UNIVERSIDAD INTERNACIONAL SEK

FACULTAD DE ARQUITECTURA E INGENIERÍAS

Trabajo de fin de Carrera Titulado:

"DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA LA GENERACIÓN DE CÓDIGO DE SOBREIMPOSICIONES DE TV DIGITAL PARA OROMARTV"

Realizado por:

Juan Fernando Baird

Director del proyecto:

Ing. Viviana Cajas, MBA

Como requisito para la obtención del título de:

INGENIERO DE SISTEMAS EN DISEÑO Y MULTIMEDIA

QUITO, MARZO DEL 2015

DECLARACIÓN JURAMENTADA

Yo, JUAN FERNANDO BAIRD BASANTES con cédula de identidad # 1722248885, declaro bajo juramento que el trabajo aquí desarrollado es de mi autoría, que no ha sido previamente presentado para ningún grado a calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración, cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la UNIVERSIDAD INTERNACIONAL SEK, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

Juan Fernando Baird Basantes

CC.:1722248885

DECLARATORIA

El presente trabajo de investigación titulado:

“DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA LA GENERACIÓN DE CÓDIGO DE SOBREIMPOSICIONES DE TV DIGITAL PARA OROMARTV”

Realizado por:

JUAN FERNANDO BAIRD BASANTES

Como requisito para la obtención del título de:

INGENIERIA DE SISTEMAS EN INFORMÁTICA Y REDES DE INFORMACIÓN

Ha sido dirigido por la docente:

ING. VIVIANA CAJAS, MBA

Quien considera que constituye un trabajo original de su autor.

Ing. Viviana Cajas, MBA

DIRECTORA

DECLARATORIA

Los Profesores informantes:

Ing. Verónica Rodríguez, MBA

Ing. Daniel Ripalda

Después de revisar el trabajo presentado,

lo han calificado como apto para su defensa oral ante

el tribunal examinador.

_____ _____

Ing. Verónica Rodríguez, MBA **Ing. Daniel Ripalda**

Quito, Marzo del 2015

Dedicatoria

Este trabajo se lo dedico a las personas que hacen de mi vida algo no solo agradable sino un sueño inmejorable, a mis padres que me apoyan y fomentan mis ideas sobre el futuro, a mi hermana que comparte juegos y aventuras épicas, a mis amigos que aunque plantillas sé que puedo contar con ellos cuando en verdad lo necesito, a Andrea Gonzales sin la cual esta tesis nunca hubiera sido completada y a quien amo con todo mi corazón. A mi abuelita Beatriz que me crió y a mi tío Galo que me recordaba todo el tiempo de hacer la tesis. Y a Pablo Andrés Grijalva (VenomEdge) amigo, mentor y ejemplo de ser humano, todos lo extrañaremos.

Agradecimientos

A mis padres que se sacrificaron inmensamente para que yo llegara aquí incluso si eso significaba perder cosas que ellos deseaban o necesitaban.

A Andrés Maldonado el mejor profesor de programación y sénior programmer que conozco sin el cual esta tesis no pasaba de una ventana sin botones.

A Andrea Gonzales por su constante apoyo y motivación durante estos años.

A los profesores de la carrera de sistemas los cuales me formaron como ingeniero y soportaron muchas veces mis ocurrencias o lógicas chuecas, gracias a todos.

A la Ing. Viviana Cajas que me ha tenido paciencia todo este tiempo y me guía para obtener mi título.

ÍNDICE GENERAL

ÍNDICE DE TABLAS Y FIGURAS

CAPÍTULO I

INTRODUCCIÓN

1.1. El Problema de Investigación

1.1.1. Planteamiento del Problema

1.1.1.1. Diagnóstico del problema

1.1. Antecedentes

OromarTV es un canal de Televisión que tiene como sitio de emisión y base de operaciones la ciudad de Manta, en la provincia de Manabí en Ecuador, es un canal que propone desde sus contenidos mostrar y valorar la cultura del pueblo Manabita que tiene como fundamento su forma de ver la vida, su pluralismo y su filosofía, utilizando contenidos de producción propia que promueven los valores éticos y morales de los Ecuatorianos, sin olvidar elementos de la construcción y el que hacer televisivo, como el entretenimiento y la diversidad en géneros y formatos audiovisuales. Es el primer canal de la provincia de Manabí que emite su señal en HD *(High Definition)* sin olvidar por ende, el constante progreso tecnológico que es fundamental e inherente a la producción de televisión.

En la actualidad el Ecuador se encuentra en una fuerte revolución tecnológica, entre los varios cambios que están siendo realizados se encuentra el "apagón analógico" (Eamon, 2008), el cual consiste en eliminar las señales analógicas para la transmisión de

radio y televisión, cambiando a una señal digital. Esto genera una gran oportunidad de innovación y renovación de la programación actual al igual que una serie de oportunidades de trabajo para ingenieros en sistemas.

"En el conjunto de América Latina, la Televisión Digital Terrestre(TDT) es –por el momento- un proyecto más que una realidad, ya que aún no se han establecido plazos de migración a la tecnología digital. Debemos mencionar la situación singular de los países más ricos del subcontinente americano, como Brasil, Argentina o México, donde se han iniciado emisiones experimentales". (Roel, 2006, pág. 4)

En el caso particular de OromarTV, la empresa se encuentra renovando infraestructura y se ve en la obligación de estar a la vanguardia del cambio, formar parte de los canales pioneros en el desarrollo de programación con aplicaciones interactivas que demuestren las ventajas y comodidades que ofrece la televisión digital al público. Debido a esto es necesario que empiece a producir programas y aplicaciones piloto que sean capaces de satisfacer las expectativas del público ecuatoriano.

Misión

Entregar una programación positiva, con espíritu innovador, joven y de calidad, que informe, eduque y entretenga con objetividad, responsabilidad social y valores, con personal calificado capaz de crear esos conceptos para la empresa y la comunidad.

Visión

Estar entre los cinco primeros canales del Ecuador en transmisión nacional e internacional en términos de rating y rentabilidad, empleando tecnología de punta, que nos proporcione una señal nítida y en HD, para lograr así la aceptación y reconocimiento de la audiencia ecuatoriana.

1.1.1.2 Pronóstico

En caso de no tomar las debidas precauciones e ignorar la inminente transformación que se aproxima a la televisión en el Ecuador, OromarTV pudo terminar convirtiéndose en un canal insostenible, que no representaría apropiadamente los intereses del público o de los posibles interesados en la compra de espacio publicitario.

1.1.1.3 Control del Pronóstico

Para evitar el pronóstico indicado, fue necesaria la inversión por parte de la empresa en el desarrollo de programación de calidad, en alta definición y con aplicaciones atractivas al público, en cada uno de sus programas, los cuales pueden ser actualizados o remplazados dependiendo del nivel de producción que cada uno representa para la empresa. Dada la dificultad que representa producir cada aplicación, o adaptarla a cada programa, en caso de ser una aplicación global, se convierte en una tarea demasiado costosa y demandante de tiempo. Por esa razón se propuso realizar una aplicación que permita la generación de código para realizar sobreimposiciones en una aplicación de Ginga. De ese modo no sería necesaria la presencia constante de expertos y el tiempo consumido al desarrollar la aplicación se reduce radicalmente.

Al generar programación lista para el cambio tecnológico, OromarTV será capaz de competir con cualquier otro canal, además de marcar tendencia al ser el pionero en el desarrollo de aplicaciones interactivas en los primeros años del apagón analógico.

1.1.2 Formulación del problema

¿El diseño e implementación de una aplicación que genere código de colocación de sobreimposiciones en TDT permitirá a la empresa OromarTV facilitar procesos, reducir costos de personal y acelerar el diseño de aplicaciones interactivas?

1.1.3. Sistematización del problema

- ¿Qué requerimientos necesita la aplicación para satisfacer las necesidades de la empresa?
- ¿Qué tipos de diseños de interfaz y código se debería analizar para la elaboración de la aplicación, será lo suficientemente amigable e intuitivo para el usuario?
- ¿Qué factores deberían considerarse para desarrollar una aplicación ligera y manejable con potencial expansivo a futuro?
- ¿Qué tan detallado debería ser el manual para que el usuario pueda usar la aplicación cómodamente?
- ¿Cómo organizar el código resultante para que el usuario pueda utilizarlo con facilidad?
- ¿Qué usuarios de la empresa OromarTV deberían probar la aplicación para un testeo apropiado de la misma?

1.1.4 Objetivo general

Diseñar, desarrollar e implementar una aplicación que permita la generación de código para la colocación de sobreimposiciones de TDT para Oromartv en la ciudad de Quito, facilitando procesos, reduciendo costos y acelerando el diseño de aplicaciones interactivas.

1.1.2. Objetivos específicos

- Realizar un levantamiento inicial de requerimientos con los ejecutivos de la empresa.

- Analizar diseños de interfaces y código para facilitar la elaboración de la aplicación, de manera que se genere una interfaz amigable e intuitiva para los usuarios.

- Desarrollar una aplicación ligera y manejable con potencial expansivo a futuro.

- Elaborar un manual de usuario con las funcionalidades del programa para una adecuada utilización de la aplicación.

- Generar un código ejecutable, asegurando la funcionalidad del código creado, simplificando así las tareas del usuario.

- Capacitar a los potenciales usuarios de la empresa OromarTV.

1.1.3. Justificaciones

Tomando en cuenta el riesgo que corría la empresa debido al inevitable e impostergable apagón analógico, el cual ya ha sido evidenciado en varios países como Japon, Brasil y Estados Unidos y ha sido anunciado por varios periódicos y blogs entre ellos EcuadorTel's Blog (Roel, 2006) y El Telegrafo (Ana, 2014). Por ello se propuso realizar una aplicación que permita la edición y generación de sobreimposiciones en una aplicación de Ginga. De ese modo no es necesaria la presencia constante de expertos y el tiempo consumido en el desarrollo se reduce radicalmente, aplicando un mínimo esfuerzo en la personalización de la misma.

Esto permitió a la empresa realizar pruebas e investigaciones con respecto al uso de dichas aplicaciones, y enfocarse en producir programación fácilmente adaptable a las mismas, o que en algún punto puedan ser enfocadas hacia la interactividad. El alcance del sistema está definido por los siguientes módulos:

- Creación de plantillas: esta es la función principal de la aplicación ya que se desea generar un código con la menor cantidad de pasos posibles por parte del usuario. Para lo cual la aplicación recibe toda la información de diseño a modo de

formulario y utilizando ciertos parámetros y estándares genera un código en GingaNCL ejecutable, listo para el uso y con la capacidad de discernir los elementos gráficos designados al mismo.

- Transformación de interfaz: a medida que el usuario usa la aplicación esta permite o bloquea posibilidades de ingreso de información, de forma que el usuario no tenga que llenar formularios para secciones que no desea programar, esta serie de funciones valida y evalúa las primeras decisiones del usuario y muestra los formularios pertinentes para una generación óptima del código deseado y evita en gran parte la generación de código basura en la aplicación resultante.

- Recuperación y configuraciones de interfaz: una vez generado el código y cerrada la aplicación toda la memoria de ésta se resetea, estas funciones permiten recuperar una aplicación antigua y configuran la interfaz para mostrar al usuario exactamente lo que hizo y los valores que ingresó, de modo que pueda personalizarlos sin tener que empezar nuevamente.

1.2. Marco Teórico

1.2.1. Estado actual del conocimiento sobre el tema

De momento el desarrollo de aplicaciones de Ginga en el Ecuador es muy limitado y se encuentra únicamente en proyectos investigativos, como por ejemplo “Sistema de prevención de catástrofes utilizando el sistema ISDB-T” y “Suite de pruebas para decodificadores ISDB-T” (ESPE, 2012) ambos proyectos en desarrollo por el grupo de tv digital de la ESPE. Los cuales pretenden prepararse para la llegada del apagón analógico, todos los involucrados en el desarrollo tienen un nivel de programación medio o avanzado, por lo cual resulta más fácil para ellos comprender y ejecutar el código necesario, el problema se da desde un punto comercial ya que de momento toma mucho tiempo, esfuerzo y conocimiento para realizar los cambios mínimos en una aplicación.

A nivel mundial varios países han comenzado a desarrollar estándares y metodologías que sirvan como norma para la transición de señales digitales, muchos de los cuales ya están en uso en su país de origen como es el caso de Japón, Estados unidos, España, México. Entre los estándares más utilizados tenemos el estándar Americano ATSC (Advance Televicion System Committie), el estándar Europeo DVB-T (Digital Video Broadcasting Terrestrial), el estándar Chino DTMB (Digital Terrestrial Multimedia Broadcasting) y el estándar Japonés ISDB-T (Integrated Service Digital Broadcasting Terrestrial) (Gamarro, 2013) al mismo que se le implementaron innovaciones realizadas en Brasil. Luego de varias deliberaciones y estudios que consideraban los aspectos técnicos, socioeconómicos y de cooperación internacional de cada uno de los estándares, el Ecuador adoptó el estándar Japonés con las modificaciones brasileñas.

Varias empresas están enfocadas en la preparación para el cambio que se avecina, algunas emisoras como EcuadorTV y Gama ya transmiten en HD (alta definición) y tienen varios canales digitales, esto marca la pauta y permite evaluar el estado de competencia en el cual se encuentran todas las empresas de televisión.

Fue necesaria una mejor comprensión de la herramienta Ginga y sus posibles alteraciones para poder desarrollar una plataforma no solo eficiente sino estable y capaz de durar uno a dos años sin necesidad de un parche o actualización, por ese motivo este proyecto sirve como pilar inicial para el desarrollo futura de una plataforma más completa y accesible.

Tomando en consideración que el proyecto de tesis es un proyecto de desarrollo de software fue necesario analizar y estudiar todas las metodologías disponibles y evaluar así que metodología se adapta de mejor manera a las necesidades del cliente y del programador. Como se menciona a continuación hay varias maneras de obtener los mismos resultados en desarrollo de software sin embargo la experiencia y estudio de cada programador lo guía a cierto modelo en particular.

Metodologías disponibles

Modelo en cascada

El modelo de la cascada, a veces llamado ciclo de vida clásico, sugiere un enfoque sistemático y secuencial para el desarrollo del software, que comienza con la especificación de los requerimientos por parte del cliente y avanza a través de planeación, modelado, construcción y despliegue, para concluir con el apoyo del software terminado. (Pressman, 2010, pág. 33)

Los principios básicos del modelo de cascada son los siguientes:

- El proyecto está dividido en fases secuenciales, con cierta superposición y splashback aceptable entre fases.

- Se hace hincapié en la planificación, los horarios, fechas, presupuestos y ejecución de todo un sistema de una sola vez.

- Un estricto control se mantiene durante la vida del proyecto a través de la utilización de una amplia documentación escrita, así como a través de comentarios y aprobación.

Incremental

El modelo incremental combina elementos de los flujos de proceso lineal y paralelo, el modelo incremental aplica secuencias lineales en forma escalonada a medida que avanza el calendario de actividades. Cada secuencia lineal produce “incrementos” de software susceptibles de entregarse de manera parecida a los incrementos producidos en un flujo de proceso evolutivo. (Pressman, 2010, pág. 35)

Al ser uno de los modelos más adoptados en la actualidad por empresas pequeñas y desarrolladores freelancers se investigó más a fondo y posee dos métodos comúnmente aceptados como los mejores SRUM y Extreme Programming(XP).

Los principios básicos son:

- Una serie de mini-cascadas se llevan a cabo en todas las fases de modelo de desarrollo para completar una pequeña parte del sistema, antes de proceder a la próxima etapa.

- El concepto inicial de software, análisis de las necesidades, y el diseño de la arquitectura y colectiva básicas se definen utilizando el enfoque de cascada, seguida por un iterativo de prototipos, que culmina en la instalación del prototipo final.

Espiral

Propuesto en primer lugar por Barry Boehm, el modelo espiral es un modelo evolutivo del proceso del software y se acopla con la naturaleza iterativa de hacer prototipos con los aspectos controlados y sistémicos del modelo de cascada. Tiene el potencial para hacer un desarrollo rápido de versiones cada vez más completas. (Pressman, 2010, pág. 39)

Los principios básicos son:

- La atención se centra en la evaluación y reducción del riesgo del proyecto, separando el proyecto en segmentos más pequeños y proporcionar mayor facilidad de cambio durante el proceso de desarrollo.

- Cada viaje alrededor de la espiral atraviesa cuatro cuadrantes básicos: determinar objetivos, alternativas, y desencadenantes de la iteración; Evaluar alternativas; Identificar y resolver los riesgos; desarrollar y verificar los resultados de la iteración, y plan de la próxima iteración.

- Cada ciclo comienza con la identificación de los interesados y sus condiciones de ganancia, y termina con la revisión y examinación.

Metodologías ágiles de desarrollo

Se refieren a métodos de ingeniería del software basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan mediante la colaboración de grupos auto-organizados y multidisciplinarios. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en lapsos cortos. (Chin, 2004)

SCRUM

Desarrollada por Ken Schwaber, Jeff Sutherland y Mike Beedle. Define un marco para la gestión de proyectos, que se ha utilizado con éxito durante los últimos 10 años. Esta especialmente indicada para proyectos con rápido cambio de requisitos. Sus principales características se pueden resumir en dos. El desarrollo de software se realiza mediante iteraciones, denominadas sprints, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración. (Canos, 2003, pág. 7)

Programación Extrema (XP)

Es una metodología ágil centrada en el potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo,

preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y capacidad para enfrentar los cambios. (Canos, 2003, pág. 3)

Lenguajes de programación

Java

Se planea el uso de este lenguaje de programación ya que está ampliamente disponible para el aprendizaje y posee varios programas que permiten su edición y compilación. Además tomando en consideración el tipo de plataforma intermediaria que se desea realizar, usar un lenguaje sencillo pero eficiente como lo es Java facilita al momento de la programación y de la migración a otras plataformas si fuera necesario.

Al ser un lenguaje común y muy utilizado, también permite dejar una ventana abierta para el momento en el cual se desee alterar la aplicación ya sea para integrarla con nuevos módulos o simplemente analizar el código y reciclarlo para diseñar una aplicación mucho más completa, lo cual sería una de las metas a futuro que se aspira al diseñar esta aplicación inicial. (Segovia, 2014)

1.2.2. Adopción de una perspectiva teórica

1.2.2.1. Selección de metodología

Dada la naturaleza de este proyecto y la cantidad de personas involucradas en el mismo se determinó la utilización de metodologías ágiles de diseño de software. Para llegar a esta conclusión se consideró: la originalidad del proyecto y su proyección

innovadora indicando una inclinación al cambio, desde un inicio el proyecto de tesis ha estado sujeto a constantes cambios y reducciones e incrementos al alcance del mismo.

Aunque todos los métodos ágiles están basados alrededor de la noción de desarrollo y entrega incremental, cada uno propone distintos procesos para conseguirlo. Sin embargo, comparten un conjunto de principios basados en la agilidad y por lo tanto tienen mucho en común, tomando eso en consideración, se decidió concentrarse únicamente en los métodos más usados y enfocarnos únicamente entre Scrum y XP(extreme programming) como referencia de los valores adoptados por todas las metodologías agiles se puede notar:

- **Involucramiento del cliente**: los clientes deben estar estrechamente relacionados con el proyecto, siendo su rol el de proveer y priorizar cualquier requerimiento esencial en el mismo y evaluar los resultados del mismo. (Letelier, 1990)
- **Entrega incremental:** el software es desarrollado incrementalmente, con el cliente indicando nuevas necesidades o funciones necesarias en el mismo. (Letelier, 1990)
- **Personas no procesos:** las habilidades del equipo de desarrollo deben ser reconocidas y explotadas, permitiéndoles trabajar a su manera sin procesos represivos. (Letelier, 1990)
- **Aceptar el cambio:** esperar que los requerimientos cambien de improvisto y diseñar el sistema para soportar dichos cambios. (Letelier, 1990)
- **Mantener simplicidad:** enfocarse en la simplicidad ya sea en el sistema siendo desarrollado y en el proceso de desarrollo, siempre que sea posible trabajar para activamente eliminar la complejidad del sistema. (Letelier, 1990)

1.2.2.2. Selección de lenguajes e interfaces

Para la programación del proyecto de tesis se debe considerar las herramientas y lenguajes que se adapten mejor a las necesidades del cliente, en este caso se tomó como

lenguaje de programación a Java, por su flexibilidad y facilidad de aprendizaje, mientras que se decidió utilizar GingaNCL como lenguaje de salida para el código resultante. Para esto se consideró que al ser orientado a la televisión y a usuarios de bajo nivel de programación, el código generado, debería ser de la mayor simplicidad posible, de modo que si alguno desease estudiarlo, no tenga problemas en la comprensión y estudio del mismo.

Al haber elegido los lenguajes previamente mencionados, se optó por el uso de dos interfaces de programación altamente aceptados y utilizados por la comunidad, los cuales son: Eclipse para la realización de pruebas y estudios de código GingaNCL y NetBeans como plataforma de desarrollo para Java, debido a su facilidad de uso y estudio. Asegurando así que la aplicación pueda ser usada para el aprendizaje de estudiantes que deseen comprender su funcionamiento, o de profesionales que deseen modificarlo.

Tabla #1: Tabla Comparativa de Lenguajes

Elaborado por: Samy Manosalvas marzo 24, 2014

LENGUAJE DE PROGRAMACIÓN	VENTAJAS	DESVENTAJAS	IDE DE DESARROLLO	SISTEMAS OPERATIVOS
C++	Es potente en cuanto a lo que se refiere a creación de sistemas complejos un lenguaje muy robusto	• No es atractivo visualmente • No soporta para creación de páginas web	• DEV C++ • BORLAND C • TURBO C	Sirve para todos los sistemas operativos pero cada uno con su respectiva versión para dicho sistema
VISUAL BASIC	• Posee una curva de aprendizaje muy rápida. • Integra el diseño e implementación de formularios de Windows. • Permite usar con facilidad la plataforma de los sistemas Windows, dado que tiene acceso prácticamente total al api de Windows, incluidas librerías actuales. • Es uno de los lenguajes de uso más extendido, por lo que resulta fácil encontrar información, documentación y	Las críticas hechas en las ediciones de visual Basic anteriores a vb.net son variadas, se citan entre ellas: • Problema de versionado asociado con varias librerías DLL, conocido como DLL HELL. • Pobre soporte para programación orientada a objetos • Incapacidad para crear aplicaciones multihilo,	Entorno de desarrollo Existe un único entorno de desarrollo para visual Basic, desarrollado por Microsoft: Microsoft visual Basic x.0 para versiones desde la 1.0 hasta la 6.0, (con las diferencias entre las versiones desde la 1.0 (MS-DOS/Windows 3.1) hasta la 3.0 (16 bits, Windows 3.1/95) y las de la 4.0 (16/32 bits, Windows 3.1/95/nt) hasta la 6.0 (32 bits, Windows 9x/me/nt/2000/XP/2003 server).	Sirve para hacer aplicaciones de escritorio

LENGUAJE DE PROGRAMACIÓN	VENTAJAS	DESVENTAJAS	IDE DE DESARROLLO	SISTEMAS OPERATIVOS
	fuentes para los proyectos.	sin tener que recurrir a llamadas del api de Windows.		
C#	• Declaraciones en el espacio de nombres: al empezar a programar algo, se puede definir una o más clases dentro de un mismo espacio de nombres. • Tipos de datos: en c# existe un rango más amplio y definido de tipos de datos que los que se encuentran en c, c++ o java. • Atributos: cada miembro de una clase tiene un atributo de acceso del tipo público, protegido, interno, interno protegido y privado.	Se tiene que conseguir una versión reciente de visual studio .net, por otra parte se tiene que tener algunos requerimientos mínimos del sistema para poder trabajar adecuadamente tales como contar con Windows nt 4 o superior, tener alrededor de 4 gigas de espacio libre para la pura instalación, etc.	Visual Studio .NET	La plataforma .Net Sirve para hacer aplicaciones de escritorio, aplicaciones web y móviles. Sistema operativo Windows
JAVA	• Se pueden realizar distintos aplicativos, como son applets, que son aplicaciones especiales,	Esperar la actualización siguiente para que sea más rápido.	• Eclipse • Netbeans	Sirve para todos los sistemas operativos y si no es la versión

LENGUAJE DE PROGRAMACIÓN	VENTAJAS	DESVENTAJAS	IDE DE DESARROLLO	SISTEMAS OPERATIVOS
	que se ejecutan dentro de un navegador al ser cargada una página HTML en un servidor web, por lo general los applets (Ya son historia) son programas pequeños y de propósitos específicos. • Puede desarrollar aplicaciones de escritorio que se ejecutan en forma independiente, es decir con la programación java, se pueden realizar aplicaciones como un procesador de palabras, una hoja que sirva para cálculos, una aplicación gráfica, etc. • Se puede realizar soluciones empresariales en un entorno web • Soporta el desarrollo de aplicaciones móviles		• Power buider	adecuada para dicho sistema, la misma aplicación java se encarga de descargas o actualizar versión para un excelente desempeño en el pc. Algunos de los sistemas operativos más destacados en los que funciona la aplicación: **Unix, Linux, Solaris,** **Windows, mac.**

LENGUAJE DE PROGRAMACIÓN	VENTAJAS	DESVENTAJAS	IDE DE DESARROLLO	SISTEMAS OPERATIVOS
PHP	Es un lenguaje multiplataforma. Orientado al desarrollo de aplicaciones web dinámicas con acceso a información almacenada en una base de datos.	Como es un lenguaje que se interpreta en ejecución, para ciertos usos puede resultar un inconveniente que el código fuente no pueda ser ocultado. La ofuscación es una técnica que puede dificultar la lectura del código pero no la impide y, en ciertos casos, representa un costo en tiempos de ejecución.	• Eclipse • Netbeans • DreamWeaver	Se usa principalmente para la interpretación del lado del servidor, páginas web y CMS Se usa en todos los sistemas operativos

1.2.3. Marco Conceptual

Ginga

"Ginga es el nombre del middleware abierto del Sistema Brasileño de TV Digital (SBTVD). Ginga es una capa de software intermedio (middleware), entre el hardware/Sistema Operativo y las aplicaciones, que ofrece una serie de facilidades para el desenvolvimiento de contenidos y aplicaciones para TV Digital, permitiendo la posibilidad de poder presentar los contenidos en distintos receptores independientemente de la plataforma de hardware del fabricante y el tipo de receptor.

Las aplicaciones ejecutadas sobre Ginga son clasificadas en dos categorías, dependiendo de la forma en la que son escritas. Las Aplicaciones de Procedimiento son escritas usando el lenguaje Java y las Aplicaciones Declarativas son escritas usando el lenguaje NCL.". (Espinoza, 2010)

Eclipse

"Es un programa que cuenta con un conjunto de herramientas para ser usado por un programador, es de código abierto multiplataforma y permite desarrollar aplicaciones de cliente enriquecido. La característica principal de eclipse, es que es extenso. La forma en que los plugins interactúan es mediante interfaces o puntos de extensión; así, los nuevos aportes se integran fácilmente.

El programa es neutro y se adapta a cualquier tipo de lenguaje, a pesar de que su mayor parte está escrita en java (excepto el núcleo). Su uso más popular es como IDE para java. Los siguientes componentes constituyen la plataforma de cliente enriquecido". (Molina, 2014)

- Plataforma principal: Inicio de Eclipse, ejecución de los plugins

- Plataforma para integrar distribuciones: OSGI (está definida con una serie de APIS básicas para el desarrollo de servicios).

- El Standard Widget Toolkit (SWT): es un conjunto de componentes para construir interfaces gráficas en Java.

- JFace : se encarga del manejo de archivos, manejo de texto, editores de texto.

- El Workbench de Eclipse: se encarga de las vistas, los editores, las perspectivas y los asistentes.

NetBeans

"NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extender el NetBeans IDE. NetBeans IDE es un producto libre y gratuito sin restricciones de uso.

También está disponible NetBeans Platform; una base modular y extensible usada como estructura de integración para crear grandes aplicaciones de escritorio. Empresas independientes asociadas, especializadas en desarrollo de software, proporcionan extensiones adicionales que se integran fácilmente en la plataforma y que pueden también utilizarse para desarrollar sus propias herramientas y soluciones." (Oracle, 2013)

GingaNCL

"El lenguaje NCL ha sido desarrollado utilizando una estructura modular, siguiendo los principios adoptados por el W3C -(WWW) World-Wide Web Consortium. Es un lenguaje declarativo que provee facilidades para especificar aspectos de interactividad, sincronismos espacial/temporal entre objetos de multimedia, adaptabilidad y soporte para múltiples dispositivos, es decir, construir aplicaciones.

La Programación Declarativa, es un paradigma de programación que está basado en el desarrollo de programas especificando o "declarando" un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución. La solución es obtenida mediante mecanismos internos de control, sin especificar exactamente cómo encontrarla." (Bachetta, 2010)

Sobreimposición

La sobreimposición es una pieza gráfica animada o estática, en donde se puede observar datos informativos como lugar de la noticia, nombres de los presentadores, etc. En el caso de TDT se considera sobreimposición a todo asset que se agregue encima del video original que está siendo transmitido, de modo que cada menú, botón, o video que se incluya es una sobreimposición al video original (Castro, 2010)

Televisión Digital Terrestre (TDT)

Esta tecnología permite la transmisión de contenidos a través de una red de repetidores terrestres, la codificación digital permite trasmitir varios canales en el espacio que antes lo utilizaba un canal analógico, por medio de la tecnología Multicasting. Esta permite comprimir la señal, lo cual nos da la posibilidad de transmitir varias señales de distintas calidades en el mismo espacio que ocupaba una sola señal analógica de calidad normal. (Bachetta, 2010)

Assets

Los assets pueden ser traducidos como elementos que serán introducidos al videojuego o aplicación. Estos elementos incluyen Modelos 3D, personajes, texturas, materiales, animaciones, scripts, sonidos, y algunos elementos específicos de cada motor. Cada motor trabaja de una manera distinta a otros lo cual puede aceptar "Assets" que otros motores no pueden manejar, sin embargo los ejemplos mencionados antes, son elementos que todos los motores de hoy en día usan. (Ward, 2008)

SetTopBox

Un set-top box o decodificador es un dispositivo que permite a un aparato de televisión para convertirse en una interfaz de usuario a Internet y también permite a un aparato de televisión para recibir y decodificar la televisión digital (TDT) emisiones. TDT set-top boxes a veces se llaman receptores. Un decodificador es necesario televidentes que deseen utilizar sus televisores analógicos actuales para recibir las emisiones digitales. (Rouse, 2006)

CAPÍTULO II

MÉTODO

2.1. Análisis

2.1.1. Levantamiento de Requerimientos

El proyecto adoptó un estudio exploratorio, que permitió demostrar los posibles usos y ventajas que se dan mediante la implementación de la aplicación desarrollada, permitiendo así abrir campo para futuros estudios relacionados, y facilitar el aprendizaje del desarrollo de aplicaciones para TDT(Televisión digital terrestre).

2.1.1.1. Levantamiento de Requerimientos

Esta investigación se basó en un proyecto de desarrollo, que busca satisfacer las necesidades específicas de la empresa OromarTV, entregando como producto una aplicación que permite la generación de un código ejecutable Ginga y la edición del mismo a modo de plantillas intercambiables.

El levantamiento de requerimientos se efectuó con entrevistas y discusiones con el personal administrativo de OroMarTv, en los cuales se aconsejó y guió al cliente para lograr recolectar los requerimientos necesarios para poder realizar la aplicación.

Los requerimientos de la aplicación fueron:

- **Facilidad de aprendizaje:** considerando que la mayoría de empleados de la empresa son diseñadores y personal de comunicación, era necesario que el programa no requiera mucho conocimiento previo de programación en Ginga.

- **Entorno Amigable e intuitivo:** una queja constante de varios usuarios, radica en la cantidad excesiva de botones y herramientas las cuales confunden y molestan al personal al tratarse de un tema del cual desconocen como lo es la programación, la aplicación debía ser lo suficientemente simple como para evitar dichas molestias.

- **Orientado a televisión:** el programa debía priorizar la experiencia televisiva, por lo cual se eliminaron posibles funciones que si bien más interactivas, no compaginan con los deseos de la empresa que son mejorar la experiencia televisiva en la programación actual.

- **Capacidad de grabado y carga de proyectos viejos:** como valor agregado se buscó la forma de permitirle al usuario no solo salvar su proyecto, sino la capacidad de tomar partes del mismo para crear otro.

- **Capacidad de expansión:** el programa tiene la arquitectura necesaria para fácilmente aumentar más módulos en caso de que la empresa lo necesite, o decidan implementar nuevas formas de promoción o cambien el tipo de programación.

2.1.2. Descripción general del Sistema propuesto

2.1.2.1 Identificación de Actores

Tabla #2: Identificación de Actores - Usuario

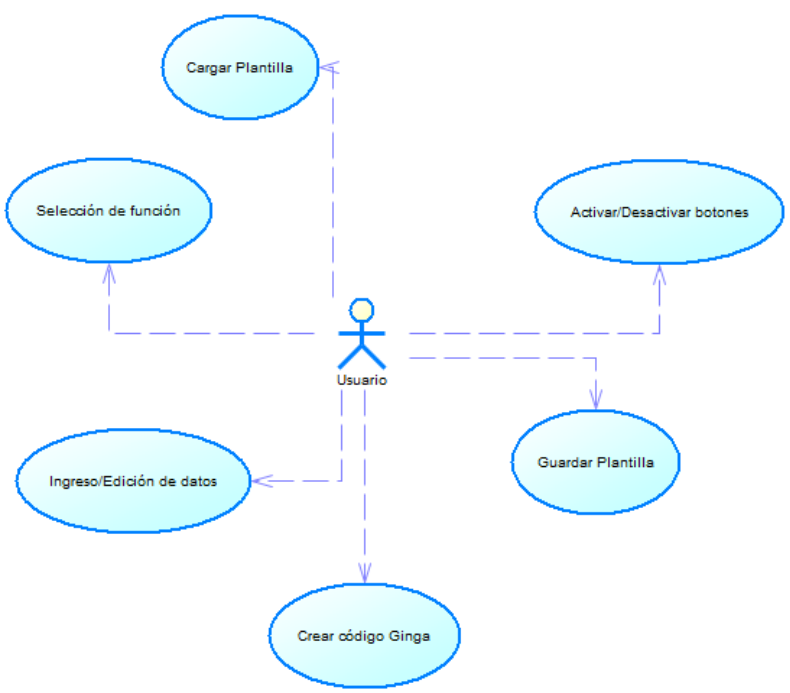
Elaborado por: Juan Fernando Baird

ACTORES	FUNCIÓN
 **Usuario**	• Iniciar programa • Cargar plantilla antigua/ Configurar plantilla nueva • Activar/Desactivar botones deseados • Selección de funciones deseadas • Ingreso de datos • Guardado de plantilla o plantillas • Generación de código Ginga

2.1.2.2 Diagrama de Casos de Uso por Actores

Figura #1 Actor Usuario

Elaborado por: Juan Fernando Baird



2.1.2.3 Casos de Uso

Tabla # 3. Casos de uso # 1: Creación de nuevo proyecto Ginga

Elaborado por: Juan Fernando Baird

Caso de uso:	Creación de un nuevo proyecto/plantilla.
Actor principal:	Usuario
Objetivo en contexto:	Crear un proyecto completamente nuevo para Ginga.
Disparador:	Inicio del programa
Escenario:	1. Usuario: inicia Mushu. 2. Sistema: despliega interfaz de usuario. 3. Usuario: ingresa un nombre clave para su nueva aplicación. 4. Usuario: selecciona el botón que desea activar. 5. Sistema: habilita/deshabilita los botones según lo indicado. 6. Usuario: selecciona la función deseada en el botón. 7. Sistema: prepara la interfaz acorde con la selección. 8. Usuario: ingresa los datos solicitados en el formulario de la función. 9. Sistema: procesa la información inicial y de ser necesario despliega una segunda etapa de interacción o se mantiene en standby. 10. Usuario verifica la información y procede a ejecutar la función Crear NCL. 11. Sistema: ingresa la información a la función y genera el código de acuerdo a los estándares establecidos.
Excepciones	En caso de un error de ingreso, el programa desplegará una ventana de error solicitando que revise la información ingresada ya que uno de los campos esta incorrecto.
Prioridad:	Esencial
Cuando estará disponible:	24 horas
Frecuencia de uso:	Bajo demanda
Canal para el actor:	Interfaz de usuario
Actores secundarios:	Ninguno

Tabla # 4. Casos de uso # 2: Guardar plantilla

Elaborado por: Juan Fernando Baird

Caso de uso:	Guardar plantilla
Actor principal:	Usuario
Objetivo en contexto:	Guardar un proyecto nuevo o modificado para ser usado a futuro.
Disparador:	Llamado a la función guardar o guardar All
Escenario:	1. Usuario: ingresa los datos solicitados en el formulario de la función. 2. Sistema: procesa la información inicial y de ser necesario despliega una segunda etapa de interacción o se mantiene en standby, habilitando el botón guardar. 3. Usuario verifica la información y procede a ejecutar la función guardar o guardar All. 4. Sistema: ingresa la información a la función y crea una carpeta en el escritorio la cual contendrá un archivo cifrado por cada botón guardado.
Excepciones	• La función guardar no se encuentra disponible hasta que el usuario ingrese toda la información necesaria en cualquiera de las funciones disponibles. • La función guardar All se encuentra siempre disponible, pero no creara ningún archivo si no existe por lo menos un botón guardar activado.
Prioridad:	Secundaria pero Altamente recomendada.
Cuando estará disponible:	24 horas, siempre que se cumpla la condición.
Frecuencia de uso:	Bajo demanda
Canal para el actor:	Interfaz de usuario
Actores secundarios:	Ninguno
Canales para los actores secundarios:	Ninguno

Tabla # 5. Casos de uso # 3: Cargar Plantilla

Elaborado por: Juan Fernando Baird

Caso de uso:	Cargar plantilla
Actor principal:	Usuario
Objetivo en contexto:	Edición/ Uso de una archivo antiguo para crear una nueva aplicación Ginga.
Disparador:	Llamado a la función abrir
Escenario:	1. Usuario: presiona uno de los botones abrir, durante cualquier etapa del desarrollo. 2. Sistema: despliega una ventana solicitando la selección de un archivo cifrado. 3. Usuario: selecciona el archivo deseado. 4. Sistema: ejecuta la función y adapta la interfaz eliminando cualquier progreso y reemplazándolo por la plantilla editable deseada.
Excepciones	En caso de un error de ingreso, el programa desplegará un pequeño mensaje de error indicando que el archivo no es el deseado.
Prioridad:	Esencial
Cuando estará disponible:	24 horas.
Frecuencia de uso:	Bajo demanda
Canal para el actor:	Interfaz de usuario
Actores secundarios:	Ninguno
Canales para los actores secundarios:	Ninguno

Tabla # 7. Casos de uso # 4: Editar Plantilla

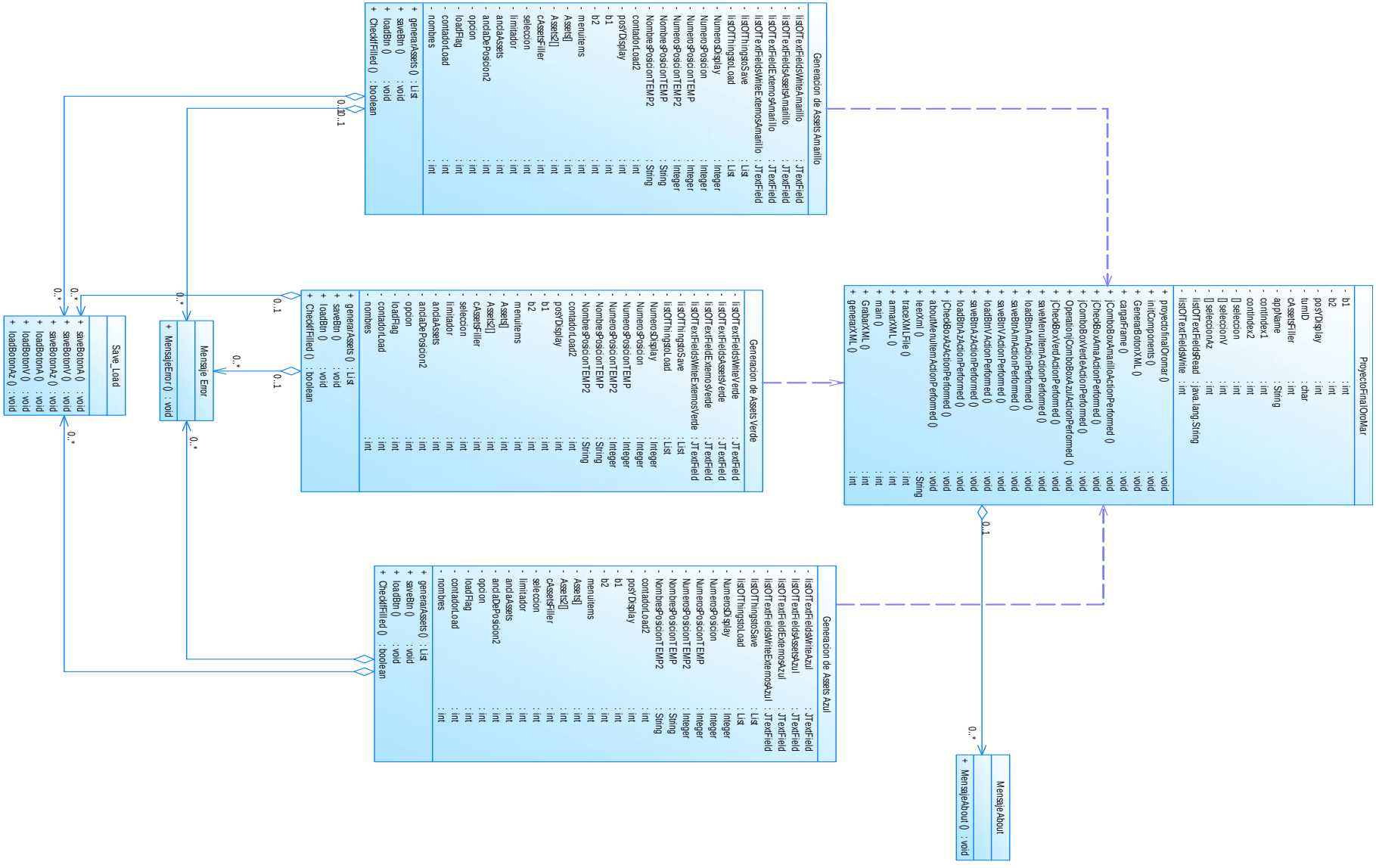
Elaborado por: Juan Fernando Baird

Caso de uso:	Editar plantilla
Actor principal:	Usuario
Objetivo en contexto:	Edición/ Uso de una archivo antiguo para crear una nueva aplicación Ginga.
Disparador:	Llamado a la función abrir
Escenario:	1. Sistema: ejecuta la función abrir y adapta la interfaz eliminando. cualquier progreso y reemplazándolo por la plantilla editable deseada. 2. Usuario: cambia la información deseada y procede a grabar el archivo bajo el mismo nombre del original. 3. Sistema: sobrescribe el archivo antiguo con la información adecuada.
Excepciones	En caso de ingresar un nombre distinto, el programa simplemente creara un archivo cifrado nuevo, en lugar de reemplazar el original.
Prioridad:	Esencial
Cuando estará disponible:	24 horas.
Frecuencia de uso:	Bajo demanda
Canal para el actor:	Interfaz de usuario
Actores secundarios:	Ninguno
Canales para los actores secundarios:	Ninguno

2.1.3. Diagramas generales

2.1.3.1. Diagramas de clase (Análisis).

Figura #2 Diagrama de clase
Elaborado por: Juan Fernando Baird


ProyectoFinalOroMar
- b1 : int
- b2 : int
- posYDisplay : int
- turnID : char
- cAssetsFiller : int
- appName : String
- contIndex1 : int
- contIndex2 : int
- [] seleccion : int
- [] seleccionV : int
- [] seleccionAz : int
- listOfTextFieldsRead : java.lang.String
- listOfTextFieldsWrite : int
+ proyectoFinalOromar () : void
+ initComponents () : void
+ GenerarBotonXML () : void
+ cargarFrame () : void
+ jComboBoxAmarilloActionPerformed () : void
+ jCheckBoxAmaActionPerformed () : void
+ jComboBoxVerdeActionPerformed () : void
+ OperationjComboBoxAzulActionPerformed () : void
+ jCheckBoxVerdActionPerformed () : void
+ saveMenuItemActionPerformed () : void
+ loadBtnAmActionPerformed () : void
+ saveBtnAmActionPerformed () : void
+ saveBtnVActionPerformed () : void
+ loadBtnVActionPerformed () : void
+ saveBtnAzActionPerformed () : void
+ loadBtnAzActionPerformed () : void
+ jCheckBoxAzActionPerformed () : void
+ aboutMenuItemActionPerformed () : void
+ leerXml () : String
+ traceXMLFile () : int
+ armarXML () : int
+ main () : int
+ GrabarXML () : int
+ generarXML () : int
Generacion de Assets Amarillo
- listOfTextFieldsWriteAmarillo : JTextField
- listOfTextFieldsAssetsAmarillo : JTextField
- listOfTextFieldsExternosAmarillo : JTextField
- listOfTextFieldsWriteExternosAmarillo : JTextField
- listOfThingstoSave : List
- listOfThingstoLoad : List
- NumerosDisplay : Integer
- NumerosPosicion : Integer
- NumerosPosicionTEMP : Integer
- NumerosPosicionTEMP2 : Integer
- NombresPosicionTEMP : String
- NombresPosicionTEMP2 : String
- contadorLoad2 : int
- posYDisplay : int
- b1 : int
- b2 : int
- menuItems : int
- Assets[] : int
- Assets2[] : int
- cAssetsFiller : int
- seleccion : int
- limitador : int
- andaAssets : int
- andaDePosicion2 : int
- opcion : int
- loadFlag : int
- contadorLoad : int
- nombres : int
+ generarAssets () : List
+ saveBtn () : void
+ loadBtn () : void
+ CheckIfFilled () : boolean
Generacion de Assets Verde
- listOfTextFieldsWriteVerde : JTextField
- listOfTextFieldsAssetsVerde : JTextField
- listOfTextFieldsExternosVerde : JTextField
- listOfTextFieldsWriteExternosVerde : JTextField
- listOfThingstoSave : List
- listOfThingstoLoad : List
- NumerosDisplay : Integer
- NumerosPosicion : Integer
- NumerosPosicionTEMP : Integer
- NumerosPosicionTEMP2 : Integer
- NombresPosicionTEMP : String
- NombresPosicionTEMP2 : String
- contadorLoad2 : int
- posYDisplay : int
- b1 : int
- b2 : int
- menuItems : int
- Assets[] : int
- Assets2[] : int
- cAssetsFiller : int
- seleccion : int
- limitador : int
- andaAssets : int
- andaDePosicion2 : int
- opcion : int
- loadFlag : int
- contadorLoad : int
- nombres : int
+ generarAssets () : List
+ saveBtn () : void
+ loadBtn () : void
+ CheckIfFilled () : boolean
Generacion de Assets Azul
- listOfTextFieldsWriteAzul : JTextField
- listOfTextFieldsAssetsAzul : JTextField
- listOfTextFieldsExternosAzul : JTextField
- listOfTextFieldsWriteExternosAzul : JTextField
- listOfThingstoSave : List
- listOfThingstoLoad : List
- NumerosDisplay : Integer
- NumerosPosicion : Integer
- NumerosPosicionTEMP : Integer
- NumerosPosicionTEMP2 : Integer
- NombresPosicionTEMP : String
- NombresPosicionTEMP2 : String
- contadorLoad2 : int
- posYDisplay : int
- b1 : int
- b2 : int
- menuItems : int
- Assets[] : int
- Assets2[] : int
- cAssetsFiller : int
- seleccion : int
- limitador : int
- andaAssets : int
- andaDePosicion2 : int
- opcion : int
- loadFlag : int
- contadorLoad : int
- nombres : int
+ generarAssets () : List
+ saveBtn () : void
+ loadBtn () : void
+ CheckIfFilled () : boolean
Save_Load
+ saveBotonA () : void
+ saveBotonV () : void
+ saveBotonAz () : void
+ loadBotonA () : void
+ loadBotonV () : void
+ loadBotonAz () : void
Mensaje Error
+ MensajeError () : void
MensajeAbout
+ MensajeAbout () : void
0..1
0..*

2.1.3.2. Diagramas de secuencia

Figura #3 Diagrama de secuencia: Crear ejecutable

Elaborado por: Juan Fernando Baird

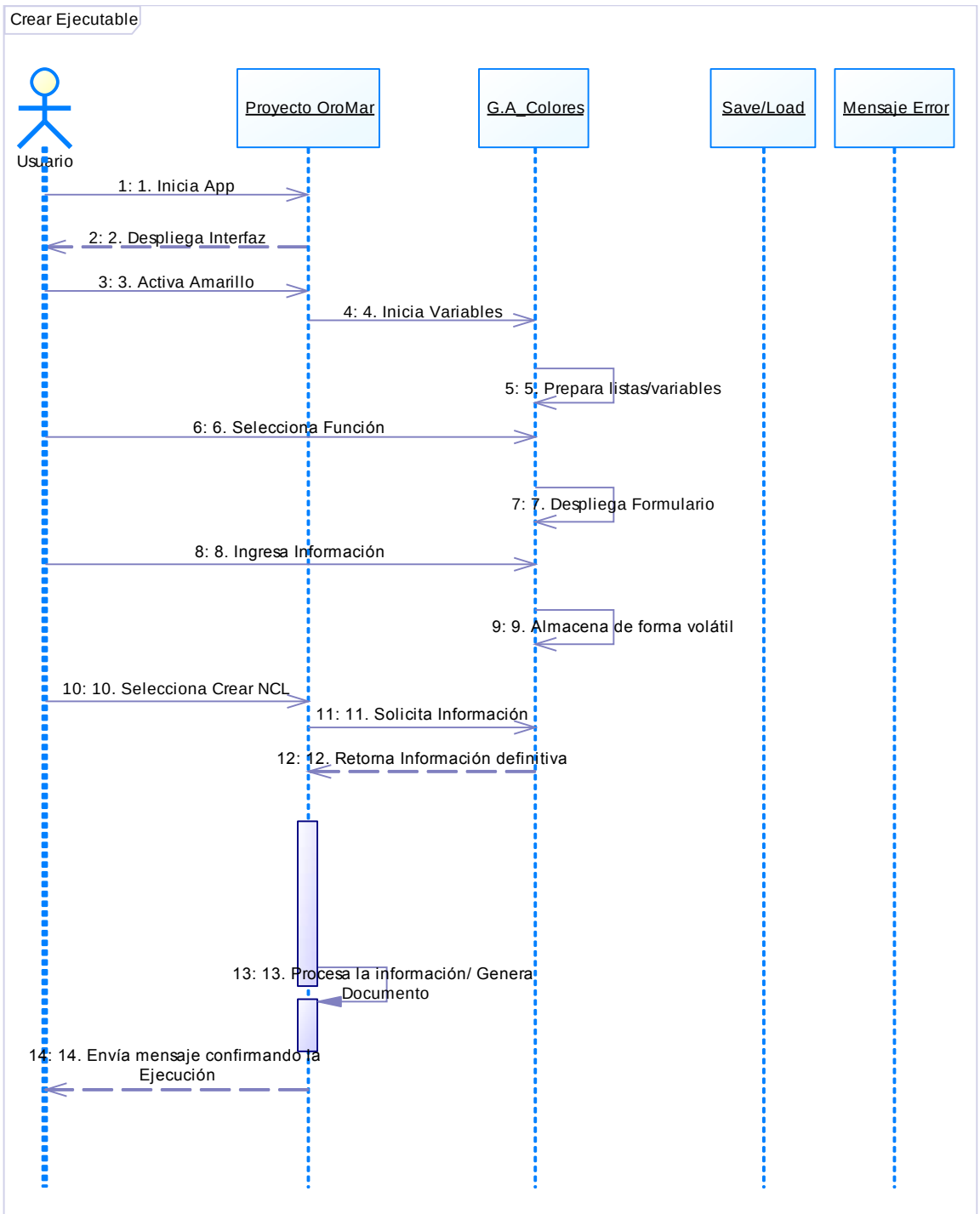


Figura #4 Diagrama de secuencia: Guardar Plantilla

Elaborado por: Juan Fernando Baird

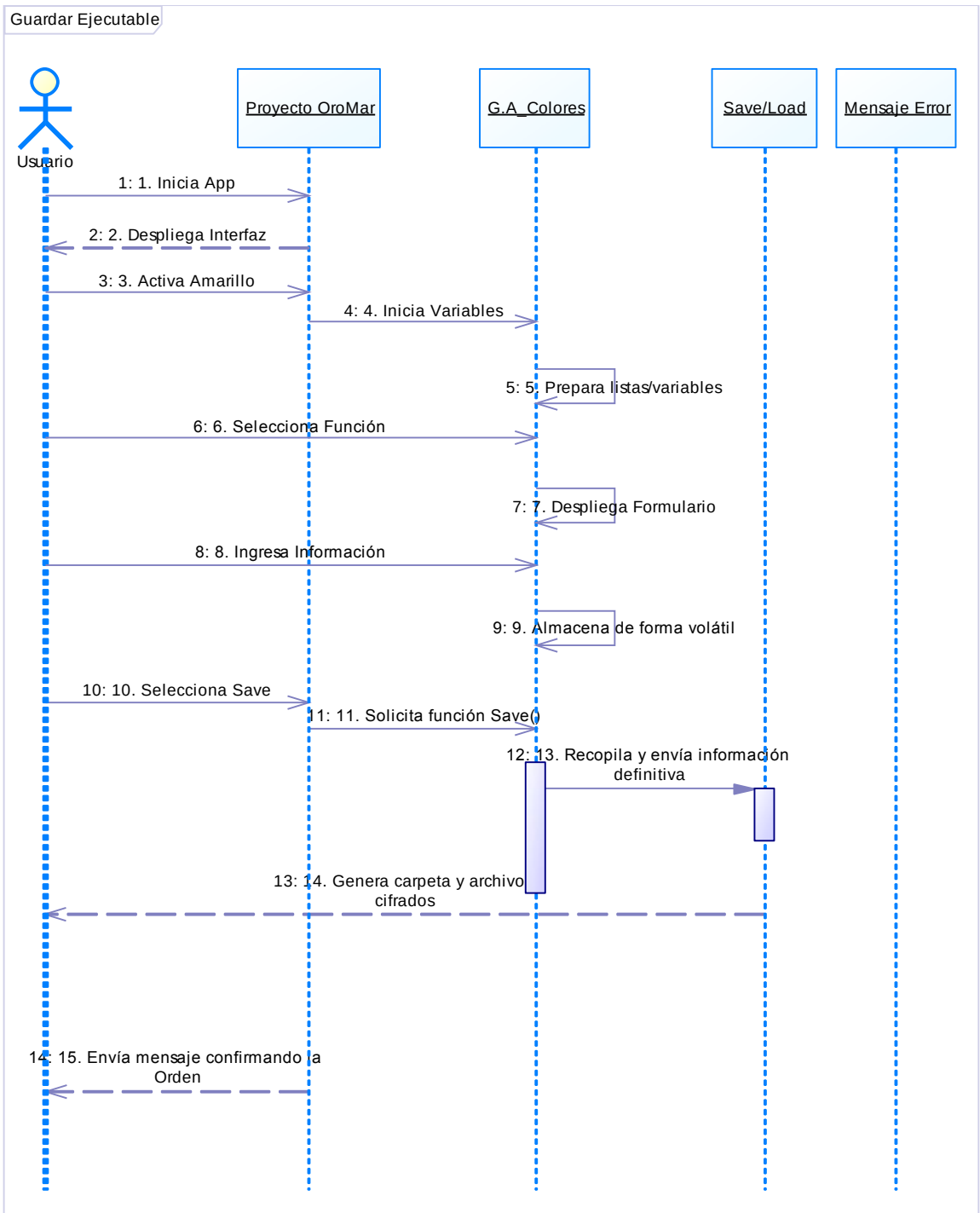
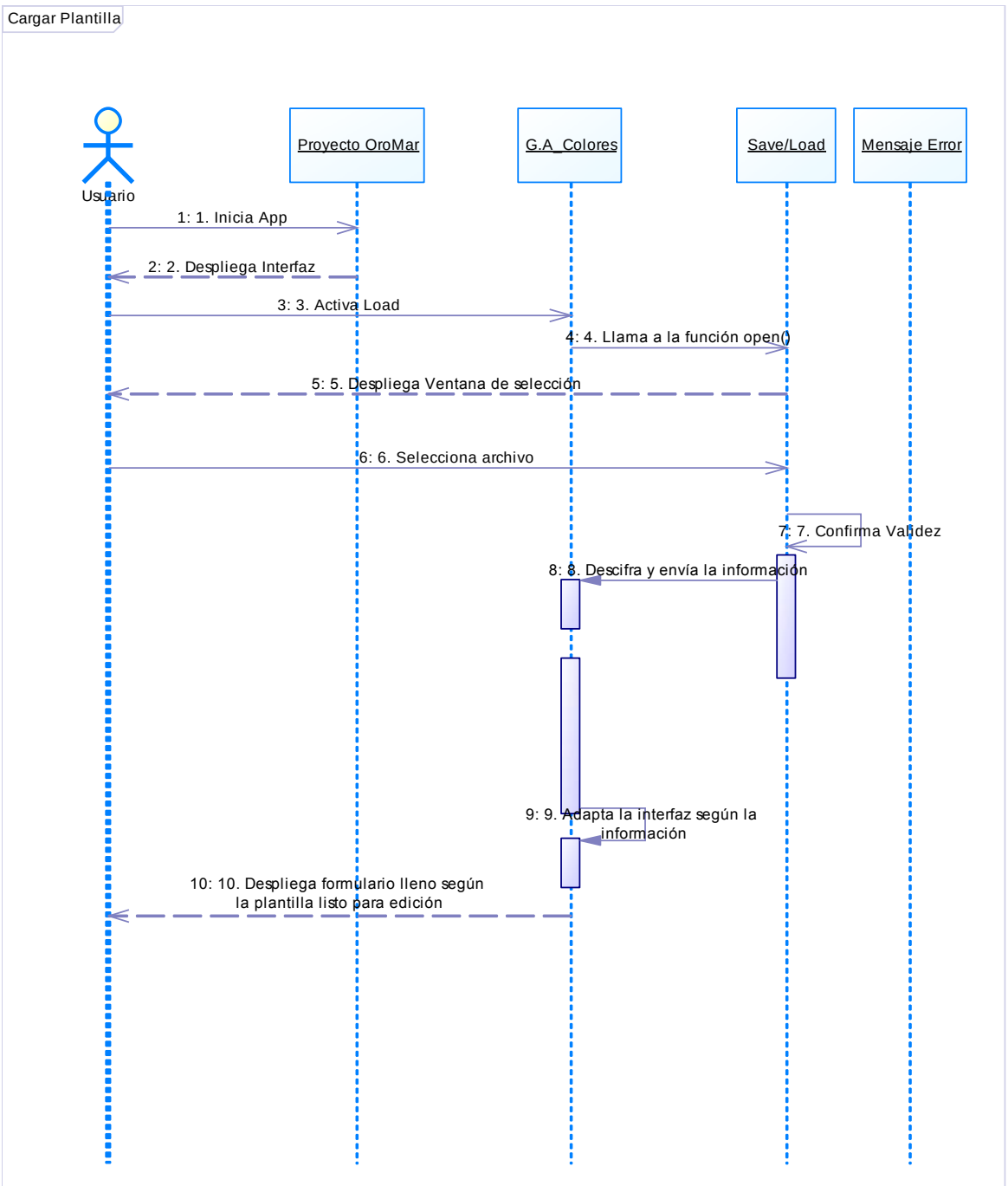


Figura #5 Diagrama de secuencia: Cargar Plantilla

Elaborado por: Juan Fernando Baird



2.1.3.3. Diagramas de actividades

Como se puede observar en los siguientes diagramas existen 3 clases hermanas las cuales poseen las mismas funciones y capacidades pero manejan sus propias variables y parámetros, estas clases se identifican mediante las letras A: amarillo, Az: azul, V: verde, al manejarse individualmente pueden ser activadas en cualquier momento durante la edición o creación de una plantilla nueva o ejecutable.

La selección de Función A, V, Az representa la activación de una de las opciones en el menú desplegable en su respectivo botón A: amarillo, V: verde, Az: azul. Indicando así la selección de la función que se le dará al botón programado, como lo sería el despliegue de simples assets o de un menú.

En caso de fallar la validación final, el usuario retorna al formulario desplegado anteriormente, pero esto no significa que no pueda regresar a la etapa de selección de función, simplemente deberá seleccionar la función deseada y la aplicación adaptará la interfaz para el nuevo formulario a desplegarse, esto causa que la información previamente ingresa sea borrada, ya que ambas funciones requieren de algoritmos distintos para su uso.

Figura #6 Diagrama de actividades: Crear Ejecutable

Elaborado por: Juan Fernando Baird

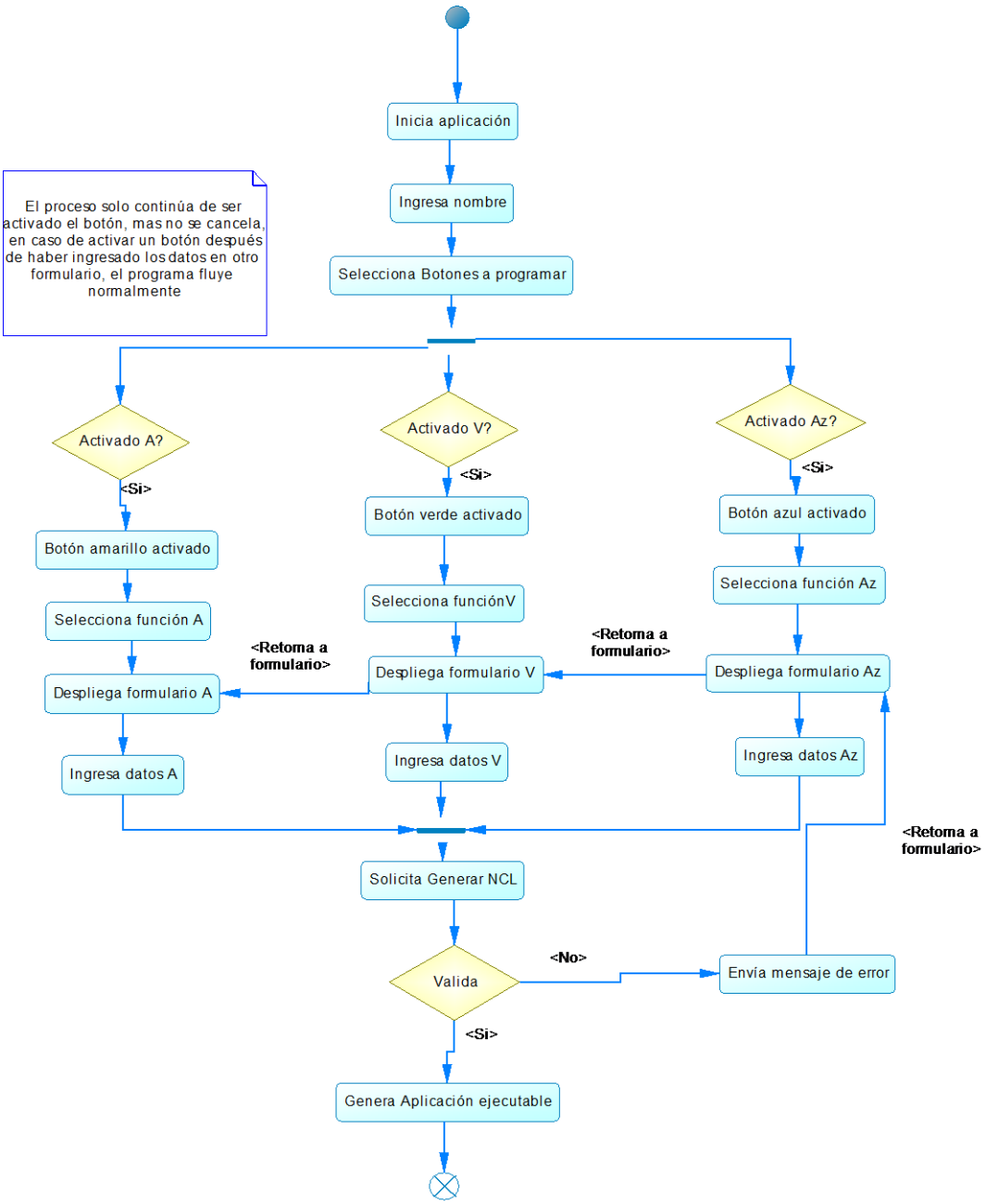


Figura #7 Diagrama de actividades: Crear Plantilla

Elaborado por: Juan Fernando Baird

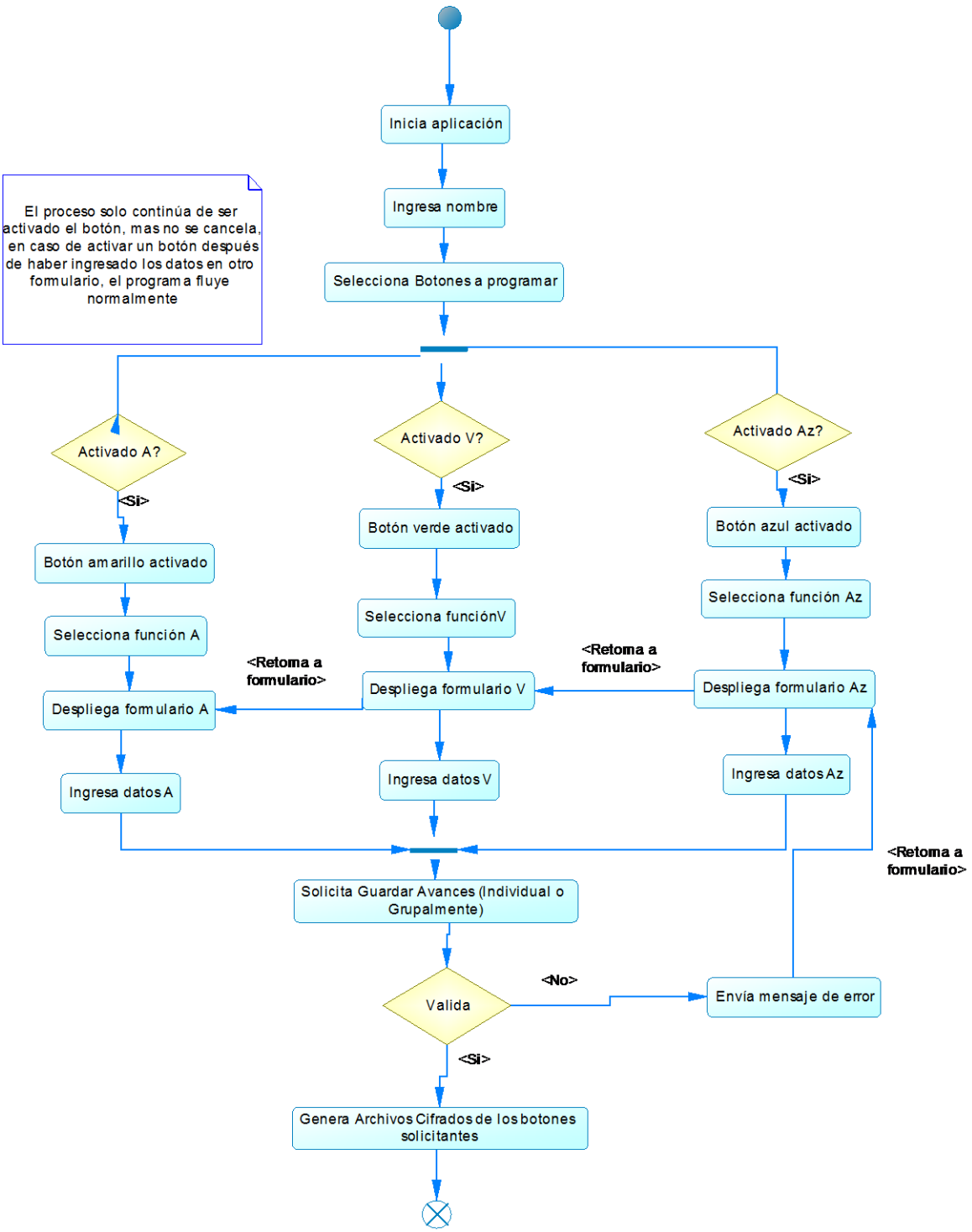
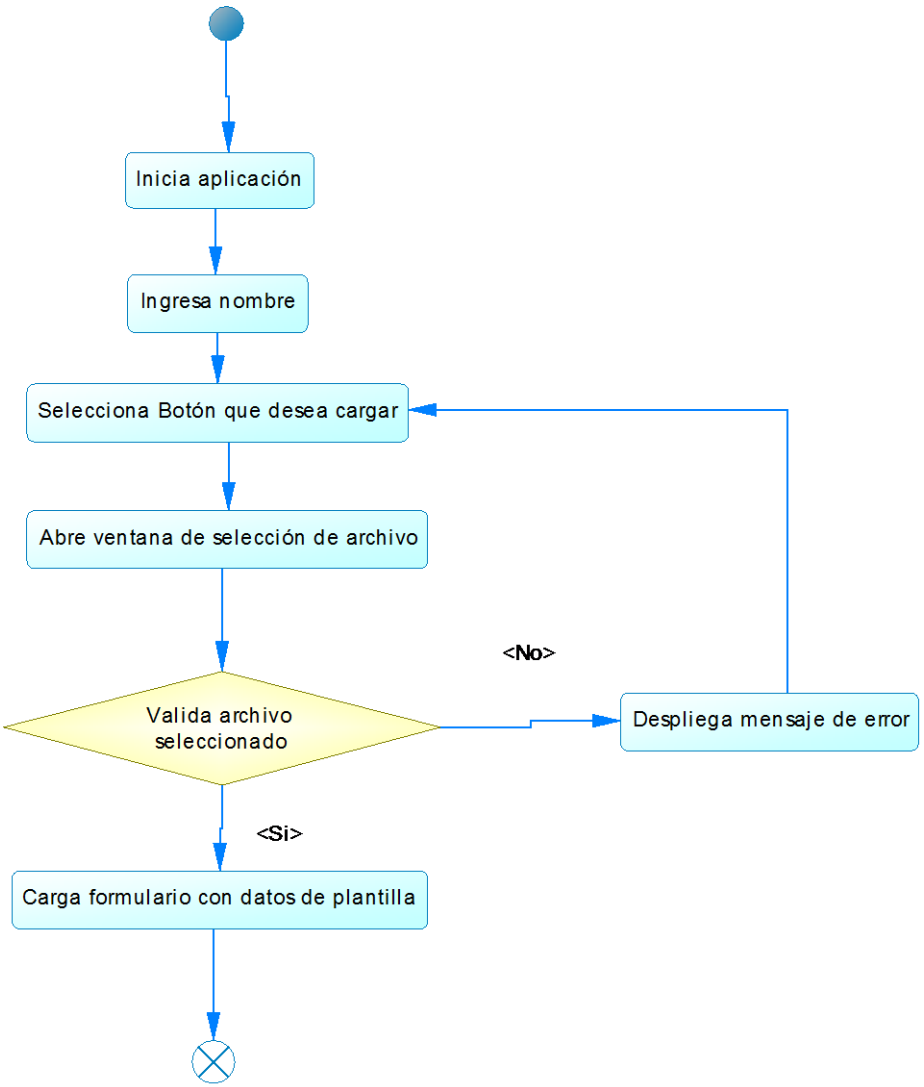


Figura #8 Diagrama de actividades: Cargar Plantilla

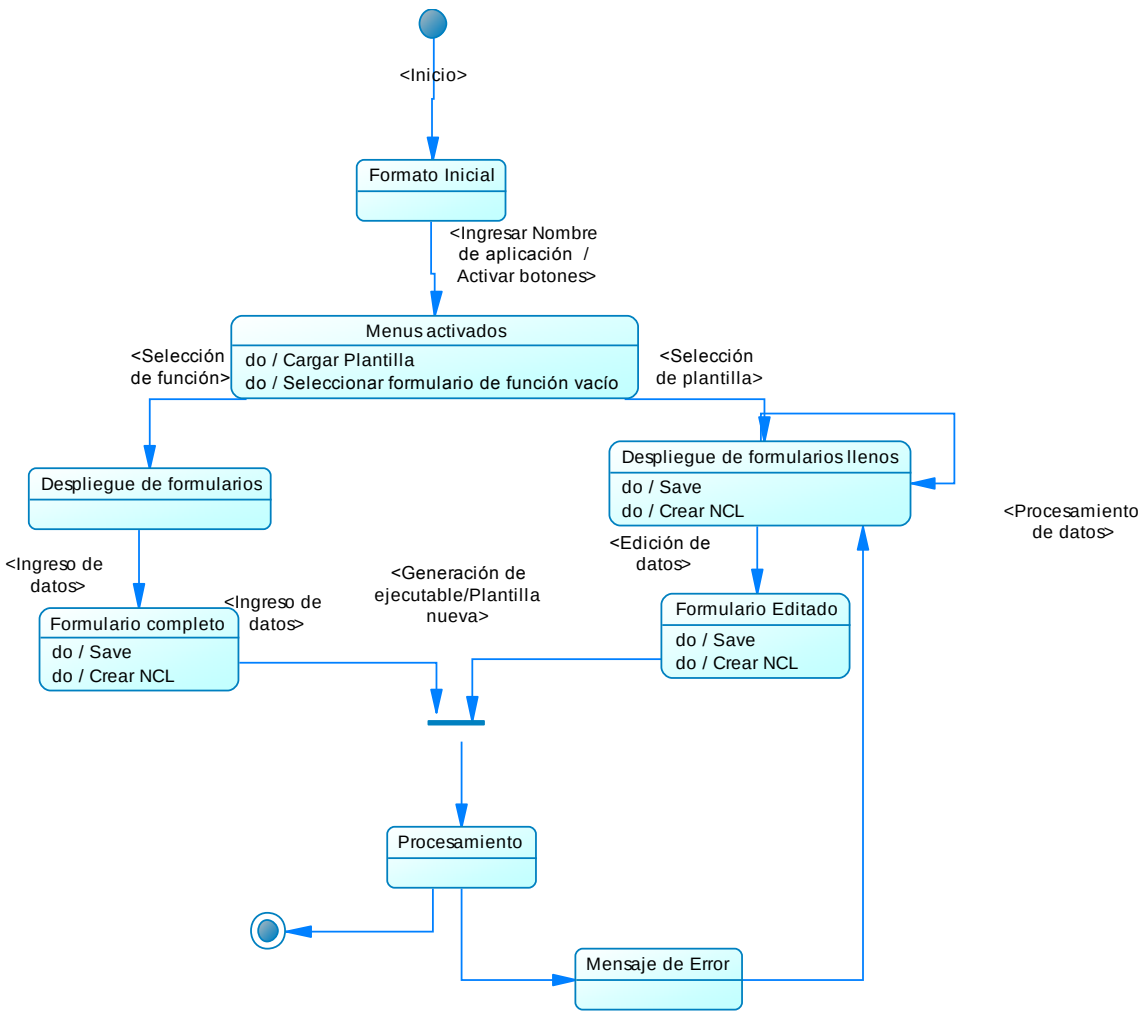
Elaborado por: Juan Fernando Baird



2.1.3.4. Diagramas de estados

Figura #9 Diagrama de estados

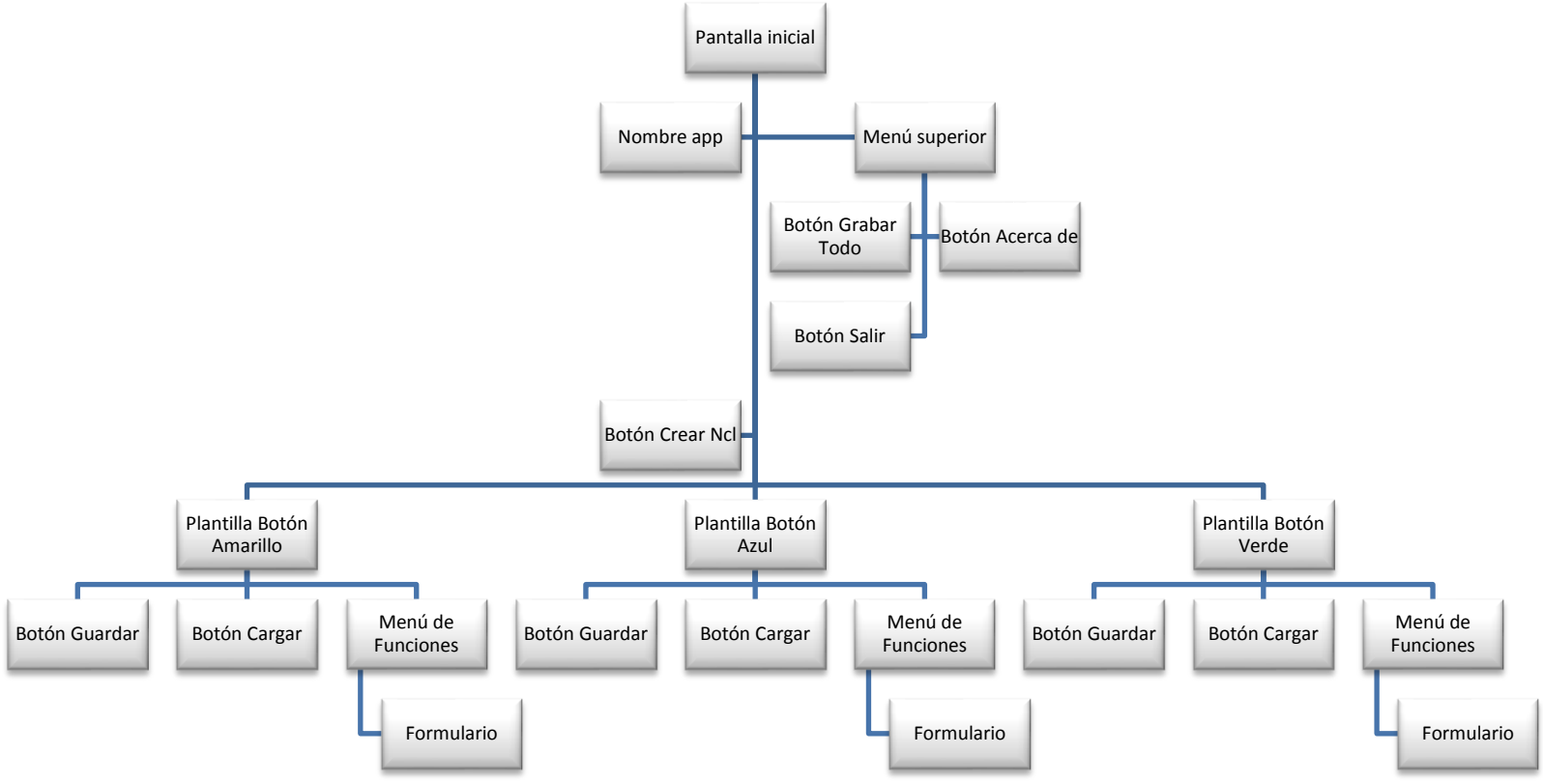
Elaborado por: Juan Fernando Baird



2.1.3.5. Diagrama de navegación

Figura #10 Diagrama de navegación

Elaborado por: Juan Fernando Baird



2.1.4. Estudio de factibilidad

2.1.4.1. Operativa

Considerando el cambio inminente en el país todas las empresas de Tv tendrán que recurrir al desarrollo de aplicaciones similares o contratar un equipo dedicado al desarrollo de aplicaciones Ginga para su programación. Esto representa un gran ahorro por parte de la empresa en valores operacionales y en tiempos de ejecución ahora los usuarios requieren mucho menos esfuerzo y conocimientos para obtener las aplicaciones orientadas a su programación televisiva.

2.1.4.2. Tecnológica

Para el desarrollo e implementación del sistema se utilizó software de código abierto y herramientas de licencia libre como:

- NetBeans
- Eclipse
- Emulador Ginga
- Java
- GingaNCL

El desarrollo de la aplicación se dio principalmente en la herramienta NetBeans usando código Java con algunas modificaciones en el uso del generador XML integrado que posee el IDE, para la generación del código Ginga. Eclipse simplemente fue usado en las etapas principales del desarrollo ya que posee una librería capaz de compilar código Ginga, el cual fue usado para pruebas iniciales del código generado.

2.1.4.3. Económica

Los costos más relevantes que la empresa cubrió para el desarrollo del sistema fueron:

Tabla # 7. Costos para el desarrollo del Sistema

Elaborado por: Juan Fernando Baird

Recursos	Costes
Portátil	$1.200,00
Cursos de programación en Lua y Ginga	$1.000,00
Materiales de oficina	$575,00
Movilización	$200,00
Microsoft Office	$50,00
Software de desarrollo (gratuito)	-------
Total	$3.025,00

2.2. Diseño

2.2.1. Entorno del Software

Para el diseño de la aplicación, se utilizó en un 100% el entorno de desarrollo NetBeans, en el cual se generó el código responsable del entorno gráfico y del análisis de la información recibida para convertirla en código Ginga el cual es similar en estructura al XML, pero muy distinto en funcionalidad y etiquetas. Además de la limitada disponibilidad en funciones que se posee, de manera abierta, actualmente en Ginga. Gracias a la libertad creativa que fue brindada por la empresa el proyecto adoptó el nombre de Mushu, la adquisición del nombre no se dio hasta casi terminado el proyecto.

Debido a su similitud estructural y conocimientos adquiridos en varios cursos externos fue posible la creación de procesos capaces de imprimir en un archivo .ncl (Nestled Context Languaje) el cual puede ser interpretado por los codificadores de TDT, o en el caso práctico para las pruebas y desarrollo, por cualquier emulador de consola TDT como es el caso en máquinas virtuales. (Crockford, 2008)

Para la creación de la aplicación se utilizó lenguaje Java para el entorno y procesos que analizan la información y la transforman a una estructura similar a XML, y GingaNCL para crear los parámetros pre codificados que se convertirán en el cuerpo de la aplicación Ginga resultante, a manera de un guion a seguir dependiendo de la información introducida.

2.2.1.1. Arquitectura del ambiente de desarrollo

Para la construcción de la aplicación se utilizó una pc con el sistema operativo Windows 7 profesional x64 bits, se requirió además, emulador virtual de un set-topbox adquirido en la página oficial de Ginga Ecuador, el emulador es capaz de simular un entorno óptimo para el desarrollo de aplicaciones Ginga por lo cual se tomó mucho en consideración las posibles limitaciones físicas que tendrán los equipos que el gobierno disponga al público y el hecho de que algunos televisores ya poseen un hardware básico lo que podría llevar a muchos usuarios a no comprar otro equipo incluso si son de mayor potencia.

Las características de la máquina utilizada para el desarrollo y pruebas de la aplicación son:

- Procesador: Intel Core i7 47-70QM 3.40GHz
- Sistema operativo: Windows 7 Ultimate x64 bits
- Memoria RAM: 8GB
- Tarjeta gráfica Nvidia GTX 610

2.2.1.2. Arquitectura del ambiente en producción

Para la implementación de la aplicación se creó un ejecutable .jar y se empaquetó en un DVD el cual contiene el ejecutable, el instalador del emulador Ginga para las pruebas que se deseen realizar y una copia digital del manual de usuario. Además se proporcionó a la empresa de ejemplos sencillos realizados por el autor. La independencia de servicios web y base de datos convierte a la aplicación Mushu a un sistema portátil y flexible con disponibilidad las 24 horas, todos los días de la semana, la aplicación pesa 243 kb y no requiere instalación.

2.2.2. Diagramas de Diseño

2.2.2.1. Diagramas de Colaboración

Figura #11 Diagrama de Colaboración- Crear NCL

Elaborado por: Juan Fernando Baird

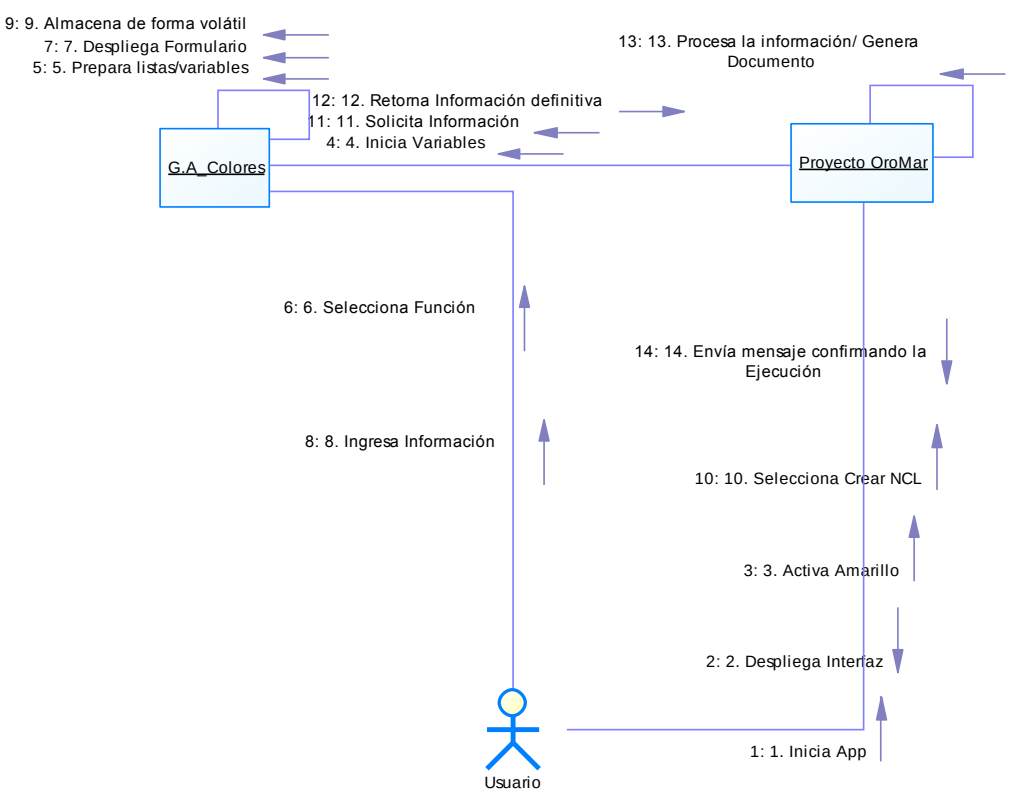


Figura #12 Diagrama de Colaboración- Crear Plantilla

Elaborado por: Juan Fernando Baird

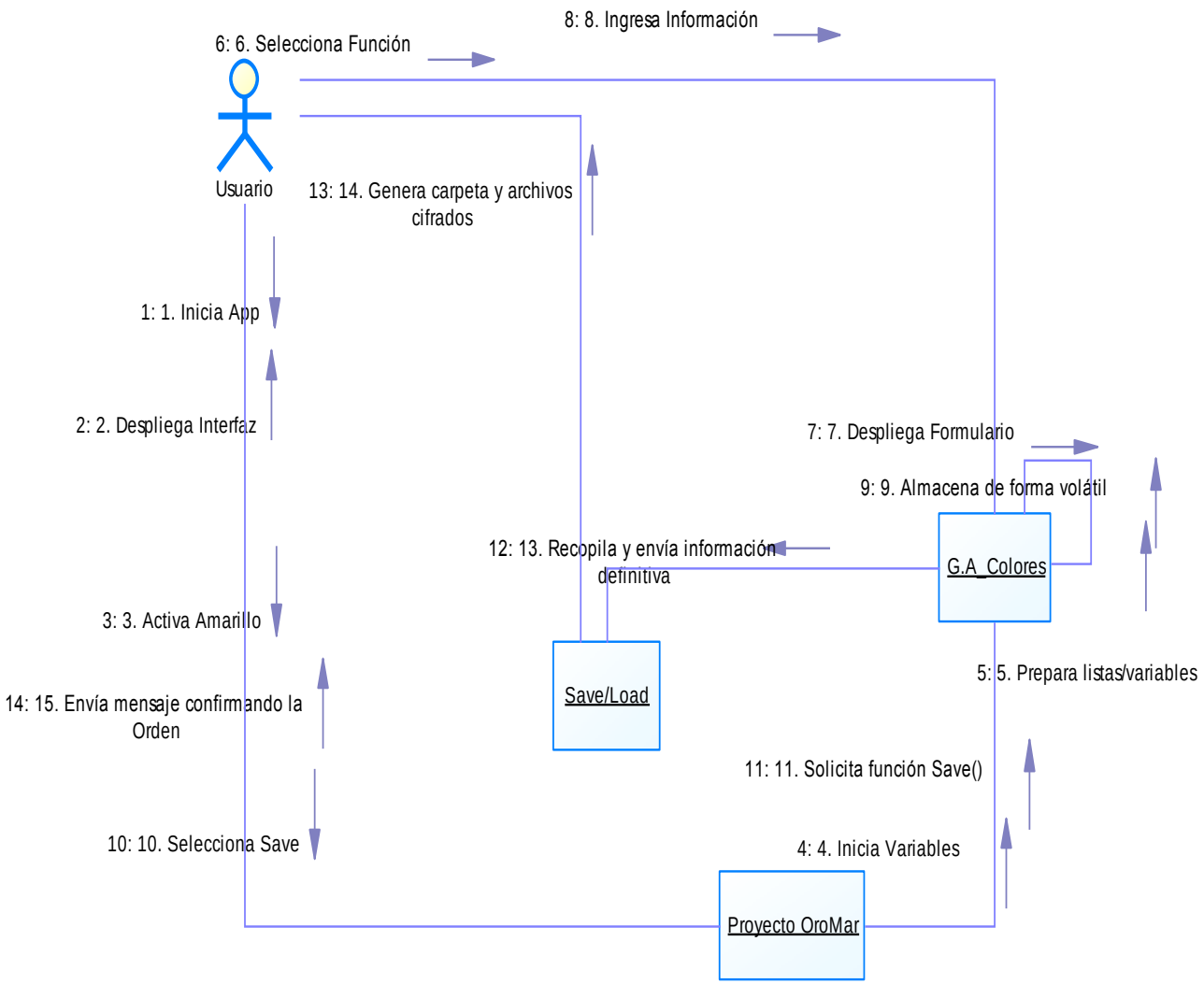
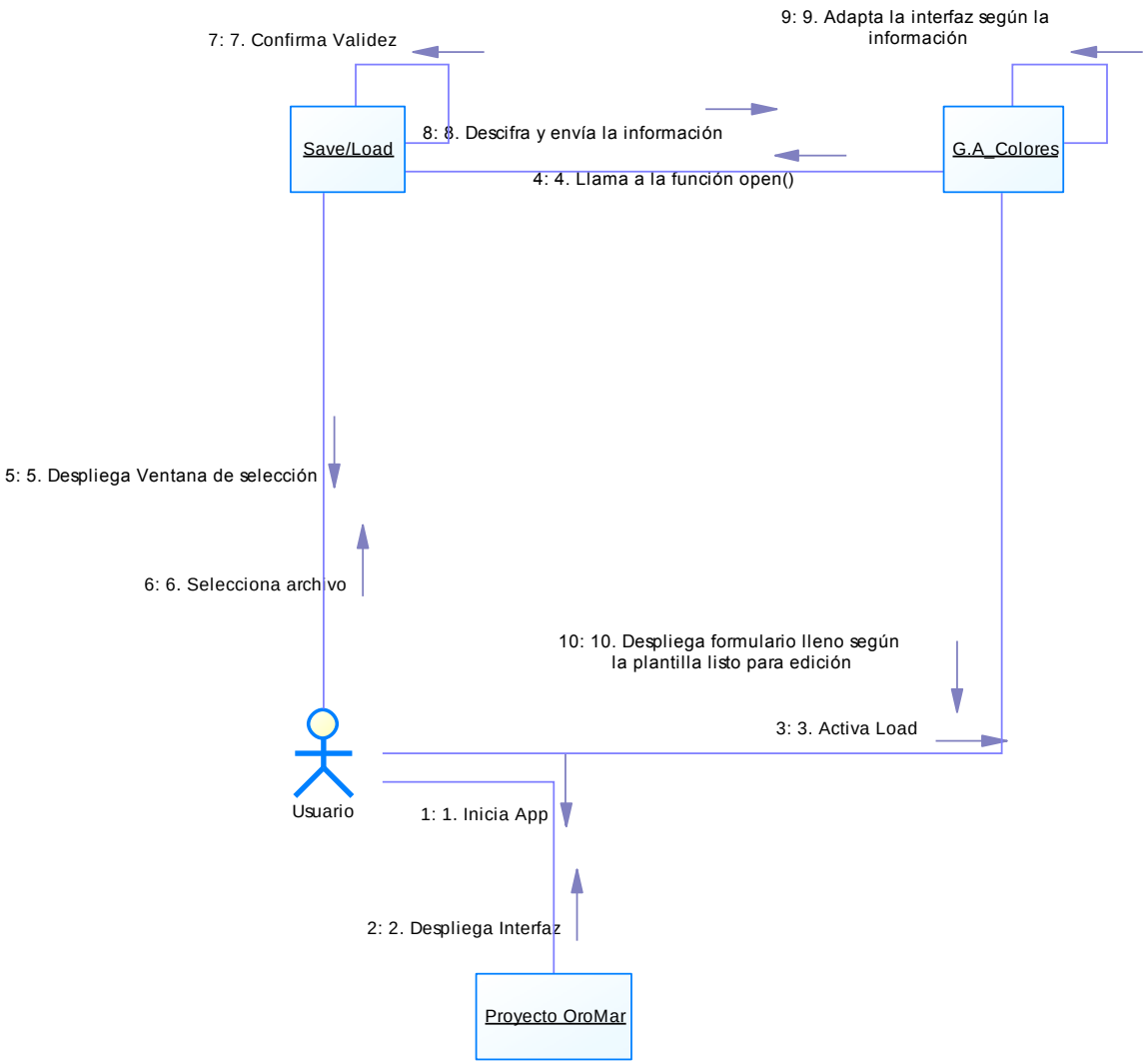


Figura #13 Diagrama de Colaboración- Cargar Plantilla

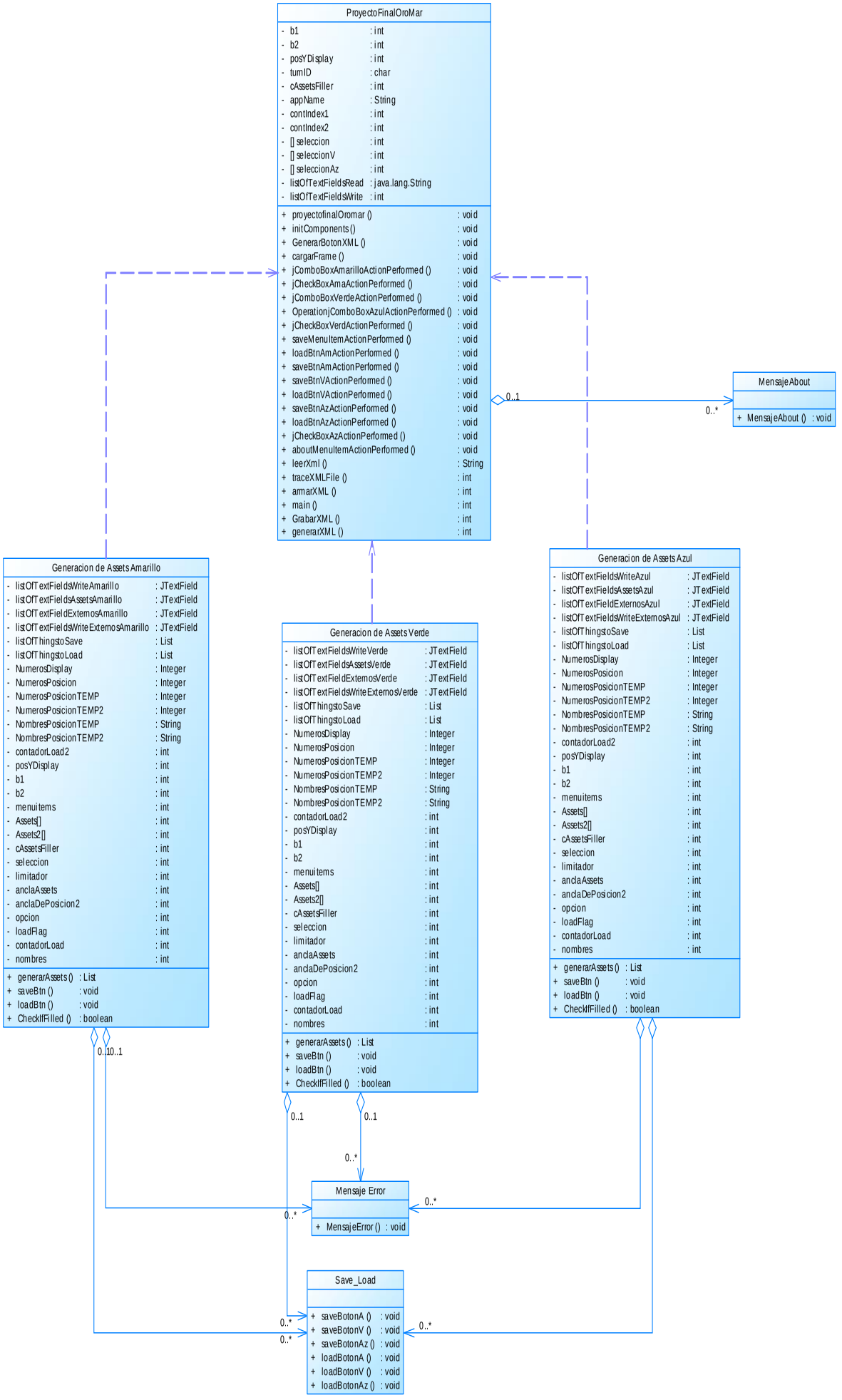
Elaborado por: Juan Fernando Baird



2.2.2.2. Diagrama de Clases (Diseño)

Figura #14 Diagrama de Colaboración- Cargar Plantilla

Elaborado por: Juan Fernando Baird


ProyectoFinalOroMar
- b1 : int
- b2 : int
- posYDisplay : int
- tumID : char
- cAssetsFiller : int
- appName : String
- contIndex1 : int
- contIndex2 : int
- [] seleccion : int
- [] seleccionV : int
- [] seleccionAz : int
- listOfTextFieldsRead : java.lang.String
- listOfTextFieldsWrite : int
+ proyectofinalOromar () : void
+ initComponents() : void
+ GenerarBotonXML () : void
+ cargarFrame () : void
+ jComboBoxAmarilloActionPerformed () : void
+ jCheckBoxAmaActionPerformed () : void
+ jComboBoxVerdeActionPerformed () : void
+ OperationjComboBoxAzulActionPerformed () : void
+ jCheckBoxVerdActionPerformed () : void
+ saveMenuItemActionPerformed () : void
+ loadBtnAmActionPerformed () : void
+ saveBtnAmActionPerformed () : void
+ saveBtnVActionPerformed () : void
+ loadBtnVActionPerformed () : void
+ saveBtnAzActionPerformed () : void
+ loadBtnAzActionPerformed () : void
+ jCheckBoxAzActionPerformed () : void
+ aboutMenuItemActionPerformed () : void
+ leerXml () : String
+ traceXMLFile () : int
+ armarXML () : int
+ main () : int
+ GrabarXML () : int
+ generarXML () : int
0..1
0..*
MensajeAbout
+ MensajeAbout () : void
Generacion de Assets Amarillo
- listOfTextFieldsWriteAmarillo : JTextField
- listOfTextFieldsAssetsAmarillo : JTextField
- listOfTextFieldExternosAmarillo : JTextField
- listOfTextFieldsWriteExternosAmarillo : JTextField
- listOfThingstoSave : List
- listOfThingstoLoad : List
- NumerosDisplay : Integer
- NumerosPosicion : Integer
- NumerosPosicionTEMP : Integer
- NumerosPosicionTEMP2 : Integer
- NombresPosicionTEMP : String
- NombresPosicionTEMP2 : String
- contadorLoad2 : int
- posYDisplay : int
- b1 : int
- b2 : int
- menuitems : int
- Assets[] : int
- Assets2[] : int
- cAssetsFiller : int
- seleccion : int
- limitador : int
- anclaAssets : int
- anclaDePosicion2 : int
- opcion : int
- loadFlag : int
- contadorLoad : int
- nombres : int
+ generarAssets () : List
+ saveBtn () : void
+ loadBtn () : void
+ CheckIfFilled () : boolean
0..1
0..1
Generacion de Assets Verde
- listOfTextFieldsWriteVerde : JTextField
- listOfTextFieldsAssetsVerde : JTextField
- listOfTextFieldExternosVerde : JTextField
- listOfTextFieldsWriteExternosVerde : JTextField
- listOfThingstoSave : List
- listOfThingstoLoad : List
- NumerosDisplay : Integer
- NumerosPosicion : Integer
- NumerosPosicionTEMP : Integer
- NumerosPosicionTEMP2 : Integer
- NombresPosicionTEMP : String
- NombresPosicionTEMP2 : String
- contadorLoad2 : int
- posYDisplay : int
- b1 : int
- b2 : int
- menuitems : int
- Assets[] : int
- Assets2[] : int
- cAssetsFiller : int
- seleccion : int
- limitador : int
- anclaAssets : int
- anclaDePosicion2 : int
- opcion : int
- loadFlag : int
- contadorLoad : int
- nombres : int
+ generarAssets () : List
+ saveBtn () : void
+ loadBtn () : void
+ CheckIfFilled () : boolean
0..1
0..1
0..*
Generacion de Assets Azul
- listOfTextFieldsWriteAzul : JTextField
- listOfTextFieldsAssetsAzul : JTextField
- listOfTextFieldExternosAzul : JTextField
- listOfTextFieldsWriteExternosAzul : JTextField
- listOfThingstoSave : List
- listOfThingstoLoad : List
- NumerosDisplay : Integer
- NumerosPosicion : Integer
- NumerosPosicionTEMP : Integer
- NumerosPosicionTEMP2 : Integer
- NombresPosicionTEMP : String
- NombresPosicionTEMP2 : String
- contadorLoad2 : int
- posYDisplay : int
- b1 : int
- b2 : int
- menuitems : int
- Assets[] : int
- Assets2[] : int
- cAssetsFiller : int
- seleccion : int
- limitador : int
- anclaAssets : int
- anclaDePosicion2 : int
- opcion : int
- loadFlag : int
- contadorLoad : int
- nombres : int
+ generarAssets () : List
+ saveBtn () : void
+ loadBtn () : void
+ CheckIfFilled () : boolean
0..*
0..*
Mensaje Error
+ MensajeError () : void
0..*
0..*
Save_Load
+ saveBotonA () : void
+ saveBotonV () : void
+ saveBotonAz () : void
+ loadBotonA () : void
+ loadBotonV () : void
+ loadBotonAz () : void
0..*
0..*
0..*

2.2.3. Estructura de archivos cifrados

Como se puede notar la aplicación carece de base de datos, la razón de ello radica en la falta de necesidad de una. Toda la información ingresada es almacenada en variables temporales de manera dinámica durante la ejecución esto a su vez le da a la aplicación mayor movilidad y disponibilidad ya que toda la información que se requiere para crear ejecutables Ginga ya se encuentra en el código.

Una vez ejecutado el código, si el usuario desea guardar su progreso a manera de una plantilla reutilizable mediante las funciones guardar o guardar All, las cuales crean una carpeta en el escritorio y en su interior creara archivos cifrados tipo .txt con códigos numéricos que establecen el orden, cantidad y tipo de assets ingresados por el usuario. Al cargar estos archivos el programa en realidad recupera la información cifrada y ejecuta automáticamente las funciones del mismo en un orden predeterminado en cuestión de segundos y desplegando la plantilla tal como el usuario la tenía al momento de grabar.

2.2.4. Diccionario de Datos

Tabla # 8. Campos del archivo cifrado

Elaborado por: Juan Fernando Baird

Campos	Contenido
int Selección	Almacena el número de función seleccionado
List NumerosDisplay	Almacena los valores numéricos insertados en cantidad de elementos internos y externos
List NumerosPosicion	Almacena las posiciones x y, el alto y el ancho de los assets.

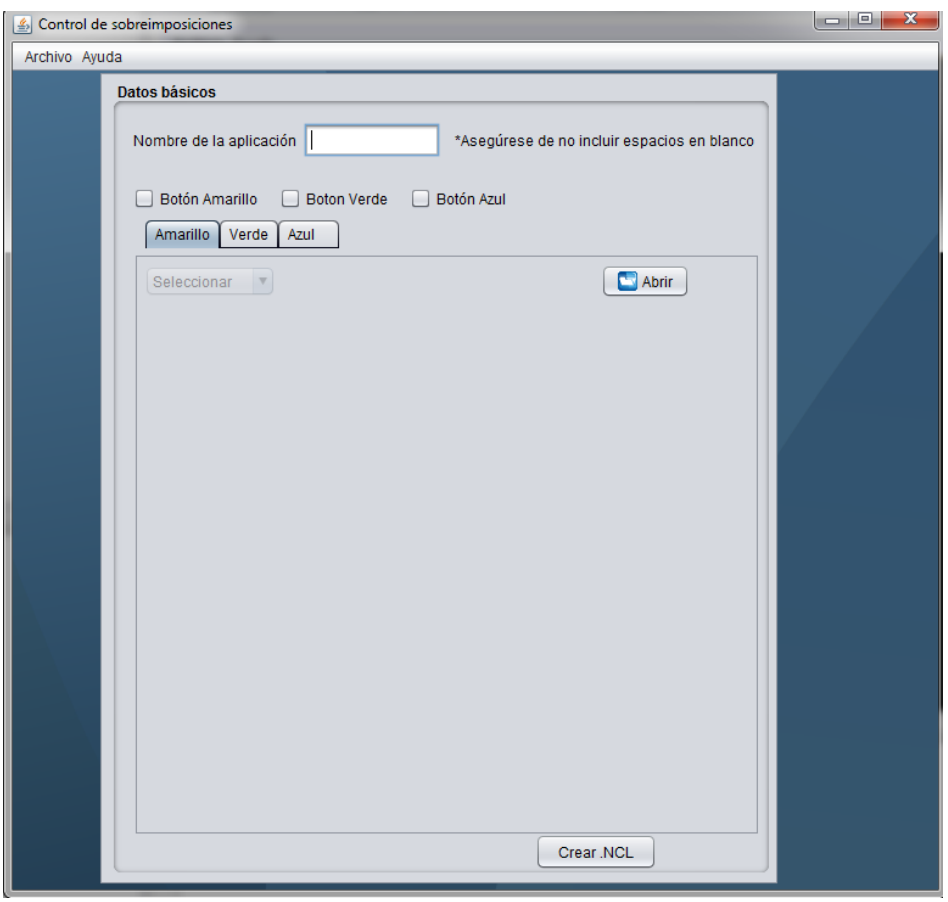
2.2.5. Descripción de Interfaces

Al ser un sistema orientado a diseñadores y programadores por igual, se creó una interfaz sencilla que requiere la menor cantidad de información posible, pero a la vez genera un código ordenado y etiquetado de una forma organizada y fácil de entender para el incremento de código personalizado por un colega.

Como se muestra en la figura #14 la aplicación cuenta con un set de elementos prediseñados y acomodados para su fácil uso por parte del usuario, cabe mencionar que aunque la parte externa se ve sencilla, el mayor reto fue manejar todas las interacciones dinámicas que dada la naturaleza del programa, son las que se encargan de manejar ese mínimo de información requerido, y convertirlo en varias líneas de código ejecutable Ginga.

Figura #15 Interfaz: Estado Inicial

Elaborado por: Juan Fernando Baird

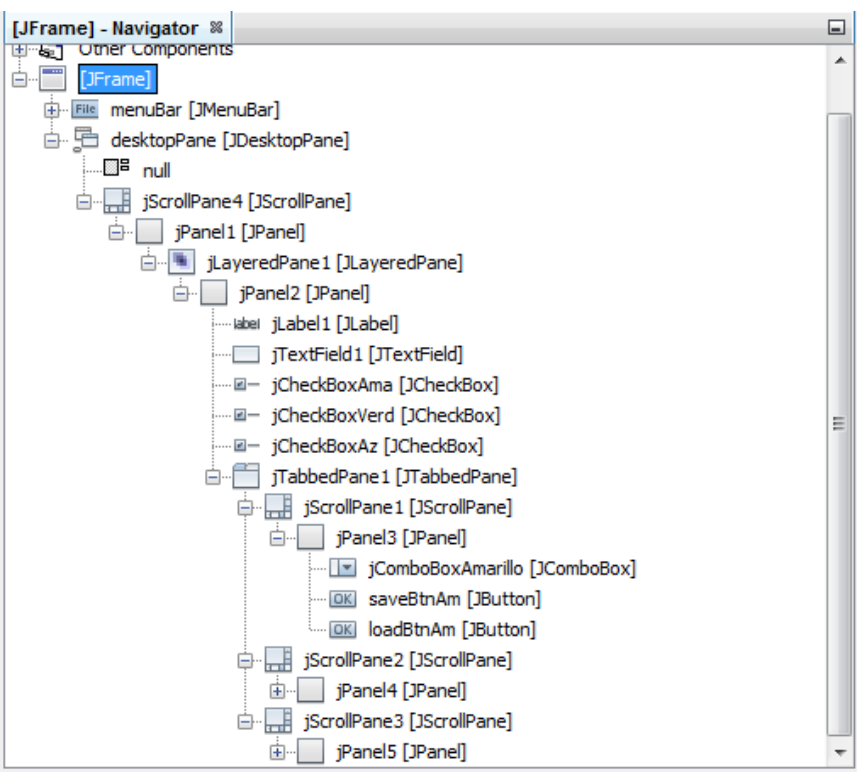


La aplicación se divide en capas individuales que representan escalones de procesos, mientras más profundo se encuentra la capa, el nivel de procesamiento y elementos que posee se vuelven más simples, de esta manera se facilita el procesamiento y almacenamiento de datos. Además esta estructura facilita el control de elementos gráficos durante la ejecución del mismo, lo cual puede ser observado al momento de solicitar el incremento o decremento de elementos en cualquiera de las funciones presentadas.

En la Figura #16 se puede observar la estructura de los elementos tal y como fueron diseñados en el IDE a manera de ilustrar la manera escalonada en la cual se optó por trabajar, de esta manera el control simultáneo de varios formularios paralelos se convirtió en una tarea más sencilla y requiere de menos capacidad de procesamiento.

Figura #16 Interfaz: Capas de la interfaz de usuario

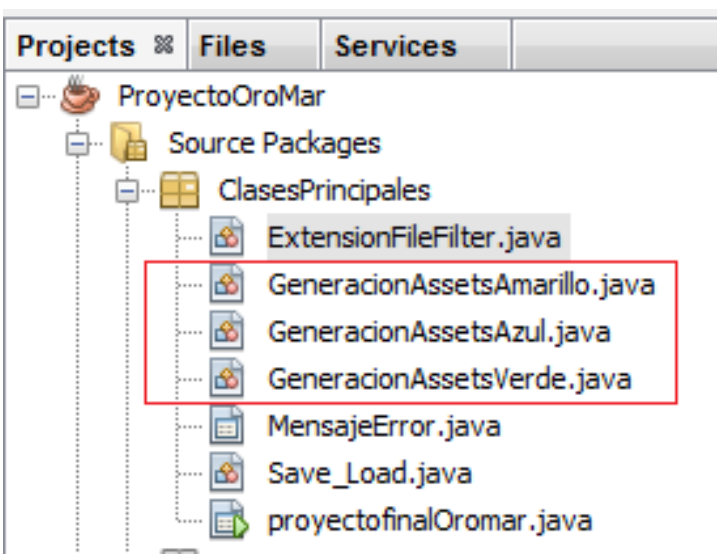
Elaborado por: Juan Fernando Baird



A medida que se navega la aplicación se ira convirtiendo dinámicamente, de acuerdo a las especificaciones del usuario, manejando cada "Botón" individualmente según sea la necesidad del mismo, para esto se manejan 3 clases hermanas en las cuales únicamente varía el nombre de ciertas variables y las funciones a las que llaman. De esta manera se mantiene cada "Botón" con sus respectivas variables intactas y permite una mayor libertad creativa al momento del diseño de la aplicación final. La cual toma la información colectiva y utilizando algoritmos previamente mapeados en una secuencia exacta, transforma los datos de posición en toda una aplicación Ginga independiente, manteniendo un estándar fácil de comprender y modificar.

Figura #17 Interfaz: Clases de control

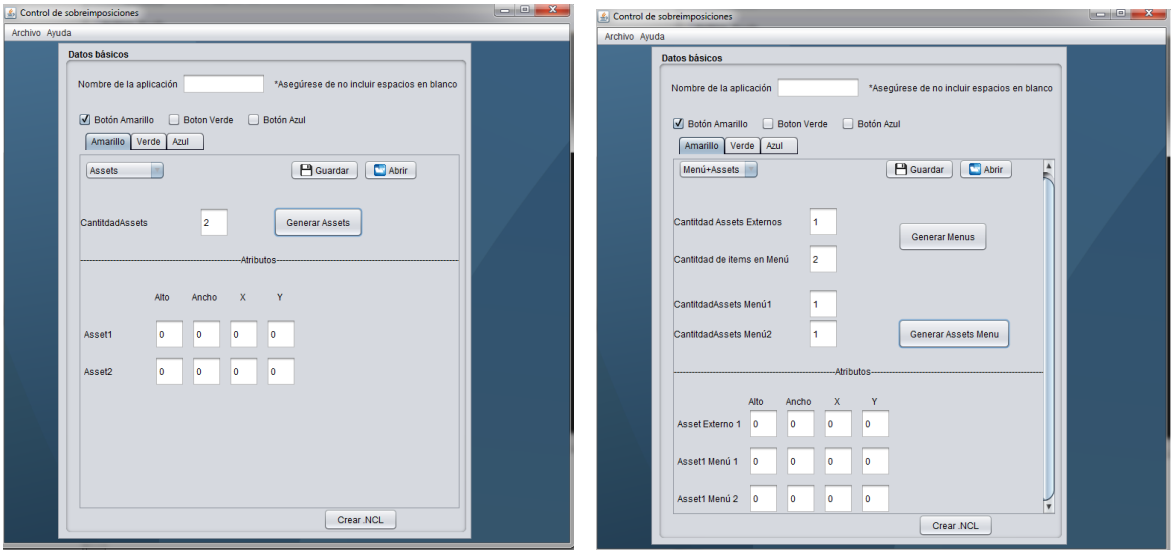
Elaborado por: Juan Fernando Baird



Como se puede observar en las figuras el uso de clases individuales permite el desarrollo de parámetros completamente distintos en cada una de las pestañas sin el riesgo de pérdida de datos al momento de su edición, el desarrollo paralelo y selectivo se considera un requisito obligatorio, ya que al tratarse de una herramienta diseñada para la industria del entretenimiento, es necesario dar ciertas libertades creativas a los usuarios.

Figura #18 Interfaz: Tipos de formulario

Elaborado por: Juan Fernando Baird

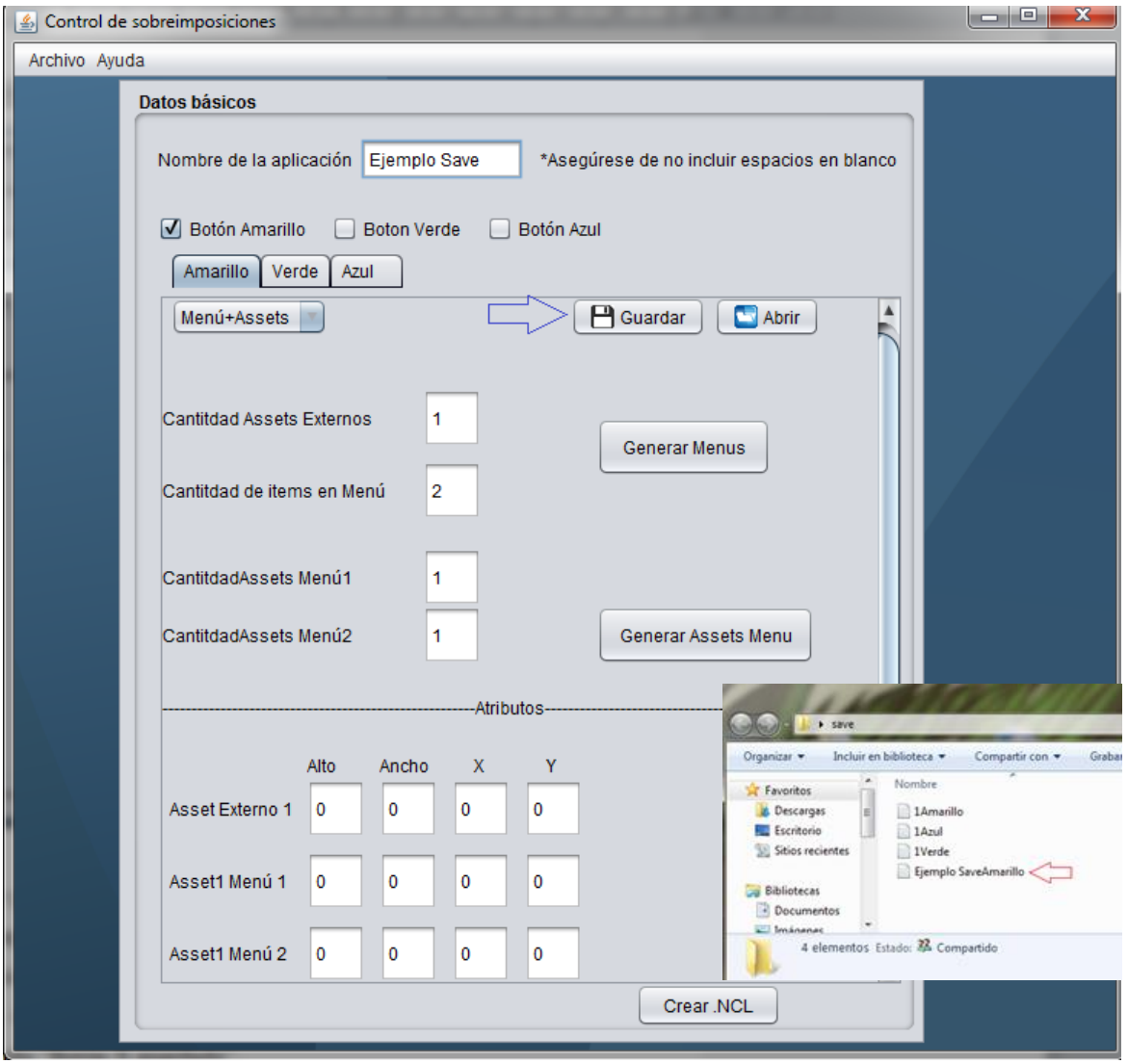


Una vez ingresada la información se puede grabar en un archivo .txt cifrado, el cual solo poseerá la información del "Botón" seleccionado y se almacenará en una carpeta ubicada en el escritorio llamada guardar o se puede compilar y combinar para generar un archivo ejecutable .ncl, el cual puede ser visualizado en cualquier consola TDT o en su defecto en los emuladores disponibles para el mismo efecto.

Esta aplicación aparecerá en el escritorio del equipo en cuestión, tomando en consideración que el usuario deberá colocarla en la carpeta que contenga los archivos pertinentes para la misma. La aplicación fue diseñada para no exigir mucho del set-top box pero se deben considerar las limitaciones del equipo físico ya que el emulador posee un mayor poder de procesamiento al estar instalado en una computadora personal.

Figura #19 Interfaz: Resultado guardar

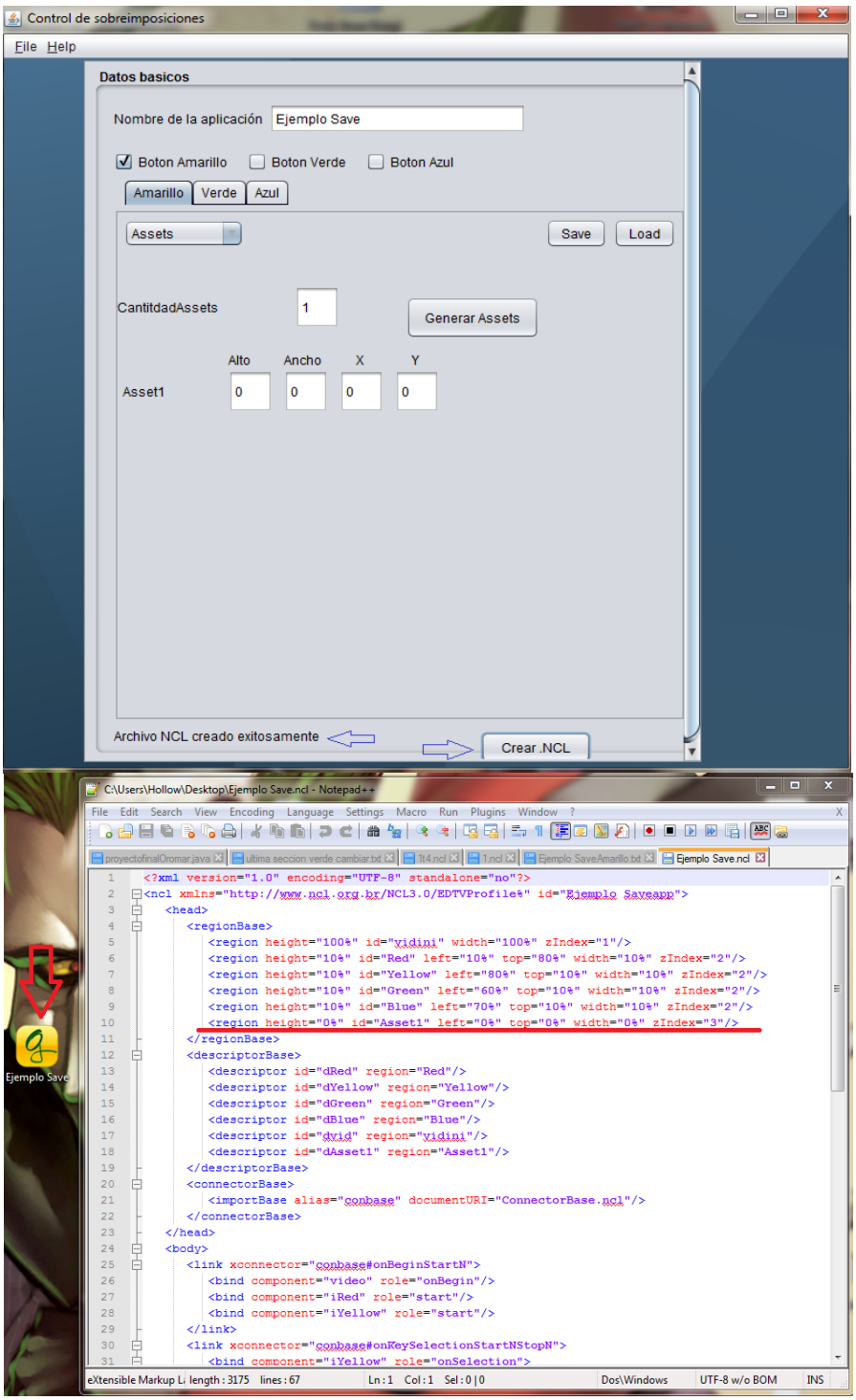
Elaborado por: Juan Fernando Baird



Como se puede observar en la Figura #19, la aplicación almacenara las plantillas creadas en la carpeta “save”, ubicada en el escritorio del usuario, en caso de no existir la misma la aplicación se encarga de crear una carpeta nueva en el escritorio llamada “save” y almacena los datos en la misma, enviando un mensaje en la parte inferior de la aplicación diciendo “Botón X guardado”.

Figura #20 Interfaz: Resultado Crear NCL

Elaborado por: Juan Fernando Baird



Como se puede observar en la figura anterior, el archivo resultante de la generación de código GingaNCL posee una estructura similar a la de un archivo XML común, pero construido según los parámetros de programación Ginga, facilitando así el trabajo de desarrollador ya que en Ginga se trabaja con etiquetas e identificadores muy delicados, esto causa una gran pérdida de tiempo y esfuerzo al momento de hacer pruebas o cambios a un código viejo, y aún más problemas al crear una aplicación desde cero ya que es muy sensible.

2.2.6. Administración y Seguridad

2.2.6.1. Niveles de Acceso

Absolutamente todos los usuarios tienen el mismo nivel de accesibilidad tanto a la aplicación como al código resultante, la aplicación fue creada como una herramienta útil para los diseñadores o programadores encargados del diseño de aplicaciones Ginga para la empresa OroMarTv, por esa misma razón es de esperar que todos tengan los mismos privilegios de uso. Además la aplicación debe ser instalada en cada máquina individualmente, por lo cual únicamente maneja archivos propios de un mismo usuario

2.2.6.2. Seguridad

En cuanto a seguridades, el sistema solo posee el cifrado en los archivos guardados, el cual fue creado con la simple razón de evitar que el usuario decida manipular los datos del mismo y posiblemente dañe la plantilla al hacerlo. La razón de esto se encuentra en que el programa no está diseñado para manejar ningún tipo de información delicada o potencialmente dañina de la empresa, no tiene conexión a internet por lo cual no puede ser hackeado remotamente, e incluso de ser ingresado mediante la computadora del usuario, el ‘Hacker’ podrá crear plantillas inofensivas en el escritorio de la computadora intervenida.

Al igual que aplicaciones como Word o Paint, la aplicación crea un archivo inofensivo y no requiere de información sensible para trabajar, las únicas seguridades implementadas son para evitar que el Usuario se perjudique a sí mismo al usar la aplicación o sus productos derivados. Para ello el programa tiene mensajes de error cuando el usuario esté por cometer alguna falla capaz de crashear el programa y los archivos de persistencia o plantillas se encuentran cifrados de modo que el usuario puede abrirlos pero solo encontrará información codificada, con el objetivo de que no sienta deseos de editarla.

CAPÍTULO III

RESULTADOS

3.1. Construcción

3.1.1.1. Generalidades

Para la construcción del código se utilizó principalmente el IDE NetBeans y fue escrito en lenguaje Java debido a su flexibilidad, facilidad de aprendizaje y accesibilidad a documentos y tutoriales en línea al igual que cursos disponibles, únicamente un módulo contiene lenguaje GingaNCL ya que este es el encargado de generar el ejecutable y armar el código final.

La aplicación pasó por varios cambios y mejoras durante su desarrollo, principalmente dado a que fue creada desde cero y con varias ideas en mente al diseñarla, no cabe duda de su originalidad ya que es la primera herramienta versátil para un código aún en desarrollo e implementación en el país.

La aplicación fue desarrollada por secciones individuales, cada una basada en una entrevista o charla con los posibles usuarios y sugerencias proporcionadas por el desarrollador, los módulos se desarrollaron de la siguiente manera: Generación código NCL, Desarrollo de Interfaz inicial, Algoritmos de estructura para el código NCL, Algoritmos de Ingreso de datos, Creación del archivo cifrado, Recuperación de plantilla

creada y finalmente se crearon las funciones unificadoras encargadas de combinar la información ingresada y construir el archivo ejecutable final. Cada módulo fue modificado y mejorado mientras se desarrollaba la aplicación para mantener una constancia en el estilo visual e interactivo de la misma.

La fase de construcción tuvo las siguientes etapas: análisis de funciones para la aplicación y recopilación de requerimientos, desarrollo de interfaz inicial, desarrollo de módulos individuales, acoplamiento de módulos, pruebas y mantenimiento.

3.1.1.2. Estándares de codificación

Para el desarrollo de la aplicación se usó estándares comunes de codificación de acuerdo con las buenas prácticas de programación con nombres claros y distintivos entre las variables más importantes al igual que notaciones sobre el uso de cada variable que parezca ambigua o simplemente no este clara, indentación, el uso de mayúsculas en palabras compuestas. (Segovia, 2014) Sin embargo para el cogido Ginga que el programa genera se optó por crear algunos parámetros y reglas de nomenclatura, para así facilitar el uso del programa y evitar errores.

Como todo código en desarrollo Ginga no es perfecto, es un lenguaje que crece y se mejora todos los días con el estudio y participación de varias universidades y sociedades interesadas en la perfección del mismo. A continuación se listan algunas de las limitaciones propias del código Ginga.

- **Restricción en el uso de memoria**, como ya se mencionó con anterioridad, la capacidad de procesamiento y memoria es muy limitada, lo cual puede causar que las aplicaciones colapsen, para eso se debe contemplar el peso y cantidad de assets que serán usados en la aplicación.

- **Limitación en el espaciado en menús,** aunque parezca extraño Ginga no reconoce a objetos de un mismo menú que estén a más de un 12% de tamaño de pantalla de distancia el uno del otro.

- **Inhabilidad para reproducir ciertos formatos de video,** si bien es capaz de reproducir los formatos más comunes como mp3, mp4, avi. Tiene problemas reconociendo otros tipos de formato más específicos como wmv, mov, flv.

- **Notable carencia de documentación y archivos en español e inglés,** debido al origen brasileño del sistema, es increíblemente difícil hallar información en cualquier otro idioma que no sea portugués, complicando así la personalización o mejora del mismo.

- **Limitación en la detección del nombres que contengan espacios en blanco o ciertos caracteres especiales,** como muchos otros lenguajes de programación Ginga tiene ciertas falencias al tratar de detectar elementos con espacios en el nombre, como por ejemplo: **Ejemplo guardar.avi.**

- **La aplicación necesita estar empacada con las imágenes y videos a utilizar, además de la biblioteca Ginga de funciones,** como la mayoría de desarrolladores que trabajan en Ginga conocen, es necesario empacar la aplicación con su librería de imágenes, y librerías para que funcione, y ya dependerá de la empresa como decidan hacerlo.

3.1.1.2.1. Limitaciones/estandarts Mushu

Con esa información y tomando en consideración la experiencia personal en lo que refiere a programación, entornos de trabajo mixtos y el nivel de conocimiento necesario para hacer ciertos cambios constantemente, se elaboró un set de reglas simples y comprensibles para asegurar una experiencia satisfactoria del programa.

- **Todas las imágenes utilizadas deben ser de extensión .PNG,** la primera consideración para este estándar es el hecho de que simplemente tienen mayor calidad y es visualmente agradable al ser usada en varios tamaños, además para una facilidad en el ámbito de diseño, los archivos png tiene capacidades de traslucidez lo que abre un espectro de posibilidades al usuario que de otra manera no es posible, con todo eso en cuenta la aplicación Mushu genera por default el código buscando por archivos png.

- **Todos los assets se deberán llamar según el estándar establecido:**

    - Para los botones básicos que representan el contenido el formato es: nombre de la aplicación + color en inglés (yellow/blue/green) + siglas 'btn' **ej: Noticieroyellowbtn.**

    - Para assets externos (fuera de un menú) el formato es: nombre de la aplicación + palabra 'Asset' + primera letra mayúscula del nombre del color en inglés (yellow=Y/blue=B/green=G) + número del asset **ej: NoticieroAssetG1**

    - **Para los botones de menú el formato es:** primera letra mayúscula del nombre del color en inglés (yellow=Y/blue=B/green=G) + palabra 'Menu' + palabra 'Item' + número de botón **ej: GMenuItem1**.

    - Para los assets internos (provenientes de botones de menú creados) el formato es: nombre de la aplicación + palabra 'Asset' + primera letra mayúscula del nombre del color en inglés (yellow=Y/blue=B/green=G) + número del asset + palabra 'Menu' + número del botón **ej: 1AssetG1Menu1 o 1AssetG2Menu1 o 1AssetG1Menu2.**

- El video a utilizar deberá estar en formato mp4 y poseer la siguiente nomenclatura: Nombre de la aplicación + Vid1 **ej: NoticieroVid1.**

- **Dentro del código Ginga generado, absolutamente todo en lo que respecta a tamaños y espaciado, será manejado en porcentaje,** esto se debe a una simple razón, la aplicación creada debe ser compatible con varios tamaños y tipos de televisores, lo cual haría el uso de medidas por pixeles increíblemente inexactas y obsoletas ya que eso requiere que todos los usuarios posean las mismas medidas en sus equipos, la intención es crear Responsive Designs (Peterson, 2014) capaces de satisfacer las necesidades del público Ecuatoriano.

- **Para las mediciones internas se optó por mediciones según "left" y "top",** en Ginga no existe un método de ubicación basado en x/y en su lugar se utilizan 4 posibles medidas, de las cuales por convención, se toma únicamente dos, las opciones disponibles son: "top", "left","right","bottom". Cada una de ellas representa un lado de la pantalla y dependiendo de su valor, que tan alejado esta X objeto de tal lado de la pantalla.

Figura #21 Estándar - Navegación de pantalla

Elaborado por: Juan Fernando Baird

- **Se puede crear un máximo de 9 botones por cada menú creado,** debido a restricciones de tamaño propias de Ginga y para evitar sobrecargar la aplicación, cada ítem de menú ocupa un 10% de la pantalla más el 10% que ocupan los iconos que representan los colores se tiene un 90% de la pantalla disponible para uso del usuario.

Figura #22 Estándar - Limitaciones de menú

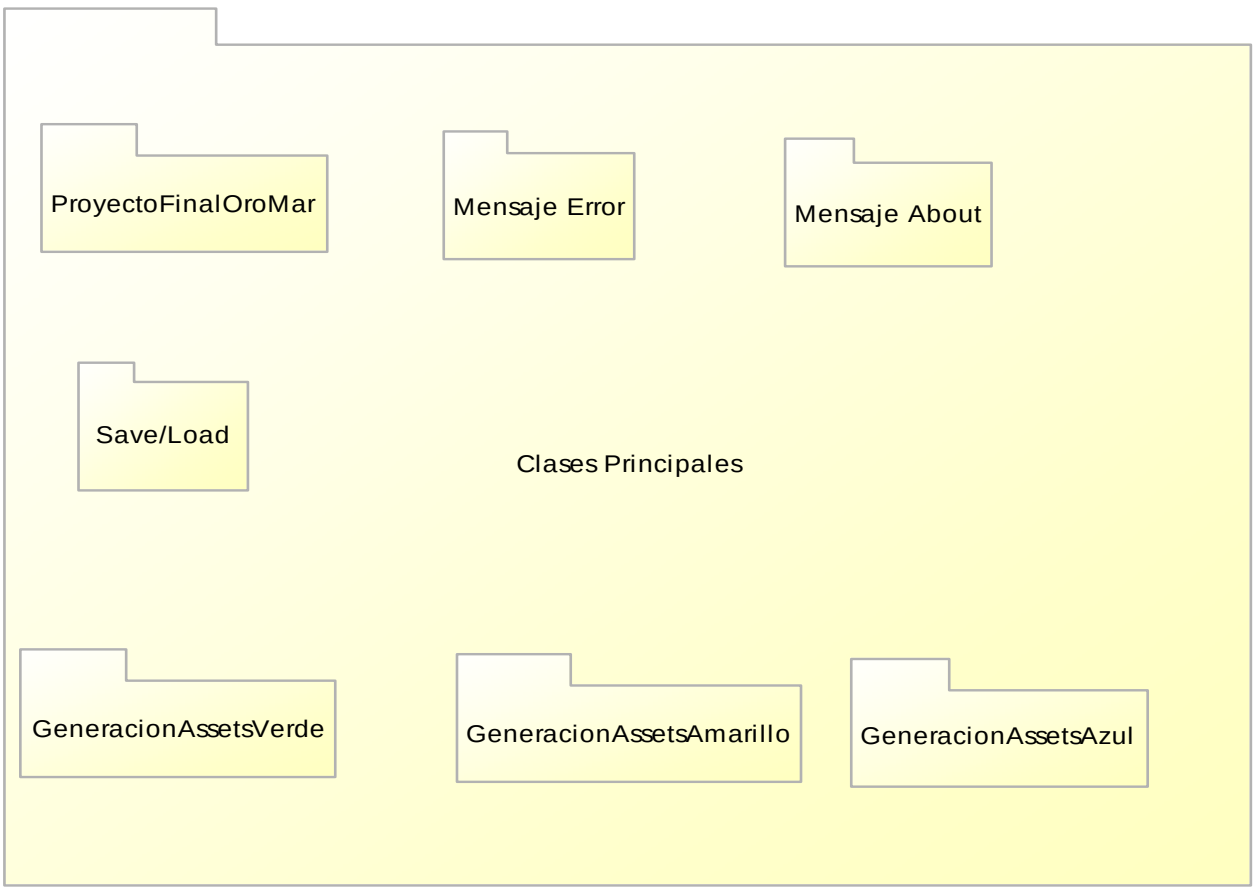
Elaborado por: Juan Fernando Baird

3.1.2 Diagramas Finales

3.1.2.1 Diagramas de Paquetes

Figura #23 Paquete de Clases

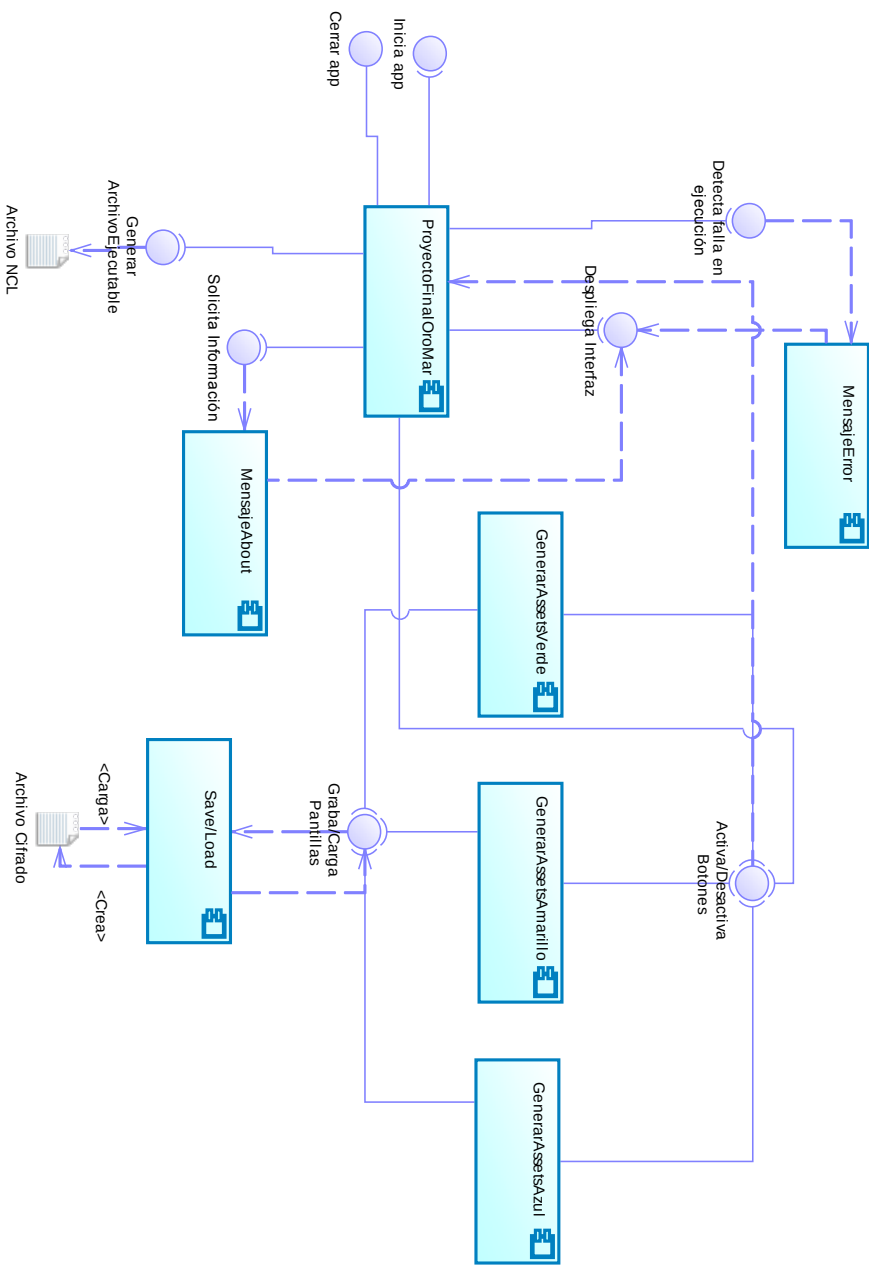
Elaborado por: Juan Fernando Baird



3.1.2.2 Diagramas de Componentes

Figura #24 Diagrama de Componentes

Elaborado por: Juan Fernando Baird



3.1.2.3 Descripción de Componentes

- **ProyectoFinalOroMar:** es el componente principal y actúa como escenario central para las interacciones del usuario con la aplicación, maneja las funciones de inicio y fin de la herramienta unificando la información del resto de componentes para crear la aplicación.

- **GeneracionAssetsAmarillo, GeneracionAssetsVerde y GeneracionAssetsAzul:** son componentes que representan a cada una de las clases hermanas encargadas del manejo de los formularios independientes por cada botón programable, cada una controla su propio set de variables y funciones.

- **guardar/abrir:** este componente se encarga de la generación y lectura de los archivos .txt cifrados usando la información provista por cualquiera de las clases de Generacion de Assets.

- **MensajeError:** se encarga de la ventana de error de la aplicación y puede ser activada por el componente principal si en algún momento la ejecución manda una señal errónea.

- **MensajeAbout:** contiene una ventana informativa sobre la aplicación y como conseguir el manual de usuario en caso de necesitarlo.

3.1.2.4 Diagramas de Despliegue

En este caso en particular no hay necesidad para un diagrama de despliegue ya que la aplicación corre independiente de internet o servidores, es completamente autónoma y todo lo que necesita es encontrarse en un equipo capaz de interpretar java.

3.3. Implementación

3.3.1 Implementación del Software

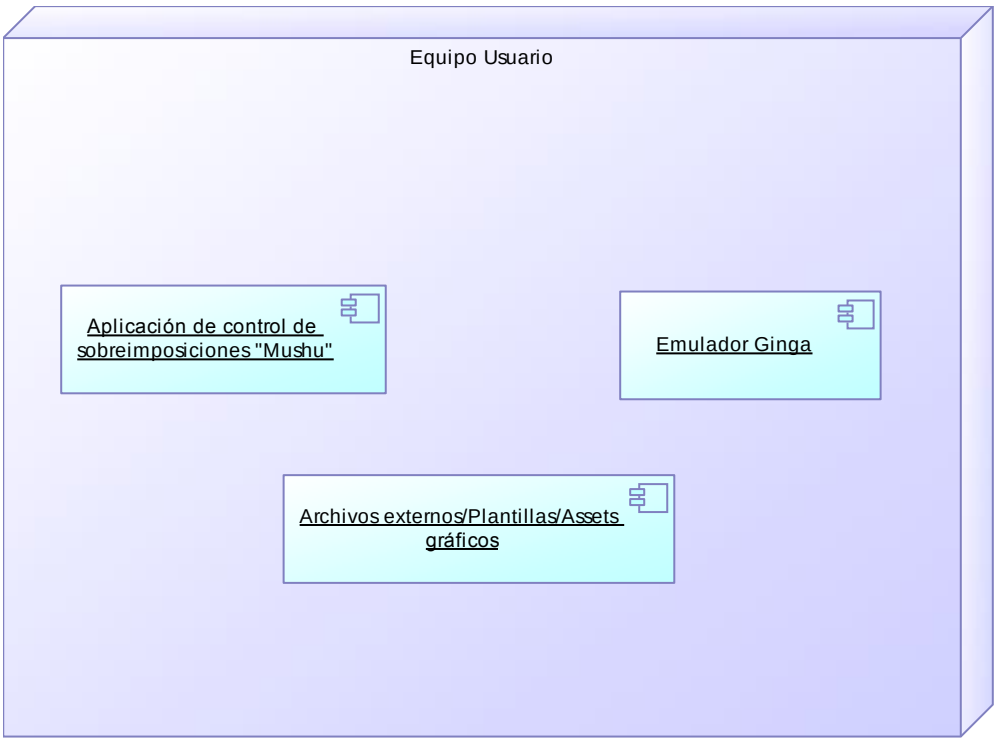
Para la implementación del software se requirió una computadora común de escritorio con la aplicación, el emulador de Ginga y varias imágenes o placeholders para las pruebas del mismo. Es importante notar que la preparación previa es lo que permite un desarrollo fluido y cómodo, ya que si el usuario no posee alguna imagen tal y como indica el manual, la aplicación Ginga no se ejecutará apropiadamente y esto puede llevar al usuario a creer que el código generado esta defectuoso. Dado a la necesidad del emulador de Ginga para las pruebas del software se aconseja utilizar la aplicación en Windows, para un óptimo rendimiento de la misma.

Como requerimientos mínimos para utilizar la aplicación de manera adecuada se recomienda:

- Sistema operativo Windows 7 o superior.
- 2Gb de ram.
- Procesador Intel core i3.
- Por lo menos 5Gb de espacio libre en el disco para poder almacenar los assets y videos necesarios para pruebas.

Figura #25 Diagrama de Arquitectura

Elaborado por: Juan Fernando Baird



3.3.2 Pruebas del Software

Dada la naturaleza del método de programación y el hecho de ser la primera aplicación de este tipo en desarrollo, las pruebas fueron constantes durante cada etapa del desarrollo de la aplicación, cada cambio y mejora del código fue revisado varias veces individualmente mediante pruebas de caja blanca, analizando cada variable y función antes de introducirla al código y una vez ingresada la nueva función se ejecutaron pruebas de caja negra para comprobar el output y comparar resultados.

El mayor reto a superar fue la creación del código Ginga de manera ordenada, para ello se estudió un curso de programación Java y al notar las similitudes con XML se

adaptó el código para crear etiquetas especiales en un orden predefinido el cual da como resultado la aplicación ejecutable Ginga.

En segundo lugar con lo referente a complejidad se encuentra el almacenamiento de datos dinámicos, ya que se permite tanta libertad al usuario en cuestiones cuantitativas era importante una arquitectura que soporte el cambio constante y ordenado de datos para almacenarlos y leerlos sin cargar información innecesaria pero si todo dato sensible del usuario. Finalmente se optó por listas dinámicas las cuales se crean, llenan y borran constantemente durante la ejecución del programa.

Una vez resueltos ambos problemas el resto de bugs y glitches se trataban de problemas gráficos en lo que concierne en forma, reacción o diseño de la aplicación, estos fueron resueltos como tareas menores mientras se implementaba el código de control de sobreimposiciones. Cabe mencionar que debido a la falta de información profesional en el tema de televisión digital por parte de la empresa y la comunidad en general, se usó como base de ejemplo la programación de TvCable, Claro y DirectTv ya que en esencia son los pioneros del cambio al que se va a someter la televisión, por lo mismo la aplicación se mantiene al margen de lo que necesitaría una empresa de transmisión como lo es OroMarTv y se perfeccionó en lo mismo, en lugar de ampliar con módulos o funcionalidades que podrían disminuir la cantidad de personas capaces de recibirla, o simplemente tornarse inútil por la falta de capacidad de procesamiento de los decodificadores implementados por el estado o público.

A continuación se presenta una tabla con los principales módulos, sus funciones y los mayores problemas encontrados en los mismos durante el desarrollo, estos problemas son solo una pequeña parte de los encontrados pero representan a los mayores obstáculos enfrentados durante la creación del programa.

Tabla # 8 Casos de prueba: Mushu

Elaborado por: Juan Fernando Baird

N°	**Módulo**	**Descripción**	**Datos**	**Problemas encontrados**	**Resultado esperado**	**Resultado Final**
1	Creación Código Ginga.	Completar correctamente el formulario y seleccionar la función Crear .NCL	Nombre app Coordenadas	• Orden de assets erróneo. • Fallas de nomenclatura. • Dilema con la creación de código anidado. • Fallas de coordinación de botones. • Perdida de datos.	Un documento con código Ginga reutilizable que permita la creación de ejecutables Ginga.	Un documento .NCL ejecutable capaz de reconocer sus atributos y diferenciarlos de atributos de otros programas Ginga.
2	Almacenamiento de información ingresada para reutilización.	Crear un archivo que contenga la información de una aplicación creada para su futuro uso en otra aplicación distinta.	Nombre app Coordenadas	• Formato de guardado. • Extensión del archivo creado. • Método de grabación. • Información errónea almacenada. • Perdida de información. • Manejo de porcentajes.	Crear una plantilla reutilizable, disponible para el usuario que mantenga la fidelidad de la información intacta.	Archivos cifrados intercambiables los cuales representan cada botón guardado, completamente editables y reutilizables independientes de la aplicación ejecutable.

N°	**Módulo**	**Descripción**	**Datos**	**Problemas encontrados**	**Resultado esperado**	**Resultado Final**
3	Reutilización de proyectos antiguos.	Usar un archivo previo para recupera la información previamente ingresada.	Plantilla	• Formato para almacenar la información. • Orden de lectura de la información. • Capacidad de análisis de la información ingresada. • Capacidad de proteger la información..	Utilizar un proyecto antiguo como herramienta o plantilla para crear un programa nuevo.	Recuperación de la información completa recuperando el estado de la aplicación en el momento de ser guardada.

3.3.2 Capacitación

Para la capacitación del personal de la empresa se solicitó una reunión formal con los posibles usuarios y directores creativos que en la actualidad planean empezar a experimentar con Ginga para prepararse para el apagón analógico, pero al estar en un estado de preparación la empresa carece de un departamento de TDT, por lo cual futuras capacitaciones pueden ser necesarias.

Además se les indicó las bases bajo las cuales fue creada la aplicación y los objetivos de la misma, se enfatizó en los límites y estándares a los cuales la empresa deberá someterse para un uso efectivo de la misma, y se realizó una serie de ejercicios prácticos para demostrar a los usuarios la compleja simplicidad de la misma. Sin mayor problema los empleados de la empresa lograron manejar la aplicación y sugirieron algunas mejoras, las cuales se implementó en el programa rápidamente. El manual de usuario entregado a la empresa puede ser observado en el Anexo A, es una copia digital que contiene toda la información de la aplicación y cómo manejar la interfaz.

Para los ejercicios realizados se utilizó los ejemplos del manual de usuario, además de una demostración básica con Assets temporales para demostrar cosas como distorsión, posición, y funcionalidad de los assets colocados. Además se mostró un video creado por el creador de la aplicación demostrando las posibles mejoras que se pueden aplicar a la aplicación si en algún futuro la empresa decide, ampliar el rango de sus aplicaciones interactivas.

3.3.3 Explotación, aprobación y beneficios

Tras la capacitación y mejora del programa los usuarios tanto diseñadores como programadores vieron los posibles usos de la aplicación, hubo una discusión sobre funciones y límites probados de las mismas, las cuales fueron conversadas y explicadas de modo que ellos estén en capacidad de considerarlas y jugar con las mismas. Los beneficios de la aplicación fueron fácilmente notados ya que la reducción en tiempo de planeación, horas de trabajo y ejecución fue notablemente corto, en comparación a lo que tomaría seguir el clásico modelo de desarrollo de software, además de la cantidad de personal necesaria es mínima ahora que una sola persona puede hacer una o varias aplicaciones ejecutables en cuestión de minutos lo que antes tomaba horas incluso días de trabajo si se toman en cuenta las pruebas y modificaciones, ya que cada elemento declarado en un documento Ginga se declara mínimo 3 veces, y requiere precisión quirúrgica para evitar desencadenar errores en todo el documento, tarea que ahora se puede realizar simplemente cambiando un número en un formulario, el personal de la empresa puede enfocar sus esfuerzos y recursos en otras tareas más orientadas a sus especialidades.

3.3.4 Mantenimiento

3.3.4.1 Preventivo

Durante el desarrollo del sistema se realizaron pruebas de caja blanca y de caja negra para poder garantizar el normal funcionamiento del sistema y evitar posibles errores. Como mantenimiento preventivo se recomienda guardar las plantillas creadas antes de crear el ejecutable, mantener respaldos de las varias plantillas creadas hasta encontrar la combinación deseada y evitar manipular los archivos cifrados.

Además es importante tener una carpeta con los archivos externos necesarios para que funcione la aplicación, muchas veces se confunde con error de programación a lo que en realidad es una falla de planeación o de nombramiento de assets, Ginga es muy

sensible y requiere de exactitud en la nomenclatura de los objetos a ser utilizados por el programa, la falla en cumplir con ese prerrequisito puede causar que la aplicación Ginga se congele al ser utilizada.

3.3.4.1 Correctivo

En caso de darse un error, se recomienda revisar en primer lugar las dimensiones ingresadas en la plantilla, asegurarse de que no existan espacios vacíos entre números ni en el nombre de la aplicación. De estar correctos los parámetros se sugiere revisar la carpeta con Assets, puede que un espacio en blanco este causando que el programa no reconozca el nombre del objeto o en el peor de los casos que dicho objeto no exista, causando asi que el programa se congele durante la ejecución.

CAPÍTULO IV

DISCUSIÓN

4.1. Conclusiones

- Se cumplió con el objetivo principal y se lo llevo más allá obteniendo no solo código sino un ejecutable listo para su uso sin mayor necesidad de procesar el código generado, como era el objetivo inicial, además de incluir capacidades de carga y almacenamiento de datos.

- El programa efectivamente reduce costos de producción y tiempos de desarrollo. Lo que tomaba horas de trabajo con el código manualmente, ahora son minutos utilizando la aplicación, lo que tomaba días ahora son horas y con la seguridad de que el código generado es completamente funcional.

- Se realizó un programa que fue moldeado durante cada paso del desarrollo ya que al carecer de experiencia en el tema, por parte del desarrollador y la empresa, la mayoría de requerimientos y funciones necesarias fueron sugeridas por el programador y aprobadas por un representante de OroMarTv

- La interfaz de usuario se creó efectivamente evitando en lo posible procesos complejos para el usuario, de forma que el mismo no tenga que preocuparse en mayor cosa que ingresar datos simples y presionar uno o dos botones.

- La aplicación no ocupa mayor cantidad de recursos de máquina ya que se la elaboró de tal manera que el código esta pre generado internamente y simplemente procesa una cantidad mínima de datos introducidos por el usuario.

- El manual de usuario fue creado de modo de ser fácil de entender y revisar, además de ser recibido por usuarios de prueba, los cuales no poseen conocimientos de programación previos.

- El código ejecutable resulto completamente independiente de otras plataformas, simplemente requiere el emulador y la carpeta con assets para funcionar correctamente.

- El producto fue probado y testeado con empleados y ejecutivos de la empresa OroMarTv al igual que posibles usuarios externos como colegas de la universidad.

4.2. Recomendaciones

- Asegurarse de leer el manual de usuario entregado a la empresa y ubicado en el Anexo A del documento de tesis, especialmente las limitaciones Ginga y los estándares del programa establecidos por el desarrollador del mismo.

- Considerar las limitaciones de los codificadores a utilizar y planear de acuerdo a eso cuantos Assets se pueden utilizar en el programa.

- A futuro sería posible implementar mejoras al programa bajo dos condiciones, que los equipos disponibles al público sean medianamente capaces de almacenar memoria o procesar datos y generar funciones propias en Ginga para aumentar el espectro de funciones posibles.

- Mantener en mente que todos los números ingresados al programa representan porcentajes de pantalla y planear los assets de acuerdo a eso.

- Se recomienda aprender algo de código Ginga ya que de esta forma se puede combinar la herramienta con código privado y generar aplicaciones impresionantes.

REFERENCIAS BIBLIOGRÁFICAS

- Canos, J., Letelier, P. y Penades, M., "SCRUM" Metodologías Ágiles en el Desarrollo de Software. Universidad Politécnica de Valencia. P 7
- Pressman, R. (2010). *Ingeniería del Software. Un enfoque práctico* (7ma ed.). México D.F: Mc Graw Hill.
- Roel, M., (2006) "EL RETO DE LA TELEVISIÓN: EL "APAGÓN ANALÓGICO" Y LA CONSOLIDACIÓN DE LO DIGITAL" Revista Latinoamericana de Comunicación CHASQUI Pag. 4
- Sommerville, I., (2011) "Agile software development" Software Engineering Addison-Wesley P. 58
- Segovia, R. (2014). Java Fundamentals Programing. En R. Segovia, Java Fundamentals Programing (págs. 23-44). Quito: Centro de Educación Continua.
- Peterson, C. (2014). Learning Responsive Web Desing. En C. Peterson, Learning Responsive Web Desing. Canada: Cambridge.

WEBGRAFIA

- Bachetta, L., (2010) "SATVD-T Sistema argentino de televisión digital terrestre" http://www.slideshare.net/luisbacchetta/ginga-ncllua
- (2014, 03). Definición sobreimposiciones. dspace.espol.edu.ec Recuperado 03 de http://www.dspace.espol.edu.ec/bitstream/123456789/24281/1/ART_RESUMEN_CICYT_CLUB%20DE%20PAPEL.pdf
- (2014, 03). Estándares de TDT. redalyc.org Recuperado 03 de http://www.redalyc.org/pdf/160/16009410.pdf

- (2014, 01). Ginga. repositorio.espe.edu.ec. Recuperado 01, 2014,de http://repositorio.espe.edu.ec/bitstream/21000/4748/2/T-ESPE-032865-A.pdf
- (2014, 01). GingaJ. comunidadgingaec.blogspot.com. Recuperado 01, 2014,de http://comunidadgingaec.blogspot.com/2011/06/middleware-ginga.html
- (2014, 01). Illustrador. es.wikipedia.org. Recuperado 01, 2014,de http://es.wikipedia.org/wiki/Adobe_Illustrator
- Molina, R (2014, 01). Que Es Eclipse?. BuenasTareas.com. Recuperado 01, 2014,de http://www.buenastareas.com/ensayos/Que-Es-Eclipse/179510.html
- (2014, 03). Proyectos en proceso de TDT. espetv.espe.edu.ec Recuperado 03 dehttp://www.espetv.espe.edu.ec/webespetv/Proyectos.html
- Gamarro, U. (2013). *Ultimas noticias*. Recuperado el 15 de 01 de 2015, de Ultimas noticias: http://www.prensalibre.com/economia/Guatemala-elige-modelo_0_929907014.html
- Eamon, R. (18 de diciembre de 2008). *waybackmachine*. Recuperado el 19 de enero de 2015, de waybackmachine: http://web.archive.org/web/20111004061330/http://www.digitaltelevision.ie/National+DTT/Digital+Switchover.htm
- Ana, C. (14 de junio de 2014). El Telegrafo. Recuperado el 02 de enero de 2015, de El Telegrafo: http://www.telegrafo.com.ec/economia/item/apagon-analogico-iniciara-el-31-de-diciembre-2016.html

ANEXOS

ANEXO A – MANUAL DE USUARIO

Manual de usuario

Si bien para el correcto uso de esta herramienta no es necesario conocer de Ginga, o de programacion en si, Mushu posee ciertas limitaciones intrínsicas que provienen del hecho de usar Ginga y simples estandarts creados para asi facilitar el uso y estudio del codigo generado. Estas limitaciones añadidas no implican que el codigo no pueda ser modificado manualmente he inclusive mejorado, pero para ello se recomienda un conocimiento moderado de GingaNCL ya que de otra forma lo unico que se conseguiria es dañar el codigo pregenerado.

Limitaciones GingaNCL

Como todo código en desarrollo Ginga no es perfecto, es un lenguaje que crece y se mejora todos los días con el estudio y participación de varias universidades y sociedades interesadas en la perfección del mismo. A continuación se listan algunas de las limitaciones propias del código Ginga.

- **Restricción en el uso de memoria**, como ya se mencionó con anterioridad, la capacidad de procesamiento y memoria es muy limitad, lo cual puede causar que las aplicaciones colapsen, para eso se debe contemplar el peso y cantidad de assets que serán usados en la aplicación.

- **Limitación en el espaciado en menús,** aunque parezca extraño Ginga no reconoce a objetos de un mismo menú que estén a más de un 12% de tamaño de pantalla de distancia el uno del otro

- **Inhabilidad para reproducir ciertos formatos de video,** si bien es capaz de reproducir los formatos más comunes como mp3, mp4, avi. Tiene problemas reconociendo otros tipos de formato más específicos como wmv, mov, flv

- **Notable carencia de documentación y archivos en español e inglés,** debido al origen brasileño del sistema, es increíblemente difícil hallar información en cualquier otro idioma que no sea portugués, complicando así la personalización o mejora del mismo.

- **Limitación en la detección del nombres que contengan espacios en blanco o ciertos caracteres especiales,** como muchos otros lenguajes de programación Ginga tiene ciertas falencias al tratar de detectar elementos con espacios en el nombre, como por ejemplo: **Ejemplo guardar.avi**

- **La aplicación necesita estar empacada con las imágenes y videos a utilizar, además de la biblioteca Ginga de funciones,** como la mayoría de desarrolladores que trabajan en Ginga conocen, es necesario empacar la aplicación con su librería de imágenes, y librerías para que funcione, y ya dependerá de la empresa como decidan hacerlo.

Limitaciones/estandarts Mushu

Con esa información y tomando en consideración la experiencia personal en lo que refiere a programación, entornos de trabajo mixtos y el nivel de expertica necesario para

hacer ciertos cambios constantemente, se elaboró un set de reglas simples y comprensibles para asegurar una experiencia satisfactoria del programa.

- **Todas las imágenes utilizadas deben ser de extensión .PNG,** la primera consideración para este estándar es el hecho de que simplemente tienen mayor calidad y es visualmente agradable al ser usada en varios tamaños, además para una facilidad en el ámbito de diseño, los archivos png tiene capacidades de traslucidez lo que abre un espectro de posibilidades al usuario que de otra manera no es posible, con todo eso en cuenta la aplicación Mushu genera por default el código buscando por archivos png.

- **Todos los assets se deberán llamar según el estándar establecido:**

    - Para los botones básicos que representan el contenido el formato es: nombre de la aplicación + color en inglés (yellow/blue/green) + siglas 'btn' **ej: Noticieroyellowbtn.**

    - Para assets externos (fuera de un menú) el formato es: nombre de la aplicación + palabra 'Asset' + primera letra mayúscula del nombre del color en inglés (yellow=Y/blue=B/green=G) + número del asset **ej: NoticieroAssetG1.**

    - **Para los botones de menú el formato es:** primera letra mayúscula del nombre del color en inglés (yellow=Y/blue=B/green=G) + palabra 'Menu' + palabra 'Item' + número de botón **ej: GMenuItem1**.

    - Para los assets internos (provenientes de botones de menú creados) el formato es: nombre de la aplicación + palabra 'Asset' + primera letra

mayúscula del nombre del color en inglés (yellow=Y/blue=B/green=G) + número del asset + palabra 'Menu' + número del botón **ej: 1AssetG1Menu1 o 1AssetG2Menu1 o 1AssetG1Menu2.**

- El video a utilizar deberá estar en formato mp4 y poseer la siguiente nomenclatura : Nombre de la aplicación + Vid1 **ej: NoticieroVid1**

- **Dentro del código Ginga generado, absolutamente todo en lo que respecta a tamaños y espaciado, será manejado en porcentaje,** esto se debe a una simple razón, la aplicación creada debe ser compatible con varios tamaños y tipos de televisores, lo cual haría el uso de medidas por pixeles increíblemente inexactas y obsoletas ya que eso requiere que todos los usuarios posean las mismas medidas en sus equipos.

- **Para las mediciones internas se optó por mediciones según "left" y "top",** en Ginga no existe un método de ubicación basado en x/y en su lugar se utilizan 4 posibles medidas, de las cuales por convención, se toma únicamente dos, las opciones disponibles son: "top", "left","right","bottom". Cada una de ellas representa un lado de la pantalla y dependiendo de su valor, que tan alejado esta X objeto de tal lado de la pantalla

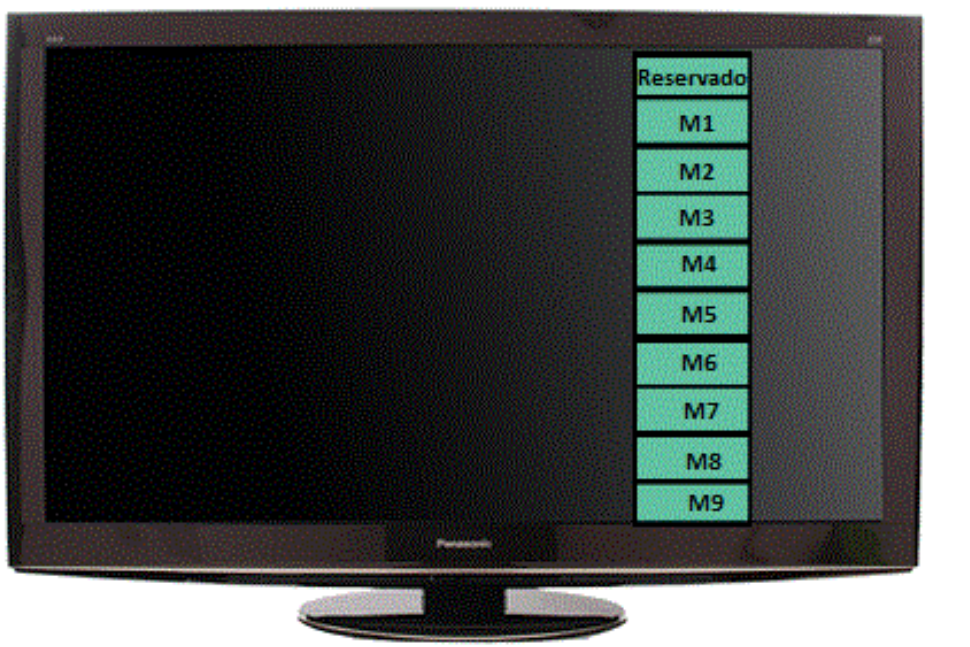
Desarrollo- Navegación de pantalla

Elaborado por: Juan Fernando Baird

- **Se puede crear un máximo de 9 botones por cada menú creado,** debido a restricciones de tamaño propias de Ginga y para evitar sobrecargar la aplicación, cada ítem de menú ocupa un 10% de la pantalla más el 10% que ocupan los iconos que representan los colores se tiene un 90% de la pantalla disponible para uso del usuario

Desarrollo- Limitaciones de menú

Elaborado por: Juan Fernando Baird



Uso de la aplicación

A continuación se detalla el uso apropiado de la aplicación con ejemplos e imágenes que facilitarán la comprensión del sistema, con un paso a paso de cada herramienta disponible, en esta guía se demostraran algunas formas de combinar las opciones disponibles de la manera más básica y entendible, un usuario avanzado puede improvisar nuevas formas de manejar la herramienta, al igual que hacer uso de conocimientos propios para mejorar el código creado en la misma siempre tomando en cuenta las posibles limitaciones a las cuales se atiene al crear aplicaciones más pesadas.

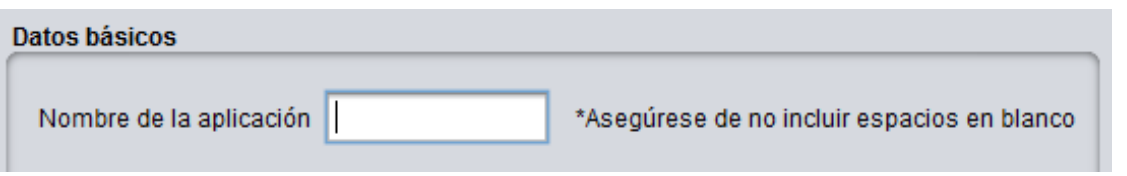
Mushu fue creado con la intención de permitir mayor flexibilidad y rapidez durante el diseño y desarrollo de aplicaciones Ginga, tomando muy en cuenta las capacidades y necesidades de los distintos tipos de usuarios que podrían necesitar usarla, principalmente diseñadores multimedia y programadores.

Arquitectura de la aplicación Mushu

El programa se divide en dos paneles visibles en todo momento, creados de tal forma que permitan una navegación intuitiva y un uso rápido, de modo que no se necesiten muchas configuraciones para obtener lo que uno desea. El panel superior contiene el nombre de la aplicación, el cual el usuario insertara por su cuenta, y tomando en consideración que Ginga no acepta espacios en blanco, se informa al usuario que recuerde no incluir espacios en blanco, no se prohíbe el uso de los mismos ya que se reconoce que el usuario puede estar diseñando una plantilla para utilizar en algún momento y por lo mismo tiene derecho a utilizar cualquier nomenclatura. La información ingresada en esta sección solo será utilizada al momento de ejecución de las funciones principales como Crear NCL y grabar.

Arquitectura- Ingreso Nombre

Elaborado por: Juan Fernando Baird

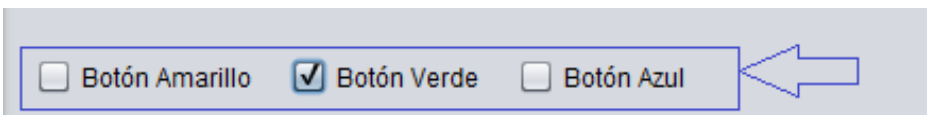


Además el panel superior contiene 3 checkboxes individuales, los cuales representan un botón de cada color disponible en los controles de TDT (amarillo, verde y

azul). El botón rojo se reservó para la cancelación de la última acción realizada ya que de esta forma la navegación de la aplicación Ginga es mucho más intuitiva. Cada selector activa o desactiva el menú selector de opción e indica al programa que esa información ingresada debe ser ignorada, esto no implica que no haya sido llenada ni la borra, solo que para la aplicación generada no necesita ser considerada, de esta forma se puede crear varios programas con la misma información cambiando el nombre únicamente.

Arquitectura- Checkboxes

Elaborado por: Juan Fernando Baird



Como tercer elemento en el panel principal se ubica un tabbed pane que a su vez contiene tres panales distintos, los cuales se especializan en cada uno de los colores disponibles, como ya se mencionó con anterioridad, esto permite una programación paralela de los botones evitando confusiones y facilitando el ingreso de dicha información. Cada panel interno contendrá la misma información inicial pero el programa permite tener distintos botones e información, de modo que el usuario no debe regirse por una plantilla rígida al momento del desarrollo sino, invita a la creatividad y lo hace de tal forma que se ve estéticamente agradable al usuario.

Arquitectura- Tabbed panes

Elaborado por: Juan Fernando Baird

En cada panel interno se encuentran los mismos elementos básicos, con los cuales se podrá pedir los elementos necesarios para crear el código a gusto del usuario, en primer lugar se encuentra el menú de selección, este dispone de dos opciones muy distintas que solicitaran información vital de tamaño y posición, para crear los assets en las secciones adecuadas del código.

Arquitectura- Menú de opciones

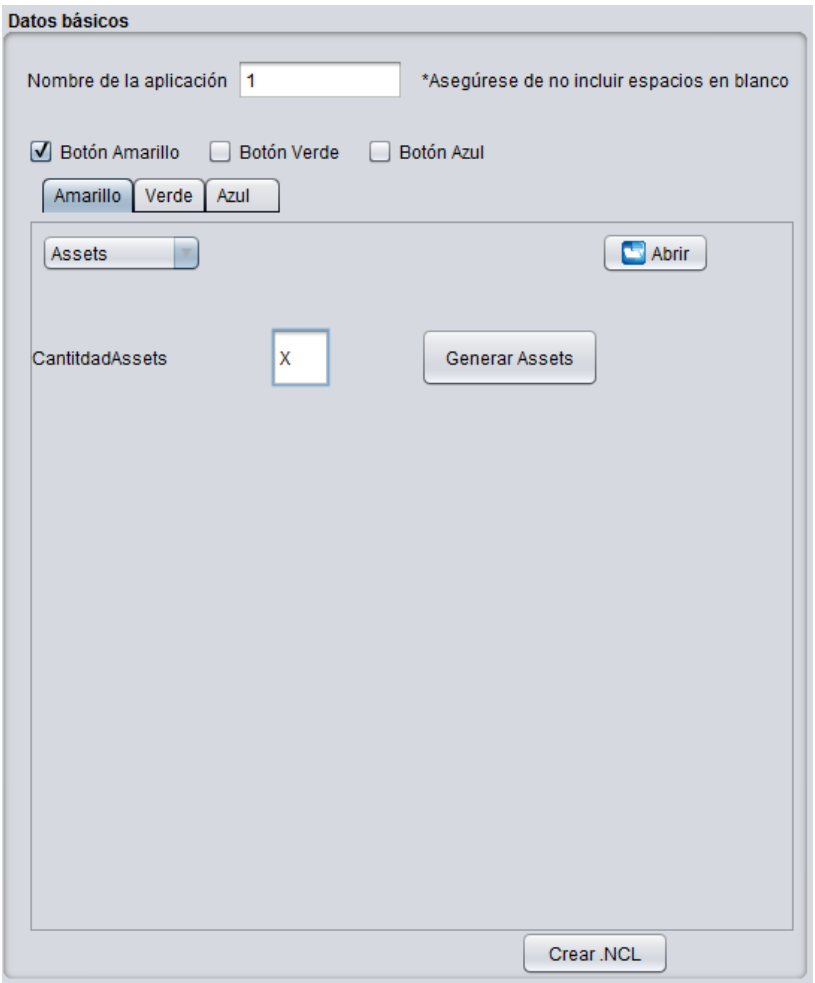
Elaborado por: Juan Fernando Baird

La primera de dichas opciones se llama “Assets”, una palabra muy usada por diseñadores para describir elementos gráficos, esta opción permite al usuario añadir una cantidad n de imágenes de distintos tamaños y en distintas locaciones de la pantalla según sea su gusto, tomando en consideración que muchas veces, no se necesita de mucha información desplegada en la aplicación, pero de esta manera el usuario puede mejorar visualmente la misma, para evitar usar una sola imagen que podría distorsionarse o degradarse en calidad, es preferible agregar varios elementos pequeños en los cuales es

más difícil notar el cambio de pantalla a pantalla. Se sugiere un límite máximo de 10 imágenes dependiendo de la calidad de las mismas para evitar aplicaciones demasiado pesadas para el usuario.

Arquitectura- Opción Assets

Elaborado por: Juan Fernando Baird

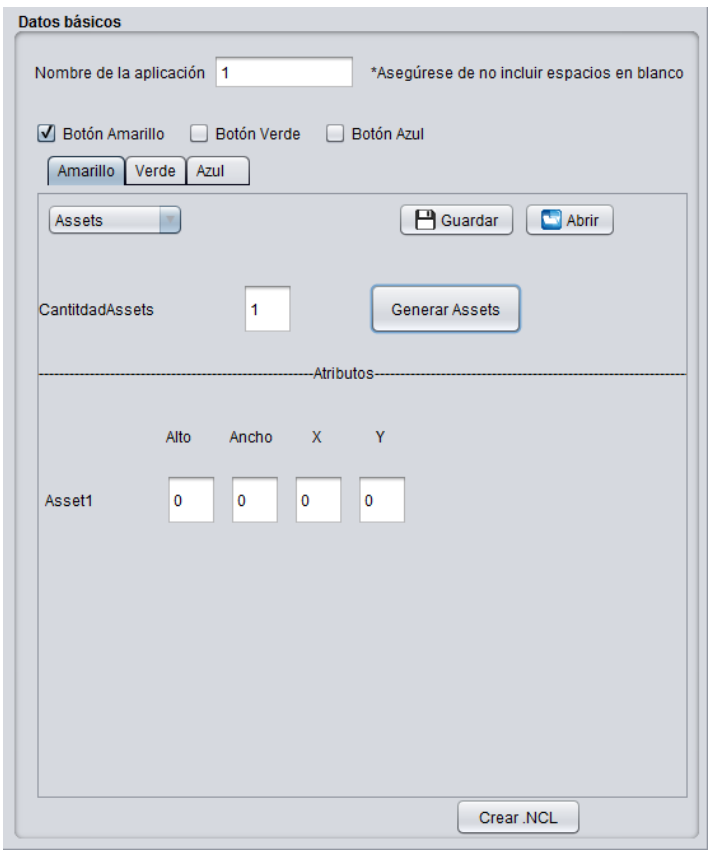


Una vez seleccionada la cantidad se puede presionar el botón “Generar Assets” y la información pasará a una función de evaluación, en la cual el programa revisara y procesará el número indicado para crear la cantidad apropiada de solicitudes, las mismas

que podrán ser incrementadas o reducidas en cualquier momento, simplemente cambiando el numero en la caja de ingreso y presionando nuevamente el botón “Generar Assets”. En caso de ser una opción inválida el programa desplegara una ventana de error de modo que el usuario sepa que ha cometido una falla.

Arquitectura- Opción Assets Activada

Elaborado por: Juan Fernando Baird



En caso de desplegarse la ventanilla de error simplemente se debe cerrar dicha ventana y proceder a cambiar la información errónea, una vez removida la causa, se puede dar click nuevamente en el botón “Generar Assets” y la aplicación continuará con su ejecución normal sin causar mayor problema.

Arquitectura- Opción Assets Error

Elaborado por: Juan Fernando Baird

La segunda opción disponible en el menú de selección es “Menu+Assets”, esta fue diseñada con la intención de permitir al usuario introducir mayor información de forma ordenada en menús interactivos dentro de su aplicación, como valor agregado se añadió la posibilidad de mostrar assets externos al igual que internos, para de esta manera permitirle al usuario crear una experiencia visual más agradable en su aplicación Ginga.

Arquitectura- Opción Menú+Assets

Elaborado por: Juan Fernando Baird

En esta función se requiere un poco más de información que la solicitada anteriormente, en primera instancia solicitará una cantidad X de assets externos, estos assets se encuentran al mismo nivel que el menú de inicio y serán visibles desde la activación del botón padre (amarillo azul o rojo). Estos assets están diseñados para ser decorativos o informativos ya que al navegar el menú se presentaran otras imágenes que en teoría deberían incluir información más detallada y organizada. En segunda instancia el

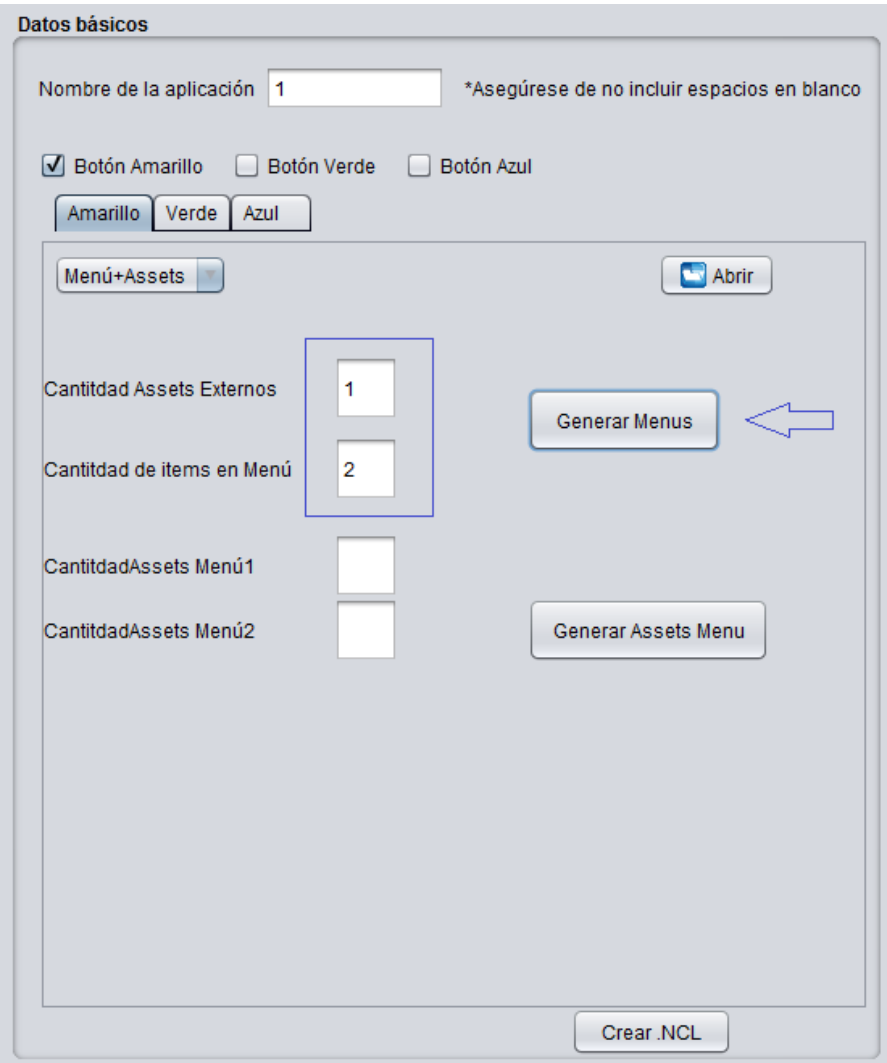
programa pide una cantidad X de ítems de menú, como ya fue mencionado con anterioridad los ítems de menú poseen un tamaño y locación fijos por lo que se recomienda no solicitar más de 9, pero es completamente factible hacerlo, simplemente al final de la creación del código se necesita alterar dichos tamaños y posiblemente locaciones a discreción del usuario.

Una vez ingresada la información requerida, se procede a presionar el botón generar menús, como el nombre lo indica, este botón se encargara de crear los espacios dinámicos para la información de cada botón de menú que se haya indicado, la información de los assets externos no será utilizada sino hasta más adelante, cada botón creado solicitara un numero de assets internos, este número no tiene límite, pero se recomienda no utilizar más de 5 a 10 assets internos por botón, ya que es necesario evaluar el tamaño de la aplicación que estamos creando, y recordar que hay limitaciones para los set-top box disponibles.

En caso de cometer un error al momento de la creación del programa, como la falta de assets o el exceso de los mismos, el programa cuenta con la capacidad de recalcular el número deseado de artículos y eliminar o aumentar n número de assets simplemente cambiando el valor numérico en la caja de texto y presionando el botón “Generar” pertinente al área que se desee cambiar.

Arquitectura- Opción Menú+Assets etapa 1

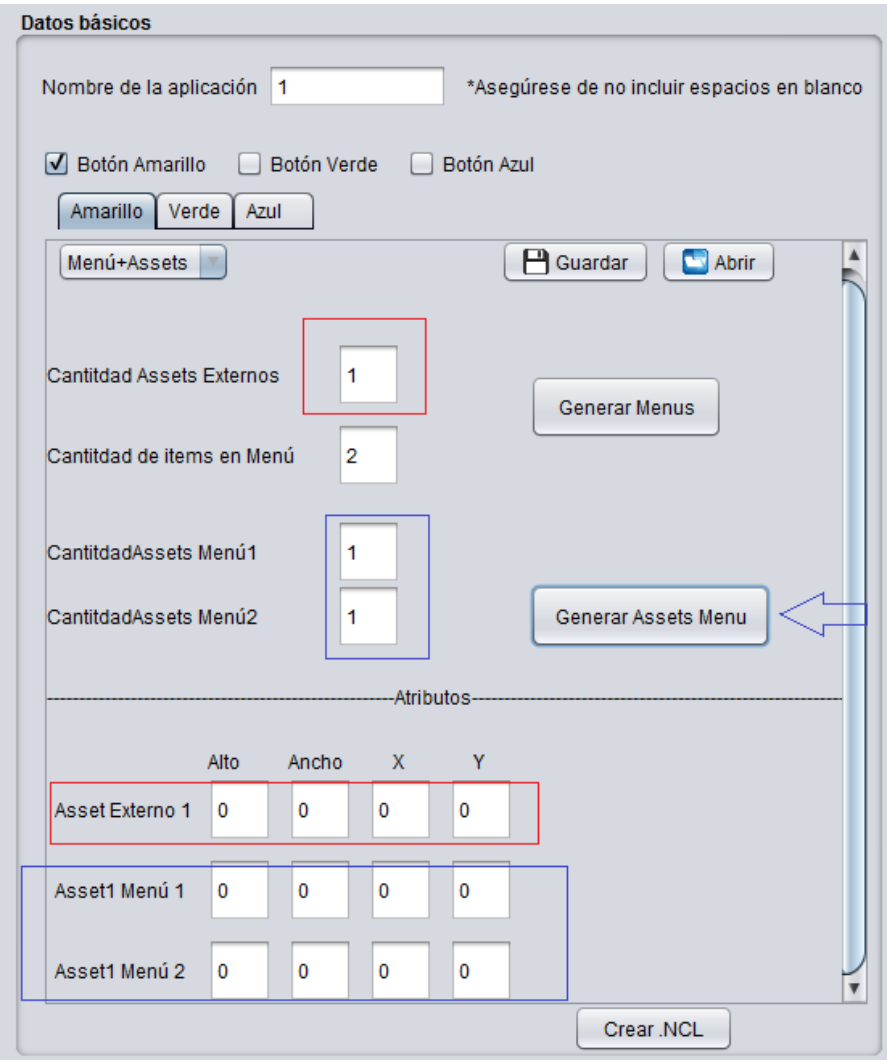
Elaborado por: Juan Fernando Baird



Una vez ingresada la segunda etapa de información se puede presionar el botón "Generar Assets Menú" el cual tomará toda la información introducida y la convertirá apropiadamente en un formulario solicitando los datos de los assets internos y externos por igual, en caso de colorar 0 en la solicitud de assets externos, únicamente pedirá la información de los assets internos, dando así mayor flexibilidad al diseñador o colega trabajando en su programa.

Arquitectura- Opción Menú+Assets etapa 2

Elaborado por: Juan Fernando Baird



Además del menú de opciones, cada panel interno posee un botón de guardar y cargar "guardar" y "abrir" respectivamente, estos botones trabajan individualmente en cada caso, solo tratando con la información del panel al que pertenecen originalmente. Este manejó de la información se creó de esa forma por una razón puramente práctica e innovadora, al igual que muchos programas, originalmente se planeaba guardar toda la información en un solo archivo cifrado el cual el usuario pueda reutilizar su código una y

otra vez, pero dada la naturaleza de la programación en Ginga, y el diseño de Mushu se optó por tres archivos individuales que se almacenen en una carpeta común.

Arquitectura- Botones guardar y abrir

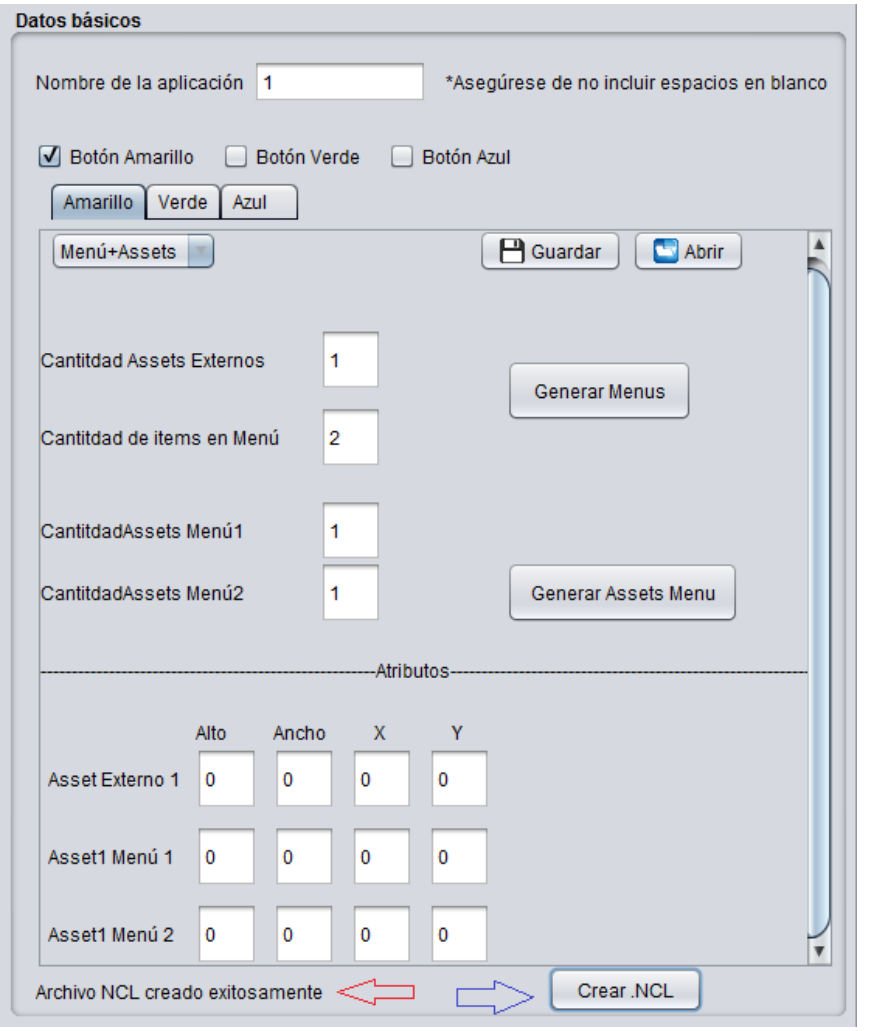
Elaborado por: Juan Fernando Baird

Esto permite al usuario no solo proteger el trabajo realizado, pero además le da la opción de intercambiarlo como desee, de modo que pueda reutilizar plantillas de formas más creativas y flexibles de lo que permitiría un solo archivo estático, la idea es de permitir al usuario usar archivos viejos como piezas intercambiables, incluso si se carga un archivo antiguo, el programa no reconoce el nombre anterior de modo que el usuario puede cambiar el nombre y algunos datos obteniendo así una nueva plantilla lista para usar cuando lo necesite.

Finalmente como último elemento interno del panel estático dispuesto para el usuario se encuentra el botón “Crear .NCL” es ahí donde se encuentra el llamado a la función central del programa. Este botón, a diferencia de lo que se creería comúnmente, se encuentra siempre habilitado para su uso, la razón de aquello es que el programa Mushu está en la capacidad de crear un código básico como esqueleto para una aplicación, el mismo que posee todas las etiquetas y variables que se pueden encontrar en todo programa Ginga. De esa forma el usuario puede crear su aplicación manualmente con los assets, menús, y aplicaciones extras que desee, sin tener que preocuparse de sintaxis y forma en el código inicial.

Arquitectura- Botón Crear .NCL

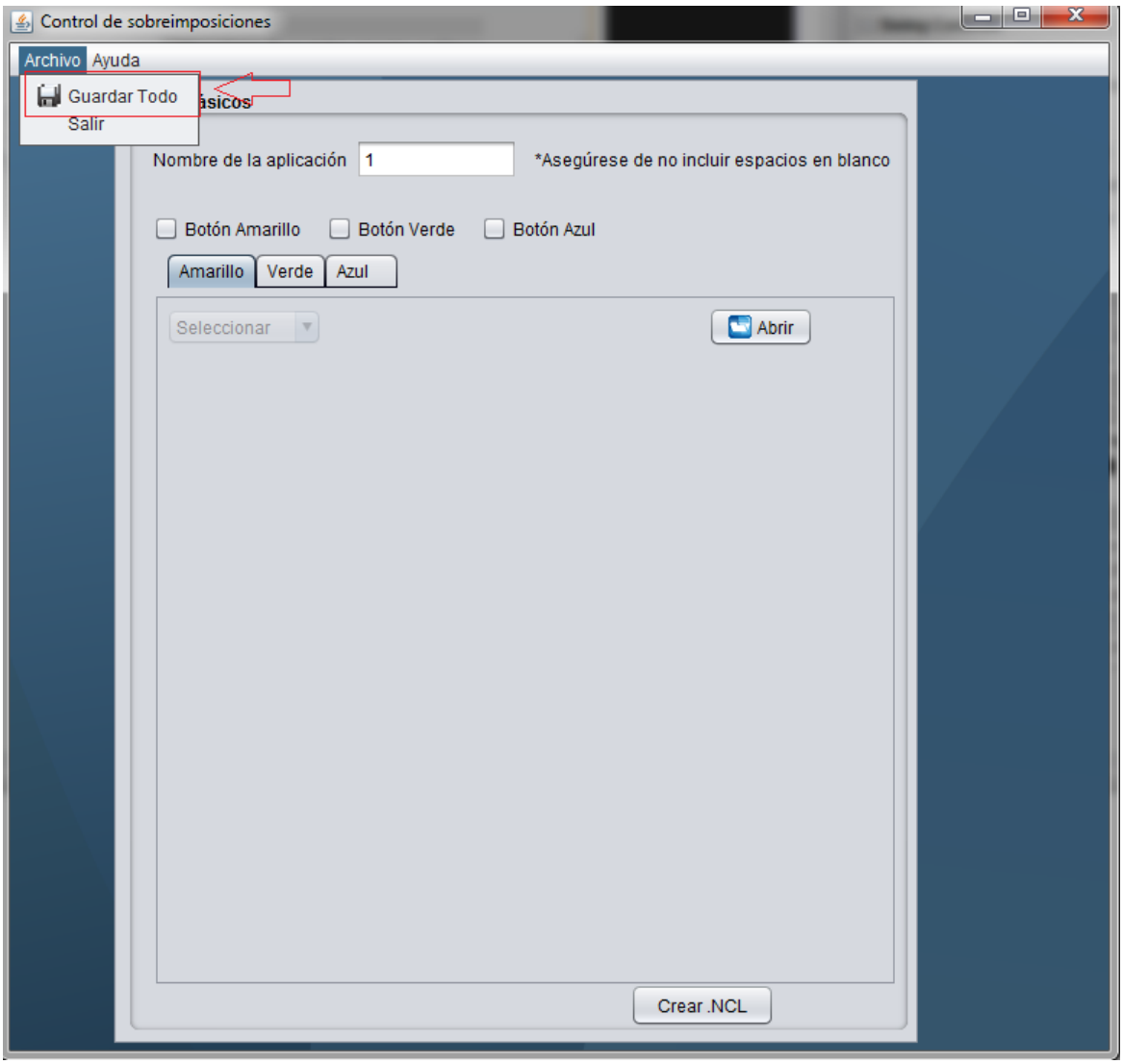
Elaborado por: Juan Fernando Baird



Con eso termina la arquitectura interna de la aplicación Mushu, adicional a las funciones previamente revisadas, existe una función adicional en la barra de menú llamada guardar All, esta permite al usuario grabar todos los botones configurados hasta el momento sin necesidad de presionar cada botón de grabado individualmente.

Arquitectura- Botón Guardar Todos

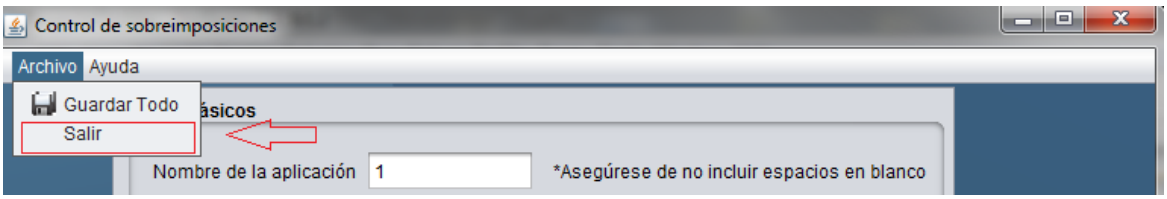
Elaborado por: Juan Fernando Baird



Además del botón "guardar All" en el menú File se encuentra el botón "Exit", como el nombre lo indica este botón únicamente cierra la aplicación Mushu al igual que lo haría el presionar el botón rojo en la esquina superior izquierda.

Arquitectura- Botón Salir

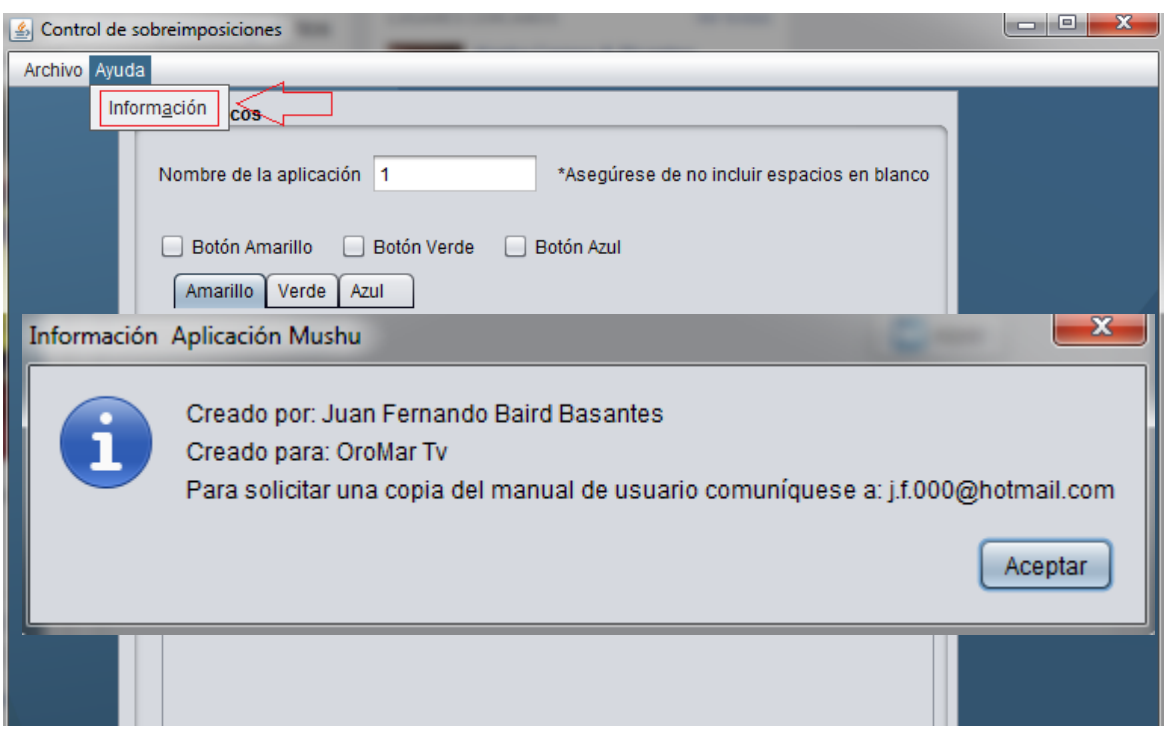
Elaborado por: Juan Fernando Baird



Por último en la barra de menú en la sección “Help” se encuentra el botón “About” el cual al ser activado despliega un mensaje informativo con un mail al cual pueden solicitar el manual de usuario en caso de haberlo perdido.

Arquitectura- Botón Ayuda

Elaborado por: Juan Fernando Baird



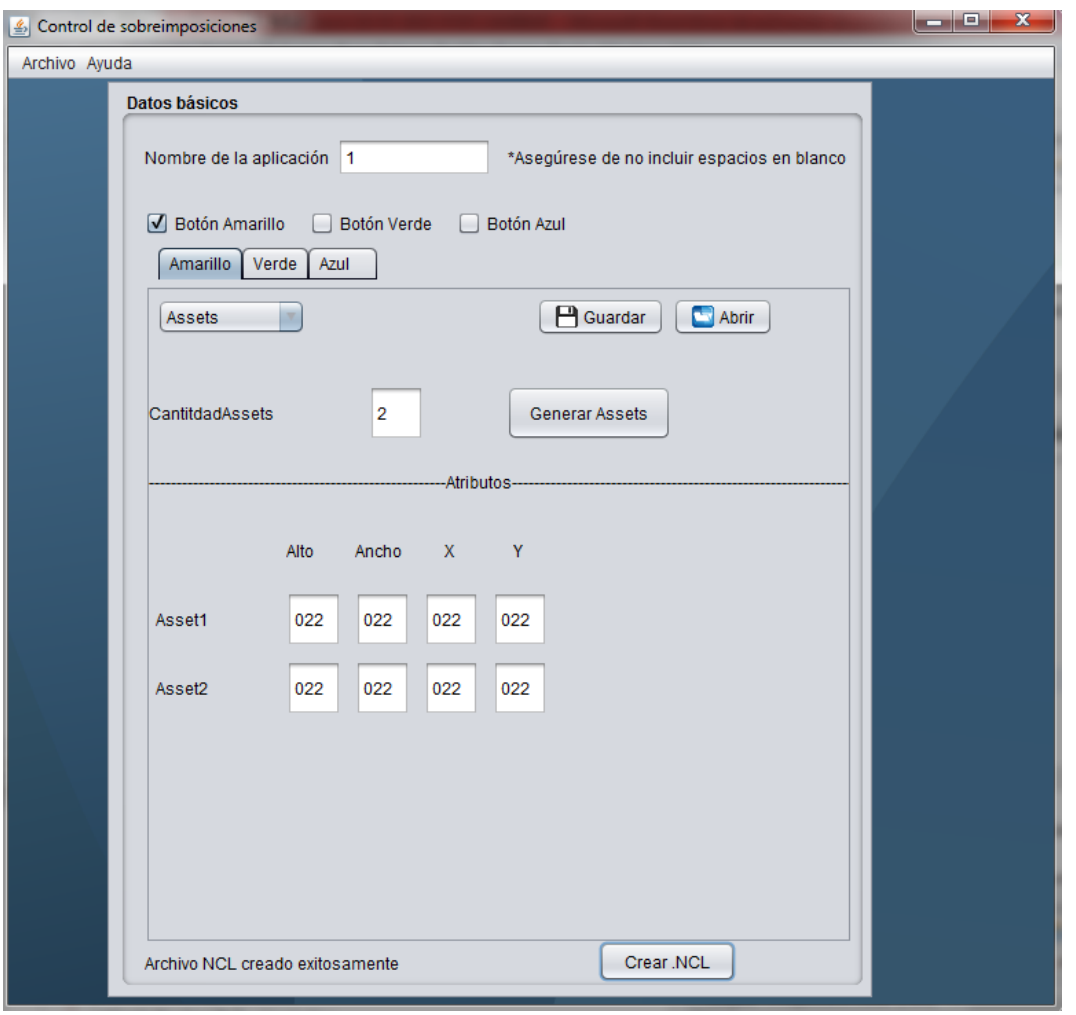
Uso de la función Assets

Como resultado de la función Assets obtendremos un código ejecutable, el cual al ser activado buscará los elementos que posean la nomenclatura adecuada y ejecutarán el video con la posibilidad de activar los botones programados. A continuación se puede observar un ejemplo del uso de la función

- Se ingresa la información y se ejecuta el comando Crear NCL

Ejemplo- Opción Assets Ingreso

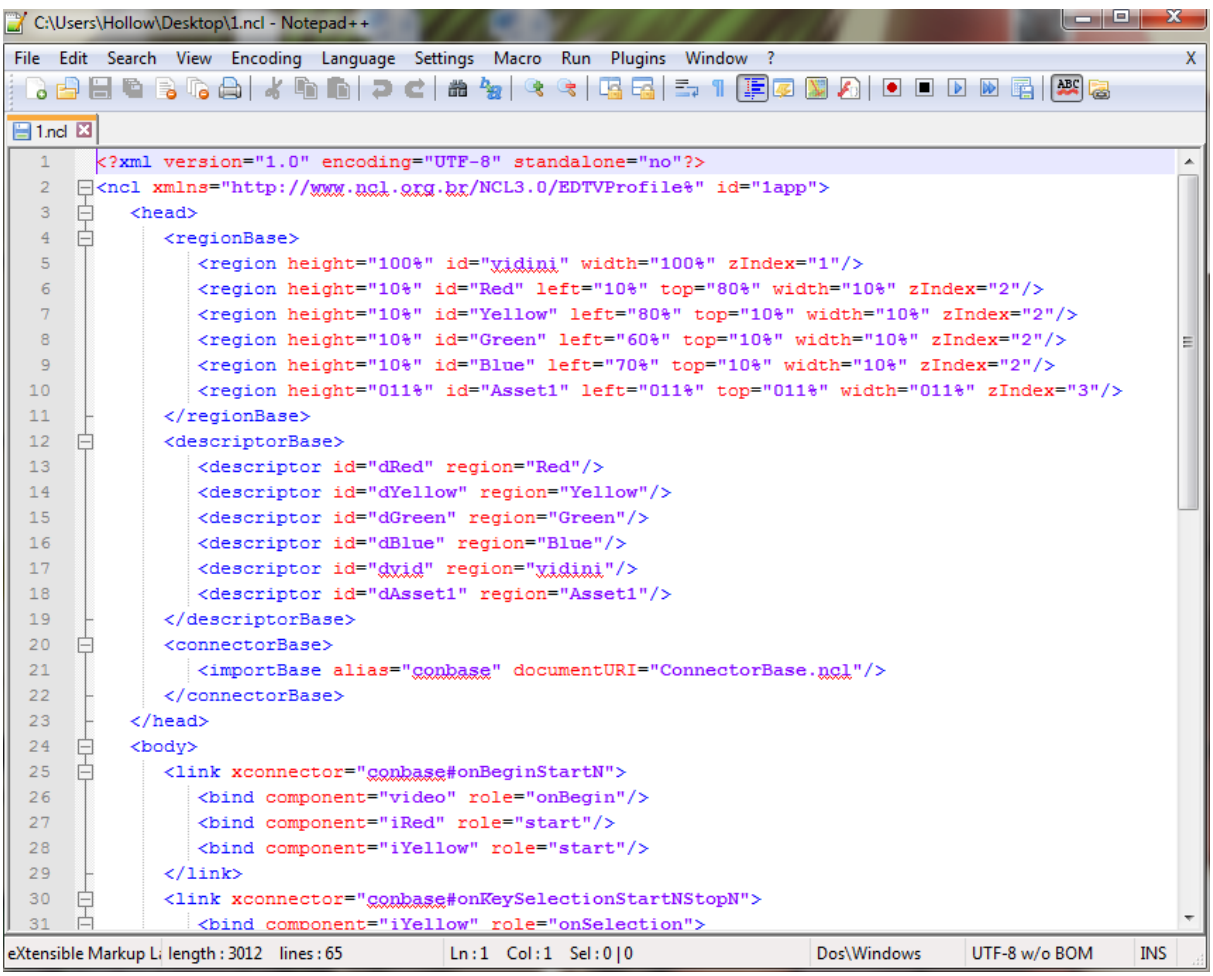
Elaborado por: Juan Fernando Baird



- De ser necesario se inspecciona el código, el cual es completamente editable, pero se sugiere precaución al hacerlo, incluso sin saber mucho de programación el código está organizado de una forma amigable al usuario, si se toman en cuenta la nomenclatura previamente indicada la edición del código será mucho mas sencilla.

Ejemplo- Opción Assets Código Generado

Elaborado por: Juan Fernando Baird



- Finalmente se coloca la aplicación en la carpeta pertinente y se ejecuta para su inspección, se recomienda mantener la aplicación abierta en caso de querer realizar un cambio rápido ya sea en tamaños o coordenadas. También se puede grabar la aplicación para uso futuro, como ya será revisado más adelante.

Ejemplo- Opción Assets Ejecución Reposo

Elaborado por: Juan Fernando Baird

Como se puede observar en la figura #19 la aplicación está corriendo en el emulador de Ginga disponible en la página oficial de Ginga ecuador, libre de cualquier costo o subscripción (Espinoza, 2010). El único botón disponible es el botón amarillo, al seleccionar dicha opción (presionando la tecla F3) el display cambia interactivamente y nos despliega los dos assets previamente programados, como muestra la figura #20 el botón amarillo es remplazado por el icono del botón rojo, esto indica al usuario que su único posible camino es presionar el botón rojo (tecla F1) para regresar al menú inicial y ocultar la información desplegada.

Ejemplo- Opción Assets Ejecución Activada

Elaborado por: Juan Fernando Baird

Uso de la función Menu+Assets

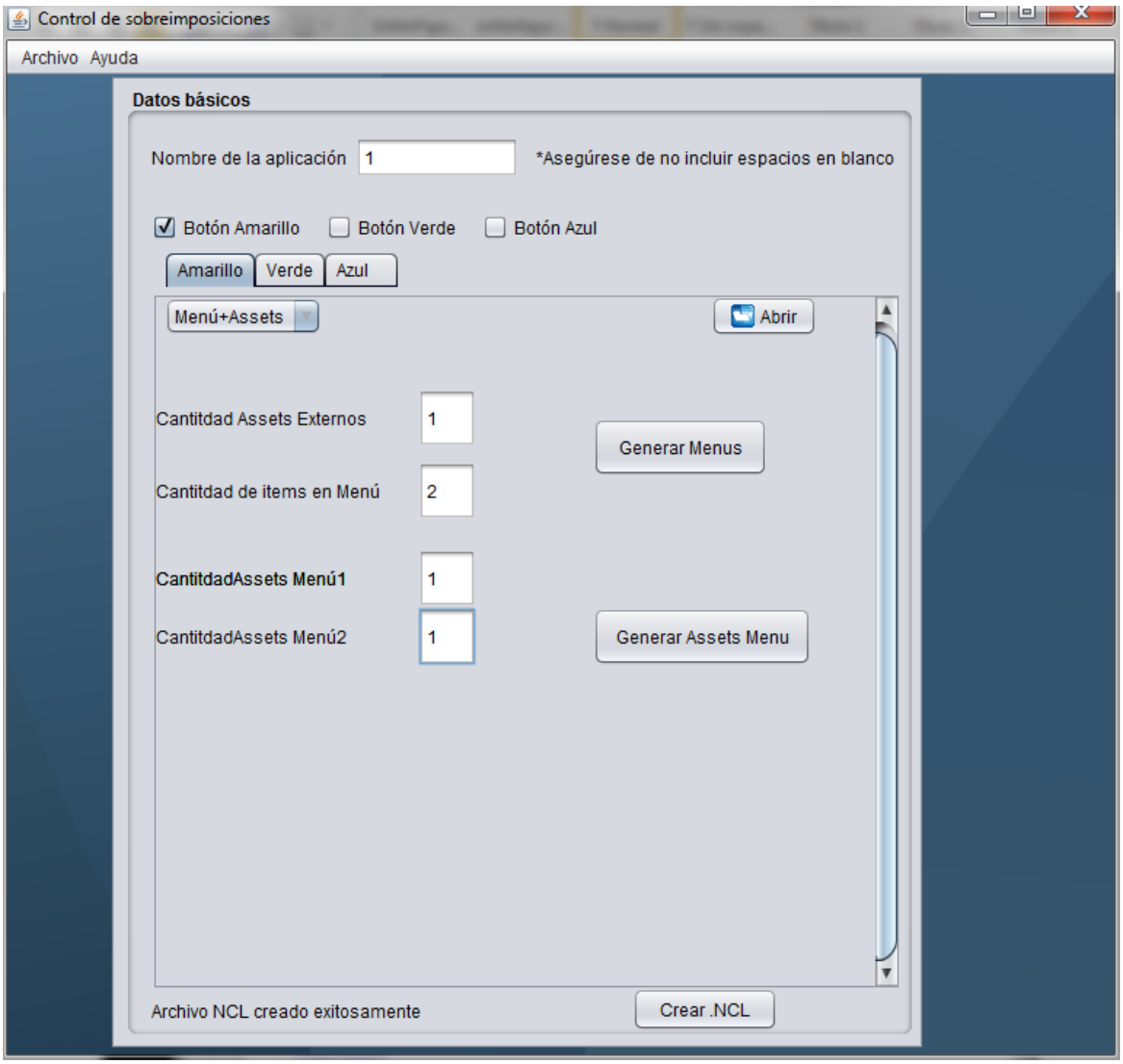
Como resultado de la función Menu+Assets obtendremos un código ejecutable, el cual al ser activado buscará los elementos que posean la nomenclatura adecuada y ejecutaran el video con la posibilidad de activar los botones programados. A continuación se puede observar un ejemplo del uso de la función, la cual permitirá el despliegue de mucha más información de manera ordenada y a su vez tomara un mayor espacio de memoria, requiriendo un mayor poder de procesamiento esto debe ser considerado ya que como ya ha sido mencionado con anterioridad hay límites a lo que el usuario será capaz de ver en su consola.

Para este ejemplo se utilizarán datos sencillos sin ningún diseño particular ya que de esta forma es más fácil visualizar, en la práctica se sugiere el uso de imágenes png con transparencia, para simular efectos visuales más detallados, que brinden al usuario una mejor experiencia. Esta opción fue creada con la intención de facilitar el desarrollo de aplicaciones mucho más visualmente atractivas en comparación a las que actualmente se pueden ver ya que automáticamente crea las regiones, descriptores y llamados a objeto de cada ítem, lo cual ha probado ser un dolor de cabeza para la mayoría de programadores que en la actualidad trabajan estudiando y mejorando Ginga.

- En primer lugar, se ingresa la información adecuada, después de seleccionar la opción Menu+Assets del menú desplegable, a diferencia de la opción anterior en esta es necesario decidir si se desea assets externos o no, lo ideal es encontrar un balance entre diseño y funcionalidad, por lo cual se permite la solicitud de 0 assets externos.

Ejemplo- Opción Menu+Assets Etapa 1

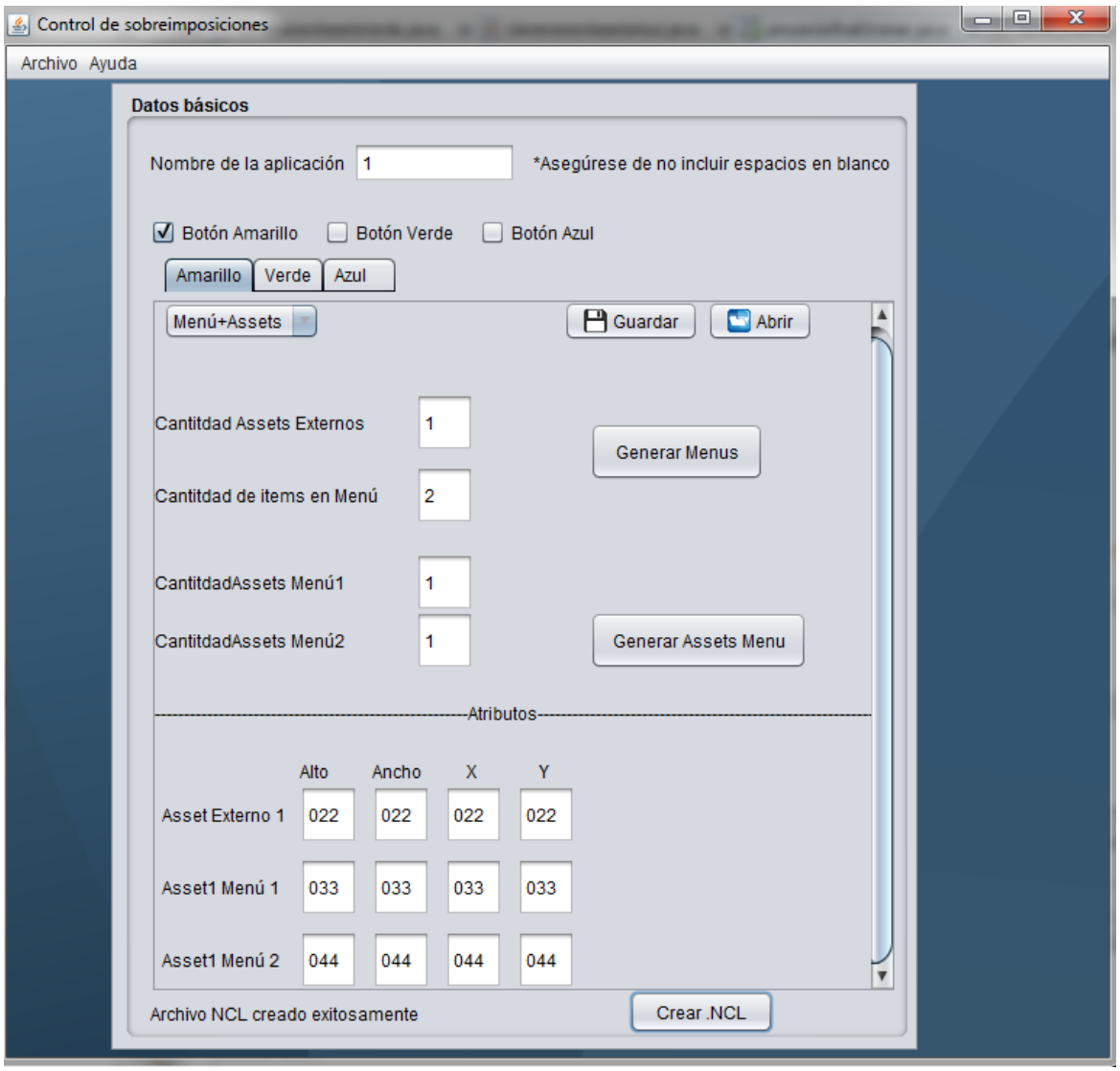
Elaborado por: Juan Fernando Baird



- Una vez creados los espacios de menú es necesario introducir cuantos assets internos tendrá cada uno, estos assets aparecerán en cuanto dicho botón sea activado y deberían contener información pertinente al mismo y de ser necesario entre dos a tres assets visuales, simplemente para dar una mejor experiencia al usuario, aunque es completamente opcional. En este ejemplo se omite esa recomendación ya que es únicamente demostrativo.

Ejemplo- Opción Menu+Assets Etapa 2

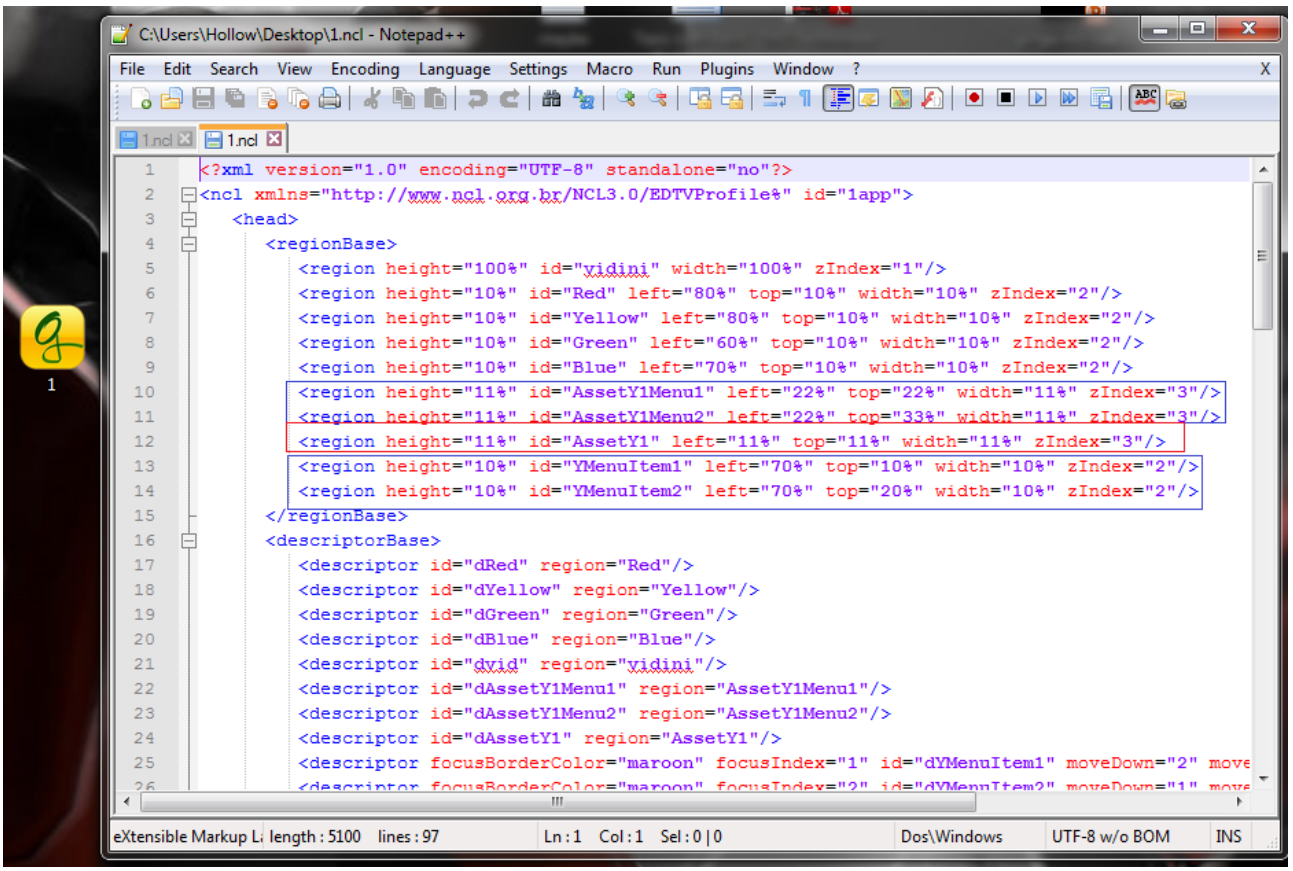
Elaborado por: Juan Fernando Baird



- Una vez ingresada la información, se recomienda revisar que se encuentre de manera correcta y se procede a llenar los datos para el resto de botones, en este ejemplo solo se desea activar el botón amarillo por lo que se procede a presionar el botón “Crear .NCL”, el programa se encargará de procesar la información y generará el código pertinente, con la nomenclatura previamente indicada.

Ejemplo- Opción Menu+Assets Codigo Generado

Elaborado por: Juan Fernando Baird



- Finalmente se coloca la aplicación en la carpeta pertinente y se ejecuta para su inspección, se recomienda mantener la aplicación abierta en caso de querer realizar un cambio rápido ya sea en tamaños o coordenadas. También se puede grabar la aplicación para uso futuro, como ya será revisado más adelante.

Ejemplo- Opción Assets Ejecución Reposo

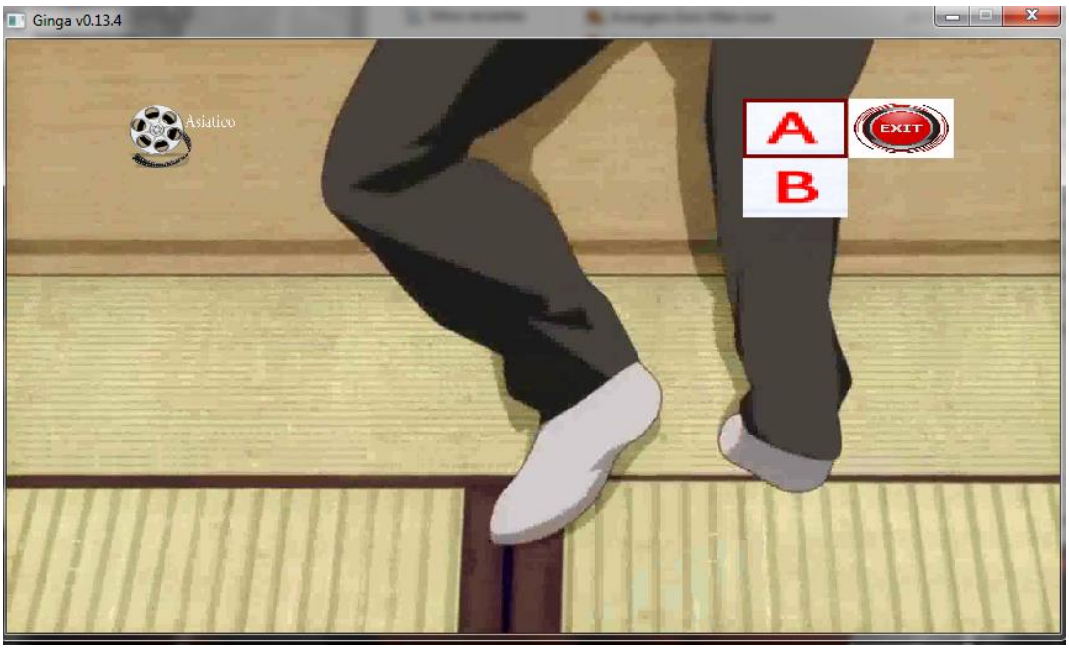
Elaborado por: Juan Fernando Baird

Al igual que en el ejemplo anterior se puede observar que únicamente el botón amarillo fue llamado, como se indica en la Figura#27 el entorno no ha cambiado, al activar el botón amarillo, se puede observar el menú y el único asset externo que se programó para el ejemplo, además del icono para el botón que indica la capacidad de retorno como indica la Figura#27.

En cuanto se activa alguna de las opciones del menú se activará la información disponible en la misma como se puede apreciar en la Figura#29, en caso de activar otra opción las imágenes anteriores desaparecen y se presenta las correspondientes al botón activado. Finalmente en caso de presionar el botón rojo (tecla F1) la aplicación vuelve al estado inicial.

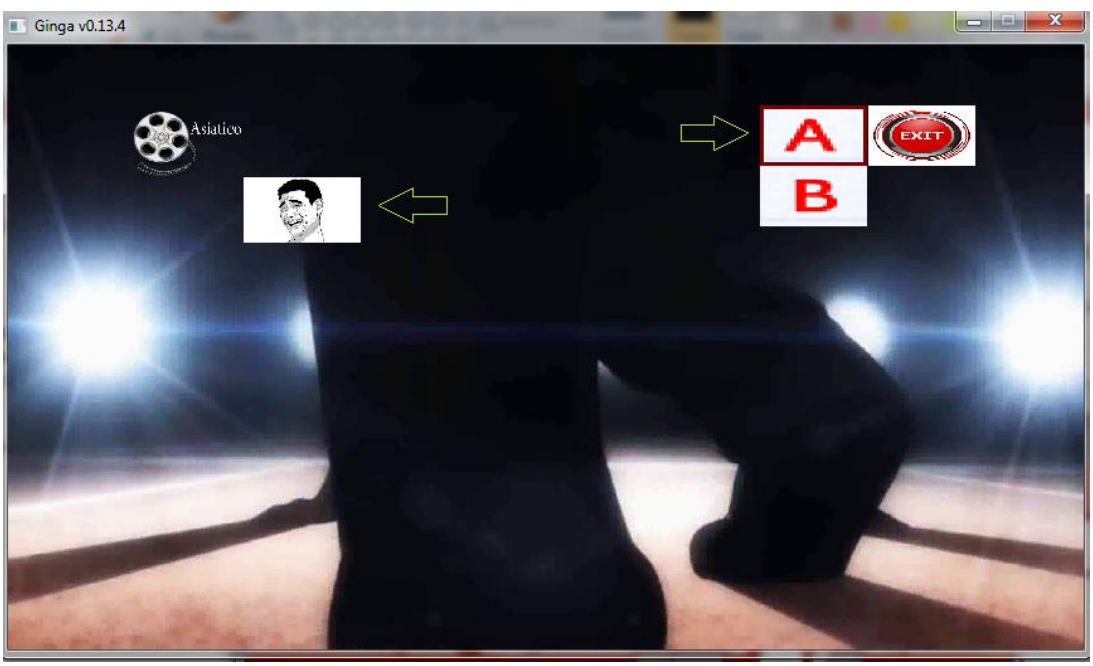
Ejemplo- Opción Assets Ejecución Menú Activado

Elaborado por: Juan Fernando Baird



Ejemplo- Opción Assets Ejecución Menú Activado Opción A

Elaborado por: Juan Fernando Baird



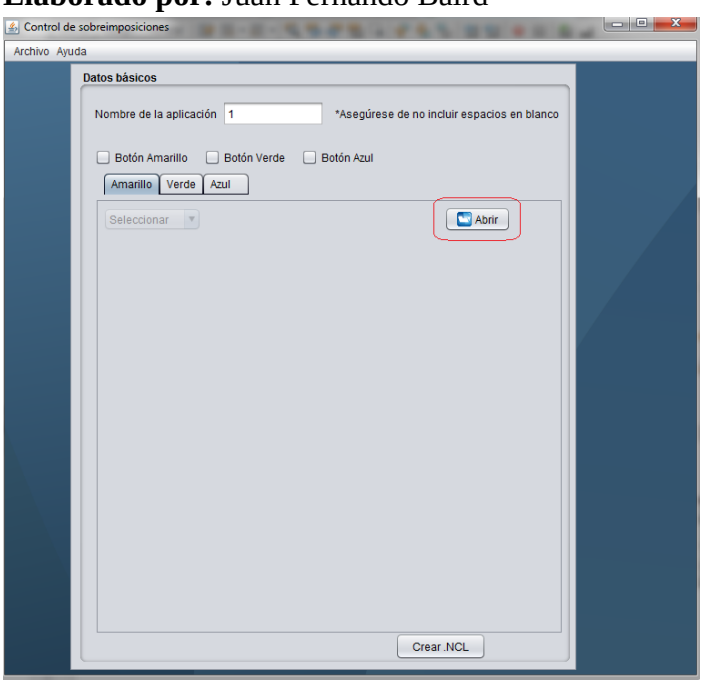
Uso de las funciones Guardar y Abrir

Dado a la flexibilidad del programa y reusabilidad que posee el código Ginga resultante, se asumió como necesidad básica la capacidad de grabar y cargar código previamente desarrollado, como se menciona en la descripción anatómica del programa, Mushu almacenará los datos de cada botón individualmente, de modo que sean intercambiables y reemplazables dando así una flexibilidad única que rara vez se observa en programas de desarrollo. Para el uso de la función guardar es necesario seguir los siguientes pasos:

- Iniciar la aplicación, como se puede observar en la Figura #35 el botón "guardar" de cada sección no se encuentra visible inicialmente, esto se debe a que el programa necesita tener un mínimo de información para percibir un perfil como salvable.

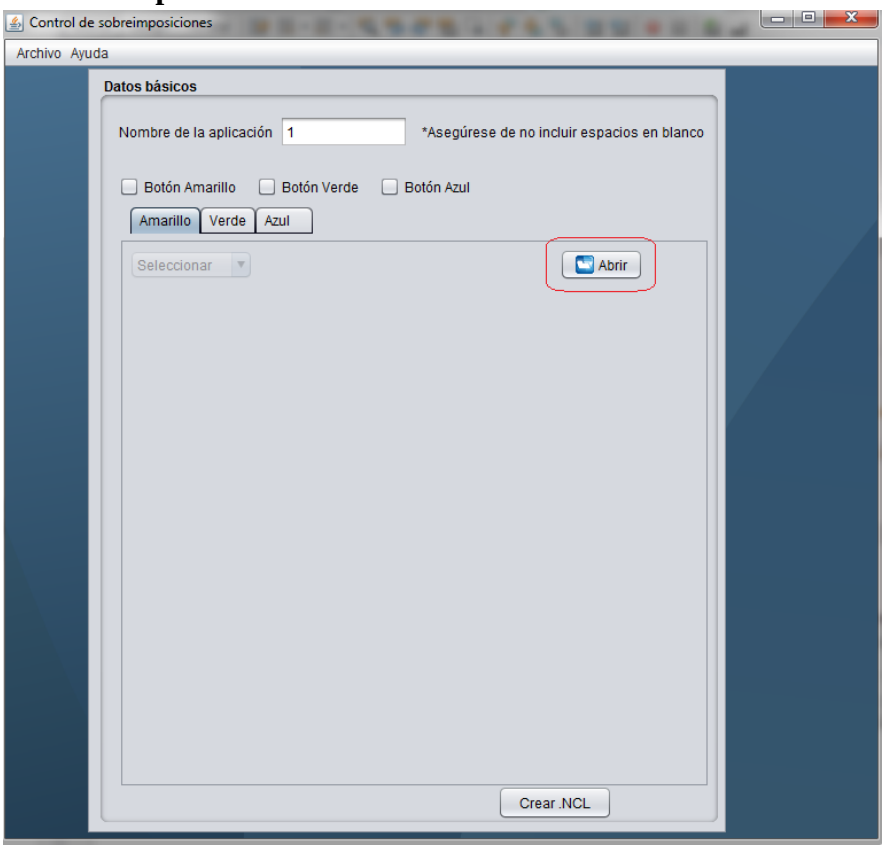
Ejemplo- Uso Botón guardar Estado Inicial

Elaborado por: Juan Fernando Baird



Ejemplo- Uso Botón guardar Estado Inicial

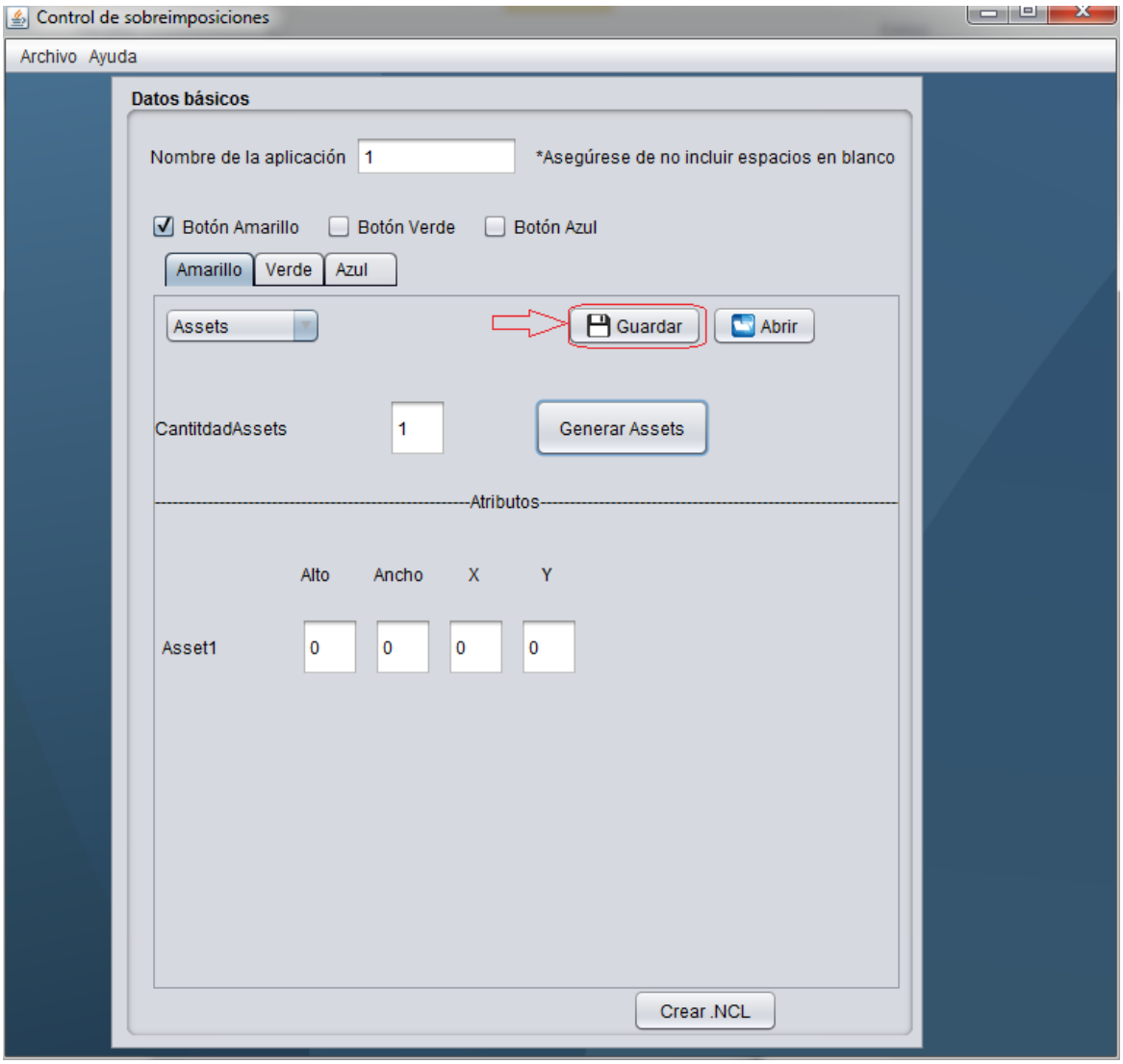
Elaborado por: Juan Fernando Baird



- A continuación, se procede a insertar la información como indica el uso de Assets o Menu+Assets, el botón "guardar" aparecerá en ambos casos en cuanto el programa haya mapeado ya los assets a utilizar, ya que en este momento se tiene una información útil del botón seleccionado.

Ejemplo- Uso Botón guardar Estado Activado

Elaborado por: Juan Fernando Baird

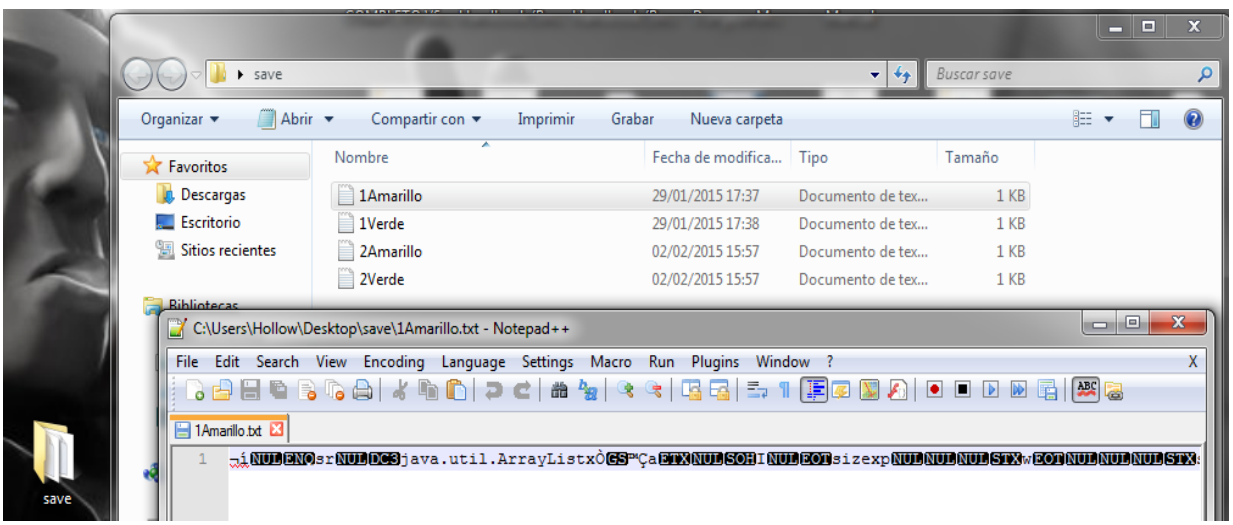


Al presionar el botón guardar el programa mostrará un pequeño mensaje de confirmación en la parte inferior de la ventana al lado izquierdo del botón “Crear NCL” y almacenara los datos en una carpeta con el nombre “save” ubicada en el escritorio del usuario, de esta forma se simplifica la necesidad de buscar el archivo guardado y mantiene de forma ordenada el código. Cada archivo se encuentra cifrado de tal forma que la

información del mismo está ligeramente protegida para evitar que el usuario manipule el archivo y termine dañando la información del mismo

Ejemplo- Uso Botón guardar Resultados

Elaborado por: Juan Fernando Baird

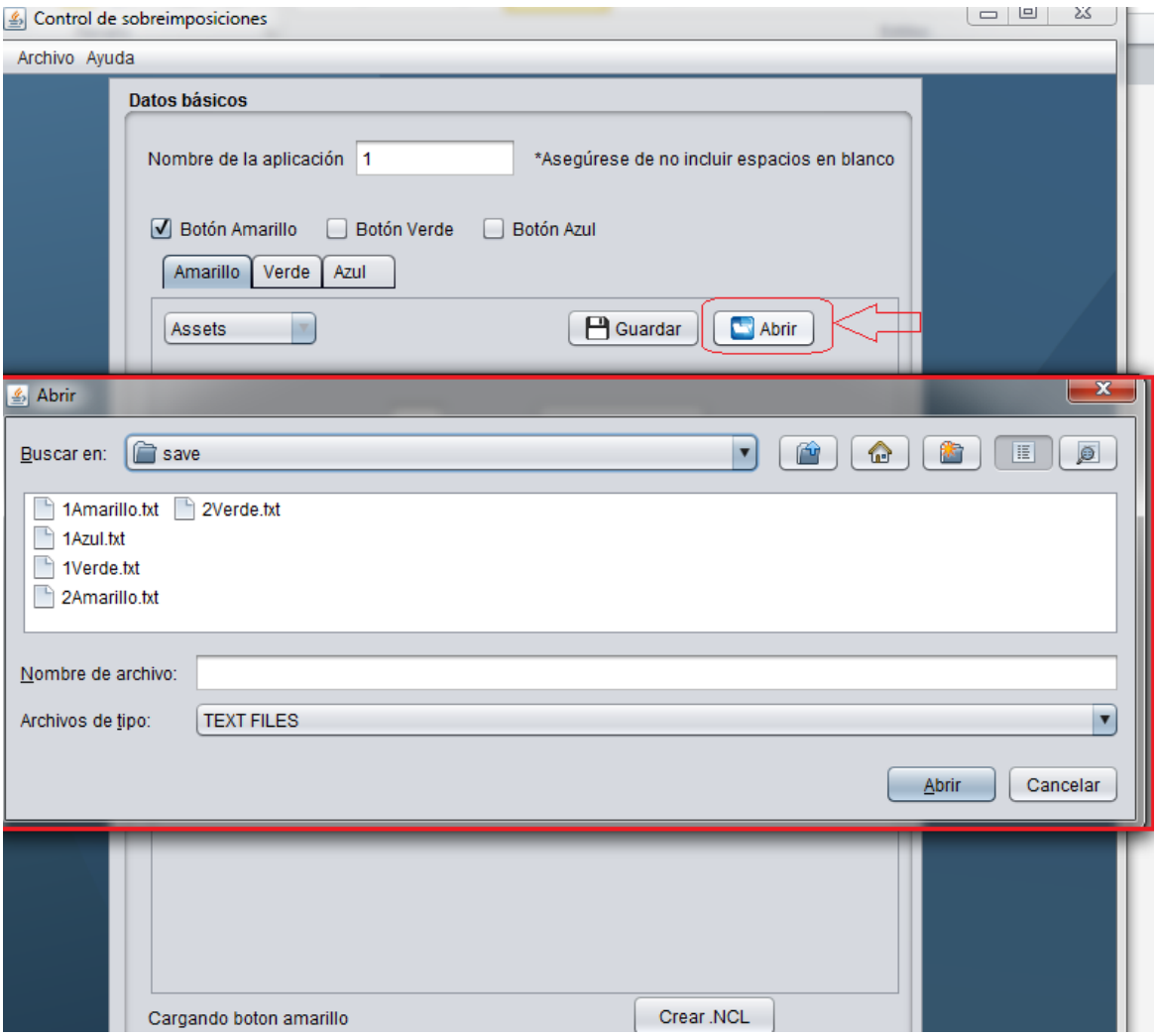


Una vez grabada información, esta se encuentra disponible para el uso del usuario en cualquier momento, incluso varias veces en el mismo programa, como por ejemplo en caso de desear que todos los botones tengan la misma cantidad de assets y elementos de un menú, simplemente se necesita crear uno, grabarlo y cargarlo en los otros dos botones, reduciendo aún más el proceso de desarrollo. A diferencia del botón guardar el botón load se encuentra siempre activo y listo para su uso, simplemente se necesita seguir los siguientes pasos:

- Con la aplicación iniciada, y en cualquier momento durante el desarrollo es posible presionar el botón “abrir”, este mostrará una ventana de selección la cual por default aparecerá en la carpeta “save” del escritorio para fácil acceso, e caso de no existir dicha carpeta, el navegador aparecerá en la carpeta “Documents” del usuario.

Ejemplo- Uso Botón abrir Activación

Elaborado por: Juan Fernando Baird

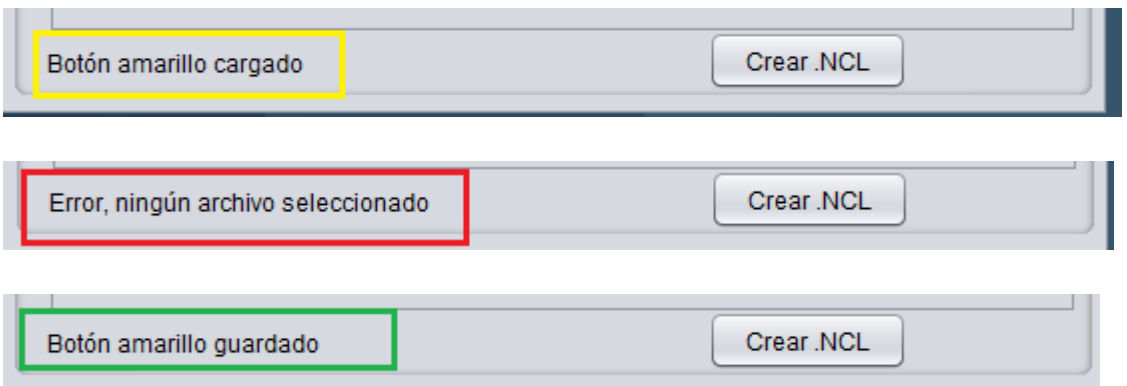


El uso del botón “abrir” indicará tres estados en el programa, dependiendo del estado del proceso de carga, al momento inicial y mientras se navegue en búsqueda del archivo el mensaje al pie de la aplicación dirá “Cargando botón X”, en caso de ser exitoso en su recuperación de información el mensaje dirá “Botón X cargado” y en caso de no seleccionar un archivo apropiado, o en su defecto ningún archivo el mensaje se lo

comunicara diciendo “Error, ningún archivo seleccionado”, tal y como indica la Figura #39.

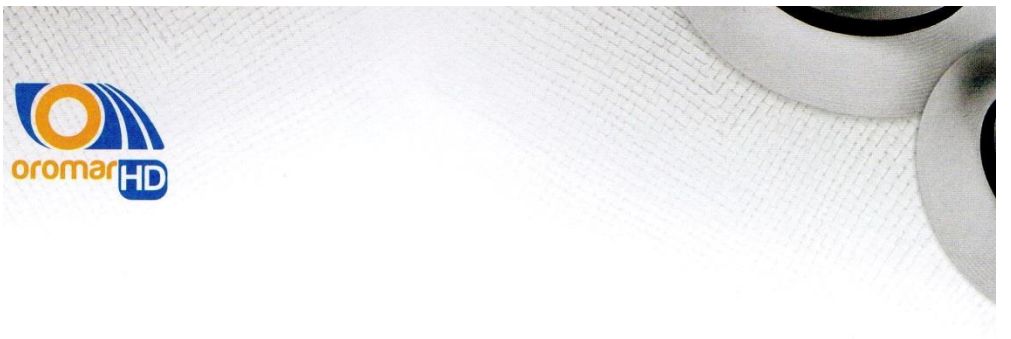
Ejemplo- Uso Botón abrir Estados

Elaborado por: Juan Fernando Baird



Una vez recuperada la información el programa se encuentra en capacidad de continuar trabajando normalmente como si el usuario hubiese introducido la información manualmente, se puede editar y manipular la información cargada sin afectar al archivo original ya que este se mantiene intacto en su carpeta de origen reduciendo así el riesgo de dañar el mismo, para sobrescribir un archivo simplemente se puede grabar el botón deseado utilizando el nombre de la aplicación antigua que se desea remplazar.

ANEXO B



CERTIFICADO

Quito, Marzo 19 del 2015

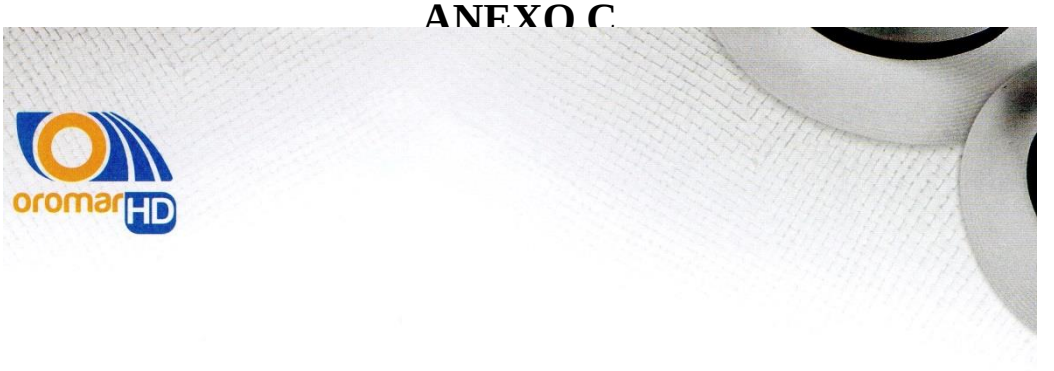
En mi calidad de VICEPRESIDENTE, CERTIFICO que el producto de software: "DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA PARA LA GENERACIÓN DE CÓDIGO DE SOBREIMPOSICIONES DE TV DIGITAL PARA OROMARTV", desarrollado individualmente por el Sr. JUAN FERNANDO BAIRD BASANTES, identificado con CI: 172224888-5, se ha implementado exitosamente a partir del: 15 de marzo del 2015.

Atentamente,

Marcos Párraga

MANTA QUITO GUAYAQUIL

ANEXO C



CERTIFICADO DE CAPACITACION

Quito, Marzo 19 del 2015

En calidad de Vicepresidente, de OroMar Tv, Certifico que:

El Señor: Juan Fernando Baird Basantes, con CI: 172224888-5 procedió a capacitar a los señores:

Nombre	Cedula de Identidad	Capacitación en	Firma
José Fabián Ureña Andrade	1707861363	Uso de la aplicación de control	
Galo Alberto Freire Landeta	1708521628	Uso de la aplicación de control	

Para el uso de: Sistema de control de sobreimposiciones para Ginga, el cual genera código ejecutable.

Atentamente,

Marcos Parraga

MANTA QUITO GUAYAQUIL