naturales en el Ecuador 18
1.1.1 Desastre naturales contemplados en la zona tres del Ecuador 19
1.2 Televisión Digital Terrestre (TDT) 19
1.2.1 Introducción 19
1.2.2 Estándares de Televisión Digital Terrestre 19
1.2.3 Interactividad 20
1.3 GINGA 21
1.3.1 Decodificadores para TDT (Set Top Box) 22
1.3.2 EiTV Smart Box 22
1.4 Sistemas de alerta temprana 25
1.4.1 Televisión digital terrestre y sistemas de alerta temprana 25
1.4.2 Tipos de aplicaciones de sistemas de alertas de emergencias 26
1.5 Servidor de televisión Digital Terrestre VillageFlow 26
1.5.1 Arquitectura 27

1.5.2 Entorno de configuración VILLAGEFLOW 29
1.6 Tarjeta DecTek DTA-115 37
1.7 Servicios Web 40
1.7.1 Ventajas de los servicios web 40
1.7.2 Servidor Web IIS 41
1.7.3 Servicios Web WCF 41
1.7.4 Base de datos SQL Server 2012 46
1.7.5 Características SQL Server 47
1.7.6 Ventajas de SQL Server 47
1.8 Android Studio 48
1.8.1 Características 48
CAPITULO II 49
2 METODOLOGÍA 49
2.1 Tipo de estudio 49
2.1.1 Descriptivo 49
2.2 Métodos, Técnicas e Instrumentos 49
2.2.1 Métodos 49
2.3 Técnicas 49
2.4 Población y muestra 49
2.4.1 Población 50
2.4.2 Muestra 50
2.5 Hipótesis 50
2.6 Operacionalización de las variables 51
2.7 Procedimientos 52
2.8 Esquema de activación remota del servidor VillageFlow con aplicación de alerta temprana. 53
2.8.1 Análisis y Configuración del servidor VillageFlow 54
2.8.2 Programación en Visual Studio para crear servicio WFC 64
2.8.3 Programación de Base de datos en SQL Server 71
2.8.4 Desarrollo de aplicación Android para realizar la activación remota del servidor VillageFlow 72
2.8.5 Pruebas de activación remota de aplicaciones de alertas de emergencias 76

2.8.6 Comprobación de hipótesis 79
2.8.7 Planteamiento de la hipótesis estadística 79
2.8.8 Establecimiento del nivel de significancia 79
2.8.9 Determinación del valor estadístico de prueba 79
CAPITULO III 82
3 RESULTADOS 82
CAPITULO IV 84
4 DISCUSIÓN 84
CAPITULO V 85
5 Conclusiones y Recomendaciones 85
5.1 Conclusiones 85
5.2 Recomendaciones 86
CAPITULO VI 87
6 PROPUESTA 87
6.1 Título de la propuesta 87
6.2 Introducción 87
6.3 Objetivos 88
6.3.1 Objetivo General 88
6.3.2 Objetivos Específicos 88
6.4 Fundamentación Científico-Técnico 88
6.5 Descripción de la propuesta 89
6.6 Diseño organizacional 89
6.7 Monitoreo y Evaluación de la propuesta 89
7 BIBLIOGRAFÍA 90
8 ANEXOS 92

ÍNDICE DE FIGURAS

Figura 1. 1 Estándares de Televisión Digital *20*
Figura 1. 2 Capas del Estándar Brasileño para TDT *21*
Figura 1. 3 Vista frontal EiTV Smart Box *23*
Figura 1. 4 Vista frontal EiTV Smart Box *23*
Figura 1. 5 Arquitectura de VillageFlow *27*
Figura 1. 6 Space de VillageFlow *30*
Figura 1. 7 Crear nueva configuración en VillageFlow *30*
Figura 1. 8 Nuevo Space creado en VillageFlow *31*
Figura 1. 9 Diagrama de bloques *31*
Figura 1. 10 TsFile In *32*
Figura 1. 11 Ventana de configuración TsFile In *33*
Figura 1. 12 Ginga Data *33*
Figura 1. 13 Búsqueda de archivo XML *34*
Figura 1. 14 Configuración de Remux *35*
Figura 1. 15 Configuración de TmccEncoder *36*
Figura 1. 16 Configuración de la salida RF *36*
Figura 1. 17 Configuración de la salida ASI *37*
Figura 1. 18 Tarjeta DecTek DTA-115 ..... *37*
Figura 1. 19 Diagrama de bloques Tarjeta DecTek DTA-115 ..... *40*
Figura 1. 20 Entorno de Visual Studio *43*
Figura 1. 21 Creación de servicio WCF *44*
Figura 1. 22 Código Fuente del Visual Studio *45*
Figura 1. 23 Inclusión de la Interface para nuestro servicio WCF *45*
Figura 1. 24 Importación de la librería Service *46*
Figura 1. 25 Entorno MySQL *47*
Figura 1. 26 Programa Android Studio 1.0.1 ..... *48*

Figura 2. 1 Proceso para el diseño del proyecto *52*
Figura 2. 2 Esquema de activación remota del servidor VillageFlow *53*
Figura 2. 3Estructura del servidor VillageFlow *54*
Figura 2. 4 Configuración por bloques del servidor VF para alerta temprana *54*
Figura 2. 5 Carpeta VillageIsland *59*

Figura 2. 6 Carpeta VillageFlow *59*
Figura 2. 7 Carpeta Current *60*
Figura 2. 8 Carpeta Gui *60*
Figura 2. 9 Archivo ajax_control_VF.php *61*
Figura 2. 10 Programación del archivo ajax_control_VF.php *62*
Figura 2. 11 Servidor VillageFlow detenido *63*
Figura 2. 12 Servidor VillageFlow Activado *63*
Figura 2. 13 Estado servidor VillageFlow no inicializado *64*
Figura 2. 14 Estado servidor VillageFlow inicializado *64*
Figura 2. 15 Diagrama de flujo del funcionamiento servicios WCF *65*
Figura 2. 16 localhost *69*
Figura 2. 17 Entorno del servicio WCF creado *70*
Figura 2. 18 Entorno de la base de datos creado *72*
Figura 2. 19 Esquema del diseño de la app móvil *73*
Figura 2. 20 Cara principal de la aplicación Master Control *75*
Figura 2. 21 Botón de Iniciar al Servidor VillageFlow *75*
Figura 2. 22 Botón de Stop al Servidor VillageFlow *76*
Figura 2. 23 Información de la aplicación *76*
Figura 2. 24 Resultado de comparación x^2prueba con el x^2tabla método del Chi-Cuadrado *81*

Figura 3. 1 Proceso de activación de la aplicación de alertas de emergencias .. *82*
Figura 3. 2 Proceso de desactivación de la aplicación de alertas de emergencias *83*

ÍNDICE DE TABLAS

Tabla 1 Desastre histórico en el Ecuador *18*
Tabla 2 Tipos de bloques de entrada, proceso y salida en VillageFlow *29*
Tabla 3 Parámetros de configuración *34*
Tabla 4 Atributos de Tarjeta DecTek DTA-115 ..... *39*
Tabla 5 Variable independiente *51*
Tabla 6 Variable dependiente *51*
Tabla 7 Parámetros de configuración para señal TDT *55*
Tabla 8 Datos configurados en la NIT *56*
Tabla 9 Datos configurados en PTM para one-seg *57*
Tabla 10 Datos configurados en TMCC Encoding *57*
Tabla 11 Datos configurados en la DEKTEC *58*
Tabla 12 Tiempo de descarga SMARTBOX *77*
Tabla 13 Tiempo de descarga EiTV Developer Box *78*
Tabla 14 Valores Obtenidos *80*
Tabla 15 Valores de frecuencias esperadas *80*
Tabla 16 Valores Críticos Método Chi-Cuadrado *80*
Tabla 17 Resultados del método estadístico del CHI-CUADRADO *81*

RESUMEN

Este trabajo presenta el diseño de una aplicación móvil en Android Studio para la activación de manera remota del servidor VillageFlow para emitir de alerta temprana para desastres naturales para televisión digital terrestre bajo el middleware GINGA y one seg , para la realización de la aplicación móvil está programada en software Android Studio y para su implementación se utiliza un servicio WCF que permite conectador a la base datos realizada en software SQL SERVER que proporcionara los datos del usuario cuando desde la aplicación móvil deseen hacer login, también el servicio WCF permite activar y desactivar servidor VILLAGEFLOW cuando desde la aplicación móvil se realiza unas de estas dos acciones , en el caso que la acción allá sido activar el servidor VillageFlow previamente configurado los parámetros de sus tres etapas como son entrada, proceso, y salida permitió por medio un decodificador SMART BOX, y un decodificar para one seg emitir las aplicaciones para alertas tempranas para desastres naturales de origen volcánicos bajo el estándar ISDB-tb.

SUMMARY

This research presents the design of a mobile application on Android Studio for activating remotely the VillageFlow server to issue early warning for natural disasters for terrestrial digital television under the middleware GINGA and one sec, for the realization of the mobile application is scheduled Android Studio software and its implementation a WCF service that allows the connector to the database made in SQL sERVER software to provide user data when the mobile app wishing to login, also the WCF service can enable and disable server VILLAGEFLOW used when the mobile app one of these two actions are performed in the case that the action there been activate the VillageFlow server previously set the parameters of its three stages such as input, process and output enabled through a decoder SMART BOX, and one for one seg decoding applications issue early warnings for natural disasters of volcanic origin under the ISDB-Tb standard.

INTRODUCCIÓN

Los primeros sistemas de alertas tempranas tienen origen en el continente asiático debido a la posición geográfica en la que se encuentra han sido víctimas de varios eventos naturales muchos de estos desastrosos.

Por este motivo ha existido la necesidad de crear una aplicación para la activación remota de las aplicaciones de alerta de emergencia en la plataforma VillageFlow, mucho más sofisticados empleados en la nuevas tecnologías existentes como televisión digital terrestre.

En Ecuador debido a la política que maneja el gobierno de la revolución ciudadana muy pronto se migrará a TDT y gracias a esto se podrá mejorar los planes de contingencia de las diferentes ciudades del país, los eventos volcánicos han dejado secuelas continuamente en el territorio nacional tanto en lo económico y social los mismos que pudieron ser disminuidos al tener sistemas de alerta temprana mucho más sofisticados como los presentados en este estudio.

- Las aplicaciones interactivas asociadas a los eventos catastróficos permitirán adicionalmente contactar con un público no acostumbrado a la utilización de sistemas on-line o de TV Interactiva, lo que servirá de "entrenamiento" y facilitará la utilización por parte del usuario de receptores interactivos en aplicaciones de la Sociedad de la Información y la administración electrónica. (Paredes, 2014)

VILLAGEFLOW es un servidor de televisión que permite emitir una señal digital con varias características como EPG (Guía de programación), EBWS (Señal de Alerta Temprana), GINGA para la interactividad, segmento de ONE-SEG entre otras, todo depende de los complementos de software y hardware que se agreguen al servidor.

Visual Studio contiene un entorno de desarrollo integrado para crear aplicaciones espectaculares para Windows, Android e IOS, además de aplicaciones web y servicios de nube innovadores, también admite varios lenguaje de programación como Visual C++, Visual C#, Visual J#, Visual Basic.NET, incluso podemos lograr entornos de desarrollo web como ASP.NET.

SQL Server 2012 maneja un sistema para gestión de base de datos producido por Microsoft el modelo está basado en almacenar y consultar datos solicitados por otras aplicaciones, sin importar si están en la misma computadora.

Android Studio es un entorno de desarrollo integrado (IDE) para la plataforma Android. Está disponible para desarrolladores para probarlo gratuitamente. Basado en IntelliJ IDEA de JetBrains, está diseñado específicamente para desarrollo en plataforma Android.

CAPITULO 1

1 FUNDAMENTACIÓN TEÓRICA

1.1 Desastre naturales en el Ecuador

El Ecuador, debido a su ubicación geográfica y al estar inmersa en "EL Cinturón de Fuego del Pacifico" lo hacen vulnerable a riegos naturales tales como: erupciones volcánicas, terremotos, temblores, deslizamientos e inundaciones.

Ver tabla 1

Desastre	Fecha	Total personas afectadas
Sequía	Marzo 1964	600 000
Inundación	08/04/1970	140 500
Inundación	Noviembre 1982	700 000
Inundación	04/08/1983	200 000
Terremoto	05/03/1987	150 000
Inundación	24/03/1992	205 000
Volcán	03/11/2002	128 150
Volcán	14/08/2006	300 013
Inundación	30/01/2008	289 122

Tabla 1 Desastre histórico en el Ecuador

Fuente: EM-DAT: THE OFDA/CRED International Database, www.emdat.be- Universidad Católica de Lovania

En el ecuador se muestra una tendencia que los desastre naturales van en un aumento gradual y cada vez aumentado su gravedad de su impacto, pero se ha logrado reducir progresivamente el número de sus víctimas, aunque exista un incremento significativo de damnificados.

1.1.1 Desastre naturales contemplados en la zona tres del Ecuador

La zona tres del Ecuador está conformada por las provincias de: Cotopaxi, Chimborazo, Pastaza y Tungurahua, en estas se contempla las siguientes amenazas naturales: inundaciones, sísmico, y de origen volcánico siendo esta la de mayor preocupación ya que se encuentra la presencia del volcán Tungurahua en actividad, situación que motiva a estar preparados, actuar adecuadamente frente a eventos adversos.

1.2 Televisión Digital Terrestre (TDT)

1.2.1 Introducción

En la actualidad el Ecuador está en una transición de la televisión analógica a la televisión digital terrestre, con la finalidad de proveer mejor calidad de transmisión de video y su mayor eficiencia de uso del espectro radioeléctrico, también con este cambio de tecnología se lograra aumentar el número de canales de televisión y así mismo aumentar su acho de banda.

1.2.2 Estándares de Televisión Digital Terrestre

A nivel mundial existe 5 tipos de transmisión de TV digital, los cual en Sudamérica se han adoptados algunos de ellos para la transmisión de Tv digital.

- Estándar Americano, ATSC
- Estándar Europeo, DVB
- Estándar Japonés, ISDB-T
- Estándar Brasileño, ISDB-Tb
- Estándar Chino, DTMB

Figura 1. 1 Estándares de Televisión Digital

Fuente: Universidad del Cauca

1.2.2.1 Estándar ISDB-T

El estándar ISDB-T (Transmisión de Servicios Integrados – Terrestre), los pioneros que han desarrollado son los japoneses y este ha sido adoptado por Brasil con algunas modificaciones.

1.2.3 Interactividad

La interactividad ofrece contenidos adicionales a los programas de televisión, dando la capacidad al usuario de ver informaciones adicionales acorde al contenido audiovisual, la programación de los canales, adquirir productos o servicios, e incluso participar en los propios programas de televisión con el control remoto. Esto es posible gracias a aplicaciones que complementan la programación. (MINISTERIO DE INDUSTRIA, 2010)

1.2.3.1 Tipos de interactividad

La interactividad puede clasificarse de la siguiente manera:

- Solo TV digital sin interactividad.
- Con Interactividad.- Es aquella que cuenta con un participación directa con el televidente, el cual deja de ser un espectador pasivo para convertirse en uno activo, los tipo de interactividad que se pueden ofrecer son los siguientes:

a. Interactividad local: La información no se envía solo se almacena.
b. Interactividad con upload: La información se envía mediante un canal de retorno.
c. Interactividad avanzada: Envío y recepción de información mediante un canal de retorno. (MINISTERIO DE INDUSTRIA, 2010)

1.3 GINGA

Middleware Abierto del Sistema Nipo-Brasileño de TV Digital (ISDB-TB). Ginga está formado por un conjunto de tecnologías estandarizadas e innovaciones brasileñas que lo convierten en la especificación de middleware más avanzada. El ambiente de presentación Ginga-NCL es el subsistema lógico necesario de Ginga, responsable de la ejecución de aplicaciones NCL.

La arquitectura Ginga permite extensiones opcionales. Por ejemplo, el ambiente de ejecución Ginga-J, el responsable de la ejecución de aplicaciones Java. Ginga ofrece servicios NCL a todas las extensiones a través de una API bien definida. (Tv Interactica GINGA, s.f.)

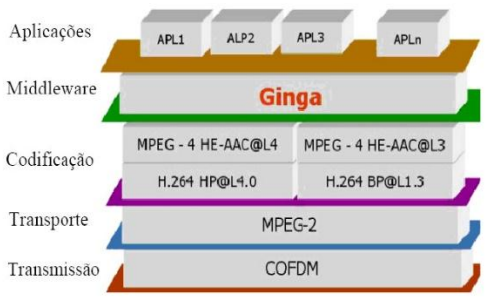


Figura 1. 2 Capas del Estándar Brasileño para TDT

Fuente: tvdginga.wordpress

El middleware abierto Ginga está subdividido en dos subsistemas principales entrelazados, que permiten el desenvolvimiento de aplicaciones siguiendo dos paradigmas de programación diferentes. Dependiendo de las funcionalidades requeridas en cada aplicación, un paradigma será más adecuado que el otro. Estos dos subsistemas son llamados Ginga-J para aplicaciones de procedimiento Java y Ginga-NCL para aplicaciones declarativas NCL. Todas las propuestas del sistema de Televisión Digital especifican middlewares sobre los cuales las aplicaciones de TV Interactiva pueden ser ejecutadas. (Guido Ovaco, 2014)

1.3.1 Decodificadores para TDT (Set Top Box)

Un Set-Top Box (de aquí en adelante STB) principalmente se encarga de recibir una señal digital, en alguno de los estándares (cable, satélite, terrestre, IPTV), y de comprobar en algunos casos que se tenga permiso para ver esta señal. Posteriormente la demodula y la envía al televisor. También permite disfrutar de todo el conjunto de ventajas que ofrece la nueva televisión digital, como pueden ser: Acceso condicional, televisión interactiva (MHP) o la televisión en alta definición. (García, 2014)

1.3.2 EiTV Smart Box

EiTV Smart Box es un decodificador con características innovadoras que lo diferencian de los convertidores digitales existentes en el mercado. Además de sintonizar canales digitales de TV que emiten en alta definición, EiTV Smart Box tiene soporte completo para la interactividad en el estándar establecido para el Sistema Brasileño de TV Digital (DTVi) que le da acceso a varias aplicaciones interactivas transmitidas por los radiodifusores.

El EiTV Smart Box es un decodificador híbrido (ISDB-T y IPTV) adecuado para su uso en soluciones OTT, IPTV y Digital Signage. (Tv, 2016)

Figura 1. 3 Vista frontal EiTV Smart Box

Fuente: EiTV

Figura 1. 4 Vista frontal EiTV Smart Box

Fuente: EiTV

1.3.2.1 Características Técnicas:

- HDTV: TV digital de alta definición para ISDB –T
- Formatos de vídeo: MPEG- 2 MP @ ML / H.264 AVC – HP@L4.0
- Formatos de audio: MPEG -1/2 capa I / II, HE- AAC
- GNU/Linux/OS
- EITV Zapper + Middleware Ginga
- Chipset: STiH205/H207
- Salida de RF: loop through
- Rango de frecuencia : UHF 470-806 MHz (canales 14 a 69)
- Full seg 5.7Mhz, One seg 0.43Mhz
- Sensibilidad: – 75dBm nivel de entrada mínimo
- RAM 512MB DDR3

- Flash 128MB NAND
- Salida HDTV
- Salida modulada de RF (canal 3 /4)
- Vídeo de la salida de componente (YPbPr)
- RCA (VIDEO / R-AUDIO- L)
- 2 puertos USB
- 1 puerto Ethernet: RJ45 – 10/ 100 base
- Guía electrónica de TV (EPG)
- Parental Control protegido con la contraseña
- Tecnología Closed Caption (subtítulos)
- Interactividad (DTVi – Ginga)
- Permite la actualización de software y la reproducción de archivos multimedia a través de los puertos USB
- Conexión a una red a través del puerto Ethernet
- Fuente de alimentación externa de 12 V de alimentación: AC 100 – 240V ~ 50/60Hz

1.3.2.2 Soporte del EiTV Smart Box

- Carga de aplicaciones DTVi – Ginga por el aire, de Internet o red local, o USB.
- Soporte simultáneo a los canales de ISDB-T e IPTV (a través de UDP y RTP).
- Aplicación gráfica (GUI) vía Web Server para instalación y configuración de aplicaciones Ginga (DTVi) y canales de IPTV.
- Exhibición del LOG de la aplicación en la interfaz web para la depuración de aplicaciones Ginga. (Tv, 2016)

1.4 Sistemas de alerta temprana

Los Sistemas de Alerta Temprana, son un conjunto de procedimientos e instrumentos, a través de los cuales se monitorea una amenaza o evento adverso (natural o antrópico) de carácter previsible, se recolectan y procesan datos e información, ofreciendo pronósticos o predicciones temporales sobre su acción y posibles efectos.

1.4.1 Televisión digital terrestre y sistemas de alerta temprana

La Televisión Digital ofrecerá la posibilidad de transmisión de alertas tempranas y oportunas a la población, en casos de emergencia, que permitirá atender estas situaciones, cuando se produzcan eventualidades como: sismos, alertas de tsunamis en el filo costero, erupción de volcanes u otras situaciones de riesgo que puedan ocurrir en el país. Del 27 al 29 de mayo de 2015, en Guayaquil se desarrolló la IV Sesión de la Plataforma Regional de las Américas para la Reducción de Riesgo de Desastres, en la que se presentaron los avances que Ecuador despliega para la reducción de riesgos en caso de desastres. Desde el Ministerio de Telecomunicaciones y de la Sociedad de la Información (MINTEL) se participó con una demostración del Sistema de Alerta de Emergencias "EWBS", a través de la Televisión Digital Terrestre (TDT). Dentro del proceso de implementación del Sistema de Alerta Temprana, con la Televisión Digital Terrestre, ya se trabaja de manera conjunta con la Secretaría de Gestión de Riesgos y otros actores del sector público y privado, en beneficio de la sociedad, utilizando la tecnología como una herramienta de información, para la protección de la vida humana, los recursos naturales y los bienes materiales de la sociedad ecuatoriana. (MINTEL, 2015)

1.4.1.1 Emergency Warning Broadcasting (EWBS)

EWBS utiliza una advertencia especial o señales de alerta incorporadas en las señales de broadcasting para cambiar automáticamente en el equipo receptor en 100 el hogar, y emitir un boletín de emergencia, alertando a la gente ante un desastre inminente, por ejemplo un tsunami o un terremoto. Es importante mencionar que las señales EWBS trabajan en sistemas análogos y digitales. Las señales EWBS embebidas en TV y radio analógica requieren un generador de señal de control de frecuencia dual. Las señales pueden ser enviadas a TV y radios convencionales sin ninguna modificación especial. La señal EWBS puede incluir códigos de área y tiempo como también códigos fijos para iniciar o terminar la operación del sistema. EWBS analógico ha estado en operación en Japón desde 1985, y en modo digital desde el año 2000. Los lineamientos para su implementación en TV Digital se definieron en el año 2006 en la Asamblea General de la ABU (Asia-Pacific Broadcasting Unión). En base al manual de la ABU se abordan los principales lineamientos para la implementación de EWBS para difusión digital. (Tiupul Urquizo, 2015)

1.4.2 Tipos de aplicaciones de sistemas de alertas de emergencias

Existen diferente tipos de aplicaciones de alertas de emergencias entre las cuales tenemos:

- Aplicaciones en One seg para alertas de emergencias
- Aplicaciones en Ginga (NCL) para alertas de emergencias
- Sistemas de alertar de emergencia EWBS

1.5 Servidor de televisión Digital Terrestre VillageFlow

VILLAGEFLOW (TM) es la plataforma de software definitiva para la generación, operación, procesamiento y seguimiento de las señales de televisión de emisión digital (Transport Stream). VILLAGEFLOW está optimizado para el

tiempo real y el funcionamiento continuo 24H / 7D y es compatible con una amplia gama de adaptadores de entrada / salida. (Dektec y otros proveedores de terceros).

Básicamente, se puede construir las instalaciones de radiodifusión más baratas, flexibles y altamente funcionales, al tiempo que permite varias señales de radiodifusión experimental y servicios desafiantes.

VILLAGEFLOW es compatible con los estándares mundiales, tales como DVB, ISDB-T, DTMB, ATSC, DVB-S / S2, DVB-T / T2, DVB-C / C2, incluyendo módulos para la modulación de RF y demodulación, la emisión de datos, EPG Generación, subtitulado, un seguimiento detallado y apoyo para todos los estándares de vídeo que van desde la televisión móvil a alta definición (y, por ejemplo .. 4K) esquemas aún más altos. (village island, 2015)

1.5.1 Arquitectura

VillageFlow se compone de varios procesos para realizar la Transmisión Digital de Televisión (Transport Stream), como nos indica la Figura 1.5

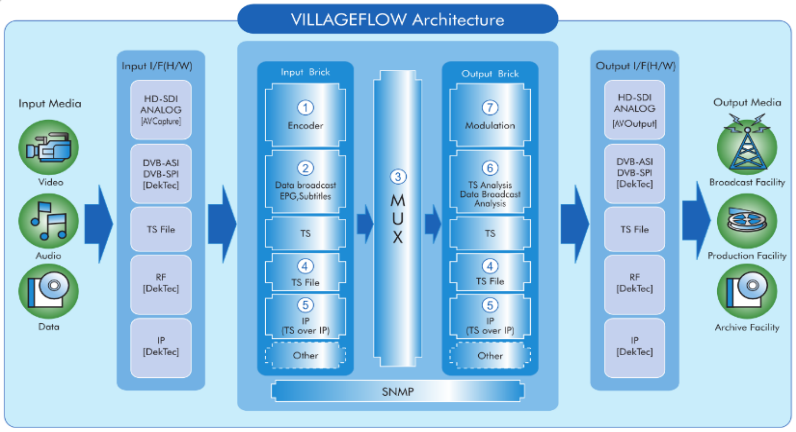


Figura 1. 5 Arquitectura de VillageFlow

Fuente: Village island

Comienza con medios de entrada como son Video, Audio y Datos, a continuación estos se dirigen por las tarjetas instaladas en el servidor. A partir de aquí se utiliza el software para realizar la configuración de los diferentes bloques de entrada, multiplexador y bloques de salida para la transmisión, estos datos generados serán enviados por las respectivas tarjetas de salida hacia los diferentes medios de recepción.

1.5.1.1 Tipos de bloques del VillageFlow

1.5.1.1.1 Bloques de entrada

Encargados de producir la conversión de una o más señales analógicas del dominio de frecuencia digital (Blackmagic HD-SDI), codificando el video, audio.

Se debe escoger de acuerdo al bloque en que se vaya a trabajar como se muestra en la tabla 2

1.5.1.1.2 Bloques de Proceso

Se realiza la multiplexación del contenido codificado proveniente de los bloques de entrada, así como también la configuración de los parámetros TMCC (Transmission Multiplexing Configuration Control) donde se debe configurar obligatoriamente la demodulación y parámetros de transmisión OFDM.

Se debe escoger de acuerdo al bloque en que se vaya a trabajar como se muestra en la tabla 2

1.5.1.1.3 Bloques de Salida

Permite la configuración de los diferentes tipos de salida tales como: RF, IP, ASI y creación de archivos TS.

Se debe escoger de acuerdo al bloque en que se vaya a trabajar como se muestra en la tabla 2

ENTRADA	*PROCESO*	*SALIDA*
ARIB-HD-HDSDI In	RemuxARIBLayerA	ARIBdata Out
ARIB1Seg-AVIfile In	RemuxARIBLayerB	Asi Out
ARIB1Seg- HDSDI In	RemuxISDBTbLayerA	Atsc-Dektec-Output-Card Out
ARIB1Seg-PCscreen In	RemuxISDBTbLayerB	Atsc Out
ARIB1Seg-Split In	RemuxSimple	AV Out
AribCaption In	RemuxWithAribSI	DtIp Out
Arib Data In	RemuxWithDvdFullSI	Dtmb Out
Asi-Decod4k In	RemuxWithDvdSimpleSI	Dvbs2 Out
Asi In	RemuxWithPSIInsertion	Dvbt Out
DtIp In	TmccEncoder	Ip Out
Dvbs2 In	Transcode – AdvEnc	IpForVideoViewer Out
Epg-1seg In	Transcode – Transcoding	Isdbs Out
Epg-EPG –In In	Transcode	Isdbt-Dektec-Output-Card Out
Epg-JNP In	TsmfDecoder	Isdbt Out
Epg In		Monitor Out
Ginga Data In		Qam Out
IP-Decod4k In		TsFile-ForContinuousRecording Out
IP In		TsFile-HLS-segmenter Out
TsFile-Decod4k In		TsFile-TSFile-Out Out
TsFile-TsFile-In In		TsFile Out
TsFile In		ULE Out
TsFilePlayList In		Video Out
TsFileSmartList In		
Video-ARIBHD In		
Video-Encod-HD In		
Video-Encoder In		
VideoK1SeglSDBTb In		

Tabla 2 Tipos de bloques de entrada, proceso y salida en VillageFlow

Fuente: Autor

1.5.2 Entorno de configuración VILLAGEFLOW

Para acceder al entorno de configuración abrimos el navegador de Internet Explorer e ingresar la dirección de host local 127.0.0.1

Figura 1. 6 Space de VillageFlow

Fuente: Autor

Crear una nueva configuración en el servidor clic en las pestañas:

- Manage Configuration files
- Create new configuration

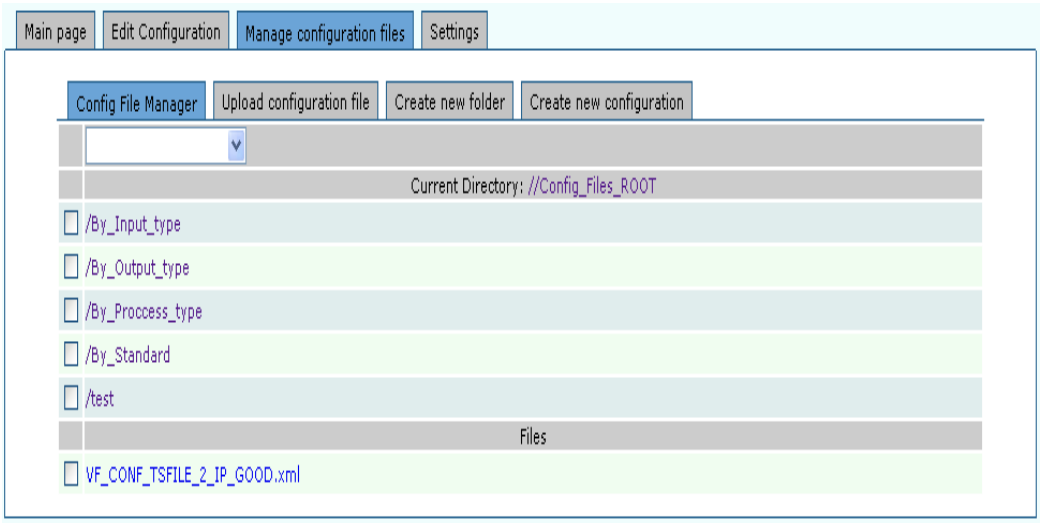


Figura 1. 7 Crear nueva configuración en VillageFlow

Fuente: Autor

Posteriormente nombramos nuestra configuración y aparecerá una pantalla como se muestra en la figura 1.8

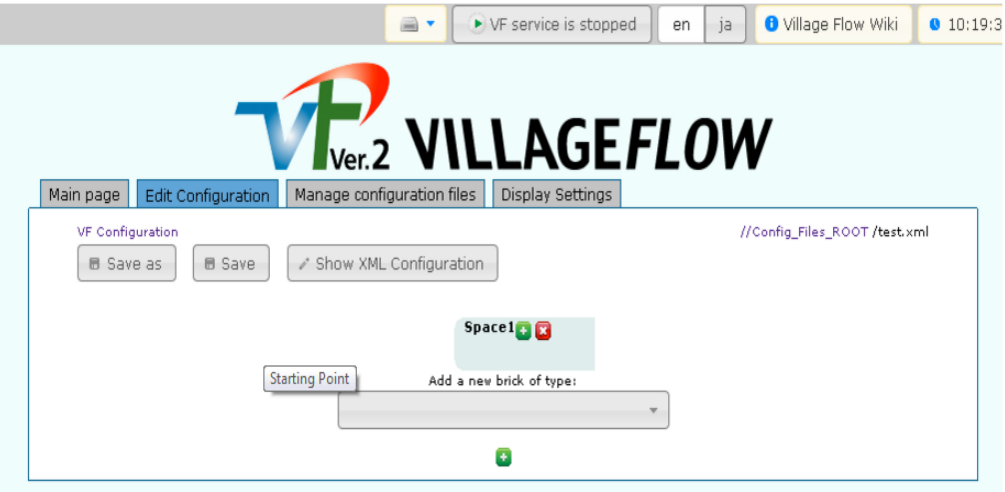


Figura 1. 8 Nuevo Space creado en VillageFlow

Fuente: Autor

Visto en el diagrama de bloques del software seria:

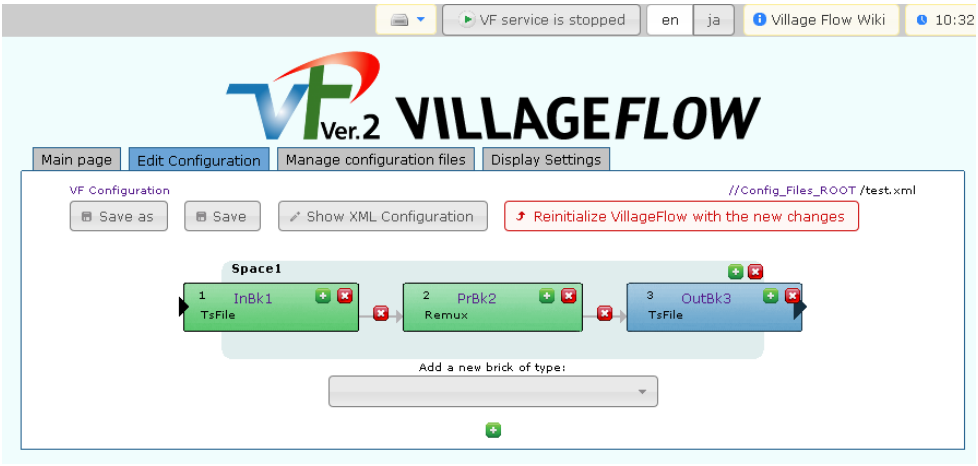


Figura 1. 9 Diagrama de bloques

Fuente: Autor

1.5.2.1 Bloques de entrada

- **TsFile In:** El bloque TsFile In es el que se encarga de la entrada de audio y video que será vista en el televisor, cabe recalcar que los archivos de video para este bloque deben estar en formato ".TS" para que la plataforma lo reconozca. En el bloque de entrada existen varios parámetros que deben ser configurados para que dicho bloque tenga un óptimo funcionamiento.

Figura 1. 10 TsFile In

Fuente: Autor

Parámetros de configuración del TsFile In:

Brick Info: son sólo para fines informativos. Pueden ser editados, pero no afectarán el proceso de VF.

TS Rate: Ajuste para el TS Rate del archivo. Se puede dejar en "0" en la mayoría de casos, ya que VF es capaz de evaluar y ajustar automáticamente el TS Rate.

TS Packet Size: TS Files se encuentran en la mayoría de los casos hechos de 188 paquetes de bytes, pero en algún momento puede ser 204 bytes. Si conoce el valor, puede ponerlo. De lo contrario, sólo hay que poner "0" y VF encontrará el valor automáticamente y configurarlo.

File Path: ruta que señala la ubicación del archivo TS.

Loop: número de veces que el archivo se reproducirá antes de detenerse. El poner "0" hace el bucle infinito (hasta que se detenga VF)

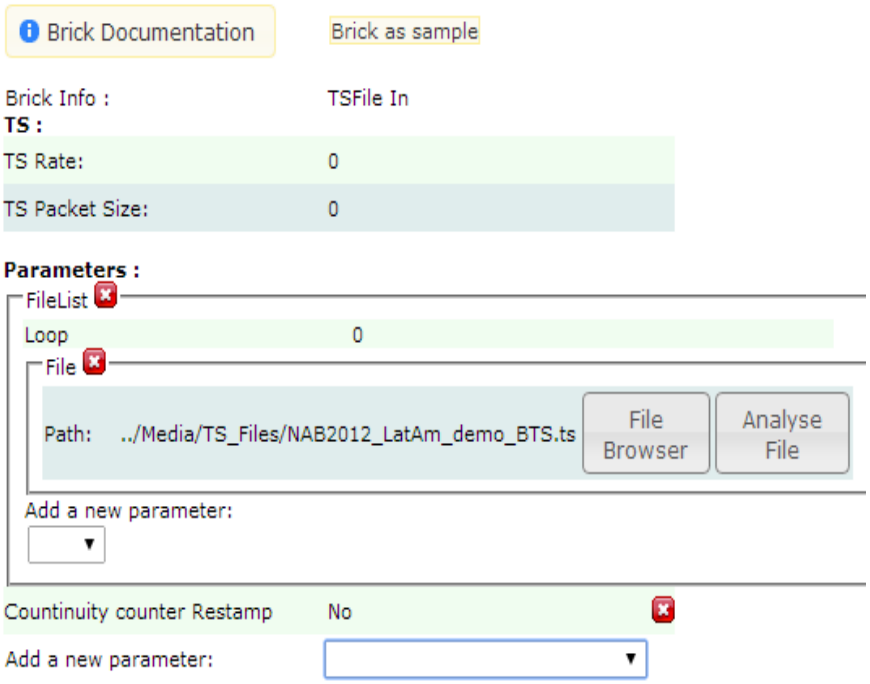


Figura 1. 11 Ventana de configuración TsFile In

Fuente: Autor

- **Ginga Data:** Este bloque de entrada codifica aplicaciones Ginga de lenguaje NCL o JAVA en un flujo de transportes TS encapsulado en un carrusel de objetos, dando como resultado un flujo de datos (Flujo de transporte MPEG).

Figura 1. 12 Ginga Data

Fuente: Autor

Parámetros de configuración preestablecidos:

NOMBRE	DESCRIPCIÓN	VALOR
TsOut	Datos de salida binarios (bytes)	
TsPacketSize	Tamaño del paquete TS de salida	188-204
TsRate	Tasa de salida de datos TS ()bps	
File	Directorio y nombre del archivo XML que contiene los parámetros de configuración de los archivos GINGA	
PID	PID de flujo de datos	1001

Tabla 3 Parámetros de configuración

Fuente: village-island

Para buscar el archivo de configuración del bloque Ginga Data clic en:

- File Browser

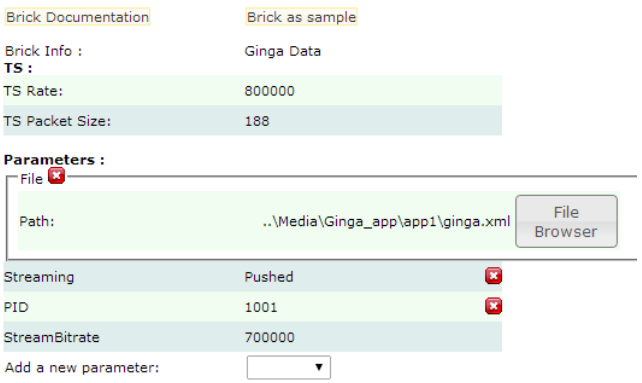


Figura 1. 13 Búsqueda de archivo XML

Fuente: Autor

1.5.2.2 Bloques de proceso

- **Remux:** Este bloque de proceso se encarga de la multiplexación del contenido audio, video y datos de las entradas con las Tablas PSI/SI correspondientes a cada uno de los servicios.

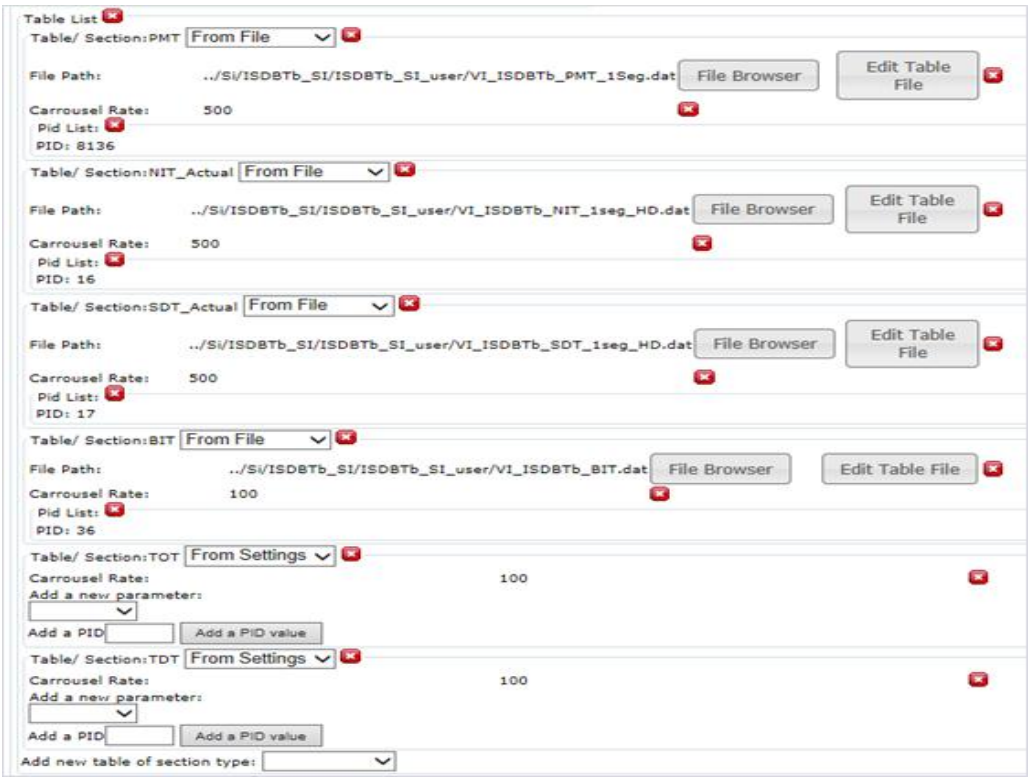


Figura 1. 14 Configuración de Remux

Fuente: Autor

- **TmccEncoder:** El bloque de proceso TMCC (Transmisión Configuration Control) realiza el control de transmisión, configuración del segmento del canal y parámetros de transmisión, su configuración puede ser manual o automática.

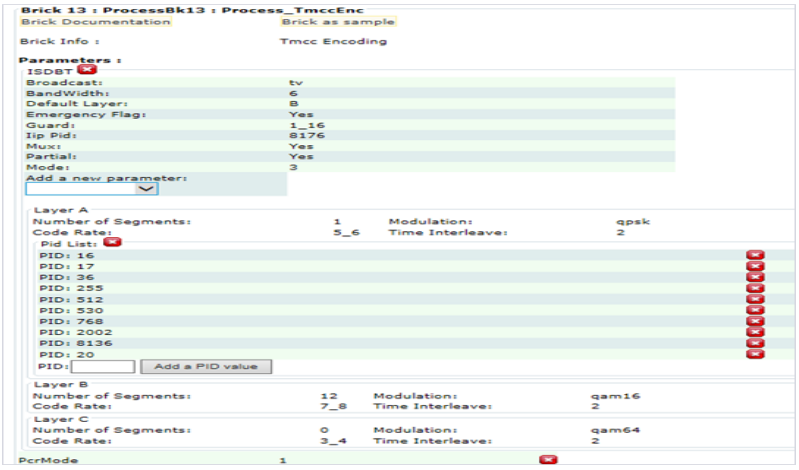


Figura 1. 15 Configuración de TmccEncoder

Fuente: Autor

1.5.2.3 Bloques de salida:

- **RF ISDB-T:** Este bloque de salida establece los parámetros de transmisión con la finalidad de añadir robustez a la señal ante las posibles condiciones que se presenten en el medio. Se establece también la tasa de datos para un ancho de banda de 6 MHz, el tamaño de los paquetes, el nivel RF a la salida, el canal de salida y su frecuencia.

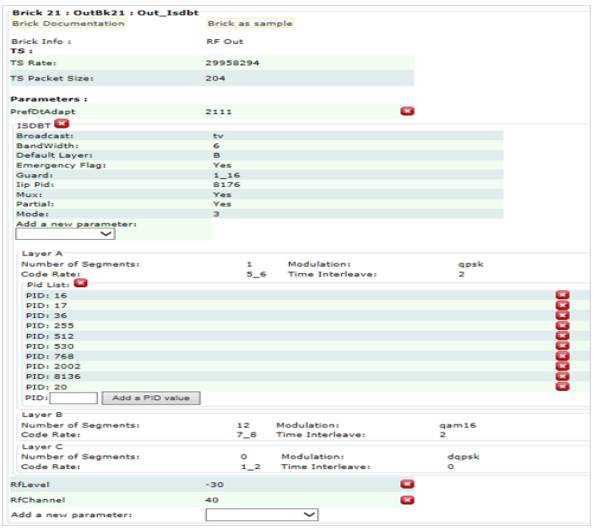


Figura 1. 16 Configuración de la salida RF

Fuente: Autor

- **Salida ASI:** Implementa un formato de transmisión de datos ASI Asynchronous Serial Interface. La señal ASI es el resultado de la compresión de vídeo MPEG2 O MPEG4 lista para su transmisión a cualquier medio físico.

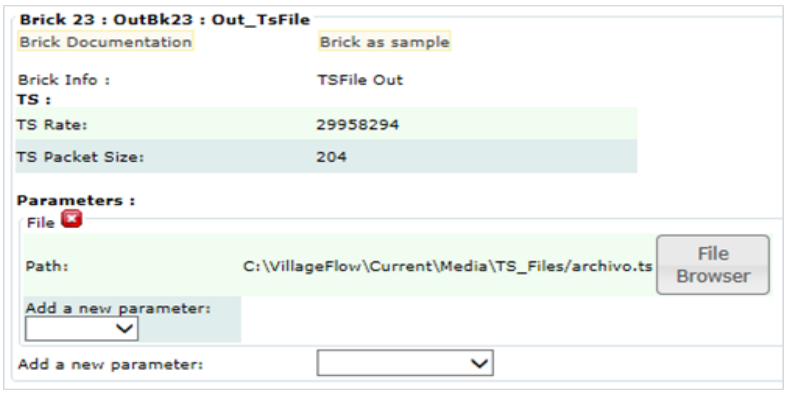


Figura 1. 17 Configuración de la salida ASI

Fuente: Autor

1.6 Tarjeta DecTek DTA-115

Modulador que maneja Multi-estándar como QAM y OFDM, tiene un nivel de salida programable con calidad de señal perfecta. Multi-estándar de propósito general modulador de ensayo en el laboratorio para el desarrollo de equipos de televisión digital, o para la experimentación con nuevos esquemas de modulación RF. (DecTek, 20)

Figura 1. 18 Tarjeta DecTek DTA-115

Fuente: DecTek

1.6.1.1 Características Técnicas:

- Convertidor elevador de todos los canales 47-862 MHz cubriendo completamente las bandas VHF y UHF.
- Amplificador de salida programable con 0-31.5dB, atenuador en pasos de 0,5 dB.
- La salida de monitor para la conexión directa a la entrada de la antena de un receptor digital.
- Puerto bidireccional DVB-ASI.
- Interfaz de programación (DTAPI) es totalmente compatible con otros adaptadores de salida Video Digital DekTec.
- PCI rev 2.2, de 32 bits, bus 33 o 66 MHz.

1.6.1.2 Estándar de Modulación

Modulación	Estándar
DTMB	GB 20600-2006
DVB-T / DVB-H	EN 300 744
ISDB-T	ARIB STD – B31
QAM	J.83 Annex A/B/C

Tabla 1. 1 Estándar de Modulación

Fuente: DecTek

1.6.1.3 Atributos

	Parámetro	Valor
	Ancho de Banda	5.0 - 8.0 MHz
Main Out-Put	Conector	50 Ω BNC
	Perdidas de retorno	≥ 15dB (47-862 MHz)
	Rango	47-862 MHz
	Tamaño de salto	Configurable, ≥100KHz
	Level QAM	-31.5 ... 0dBm ± 1dB
	Level OFDM	-34.5 ... -3dBm ± 1dB
	Fase de ruido	<-90dBc 10KHz
Monitor Out-Put	Conector	75 Ω
	Perdidas de retorno	≥ 15dB (47-862 MHz)
	Level QAM	-27.5dBm ± 2dB
	Level OFDM	-30.5dBm ± 2dB
DVB-ASI	Conector	75 Ω BNC
	Perdidas de retorno	≥ 15dB (5-270MHz)

Tabla 4 Atributos de Tarjeta DecTek DTA-115

Fuente: DecTek

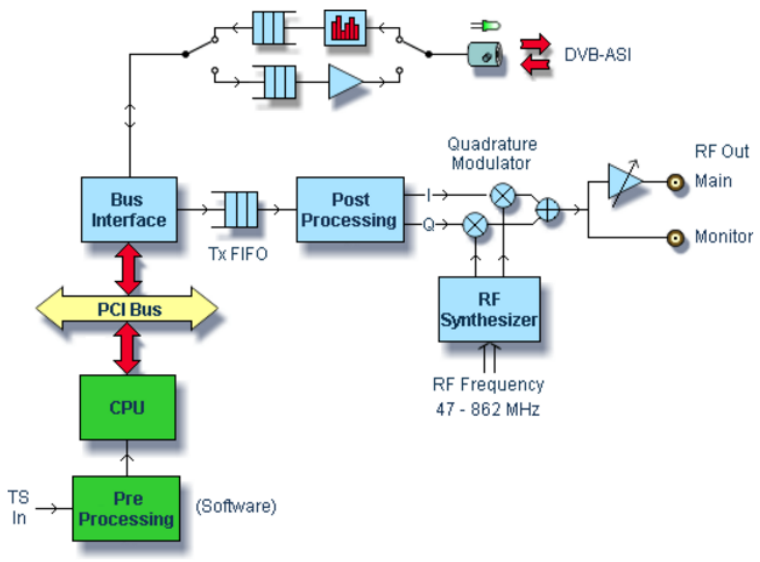


Figura 1. 19 Diagrama de bloques Tarjeta DecTek DTA-115

Fuente: DecTek

1.7 Servicios Web

El consorcio W3C define los Servicios Web como sistemas software diseñados para soportar una interacción interoperable maquina a máquina sobre una red. Los Servicios Web suelen ser APIs Web que pueden ser accedidas dentro de una red (principalmente Internet) y son ejecutados en el sistema que los aloja. La definición de Servicios Web propuesta alberga muchos tipos diferentes de sistemas, pero el caso común de uso de refiere a clientes y servidores que se comunican mediante mensajes XML que siguen el estándar SOAP. (Marset, 2006)

1.7.1 Ventajas de los servicios web

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.

- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados.

1.7.2 Servidor Web IIS

IIS es un conjunto de servicios para servidores usando Microsoft Windows. Es especialmente usado en servidores web, que actualmente es el segundo más popular sistema de servidor web (funciona en el 35% de los servidores de todos los sitios web).

De hecho, el IIS viene integrado con Windows NT 4.0. Dado que el IIS está tan íntimamente integrado con el sistema operativo, es relativamente fácil de administrar. Sin embargo, actualmente el IIS está disponible sólo para el uso en la plataforma Windows NT, mientras que los servidores Web de Netscape corren en la mayoría de las plataformas, incluyendo Windows NT, OS/2 y UNIX. (wiwiloz, 2006)

1.7.2.1 Principales características de servidores web IIS

- Tiene la capacidad de servidor web integrado.
- Posee buena seguridad y es administrable en internet.
- Tiene la presencia del protocolo HTTP 1.1 que ofrece sensibles mejoras de las prestaciones, por lo cual disminuyendo los tiempos de respuesta en la trasmisión.
- Este servidor web IIS es propio de Windows

1.7.3 Servicios Web WCF

Windows Communication Foundation (WCF) es un framework para la creación de aplicaciones orientadas a servicios. Con WCF, es posible enviar datos como mensajes asincrónicos de un extremo de servicio a otro.

Un extremo de servicio puede formar parte de un servicio disponible continuamente hospedado por IIS, o puede ser un servicio hospedado en una aplicación. Un extremo puede -37- ser un cliente de un servicio que solicita datos de un extremo de servicio. Los mensajes pueden ser tan simples como un carácter o una palabra que se envía como XML, o tan complejos como una secuencia de datos binarios. (Patiño, 2014)

1.7.3.1 Características de WCF

WCF incluye un conjunto de Características entre las cuales tenemos:

- Orientación a servicios con lo cual permite crear aplicaciones con los estándares ws, wcf.
- WCF implementa los estándares del sector modernos para la interoperabilidad de servicios web
- Los mensajes se intercambian mediantes el de solicitud/respuesta, un extremo solicita datos y el otro extremo responde y viceversa.
- Posee metadatos de servicios que permiten publicar sobre HTTP Y HTTPS.
- WCF también admite las transacciones por medio de modelos ws-atomic, las API del espacio de nombre System.

1.7.3.2 Crear una clase de servicios WCF

Para crear una clase de servicios WCF se compondrá de cuatro elementos:

- Contrato de datos: estructuras de datos que será intercambiada por el servicio.
- Contrato de servicio: es la interfaz que visualizara los diferentes métodos que puedan ser invocados por el cliente.
- Implementación de servicio: realiza la codificación del contrato de servicio y añade la lógica interna del servicio.

1.7.3.3 Visual Studio

Visual Studio contiene un entorno figura 1.20 de desarrollo integrado para crear aplicaciones espectaculares para Windows, Android e IOS, además de aplicaciones web y servicios de nube innovadores, también admite varios lenguaje de programación como Visual C++, Visual C#, Visual J#, Visual Basic.NET, incluso podemos lograr entornos de desarrollo web como ASP.NET.

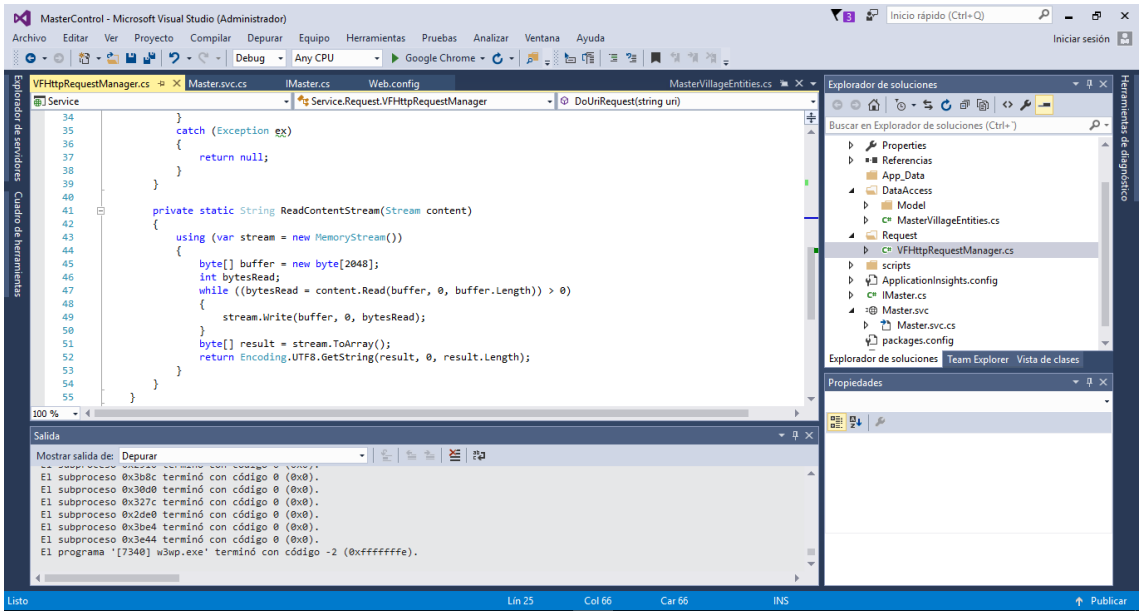


Figura 1. 20 Entorno de Visual Studio

Fuente: Autor

1.7.3.3.1 Características de Microsoft Visual Studio 2015

- Permite la creación de aplicaciones web.
- Permite crear servicios web en cualquier entorno que soporte la plataforma .NET, con esto se puede crear aplicaciones que se intercomuniquen entre estaciones trabajo, pagina web y dispositivo móviles

1.7.3.3.2 Crear servicio WCF en Visual Studio 2015

Para crear el servicio WCF es necesario los siguientes pasos.

En primer escenario será ingresar al software Visual Studio con versión VS2015 y se procedió a crear un nuevo proyecto del tipo WCF >> WCF Service Library tal y como podrán observar en la siguiente Figura 1.21, a la cual se asignó el nombre de WcfMasterControl.

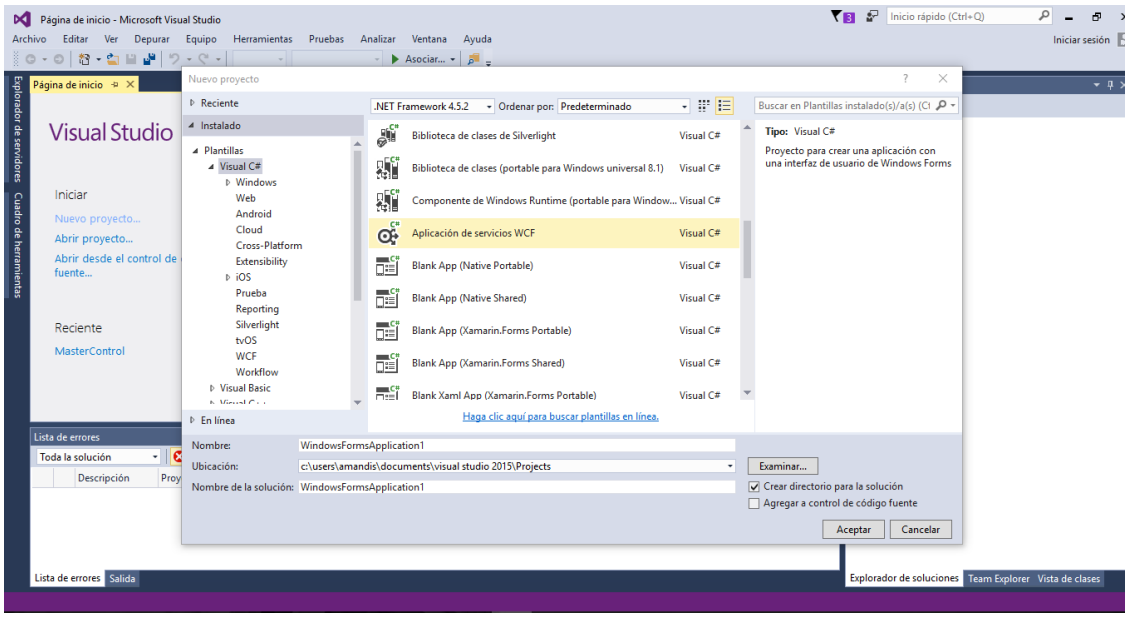


Figura 1. 21 Creación de servicio WCF

Fuente: Autor

Por defecto el proyecto WCF crea un código fuente, el cual se eliminó porque se realizó un servicio WCF desde cero, para ello se elimina la interfaz y clase que a generado el proyecto, como se observa en la figura 1.22

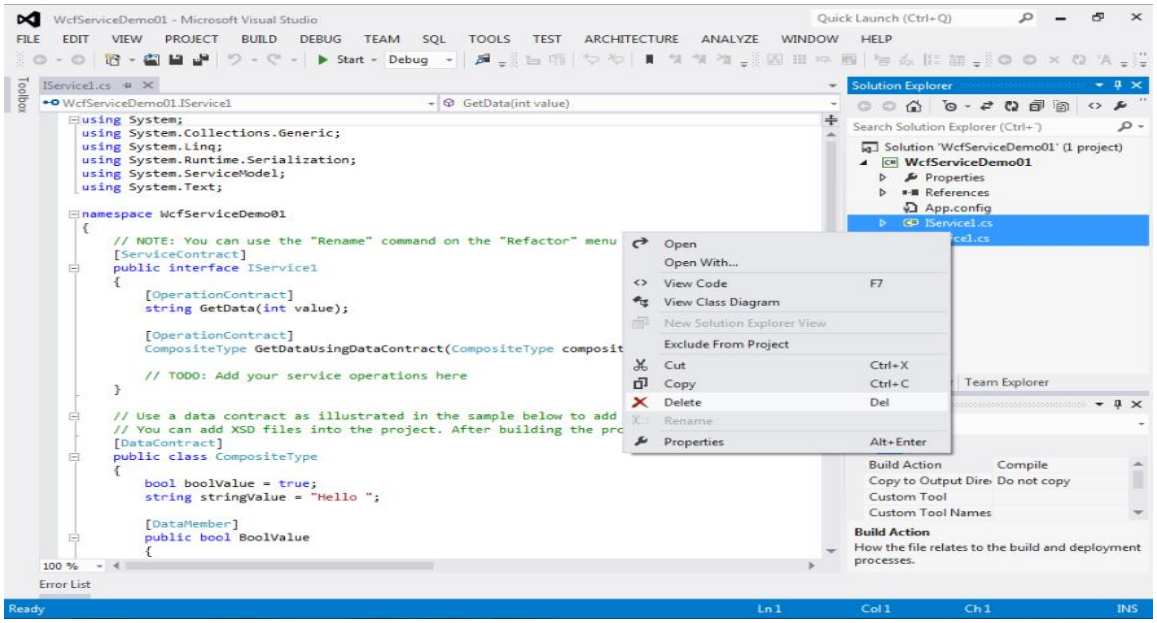


Figura 1. 22 Código Fuente del Visual Studio

Fuente: Autor

Seguido se añadió al proyecto un interface a la cual se le asignó el nombre de Service.

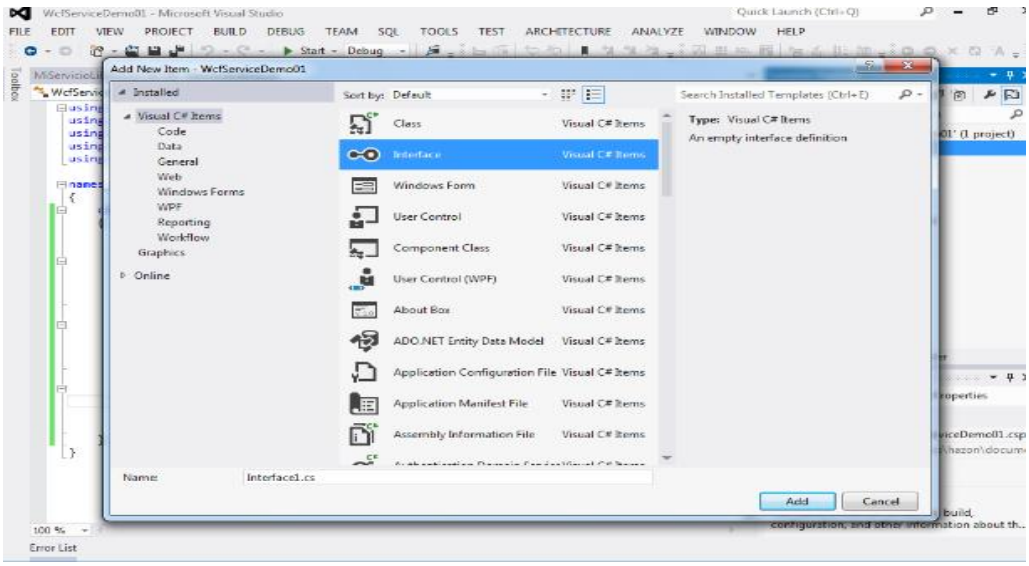


Figura 1. 23 Inclusión de la Interface para nuestro servicio WCF

Fuente: Autor

Para trabajar con WCF, sé importa una librería cuyo nombre es "Service" la cual se importó con la cláusula siguiente usingSystem.Service, tal y como podrán observar en la siguiente Figura 1.24

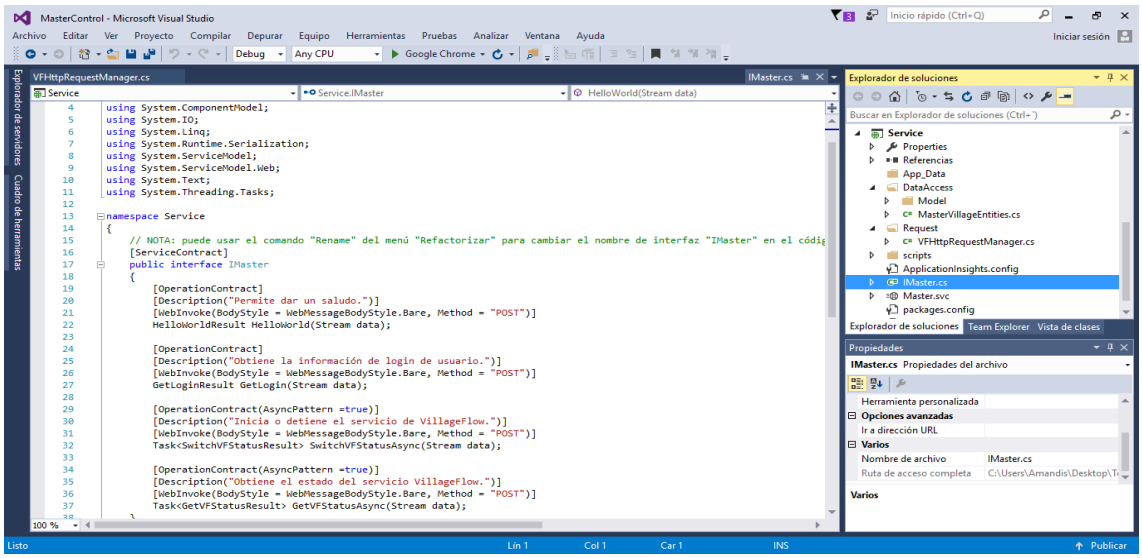


Figura 1. 24 Importación de la librería Service

Fuentes: Autor

1.7.4 Base de datos SQL Server 2012

SQL Server 2012 maneja un sistema para gestión de base de datos producido por Microsoft el modelo está basado en almacenar y consultar datos solicitados por otras aplicaciones, sin importar si están en la misma computadora, SQL Server permite diferente tipos de datos como integer, float, decimal, char etc.

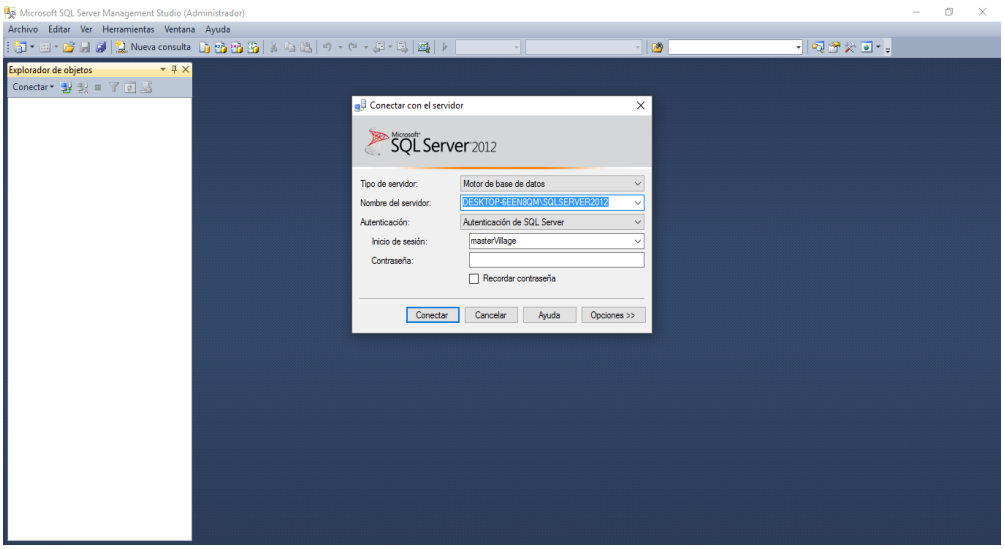


Figura 1. 25 Entorno SQL Server

Fuente: Autor

1.7.5 Características SQL Server

- Proporciona un almacenamiento fiable para datos.
- Aporta medios de recuperar rápidamente los datos.
- Permite trabajar en modo cliente-servidor, ya que la información y datos se aloja en el servidor y los clientes de la red solo acceden a la información.
- Incluye un potente ambiente grafico de administración.

1.7.6 Ventajas de SQL Server

- Puede ejecutar consultas contra una base de datos
- Recuperar datos de una base de datos
- Insertar registros en una base de datos
- Actualizar registros en una base de datos
- Eliminar registros de una base de datos
- Crear nuevas bases de datos
- Crear nuevas tablas en una base de datos

1.8 Android Studio

Android Studio es un entorno de desarrollo integrado (IDE) para la plataforma Android como indica la figura. Está disponible para desarrolladores para probarlo gratuitamente. Basado en IntelliJ IDEA de JetBrains, está diseñado específicamente para desarrollo en plataforma Android.

Figura 1. 26 Programa Android Studio 1.0.1

Fuente: Autor

1.8.1 Características

- Generación de imagen y video en tiempo real
- Consola de desarrollador: consejos de optimización, ayuda para la traducción, estadísticas de uso.
- Soporte para construcción basada en la herramienta Gradle.
- Refactorización especifica de Android y arreglos rápidos.
- Herramientas Lint para detectar problemas de rendimiento, usabilidad, compatibilidad de versiones, y otros problemas.
- Plantillas para crear diseños comunes de Android y otros componentes.
- Soporte para programar aplicaciones para Android Wear

CAPITULO II

2 METODOLOGÍA

2.1 Tipo de estudio

2.1.1 Descriptivo

Para el análisis de la aplicaciones de alertas tempranas para desastres naturales de origen volcánicos utilizando la plataforma VillageFlow y su activación remotamente mediante una aplicación en Android studio ,será una investigación experimental debido a que realiza pruebas para comprobar la activación remotamente de una señal TDT con interactividad y one seg

2.2 Métodos, Técnicas e Instrumentos

2.2.1 Métodos

2.2.1.1 Experimental

El método utilizado en este proyecto es experimental, puesto que se hizo un experimento en un ambiente controlado en el cual se realizó la activación remota de aplicaciones de alertas tempranas que fueron receptadas en un decodificar SMART BOX y proyectadas en un televisor y otra en un receptor de one seg.

2.3 Técnicas

2.4 Población y muestra

2.4.1 Población

La población es cualquier conjunto de elementos de los que se quiere conocer o investigar alguna o algunas de sus características. La población será la activación remota para la transmisión de señales para TDT con una aplicación de alerta temprana para desastres naturales.

2.4.2 Muestra

La muestra es un subconjunto representativo y finito que se extrae de la población accesible. La población carece de registro definido, es decir que es desconocida ya que el número de pruebas puede ser infinito, por lo tanto, la muestra es establecida de acuerdo al cálculo con la siguiente fórmula:

$$n = \frac{Zc^2 * p * q}{e^2}$$

n = Tamaño de la muestra

Zc = Distribución de Gauss donde Zc = 2.575

e = Error muestra falla que se produce al extraer la muestra de la población. Oscila entre 1% y 5%.

pq = Constante de la varianza población (0.25)

$$n = \frac{0.9}{0.0025}$$

$$n = 36$$

2.5 Hipótesis

La configuración de la plataforma VillageFlow y su activación remota permitirá el ahorro de tiempo para la trasmisión de señal TDT con aplicaciones de alerta temprana para desastres naturales.

2.6 Operacionalización de las variables

VARIABLE	DIMENSIÓN	INDICADORES	TÉCNICA
Servidor Villageflow	• Configuración Villageflow	• Configuración de tramas ISDB-Tb para EWB. • Acceso a servidor villageflow de forma remota. • Aplicación para activación de EWB	• Lista de cotejo • Observación

Tabla 5 Variable independiente

Fuente: Autor

VARIABLE	DIMENSIÓN	INDICADORES	TÉCNICA
Generación del sistema de transmisión de alerta de emergencia (EWBS) para desastre naturales.	• Alerta temprana	• Adición del Descriptor de Información de Emergencia • Adición de la señal de activación para EWB (Emergency Warning Broadcast)	• Lista de cotejo • Observación • Lista de cotejo • Observación

Tabla 6 Variable dependiente

Fuente: Autor

2.7 Procedimientos

Figura 2. 1 Proceso para el diseño del proyecto

Fuente: Autor

2.8 Esquema de activación remota del servidor VillageFlow con aplicación de alerta temprana.

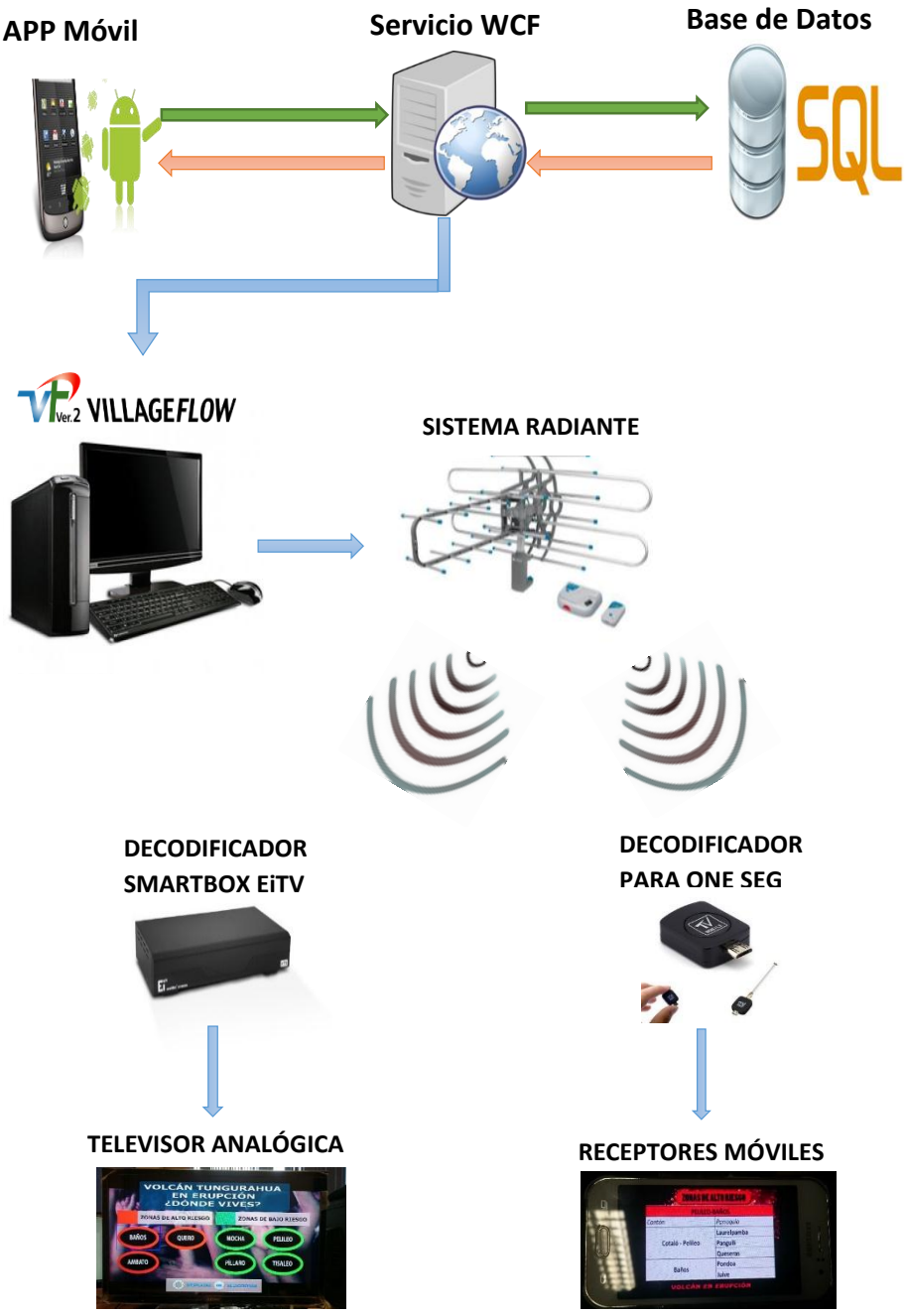


Figura 2. 2 Esquema de activación remota del servidor VillageFlow

Fuente: Autor

2.8.1 Análisis y Configuración del servidor VillageFlow

La configuración del servidor de VF está orientada a la programación por objetos, basada en lenguaje XML, y su estructura está compuesta por tres etapas:

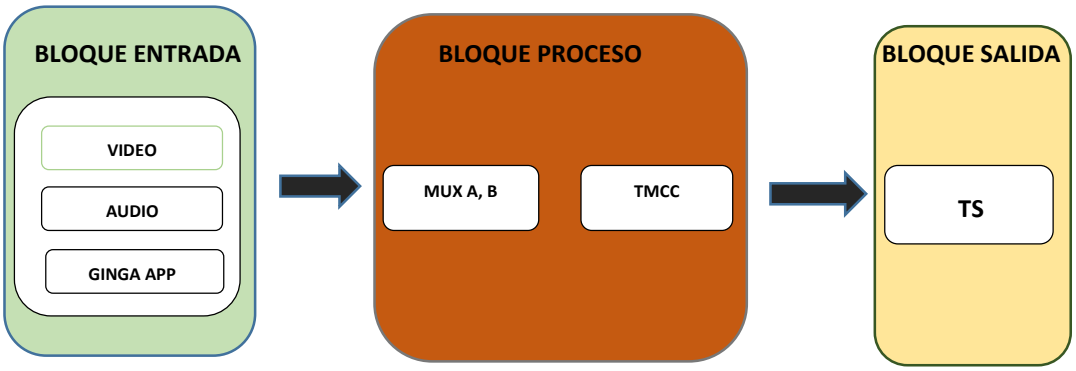


Figura 2. 3Estructura del servidor VillageFlow

Fuente: Autor

En cada bloque se configuró parámetros para generar el TS, para televisión en HD con interactividad, dentro del bloque de entrada se encuentra los Encoder, en el bloque de proceso los Remux y TMCC, y por último en el bloque de salida el DekTec Output Card y TS file Out.

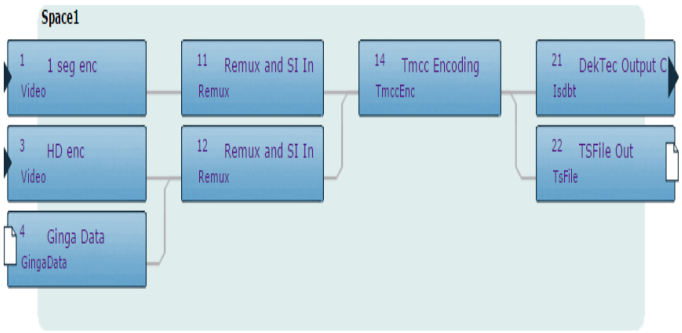


Figura 2. 4 Configuración por bloques del servidor VF para alerta temprana

Fuente: Autor

2.8.1.1 Bloque de entrada - Encoder

En el servidor VillageFlow para transmitir aplicaciones de sistemas de alertas de emergencia se configuró videocodec en H.264, audio AAC y formato MPEG-4 definidos en el estándar ISDB-Tb y mediante pruebas de laboratorio se ha definido un video rate de 8Mbps y audio rate 128Kbps, para One-Seg un video rate de 170Kbps y audio rate de 56Kbps. La resolución para HD es de 1920x1080 píxeles, mientras que One-Seg tiene una resolución LD de 320x240 píxeles a 30 cuadros por segundo. Ver Tabla

Parámetro	**UNACH_1Seg**	**UNACH_HD**
TS Rate	390 Kbps	12 Mbps
TS Packet Size	188 Bytes	188 Bytes
Vid Enc Format	H.264	H.264
Aud Enc Format	AAC	AAC
Video Rate	170 Kbps	8 Mbps
Audio Rate	56 Kbps	128 Kbps
TS Id	8	8
ProgNb	280	256
PmtPid	8136	80
PcrPid	255	256
VidPid	768	769
AudPid	512	513
PreSet	1seg_jp	HD_H.264
PrefVidAdapt	File	File
Aspect Ratio	4:3	16:9
Vid Convertion	All	All
Aud Mpeg Version	MPEG4	MPEG4
File	/avi_files/alerta	/avi_files/alerta
Video Format	320x240_2997	1920x1080_2997
AudSampleRate	48 Kbps	48 Kbps
Vid Rate Control	CBR	CBR
AudProfile	HEAACv2	HEAACv2
AudHeader	ADTS	ADTS

Tabla 7 Parámetros de configuración para señal TDT

Fuente: Autor

2.8.1.2 Bloque de Proceso - Remux

En el remux se configuró las tablas de información específica de los programas (PSI/SI), los servicios HD, One-Seg y la identificación del paquete (PID) necesarios en la capa A y B.

- En la Tabla 8 Network Information Table (NIT) se configuró el tipo de servicio, el modo de transmisión, el canal de guarda y la frecuencia de transmisión.

TABLA NIT		
Nombre	**Valor Dec**	**NIT actual**
Table ID	64	
Service_id	256	
Netwrk_id	8	
Descriptor_tag	65	Servicie List Descriptor
Service_type	1	Digital_Television
Guard_interval	1	1/16
Transmission_mode	2	Mode 3
Centre Frequency	3522	503.142 MHz

Tabla 8 Datos configurados en la NIT

Fuente: Autor

- En la Tabla 9 PMT, se modificó el número de programa, los PID de audio y video ingresados en el bloque de encoder.

Tabla PMT One-Seg		
Name	**Value Dec**	**Interpretation**
Table ID	2	PMT
Program_number	280	
ES Video		
Stream_type	27	H.264_Video
Elementary_PID	768	
ES Audio		
Stream_type	17	MPEG4_Audio
Elementary_PID	512	

Tabla 9 Datos configurados en PTM para one-seg

Fuente: Autor

2.8.1.3 Tmcc Encoding

En el componente TMCC Encoding se configuró las modulaciones, time interleave, números de segmentos y también se configura nuestro sistema de alerta de emergencia EWBS. Ver Tabla 10

Tmcc Encoding			
Broadcast	Tv		
Bandwidth	6		
Defaul Layer	B		
Emergency Flag	No		
Guard	1_16		
Mux	Yes		
Partial	Yes		
Mode	3		
Modulation	Number of Segments	Code Rate	Time Intervale
Qpsk	1	2_3	2
qam64	12	3_4	2
qam16	0	7_6	2

Tabla 10 Datos configurados en TMCC Encoding

Fuente: Autor

2.8.1.4 Bloque de Salida

En el bloque de salida se configuraron los componentes DekTec Output Card, para la transmisión de la señal TDT se configuró la tarjeta de salida DekTec con los siguientes parámetros (Tabla 11), en el componente TS

DekTec Output Card	
TS	**Valor**
TS Rate	29958294 bps
TS Packet Size	204 bytes
Parameters	
PrefDtAdapt	115
RfLevel	-180 dB
RfFrequency	503143000 Hz
Canal	19

Tabla 11 Datos configurados en la DEKTEC

Fuente: Autor

2.8.1.5 Análisis de la plataforma de VillageFlow para su activación remota

Para el análisis del servidor VillageFlow se ingresa en el lenguaje básico que está programado el servidor, el cual es PHP que un lenguaje de programación de uso general de código de lado del servidor, esta tecnología consiste en el procesamiento por lotes en servidor para general paginas HTML, para poder analizar el servidor VillageFlow se realiza los siguientes pasos.

Se abre a la carpeta VillageIsland donde está instalado el servidor VillageFlow

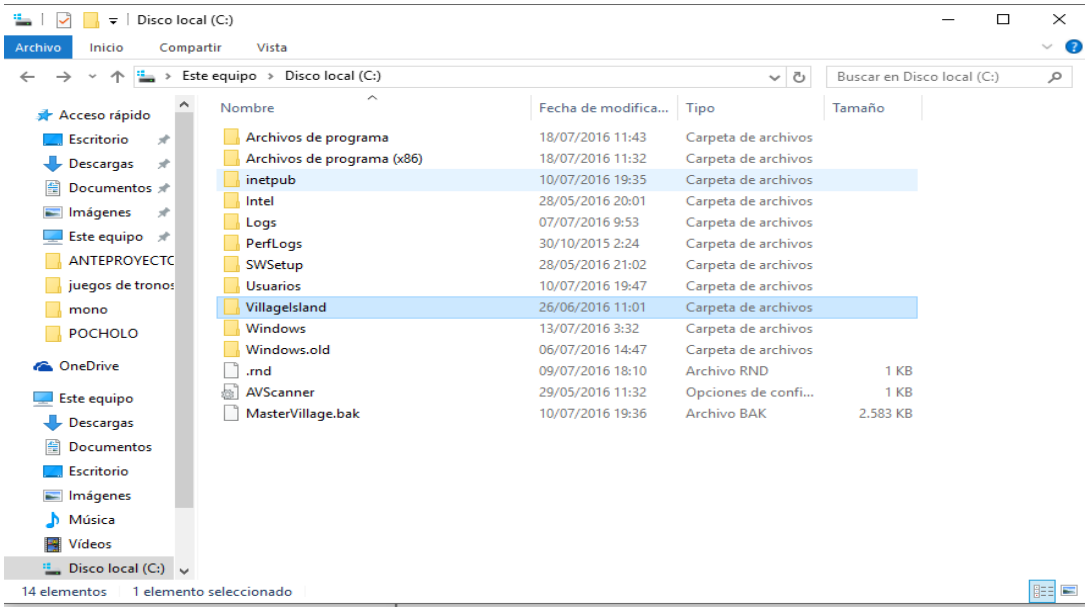


Figura 2. 5 Carpeta VillageIsland

Fuente: Autor

Seguidamente se abre la carpeta villageisland donde nos aparece otra carpeta llamada VillageFlow

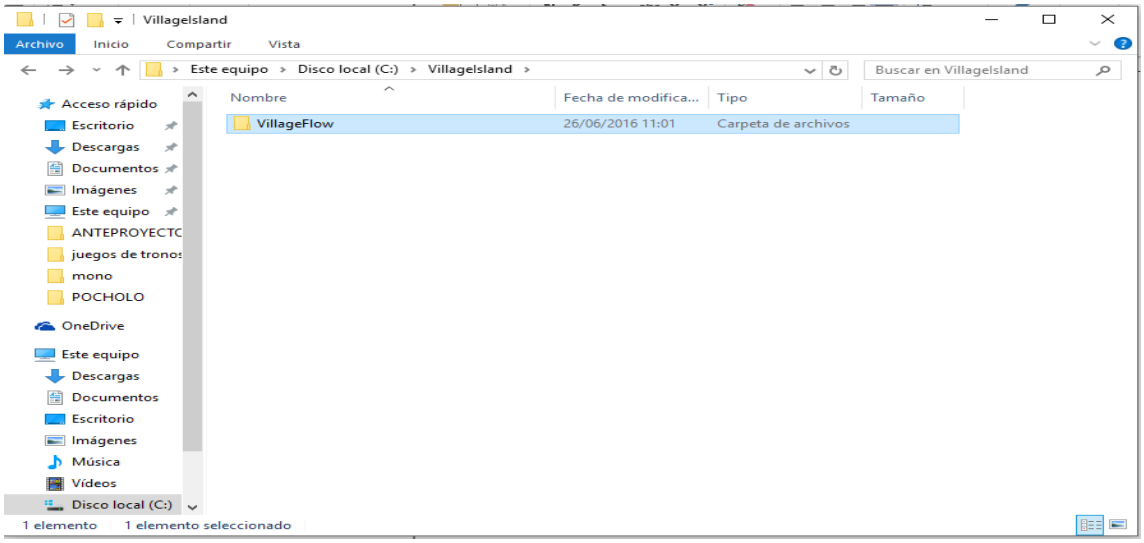


Figura 2. 6 Carpeta VillageFlow

Fuente: Autor

Después se abre la carpeta VillageFlow donde aparecer cuatro carpetas más de las cuales se abre la que dice Current.

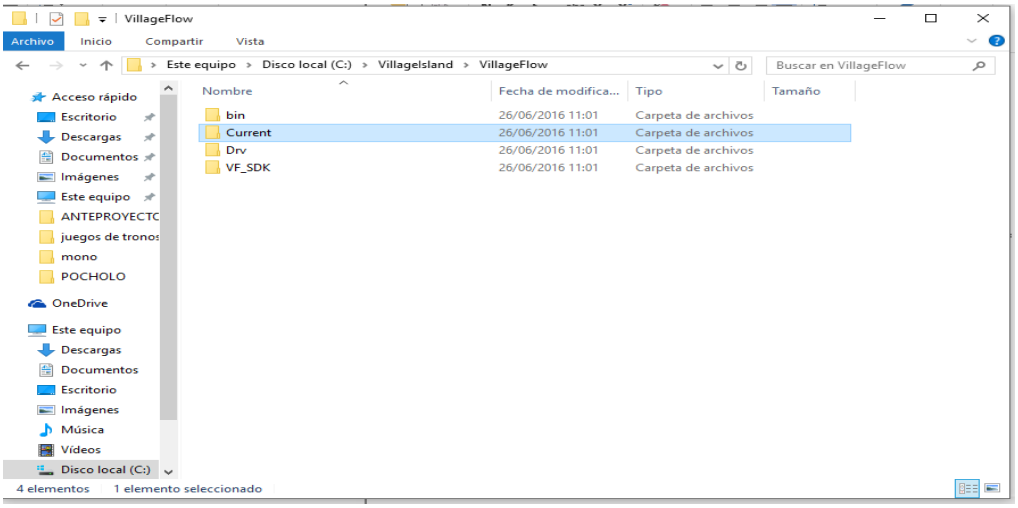


Figura 2. 7 Carpeta Current

Fuente: Autor

Seguidamente se despliega otro grupo de carpetas de las cuales se abre la de nombre Gui

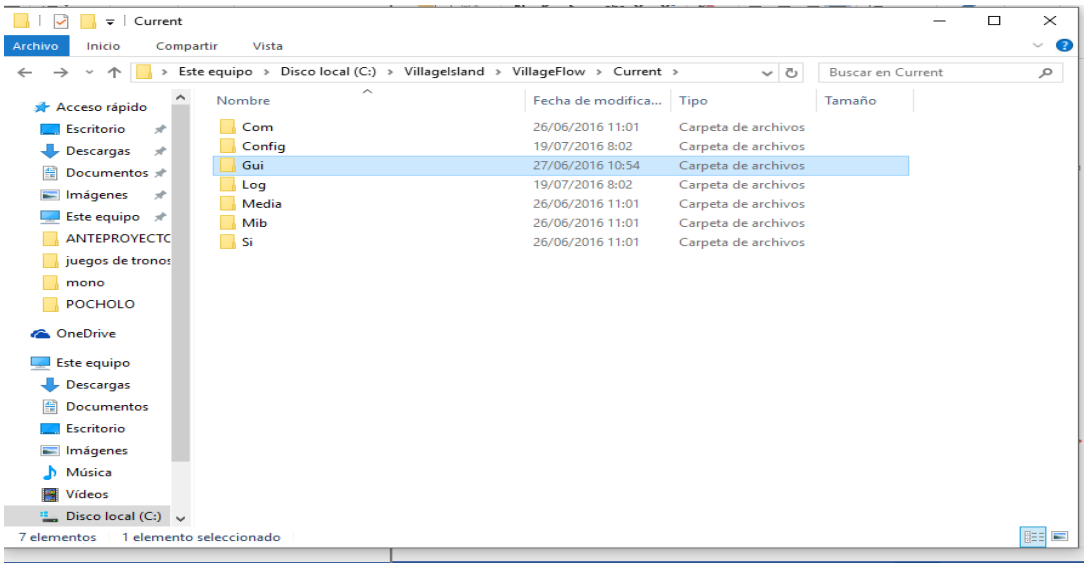


Figura 2. 8 Carpeta Gui

Fuente: Autor

En la cual aparecen un grupo de carpetas y algunos archivos con extensión PHP del cual analiza el que posee el nombre de **ajax_control_VF.php** que es el que permitirá realizar la activación de manera remota del servidor VillageFlow.

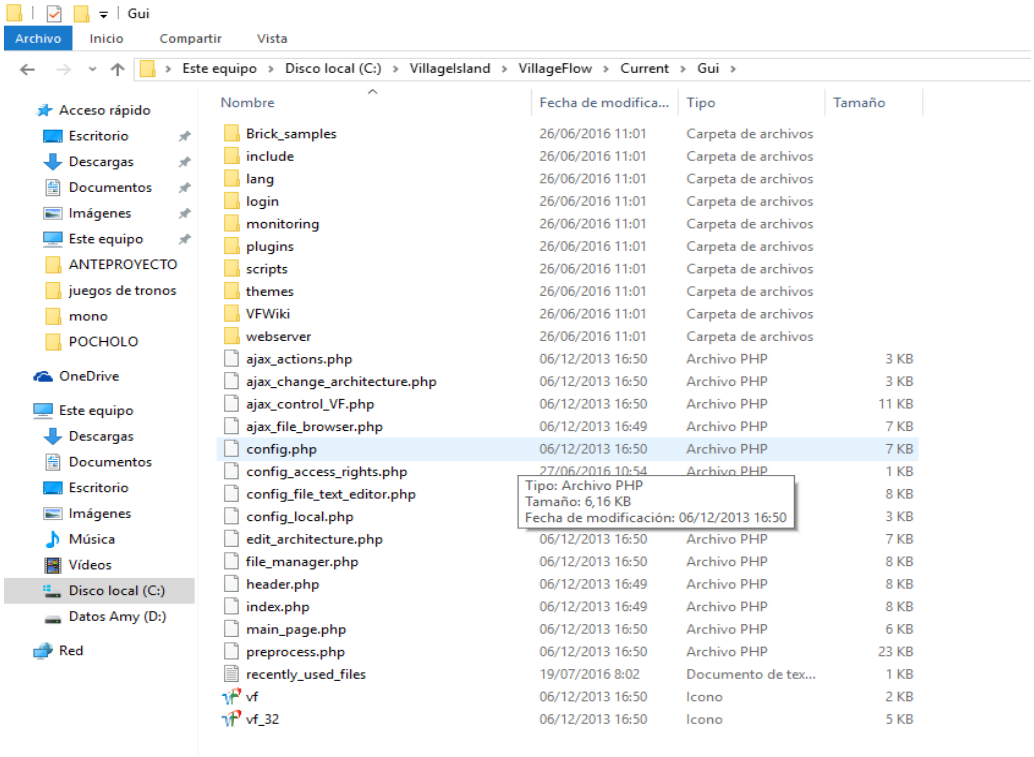


Figura 2. 9 Archivo ajax_control_VF.php

Fuente: Autor

Para analizar el archivo **ajax_control_VF.php,** se abre por un blog de nota para ver su diferente programación que contiene, en las cuales tienes diferentes tipos de acciones que realiza dicho archivo, en la cual se enmarca dos líneas de programación diferente las cuales permite la activación del servidor VillageFlow. Ver Anexo 1

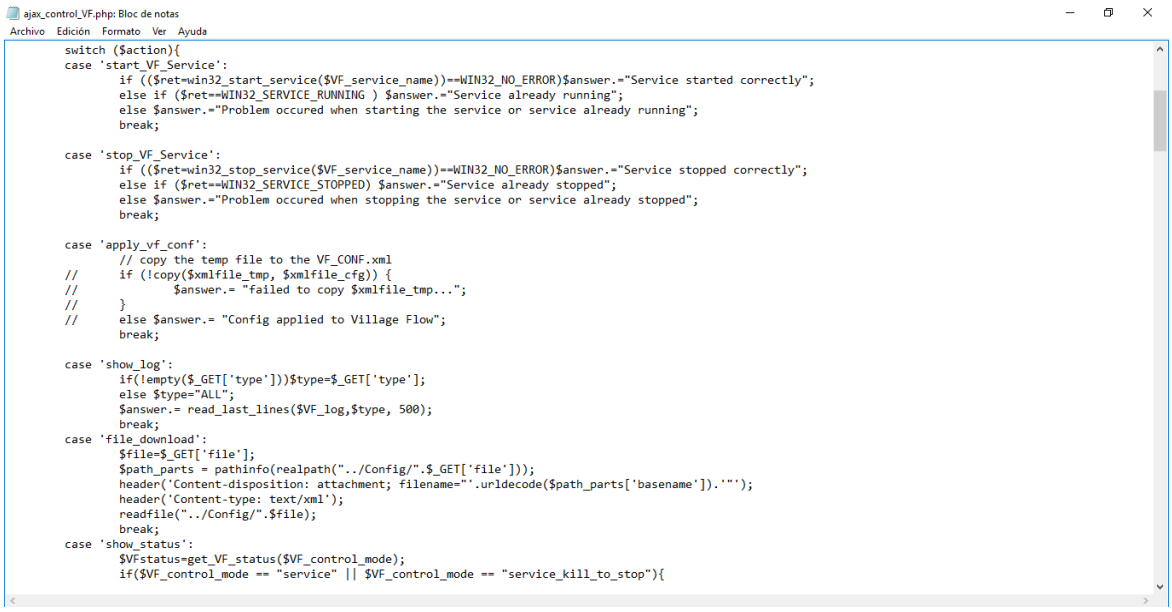


Figura 2. 10 Programación del archivo ajax_control_VF.php

Fuente: Autor

Después de analizar la programación del archivo **ajax_control_VF.php** las líneas de programación que permite manejar de manera remota el servidor VillageFlow es la siguiente.

En la siguiente línea de programación permite activar y desactivar el servidor VillageFlow.

```
case 'switch_status':
//        switch_VF_status($VF_control_mode);
$VFstatus=get_VF_status($VF_control_mode);
```

Mediante pruebas en el navegador lo que se hace es darle una dirección web **http://localhost/ajax_control_VF.php?action=switch_status** para ver el estado que se encuentra el servidor VillageFlow.

Como se observa ver en la figura 2.10 el servidor no se encuentra inicializado.

Figura 2. 11 Servidor VillageFlow detenido

Fuente: Autor

En siguiente figura 2.11 se aprecia que el servidor ya se encuentra inicializado.

Figura 2. 12 Servidor VillageFlow Activado

Fuente: Autor

La siguiente programación lo que permite saber con un valor numérico el estado del servidor VillageFlow.

```
case 'show_status_space':

        $VFstatus=get_VF_status($VF_control_mode);

        $Spaces=explode("_",$_GET['spacename']);
```

Mediante pruebas en el navegador se da una dirección web **http://localhost/ajax_control_VF.php?action=show_status** en la cual se observa que varían con valores numéricos cada vez que cambia el estado del servidor VillageFlow.

Cuando el valor es 1 quiere decir que nuestro servidor VillageFlow no se encuentra inicializado como se observa en la figura 2.12 esto sirve para la aplicación en Android al momento de ejecutar la activación del servidor VillageFlow.

Figura 2. 13 Estado servidor VillageFlow no inicializado

Fuente: Autor

En la siguiente figura 2.13 muestra un valor es 4 lo cual significa que el servidor VillageFlow se encuentra inicializado.

Figura 2. 14 Estado servidor VillageFlow inicializado

Fuente: Autor

2.8.2 Programación en Visual Studio para crear servicio WFC

En el lenguaje de programación Visual Studio se creó un servicio WCF el cual permitió la comunicación de nuestra base de datos el servidor VillageFlow y la aplicación en Android.

2.8.2.1 Diagrama de flujo del funcionamiento del servicios WCF

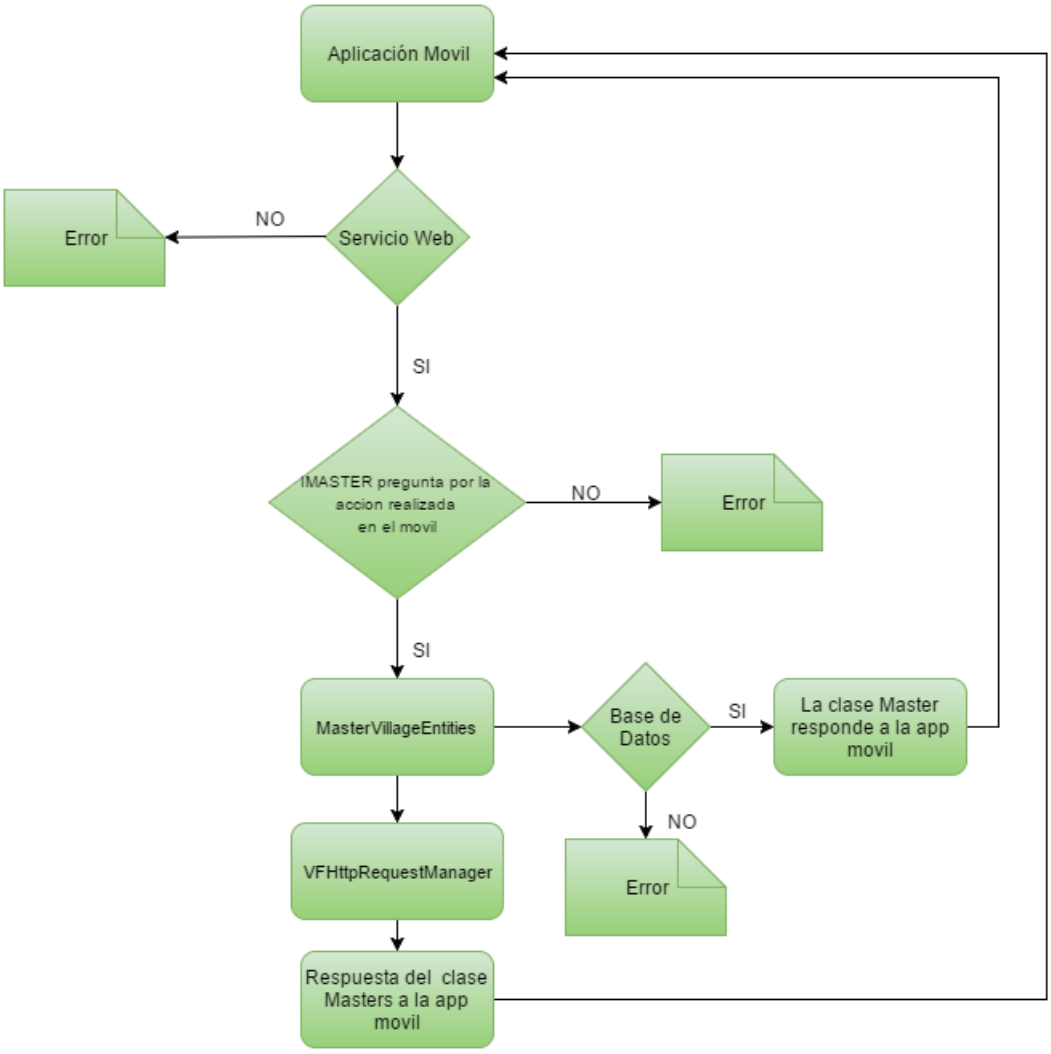


Figura 2. 15 Diagrama de flujo del funcionamiento servicios WCF

Fuente: Autor

Para realizar el servicio WCF se programó una clase con el nombre IMaster la cual es la que permite proveer de la diferente información que solicite la aplicación en Android.

La clase IMaster invocara tres métodos diferentes pero juntos formaran el servicio WCF.

- MasterVillageEntities
- Master
- VFHttpRequestManager

Cuando el usuario desea loguiarse en la aplicación en Android verifica que la información ingresada sea correcta por medio de la clase IMaster, en la siguiente programación pregunta por el método MasterVillageEntities

```
[OperationContract]
        [Description("Obtiene la información de
login de usuario.")]
        [WebInvoke(BodyStyle =
WebMessageBodyStyle.Bare, Method = "POST")]
            GetLoginResult GetLogin(Stream data);
```

Por medio del método MasterVillageEntities lo que se hace es consumir los recursos de la base datos en la cual le pregunta por todas las características de cómo está registrado el usuario en la base de datos.

```
Namespace Service.DataAccess.Model

{
    public partial class MasterVillageEntities : DbContext
    {
        public static MasterVillageEntities NewInstance()
        {
            return new MasterVillageEntities();
        }
        public static UsuarioBase GetLogin(GetLoginParams param)
        {
            using (MasterVillageEntities db = NewInstance())
            {
                return (from U in db.USUARIO
                        where U.user.Equals(param.User) &&
U.password.Equals(param.Password)
                        select new UsuarioBase
                        {
                            Apellidos = U.apellidos,
                            Cedula = U.cedula,
                            Celular = U.celular,
                            Codigo = U.codigo,
                            Direccion = U.direccion,
                            Email = U.email,
                            Nombres = U.nombres,
                            Telefono = U.telefono,
                            User = U.user
                        }).FirstOrDefault();
            }
        }

    }
}
```

Por medio del método Master se realiza devolución de una respuesta si es correctamente el ingreso del usuario o no en la aplicación en Android.

```
{
GetLoginParams p =
JsonSerializerDeserializer.Deserialize<GetLoginParams>(Streams.StreamToString(dat
a));
                ret.Login = MasterVillageEntities.GetLogin(p);

                if(ret.Login == null)
                {
                    ret.Result = ResultCodes.LoginWrong;
                }
                return ret;
            }
            catch
            {
                ret.Result = ResultCodes.Error;
                return ret;
```

Si el ingreso del usuario fue correcto la clase IMaster por medio de la siguiente programación.

```
[OperationContract(AsyncPattern =true)]
        [Description("Inicia o detiene el servicio de VillageFlow.")]
        [WebInvoke(BodyStyle = WebMessageBodyStyle.Bare, Method =
"POST")]
        Task<SwitchVFStatusResult> SwitchVFStatusAsync(Stream data);

        [OperationContract(AsyncPattern =true)]
        [Description("Obtiene el estado del servicio VillageFlow.")]
        [WebInvoke(BodyStyle = WebMessageBodyStyle.Bare, Method =
"POST")]
            Task<GetVFStatusResult> GetVFStatusAsync(Stream data);
```

Invoca al método **VFHttpRequestManager** el cual permite la activación y el estado del servidor VillageFlow en el que se encuentre en ese momento.

```
namespace Service.Request
{
    public class VFHttpRequestManager
    {
        private const String VillageFlowUri =
"http://192.168.0.100/ajax_control_VF.php?action=";
        public static async Task<string> GetVFStatus()
        {
            return await DoUriRequest($"{VillageFlowUri}show_status");
        }

        public static async Task<String> SwitchVFStatus()
        {
            return await DoUriRequest($"{VillageFlowUri}switch_status");
        }

        public static async Task<String> DoUriRequest(String uri)
                                        {
```

De la misma manera por medio del método Master respondemos a la aplicación en Android si se realizó la activacon y desactivación del servidor VillageFlow.

```
public async Task<SwitchVFStatusResult> SwitchVFStatusAsync(Stream
data)
        {
            SwitchVFStatusResult ret = new SwitchVFStatusResult();
            try
            {
                var response = await
VFHttpRequestManager.SwitchVFStatus();
                if(response == null)
                {
                    ret.Result =
ResultCodes.VillageFlowConnectionError;
                }
                return ret;
            }
            catch
            {
                ret.Result = ResultCodes.Error;
                return ret;
            }
        }
```

Una vez ya hechas todas programación de las clases lo que hace es publicar el servicio WCF si es publicado correctamente nos aparecerá en el navegador la siguiente figura.2.15

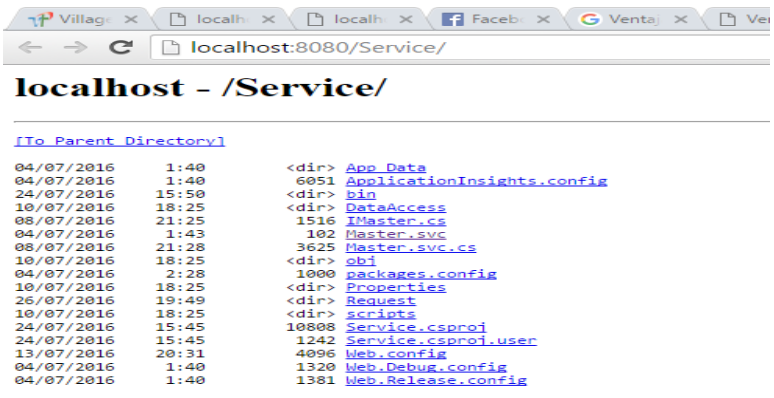


Figura 2. 16 localhost

Fuente: Autor

Después que aparece esa página web se procede a dar clic en el nombre Master.svc para poder publicar el servicio web como se ve en la figura 2.16

Figura 2. 17 Entorno del servicio WCF creado

Fuente: Autor

2.8.3 Programación de Base de datos en SQL Server

Para realización de nuestra base de datos la crearemos por medio del método de tablas la cual nos permitirá agregar los diferentes campos de los usuarios que necesitemos llenar, para lograr eso la programación necesaria es la siguiente.

En la cual realiza el registro del usuario para la base de datos con toda la información requerida, que después será consumida por el servicio web o WCF

```
INSERT INTO [dbo].[USUARIO]
           ([cedula]
           ,[nombres]
           ,[apellidos]
           ,[telefono]
           ,[celular]
           ,[direccion]
           ,[email]
           ,[user]
           ,[password])
     VALUES
           ('Darwin'
           ,'Navarrete'
           ,'032756890'
           ,'0987654321'
           ,'Chiriboga'
           ,'djnp@gmail.com'
           ,'JJJJ'
           ,'HHGHGH'
           ,'GHGHGH')
GO
```

Con el siguiente comando sirve para poder observar todos los usuarios registrados en la base de datos

```
SELECT * FROM [dbo].[USUARIO]
```

La siguiente línea de comando permite eliminar usuarios de la base de datos creada

```
DELETE FROM USUARIO WHERE CEDULA=''
```

En la siguiente figura podemos ver ya creada la base de datos que permitirá poder loguiarme desde la aplicación en Android Studio

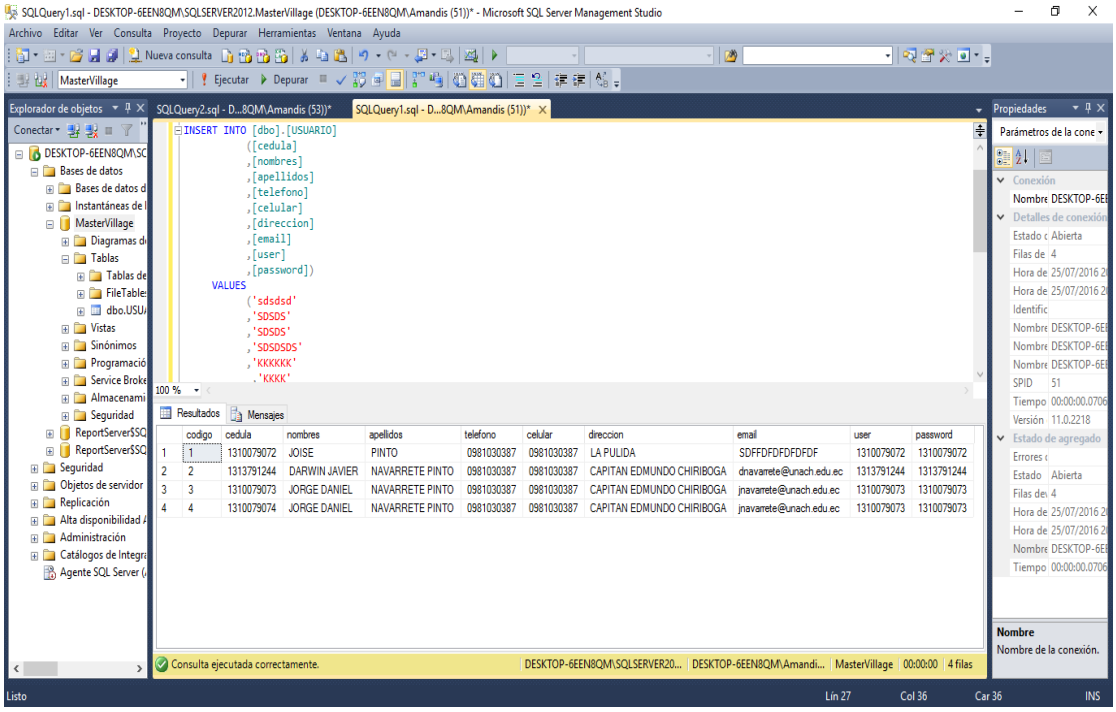


Figura 2. 18 Entorno de la base de datos creado

Fuente: Autor

2.8.4 Desarrollo de aplicación Android para realizar la activación remota del servidor VillageFlow

Por medio de la programación en Android Studio se creó la aplicación para acceder de manera remota al servidor VillageFlow para ello se lo realizó de la siguiente forma como se puede ver en la Figura 2.18

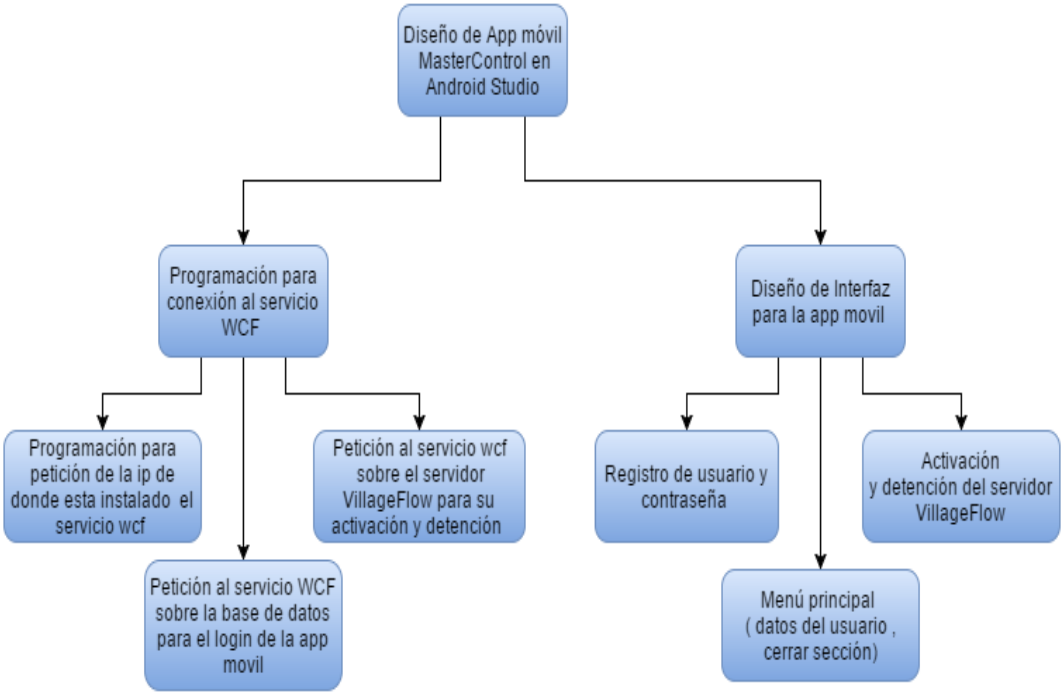


Figura 2. 19 Esquema del diseño de la app móvil

Fuente: Autor

Para poder manejar el servidor VillageFlow de manera remota tuvimos que acceder a al servicio WCF ya publicado en el sitio web, donde pregunta por la dirección ip de donde se encuentra el servicio WCF para eso utilizó las siguientes líneas de programación. Ver Anexo

```
public class MasterVillageRequestManager {
    private static final String BASE_URI =
"http://192.168.0.100:8080/Service/Master.svc/rest/";
    private static final String BASE_RESULT_ON_EXCEPTION =
"{\"Result\":\"ClientError\"}";
    public static final String
BASE_RESULT_ON_SERVER_CONNECTION_ERROR =
"{\"Result\":\"ServerConnectionError\"}";

    private static <T extends BaseResult> T
getBaseResult(String result, Class<T> type) {
        return JsonSerializerDeserializer.deserialize(result,
type);
```

Para poder loguiarnos en de manera correcta en la aplicación en Android Studio se utilizó las siguientes líneas de comandos en la cual pregunta por el usuario y la contraseña a servicio WCF ya creado.

```
public static GetLoginResult GetLogin(GetLoginParam
param, AsyncTask task)
```

Con la siguiente línea de comando se ejecuta la activación del servidor VillageFlow y además se pregunta por el estado que se encuentra el mismo. Ver Anexo

```
public static GetVFStatusResult GetVFStatus(AsyncTask
task)
```

```
public static SwitchVFStatusResult
SwitchVFStatus(AsyncTask task)
```

2.8.4.1 Interfaz de la aplicación móvil realizada en Android Studio.

La interfaz principal de la aplicación Android para la activación remota de alertas de emergencia contiene el nombre del usuario y la contraseña como se muestra en la figura 2.18.

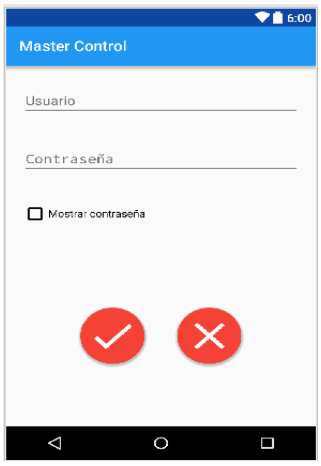


Figura 2. 20 Cara principal de la aplicación Master Control

Fuente: Autor

Una vez ingresado correctamente el usuario y la contraseña aparece un botón el cual permite ver estado del servidor VillageFlow y dando un clic para activar dicho servidor como se observa en la figura 2.18.

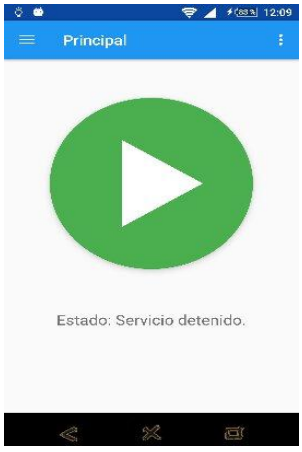


Figura 2. 21 Botón de Iniciar al Servidor VillageFlow

Fuente: Autor

Realizada la acción de activar el estado del botón cambia como se ve en la figura donde indica que se está ejecutando, dando otro clic permitirá detener el servidor VillageFlow

Figura 2. 22 Botón de Stop al Servidor VillageFlow

Fuente: Autor

Para poder ver más información de la aplicación, ingresando al menú principal donde se observar quien inicio sesión y poder cerrar la aplicación si el caso lo amerita.

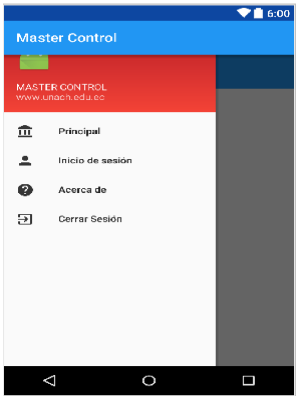


Figura 2. 23 Información de la aplicación

Fuente: Autor

2.8.5 Pruebas de activación remota de aplicaciones de alertas de emergencias

Para realizar las pruebas de activación remota de aplicación de alerta temprana para desastres naturales en el servidor VillageFlow, se transmite mediante la tarjeta DekTec a los decodificadores uno a la vez, se tomara el tiempo de descarga

de la aplicación en cada decodificador desde que el servidor inicia su funcionamiento.

SMART BOX	INICIO DE DESCARGA (Segundos)	DESCARGA (Segundos)	TIEMPO DE AR(segundos)
1	28,495	62,982	4,534
2	30,023	63,654	5,733
3	35,546	64,169	6,009
4	30,166	51,271	4,005
5	29,025	54,073	5,125
6	29,159	53,688	4,917
7	29,897	60,354	4,863
8	29,753	62,874	4,808
9	29,609	61,965	4,754
10	29,464	62,785	4,699
11	29,319	63,529	4,644
12	29,176	61,329	4,590
13	29,031	61,531	4,535
14	28,887	61,732	4,481
15	28,742	61,933	4,426
16	28,598	62,134	4,371
17	28,454	62,336	4,317
18	28,309	62,537	4,262
19	28,165	62,738	4,208
20	28,021	62,939	4,153
21	27,876	63,140	4,098
22	27,732	63,342	4,044
23	28,588	63,543	4,989
24	28,444	63,744	4,935
25	28,299	63,945	4,880
26	28,154	64,147	4,825
27	28,011	64,348	4,771
28	27,866	64,549	4,716
29	27,722	64,750	4,662
30	29,577	64,951	4,607
31	29,433	65,153	4,552
32	27,289	65,354	4,498
33	27,144	65,555	4,443
34	28,000	65,756	4,389
35	27,856	65,958	4,334

Tabla 12 Tiempo de descarga SMARTBOX

Fuente: Autor

EiTV Developer Box	INICIO DE DESCARGA (Segundos)	DESCARGA (Segundos)	TIEMPO DE AR(segundos)
1	37,080	114,041	4,534
2	39,302	112,906	5,733
3	39,722	114,567	6,009
4	35,553	114,364	4,005
5	37,639	114,627	5,125
6	37,069	114,89	4,917
7	36,807	115,153	4,863
8	36,543	115,416	4,808
9	36,281	115,679	4,754
10	36,017	115,942	4,699
11	35,753	116,205	4,644
12	35,491	116,468	4,590
13	35,227	116,731	4,535
14	34,965	116,994	4,481
15	37,701	117,257	4,426
16	37,437	117,52	4,371
17	36,175	117,783	4,317
18	36,911	118,046	4,262
19	36,649	118,309	4,208
20	36,385	118,572	4,153
21	38,121	118,835	4,098
22	36,859	119,098	4,044
23	39,595	119,361	4,989
24	39,333	119,624	4,935
25	39,069	119,887	4,880
26	38,805	120,15	4,825
27	38,543	120,413	4,771
28	38,279	120,676	4,716
29	36,017	120,939	4,662
30	35,753	121,202	4,607
31	36,489	121,465	4,552
32	37,227	121,728	4,498
33	36,963	121,991	4,443
34	36,701	122,254	4,389
35	36,437	122,517	4,334

Tabla 13 Tiempo de descarga EiTV Developer Box

Fuente: Autor

2.8.6 Comprobación de hipótesis

Para la comprobar la hipótesis se utiliza el método estadístico CHI-CUADRADO, que accede a dos grados de posibilidad, alternativa la que se quiere comprobar y nula (rechaza la hipótesis alternativa).

2.8.7 Planteamiento de la hipótesis estadística

Hipótesis nula (Ho): La configuración de la plataforma VillageFlow y su activación remota no permitirá el ahorro de tiempo para la trasmisión de señal TDT con aplicaciones de alerta temprana para desastres naturales.

Hipótesis alternativa (HI): La configuración de la plataforma VillageFlow y su activación remota permitirá el ahorro de tiempo para la trasmisión de señal TDT con aplicaciones de alerta temprana para desastres naturales.

2.8.8 Establecimiento del nivel de significancia

Las pruebas se realizaron con un 95% de confiabilidad, es decir, se trabajó con un nivel de significancia de $\alpha=0.05$.

2.8.9 Determinación del valor estadístico de prueba

Si el valor de CHI-CUADRADO es menor o igual que el CHI-CUADRADO critico entonces se acepta la hipótesis nula, caso contrario se la rechaza.

$$X^2 \leq \text{Valor Crítico}$$

Para aceptar o rechazar esta hipótesis se tomaron en cuenta 2 escenarios, un escenario A, medición de tiempos de activación remota con el EiTV Developer Box y un escenario B, medición de tiempos de activación remota con el decodificador SMARTHBOX.

En la Tabla 14 se muestran los valores obtenidos en cada uno de los escenarios en los cuales se realizaron las pruebas.

	Descarga(s)	Inicio (s)	Total	Proporción de Muestra
Escenario A	118,046	37,1110667	155,157067	0,775785333
Escenario B	62,5368	28,8522857	91,3890857	0,456945429
Total	180,5828	65,9633524	246,546152	1,232730762

Tabla 14 Valores Obtenidos
Fuente: Autor

Para obtener las frecuencias esperadas se multiplica el total de cada columna, por el total de cada fila, y se divide entre el total de cada fila y columna, puede apreciarse en la Tabla 15 y la Tabla 16 representa valores críticos definidos por este método.

	Descarga(s)	Inicio (s)	Proporción de Muestra
Escenario A	113,644838	41,5122287	0,775785333
Escenario B	66,937962	24,4511237	0,456945429

Tabla 15 Valores de frecuencias esperadas
Fuente: Autor

PROBABILIDAD

GRADOS DE LIBERTAD	0,995	0,990	0,975	0,950
1	0,000	0,000	0,001	0,004
2	0,010	0,020	0,051	0,103
3	0,720	0,115	0,216	0,352
4	0,207	0,297	0,484	0,711
5	0,412	0,554	0,831	1,145
6	0,676	0,872	1,237	1,635
7	0,989	1,239	1,690	2,167
8	1,344	1,646	2,180	2,733

Tabla 16 Valores Críticos Método Chi-Cuadrado
Fuente: Autor

La Tabla 17 muestra los valores obtenidos para nuestra comprobación de hipótesis a través del método CHI-CUADRADO.

# de filas	2
# de columnas	2
X^2=	1,718638
Grados de Libertad	1
Nivel de significación	0,05
Probabilidad	0,95
VALOR CRITICO	0.004

Tabla 17 Resultados del método estadístico del CHI-CUADRADO

Fuente: Autor

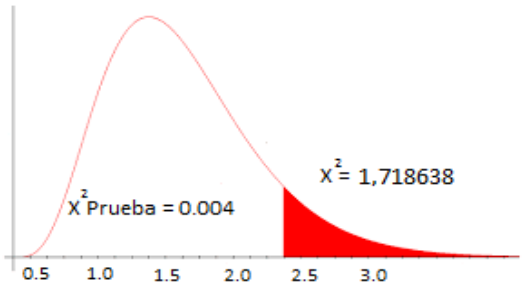


Figura 2. 24 Resultado de comparación x^2prueba con el x^2tabla método del Chi-Cuadrado

Fuente: Autor

En la Figura 2.24 se muestra la prueba Chi-Cuadrado, requiere la comparación del Valor Crítico (X^2tabla) con X^2(X^2prueba).

De acuerdo con el resultado se obtiene que X^2 es mayor que el valor crítico lo cual lleva a rechazar la hipótesis nula y aceptar la hipótesis alternativa, es decir:

"Hl: La configuración de la plataforma VillageFlow y su activación remota permitirá el ahorro de tiempo para la trasmisión de señal TDT con aplicaciones de alerta temprana para desastres naturales"

CAPITULO III

3 RESULTADOS

Se implementó un sistema de activación remota para transmisión de alerta de emergencia en la plataforma Villageflow para activar las aplicaciones TDT de alertas tempranas.

Los resultados que se obtuvo por medio de la aplicación móvil a la hora de su activación fue que se demora cuatro segundo en activar al servidor VillageFlow, y en mostrar la aplicación de alertas tempranas antes desastres naturales de origen volcánicos en nuestro televisor analógico es de 63 segundo y en los receptores móviles demora 20 segundo es más rápido ya que en el receptor móvil solo se manda video de alertas de emergencias en cambio en el televisor analógico se puede interactuar con el usuario por medio del control remoto.

Como se observa en a figura 3.1 en la aplicación móvil se nota que el servidor VillageFlow se encuentra activado y en el televisor analógico ya muestra la información de que está sucediendo un desastre natural de origen volcánico.

Figura 3. 1 Proceso de activación de la aplicación de alertas de emergencias

Fuente: Autor

De la misma manera cuando se realizó la desactivación por medio de la aplicación móvil que tarde cuatro segundo en conectarse servidor VillageFlow y desactivar para no emitir ninguna señal de emergencia a ningunos de nuestros dispositivos como son el televisor analógico y el receptor móvil como se muestra en la figura 3.2

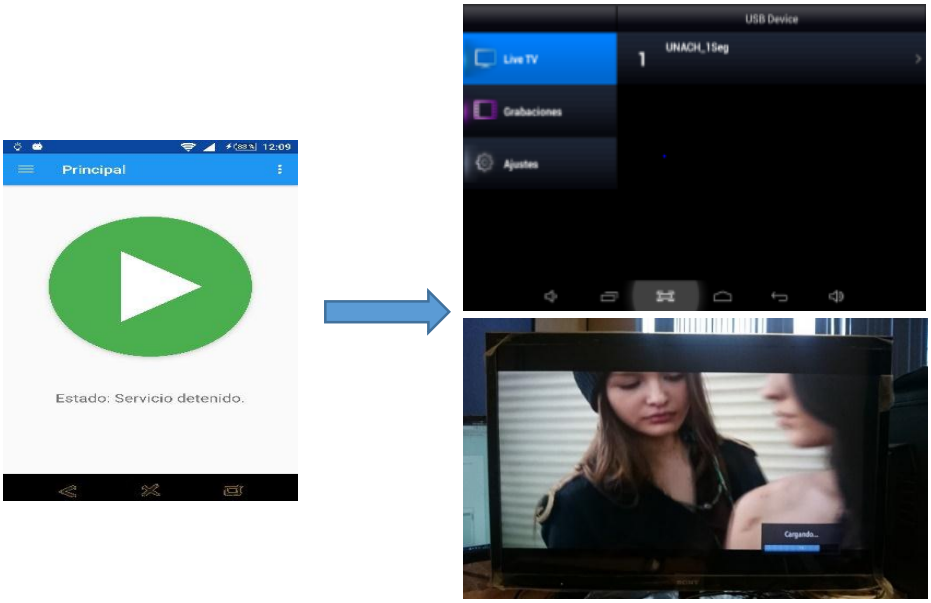


Figura 3. 2 Proceso de desactivación de la aplicación de alertas de emergencias

Fuente: Autor

CAPITULO IV

4 DISCUSIÓN

Nuestra aplicación en Android Studio permite la activación de servidor VillageFlow de manera remota, transmitiendo aplicaciones de alertas de emergencia contra desastres naturales de origen volcánicos, debido a que en nuestro país son los únicos eventos naturales que pueden ser predecibles, la aplicación permite enviar un mensaje de forma masiva a los televisores digitales y de esta manera todas las personas tengan a su alcance la información ante este tipo de emergencias. Esta aplicación permite ahorrar tiempos a la hora que ocurra un desastre natural de origen volcánico, porque no habrá la necesidad que las estaciones televisoras validen la información de que está ocurriendo un desastres natural de origen volcánico, porque la Secretaria Nacional de Riesgo o los COE provinciales contarían con la aplicación en Android y lograrías activar los sistemas de alertas de emergencia sin importar la información que estuvieses transmitiendo en ese momento las estaciones televisoras.

CAPITULO V

5 Conclusiones y Recomendaciones

5.1 Conclusiones

- La configuración y análisis desarrollado en el servidor VillageFlow permite realizar su control de activación remota y además de transmitir una señal de TDT que contiene audio, video e interactividad la misma que es receptada por los diferentes decodificadores, que hace posible la visualizar las aplicación de alerta tempranas en televisor analógico y receptores móviles.

- Por medio de la aplicación móvil de activación remota de sistemas de alertar de emergencias se aprovechó las potencialidades del servidor VillageFlow ya que emite señal de televisión digital terrestre - ISDB-Tb con alertas de emergencias con el fin ahorrar tiempos a la hora de un desastres naturales de origen volcánico.

- Los servicios WCF tienen una gran ventaja al momento de realizar servicios web ya que permiten consumir recursos mediante una aplicación móvil, y se basa en el método solicitud y respuesta, los envíos de mensaje se basan en el protocolo HTTP que son compatibles con el software Android Studio.

- El lenguaje Android Studio facilita el desarrollo de aplicaciones móviles es un lenguaje declarativo, descriptivo y fácilmente interpretado por el programador y además es un software gratuito, nos permitió realizar la aplicación de forma que el usuario no tenga inconveniente al momento de utilizarla.

5.2 Recomendaciones

- Para instalar la aplicación móvil es necesario que el software cuente con una versión Android 4.4 o mayor para que se ejecute la aplicación correctamente en móvil.

- Utilizar en el servidor VillageFlow con una dirección ip estática por motivos que el servidor WCF para el proceso de la activación remota pregunta por la dirección ip de nuestra plataforma VillageFlow

- Actualizar el hardware y software del servidor VillageFlow para aumentar su desempeño en la ejecución de la aplicación de alerta temprana.

- Ubicar todo el sistema de TDT en espacio apropiado para realizar pruebas de transmisión y recepción de señales de televisión digital.

CAPITULO VI

6 PROPUESTA

6.1 Título de la propuesta

SISTEMA DE ACTIVACIÓN REMOTA PARA TRANSMISIÓN DE ALERTA DE EMERGENCIA EN LA PLATAFORMA VILLAGEFLOW PARA ACTIVAR LAS APLICACIONES TDT DE ALERTAS TEMPRANAS.

6.2 Introducción

Los primeros sistemas de alertas tempranas tienen origen en el continente asiático debido a la posición geográfica en la que se encuentra han sido víctimas de varios eventos naturales muchos de estos desastrosos por este motivo ha existido la necesidad de crear sistemas de alerta mucho más sofisticados empleados en la nuevas tecnologías existentes como televisión digital terrestre, en países de centro y Sur América han adoptado estos sistemas con la finalidad de reducir la vulnerabilidad frente a estos eventos físicos, ahora en Ecuador debido a la política que maneja el gobierno de la revolución ciudadana muy pronto se migrará a TDT y gracias a esto se podrá mejorar los planes de contingencia de las diferentes ciudades del país.

El mundo está en proceso de transición de televisión analógica a digital, y en Ecuador se adoptado el estándar ISDB-Tb, con esto se prevé que el apagón analógico se ejecute en el año 2018.

La Universidad Nacional de Chimborazo por medio de la carrera de Ingeniería Electrónica y Telecomunicaciones, en sus proyectos de investigación siempre se encuentra inmersa en aportar con dichos proyectos a la sociedad en general. Para seguir dándole un aporte a la sociedad se realizó un proyecto que permite por medio

de una aplicación móvil la activación remota de un servidor VillageFlow, que permite general contenido TDT antes y durante un desastre natural de origen volcánico con el fin de mantener informada a la población durante una erupción volcánica

6.3 Objetivos

6.3.1 Objetivo General

- Implementar sistema de activación remota para transmisión de alerta de emergencia en la plataforma Villageflow para activar las aplicaciones TDT de alertas tempranas.

6.3.2 Objetivos Específicos

- Analizar la plataforma Villageflow para activación remota.
- Configurar la plataforma Villageflow para transmitir señal TDT de alertas de emergencia.
- Desarrollar un servicio WCF
- Desarrollar la aplicación móvil para activación remota del servidor VillageFlow.

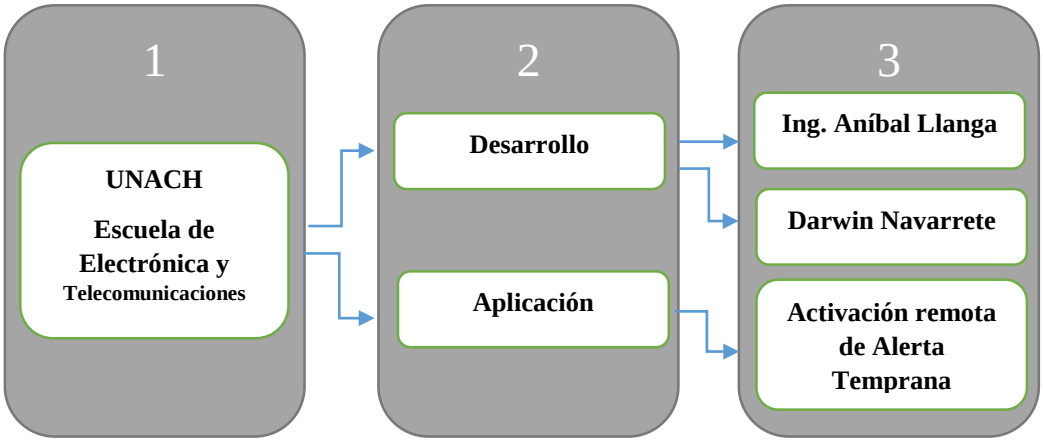
6.4 Fundamentación Científico-Técnico

La implementación de un sistema de activación remota de alerta temprana para desastres naturales de origen volcánico está encaminada a la televisión digital terrestre, mediante esta tecnología se contribuye al plan de emergencia ante desastres naturales

6.5 Descripción de la propuesta

La aplicación para activación remota del servidor de televisión digital terrestre VillageFlow está desarrollado para emitir información de alertas tempranas antes desastre naturales de origen volcánico.

6.6 Diseño organizacional



6.7 Monitoreo y Evaluación de la propuesta

La evaluación de la propuesta se realizara en base a pruebas, para esto se midió el tiempo de conexión desde la aplicación móvil al servidor VillageFlow.

La aplicación móvil para activación remota de alertas temprana para desastres naturales, utilizando plataforma Villageflow, para la zona tres del Ecuador, contribuye con emisión de una señal TDT para alertas emergencia e información de rutas de evacuación.

7 BIBLIOGRAFÍA

DecTek. (2007 de Marzo de 20). *DecTek*. Obtenido de DecTek: http://www.bjpace.com.cn/DEKTEC/english/DTA-115.pdf

Espin, P. L. (4 de Agosto de 2014). *UNIVERSIDAD CATÓLICA.* Obtenido de http://repositorio.ucsg.edu.ec/bitstream/123456789/2878/1/T-UCSG-PRE-TEC-ITEL-76.pdf

García, R. O. (2014). *Instituto de Microelectronica Aplicada.* Obtenido de Instituto de Microelectronica Aplicada: http://www.iuma.ulpgc.es/~nunez/clases-micros-para-com/mpc0809-trabajos/mpc0809RuymanOjedaSTBs.pdf

Guido Ovaco, Y. P. (2014). *Dspace Espol.* Obtenido de Dspace Espol: https://www.dspace.espol.edu.ec/bitstream/123456789/25417/1/Resumen%20de%20tesis%20GOvaco%20y%20YPilco,%20director%20de%20tesis%20%20%20M.Sc.%20C%C3%A9sar%20Yepez%20F.%2023%20dic%202013.pdf

Marset, N. R. (junlio de 2006). *users.dsic.upv.* Obtenido de users.dsic.upv: http://users.dsic.upv.es/~rnavarro/NewWeb/docs/RestVsWebServices.pdf

MINISTERIO DE INDUSTRIA, E. Y. (3 de Abril de 2010). *Ministerio de Industrio, Energia y Turismo*. Obtenido de http://www.televisiondigital.gob.es/tecnologias/Interactividad/Paginas/interactividad.aspx

MINTEL. (29 de Mayo de 2015). *MINTEL*. Obtenido de MINTEL: http://www.telecomunicaciones.gob.ec/mintel-propondra-politica-publica-sobre-la-emision-de-alertas-de-emergencia-a-traves-de-la-senal-de-tdt/

National Geographic. (2013). *National Geographic*. Obtenido de National Geographic: http://www.nationalgeographic.es/medio-ambiente/desastres-naturales/volcanes-definicion

Paredes, D. (2014). *La TVdi: concepto y factor de impulso de los Sistemas de Alertas Tempranas. Un caso práctico: Proyecto Remediando.* Mérida - Venezuela.

Parreño, J. (2014). *CREACIÓN DE NUEVOS SERVICIOS DE TELEVISIÓN DIGITAL TERRESTRE BAJO EL ESTÁNDAR ISDB-Tb PARA LA PLATAFORMA VILLAGEFLOW MEDIANTE EL ANÁLISIS DE SU ESTRUCTURA.* Sangolquí.

Patiño, C. L. (2014). *dspace.espoch.* Obtenido de dspace.espoch: http://dspace.espoch.edu.ec/bitstream/123456789/3325/1/18T00548.pdf

Tiupul Urquizo, V. k. (2015). *dspace.unach.edu.ec.* Obtenido de dspace.unach.edu.ec:

http://dspace.unach.edu.ec/bitstream/51000/602/1/UNACH-EC-IET-2015-0003.pdf

Tv Interactica GINGA. (s.f.). *Tv Interactica GINGA*. Obtenido de Tv Interactica GINGA: http://www.ginga.org.br/es

Tv, E. (2016). *Ei Tv Entretenimiento e Interactividad para Televisión*. Obtenido de Ei Tv Entretenimiento e Interactividad para Televisión: http://www.eitv.com.br/es/produtos/eitv-smartbox/

village island. (2015). Obtenido de http://village-island.com/en/villageisland/software/villageflow/

wiwiloz. (2006). *wiwiloz*. Obtenido de wiwiloz: https://wiwiloz.wordpress.com/iis-internet-information-server/

8 ANEXOS

Análisis de la plataforma VillageFlow

```
<?php
if(is_file("./include/FirePHPCore/fb.php")){
include('./include/FirePHPCore/fb.php');
ob_start();
}

include_once "config.php";
include_once "include/functions.php";
include_once "include/functions_com.php";

function handle_request($action){
global $VF_executable_name;
global $VF_service_name;
global $GUI_access_limited_to_1_window;

if(!isset($VF_control_mode)) global $VF_control_mode;
global $VF_autorecovery;
global $VF_log;
global $VF_dir;
$answer="";

switch ($action){
case 'start_VF_Service':
if
(($ret=win32_start_service($VF_service_name))==WIN32_NO_ERROR)$answe
r.="Service started correctly";
else if ($ret==WIN32_SERVICE_RUNNING ) $answer.="Service already
running";
```

```
else $answer.="Problem occured when starting the service or service already running";
break;

case 'stop_VF_Service':
if ((($ret=win32_stop_service($VF_service_name))==WIN32_NO_ERROR)$answer.="Service stopped correctly";
else if ($ret==WIN32_SERVICE_STOPPED) $answer.="Service already stopped";
else $answer.="Problem occured when stopping the service or service already stopped";
break;

case 'apply_vf_conf':
// copy the temp file to the VF_CONF.xml
//	if (!copy($xmlfile_tmp, $xmlfile_cfg)) {
//		$answer.= "failed to copy $xmlfile_tmp...";
//	}
//	else $answer.= "Config applied to Village Flow";
break;

case 'show_log':
if(!empty($_GET['type']))$type=$_GET['type'];
else $type="ALL";
$answer.= read_last_lines($VF_log,$type, 500);
break;
case 'file_download':
$file=$_GET['file'];
$path_parts = pathinfo(realpath("../Config/".$_GET['file']));
header('Content-disposition: attachment; filename="'.urldecode($path_parts['basename']).'"');
header('Content-type: text/xml');
```

```
readfile("../Config/".$file);
break;
case 'show_status':
$VFstatus=get_VF_status($VF_control_mode);
if($VF_control_mode == "service" || $VF_control_mode ==
"service_kill_to_stop"){
if($VF_autorecovery && $VFstatus==1){
// Autorecovery
if(!isset($_COOKIE['VF_cookie']['VF_started'])){
setcookie("VF_cookie[VF_started]", "FALSE", time()+2600000);
}
// Check if cookie says it's a crash
elseif($_COOKIE['VF_cookie']['VF_started']=="TRUE"){
// try to restart VF
win32_start_service($VF_service_name);
}

}
$answer.= $VFstatus;
}
else if($VF_control_mode == "application"){
// Autorecovery
if($VF_autorecovery && $VFstatus==1){
if(!isset($_COOKIE['VF_cookie']['VF_started'])){
setcookie("VF_cookie[VF_started]", "FALSE", time()+2600000);
}
// Check if cookie says it's a crash
elseif($_COOKIE['VF_cookie']['VF_started']=="TRUE"){
// try to restart VF
runAsynchronously("".$VF_dir.'/bin/'.$VF_executable_name."");
}
```

```
}
$answer.= $VFstatus;
}
break;

case 'init_brick':
$VFstatus=get_VF_status($VF_control_mode);
$brick_params=explode("_",$_GET['brick_params']);
$result="";
if($VFstatus==4){
$name="VillageFlow_Spaces_".$brick_params[0]."_Bricks_".$brick_params[1]."
_Controls_Init";
$VFstatus_brick= comset_from_name($name, "1");
$result=$VFstatus_brick;
}
$answer.=$result;
break;

case 'show_status_space':
$VFstatus=get_VF_status($VF_control_mode);
$Spaces=explode("_",$_GET['spacename']);
$result="";
if($VFstatus['CurrentState']){
foreach($Spaces as $Space){
$names[]="VillageFlow_Spaces_".$Space."_Controls_Start";
}
$VFstatus_space= comget_from_name($names);
foreach($VFstatus_space as $Space_status){
$result.=$Space_status."_";
}
```

```
}
else{
foreach($Spaces as $Space){
$result.="2_";
}
}
$answer.= substr($result,0,-1);
break;

case 'set_lock':
if($GUI_access_limited_to_1_window){
if(!empty($_GET['lockid']))$lockid=$_GET['lockid'];
if(is_file("lock.txt")){
$lockfile=explode("|",file_get_contents("lock.txt"));
$now=mktime(date("H"), date("i"), date("s"), date("m")  , date("d"), date("Y"));
if($now<$lockfile[0]){
// The "server cookie" is still valid
if($lockfile[1]==$lockid){
$answer.= "ok";
}
else{
$answer.= "locked";
}
}
else{
// server cookie expired
$expire_date  = mktime(date("H"), date("i"), date("s")+30, date("m")  , date("d"), date("Y"));
$locktxt=$expire_date."|".$lockid;
file_put_contents("lock.txt", $locktxt);
$answer.= "ok";
```

```
}
}
else{
// First time to create the server cookie
touch("lock.txt");
$expire_date  = mktime(date("H"), date("i"), date("s")+30, date("m")  , date("d"), date("Y"));
$locktxt=$expire_date."|".$lockid;
file_put_contents("lock.txt", $locktxt);
$answer.= "ok";
}
}
else{
$answer.= "ok";
}
break;

case 'refresh_bitrate':
$bitrates=array("VillageFlow_Spaces_Space_Bricks_".$_GET['brickname']."_InCons_TsIn_Probe_TotalRate");
$VF_bitrate= comget_from_name($bitrates);
$answer.= $VF_bitrate[0];
break;

case 'switch_status':
//        switch_VF_status($VF_control_mode);
$VFstatus=get_VF_status($VF_control_mode);

if(!isset($xmlfile_tmp)) global $xmlfile_tmp;
if(!isset($xmlfile_running)) global $xmlfile_running;
if(!isset($path_current)) global $path_current;
```

```
if(!isset($xmlfile_current)) global $xmlfile_current;

if($VFstatus==1){  //  Starting VF
//  Service stopped. try to start it
//                      com_clean_folder();
// copy the config file to the VF_CONF_tmp.xml (running file)
if (!copy($xmlfile_tmp, $xmlfile_running)) {
$answer.= "failed to copy $xmlfile_tmp...\n";
}
// set the current running file in the "recently_used_files.txt"
add_current_to_recently_used($path_current."/".$xmlfile_current);
}

if($VF_control_mode == "service"){
                    switch ($VFstatus){
                    case 1:
                        // Start the Windows service
                        if
(($ret=win32_start_service($VF_service_name))==WIN32_NO_ERROR){
                            setcookie("VF_cookie[VF_started]",
"TRUE", time()+2600000);
                            $answer.= "Service started correctly";
                        }
                        else if ($ret==WIN32_SERVICE_RUNNING )
$answer.= "Service already running";
                        else $answer.= "Problem occured when starting the
service or service already running";
                        break;
                    case 2:
                        //  Service starting
                        break;
                    case 4:
```

```
                    //  Service running. try to stop it
                    if
(($ret=win32_stop_service($VF_service_name))==WIN32_NO_ERROR){
                        setcookie("VF_cookie[VF_started]",
"FALSE", time()+2600000);
                        $answer.= "Service stopped correctly";
                    }
                    else if ($ret==WIN32_SERVICE_STOPPED)
$answer.= "Service already stopped";
                    else $answer.= "Problem occured when stopping the
service or service already stopped";

    if(is_file($xmlfile_running))unlink($xmlfile_running);
                    break;
                };
            }
            else if($VF_control_mode == "service_kill_to_stop"){
                switch ($VFstatus){
                case 1:
                    // Start the Windows service
                    if
(($ret=win32_start_service($VF_service_name))==WIN32_NO_ERROR){
                        setcookie("VF_cookie[VF_started]",
"TRUE", time()+2600000);
                        $answer.= "Service started correctly";
                    }
                    else if ($ret==WIN32_SERVICE_RUNNING )
$answer.= "Service already running";
                    else $answer.= "Problem occured when starting the
service or service already running";
                    break;
                case 2:
                    break;
```

```
                        case 4:
                                //      $answer.= "try to kill Wget \n";
        //                      exec("tskill VillageFlowService /a /v", $return);
                                setcookie("VF_cookie[VF_started]", "FALSE",
time()+2600000);
                                exec("taskkill /f /im VillageFlowService.exe",
$return);
                                $return_utf8 = iconv("SHIFT-JIS", "UTF-
8//IGNORE", $return[0]);
        //                      $answer.= $return_utf8;
                                        $answer.= "Service stopped correctly";
                                if
(($ret=win32_stop_service($VF_service_name))==WIN32_NO_ERROR){
                                        $answer.= "Service stopped correctly";
                                }
                                else if ($ret==WIN32_SERVICE_STOPPED) {
                                        $answer.= "Service already stopped";
                                }
                                else $answer.= "Problem occured when stopping the
service or service already stopped";

        if(is_file($xmlfile_running))unlink($xmlfile_running);
                                break;
                        };


                }
                else if($VF_control_mode == "application"){
                        if($VFstatus==4){
                                // Try to kill VF
                                if(strpos(PHP_OS,"WIN")!==false){
                                        // Windows
                                        exec("taskkill /f /im
".$VF_executable_name, $return);
```

```
                                        setcookie("VF_cookie[VF_started]",
"FALSE", time()+2600000);
                                        $answer.= "Stopping VF application";

        if(is_file($xmlfile_running))unlink($xmlfile_running);
                                }
                                else if(strpos(PHP_OS,"Linux")!==false){
                                        // Linux
                                        exec('killall -9 vf', $return2, $output2);
                                        setcookie("VF_cookie[VF_started]",
"FALSE", time()+2600000);
                                        $answer.= "Stopping VF application";

        if(is_file($xmlfile_running))unlink($xmlfile_running);
                                }
                        }
                        else{
                                $answer.= "Starting VF application";
                                setcookie("VF_cookie[VF_started]", "TRUE",
time()+2600000);
                                if(strpos(PHP_OS,"WIN")!==false){
                                        // Windows

        $command=$VF_dir.'/bin/'.$VF_executable_name;
                                        $args='../Current/Config/VF_CONF.xml';
                                        runAsynchronously($command, $args);

                                }
                                else if(strpos(PHP_OS,"Linux")!==false){
                                        // Linux
                                        $command=$VF_dir."/bin/vf >log.txt &";
                                        system($command);
                                }
```

```
                }
        }
        break;


    case 'switch_status_space':
        $VFstatus=get_VF_status($VF_control_mode);
        if($VFstatus==4){
            $stateto=$_GET['stateto'];
            switch ($stateto){
            case 1:  // start/reinitialize space
                com_clean_folder();
                if
(comset_from_name("VillageFlow_Spaces_".$_GET['spacename']."_Controls_St
art","1")) $answer.= "Config started correctly";
                else $answer.= "Problem occured when reinitializing
the config file";
                break;
            case 0:  // stop space
                if
(comset_from_name("VillageFlow_Spaces_".$_GET['spacename']."_Controls_St
art","0")){
    //
    setcookie("VF_cookie[Space_started][".strval($_GET['spacename'])."]", 0,
time()+2600000);
                        $answer.= "Space stopped correctly";
                }
                else $answer.= "Problem occured when stopping the
space";
                break;
            };
            break;
        }
        else $answer.= "service not started";
```

```
                break;
        }
        return $answer;
}

if(isset($_GET['action'])){
        echo handle_request($_GET['action']);
}

?>
```

Anexo 2

Servicio WCF

Master

```
namespace Service
{
    // NOTA: puede usar el comando "Rename" del menú "Refactorizar" para
cambiar el nombre de clase "Master" en el código, en svc y en el archivo de
configuración a la vez.
    // NOTA: para iniciar el Cliente de prueba WCF para probar este
servicio, seleccione Master.svc o Master.svc.cs en el Explorador de
soluciones e inicie la depuración.
    public class Master : IMaster
    {
        public GetLoginResult GetLogin(Stream data)
        {
            GetLoginResult ret = new GetLoginResult();
            if (data == null)
            {
                ret.Result = ResultCodes.MissingParams;
                return ret;
            }

            try
            {
                GetLoginParams p =
JsonSerializerDeserializer.Deserialize<GetLoginParams>(Streams.StreamToStri
ng(data));
                ret.Login = MasterVillageEntities.GetLogin(p);

                if(ret.Login == null)
                {
                    ret.Result = ResultCodes.LoginWrong;
                }
                return ret;
            }
```

```
            catch
            {
                ret.Result = ResultCodes.Error;
                return ret;
            }
        }

        public HelloWorldResult HelloWorld(Stream data)
        {
            HelloWorldResult ret = new HelloWorldResult();
            if(data == null)
            {
                ret.Result = ResultCodes.MissingParams;
                return ret;
            }

            try
            {
                HelloWorldParams p =
JsonSerializerDeserializer.Deserialize<HelloWorldParams>(
Streams.StreamToString(data));
                ret.Greeting = $"HelloWorld {p.Name}";
                return ret;
            }
            catch
            {
                ret.Result = ResultCodes.Error;
                return ret;
            }
        }

        public async Task<SwitchVFStatusResult> SwitchVFStatusAsync(Stream
data)
        {
            SwitchVFStatusResult ret = new SwitchVFStatusResult();
            try
            {
                var response = await VFHttpRequestManager.SwitchVFStatus();
                if(response == null)
                {
                    ret.Result = ResultCodes.VillageFlowConnectionError;
                }
                return ret;
            }
            catch
            {
                ret.Result = ResultCodes.Error;
                return ret;
            }
        }

        public async Task<GetVFStatusResult> GetVFStatusAsync(Stream data)
        {
            GetVFStatusResult ret = new GetVFStatusResult();
            try
            {
                ret.Status = await VFHttpRequestManager.GetVFStatus();
                if (ret.Status == null)
                {
```

```
                    ret.Result = ResultCodes.VillageFlowConnectionError;
                }
                else
                {
                    ret.Status = VillageFlowStatus.GetStatus(ret.Status);
                }
                return ret;
            }
            catch
            {
                ret.Result = ResultCodes.Error;
                return ret;
            }
        }


    }
}
```

ANEXO 3

VFH

```
namespace Service.Request
{
    public class VFHttpRequestManager
    {
        private const String VillageFlowUri =
"http://192.168.1.117/ajax_control_VF.php?action=";
        public static async Task<string> GetVFStatus()
        {
            return await DoUriRequest($"{VillageFlowUri}show_status");
        }

        public static async Task<String> SwitchVFStatus()
        {
            return await DoUriRequest($"{VillageFlowUri}switch_status");
        }

        public static async Task<String> DoUriRequest(String uri)
        {
            try
            {
                HttpClient Client = new HttpClient();
                HttpRequestMessage Request = new
HttpRequestMessage(HttpMethod.Get, new Uri(uri));
                HttpResponseMessage Response = null;
                    Response = await Client.SendAsync(Request);
                return ReadContentStream(await
Response.Content.ReadAsStreamAsync());
            }
            catch (Exception ex)
            {
                return null;
            }
        }

        private static String ReadContentStream(Stream content)
```

```
        {
            using (var stream = new MemoryStream())
            {
                byte[] buffer = new byte[2048];
                int bytesRead;
                while ((bytesRead = content.Read(buffer, 0, buffer.Length))
> 0)
                {
                    stream.Write(buffer, 0, bytesRead);
                }
                byte[] result = stream.ToArray();
                return Encoding.UTF8.GetString(result, 0, result.Length);
            }
        }
    }
}
```

IMASTER

```
namespace Service
{
    // NOTA: puede usar el comando "Rename" del menú "Refactorizar" para
cambiar el nombre de interfaz "IMaster" en el código y en el archivo de
configuración a la vez.
    [ServiceContract]
    public interface IMaster
    {
        [OperationContract]
        [Description("Permite dar un saludo.")]
        [WebInvoke(BodyStyle = WebMessageBodyStyle.Bare, Method = "POST")]
        HelloWorldResult HelloWorld(Stream data);

        [OperationContract]
        [Description("Obtiene la información de login de usuario.")]
        [WebInvoke(BodyStyle = WebMessageBodyStyle.Bare, Method = "POST")]
        GetLoginResult GetLogin(Stream data);

        [OperationContract(AsyncPattern =true)]
        [Description("Inicia o detiene el servicio de VillageFlow.")]
        [WebInvoke(BodyStyle = WebMessageBodyStyle.Bare, Method = "POST")]
        Task<SwitchVFStatusResult> SwitchVFStatusAsync(Stream data);

        [OperationContract(AsyncPattern =true)]
        [Description("Obtiene el estado del servicio VillageFlow.")]
        [WebInvoke(BodyStyle = WebMessageBodyStyle.Bare, Method = "POST")]
        Task<GetVFStatusResult> GetVFStatusAsync(Stream data);
    }
}
```

Anexo 4

Programación general de la aplicación MasterControl

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
```

```
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:orientation="vertical"
        android:gravity="center"
        android:layout_weight="1">

            <EditText
                android:id="@+id/editTextUsername"
                android:hint="Usuario"
                android:layout_margin="20dp"
                android:inputType="number"
                android:layout_width="match_parent"
                android:layout_height="wrap_content" />
            <EditText
                android:id="@+id/editTextPassword"
                android:hint="Contraseña"
                android:layout_margin="20dp"
                android:inputType="textPassword"
                android:layout_width="match_parent"
                android:layout_height="wrap_content" />
        <CheckBox
            android:id="@+id/checkBoxShowPassword"
            android:layout_width="match_parent"
            android:layout_margin="20dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="16dp"
            android:text="Mostrar contraseña" />
    </LinearLayout>

    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:gravity="center"
        android:layout_weight="1">
        <android.support.design.widget.FloatingActionButton
        android:id="@+id/fabLogin"

            android:layout_width="80dp"
            android:layout_height="80dp"
            android:layout_margin="20dp"
        android:src="@drawable/ic_check_white_48px" />
        <android.support.design.widget.FloatingActionButton
            android:id="@+id/fabCancel"
            android:layout_width="80dp"
            android:layout_height="80dp"
            android:layout_margin="20dp"
            android:src="@drawable/ic_close_white_48px" />
    </LinearLayout>
</LinearLayout>
```

Programación para conexión al servicio WCF

```
public class MasterVillageRequestManager {
    private static final String BASE_URI =
"http://172.30.30.116:8080/Service/Master.svc/rest/";
    private static final String BASE_RESULT_ON_EXCEPTION =
"{\"Result\":\"ClientError\"}";
    public static final String
BASE_RESULT_ON_SERVER_CONNECTION_ERROR =
"{\"Result\":\"ServerConnectionError\"}";

    private static <T extends BaseResult> T getBaseResult(String
result, Class<T> type) {
        return JsonSerializerDeserializer.deserialize(result,
type);
    }

    public static GetLoginResult GetLogin(GetLoginParam param,
AsyncTask task) {
        try {
            HttpRequestManager httpRequestManager = new
HttpRequestManager();
            String response = httpRequestManager.doRequest(new
URL(Strings.format("{0}{1}", BASE_URI,
MasterVillageRequestActions.GET_LOGIN)),
HttpRequestManager.METHOD_POST, param, true, null, task);
            return
JsonSerializerDeserializer.deserialize(response,
GetLoginResult.class);
        } catch (Exception ex) {
            return getBaseResult(BASE_RESULT_ON_EXCEPTION,
GetLoginResult.class);
        }
    }

    public static GetVFStatusResult GetVFStatus(AsyncTask task) {
        try {
            HttpRequestManager httpRequestManager = new
HttpRequestManager();
            String response = httpRequestManager.doRequest(new
URL(Strings.format("{0}{1}", BASE_URI,
MasterVillageRequestActions.GET_VF_STATUS)),HttpRequestManager.MET
HOD_POST, null, true,2000,1000,null, task);
            return
JsonSerializerDeserializer.deserialize(response,
GetVFStatusResult.class);
        } catch (Exception ex) {
            return getBaseResult(BASE_RESULT_ON_EXCEPTION,
GetVFStatusResult.class);
        }
    }

    public static SwitchVFStatusResult SwitchVFStatus(AsyncTask
task) {
```

```
        try {
            HttpRequestManager httpRequestManager = new
HttpRequestManager();
            String response = httpRequestManager.doRequest(new
URL(Strings.format("{0}{1}", BASE_URI,
MasterVillageRequestActions.SWITCH_VF_STATUS)),HttpRequestManager.
METHOD_POST, null, true,2000,1000,null, task);
            return
JsonSerializerDeserializer.deserialize(response,
SwitchVFStatusResult.class);
        } catch (Exception ex) {
            return getBaseResult(BASE_RESULT_ON_EXCEPTION,
SwitchVFStatusResult.class);
        }
    }
```