Diseño e implementación de un sistema de televisión digital interactiva aplicada a internet de las cosas para un sistema de Smart Agro

Guerrero Tamayo, Alexander Maximiliano y Samaniego Chamba, Daniel Alexander

Departamento de Eléctrica, Electrónica y Telecomunicaciones

Carrera de Ingeniería en Electrónica y Telecomunicaciones

Trabajo de titulación, previo a la obtención del título de Ingeniero en Electrónica y Telecomunicaciones

Ing. Alulema Flores, Darwin Omar, PhD.

10 de febrero del 2022

COPYLEAKS

Tesis_Guerrero_Samaniego_VerificarPlagio.pdf

Scanned on: 0:16 February 11, 2022 UTC

Firmado electrónicamente por:
DARWIN OMAR
ALULEMA
FLORES

7.9%	78	28347
Overall Similarity Score	Results Found	Total Words in Text

Identical Words	1049
Words with Minor Changes	387
Paraphrased Words	791
Ommited Words	0

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

CERTIFICACIÓN

Certifico que el trabajo de titulación, "**Diseño e implementación de un sistema de televisión digital interactiva aplicada a internet de las cosas para un sistema de Smart Agro**" fue realizado por los señores **Guerrero Tamayo, Alexander Maximiliano** y **Samaniego Chamba Daniel Alexander** el cual ha sido revisado y analizado en su totalidad por la herramienta de verificación de similitud de contenido; por lo tanto cumple con los requisitos legales, teóricos, científicos, técnicos y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, razón por la cual me permito acreditar y autorizar para que lo sustente públicamente.

Sangolquí, 10 de febrero del 2022

Firmado electrónicamente por:
DARWIN OMAR
ALULEMA
FLORES

.....

Ing. Alulema Flores, Darwin Omar, PhD

C. C 1002493334

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

RESPONSABILIDAD DE AUTORÍA

Nosotros, **Guerrero Tamayo, Alexander Maximiliano** y **Samaniego Chamba Daniel Alexander**, con cédulas de ciudadanía n° 1726779372 y n° 1723109573, declaramos que el contenido, ideas y criterios del trabajo de titulación: **Diseño e implementación de un sistema de televisión digital interactiva aplicada a internet de las cosas para un sistema de Smart Agro** es de nuestra autoría y responsabilidad, cumpliendo con los requisitos legales, teóricos, científicos, técnicos, y metodológicos establecidos por la Universidad de las Fuerzas Armadas ESPE, respetando los derechos intelectuales de terceros y referenciando las citas bibliográficas.

Sangolquí, 10 de febrero del 2022

.....	.....
Guerrero Tamayo, Alexander Maximiliano	**Samaniego Chamba Daniel Alexander**
C.C.: 1726779372	C.C.: 1723109573

DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

AUTORIZACIÓN DE PUBLICACIÓN

Nosotros, **Guerrero Tamayo, Alexander Maximiliano y Samaniego Chamba Daniel Alexander** con cédulas de ciudadanía n° 1726779372 y n° 1723109573, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar el trabajo de titulación: **Diseño e implementación de un sistema de televisión digital interactiva aplicada a internet de las cosas para un sistema de Smart Agro** en el Repositorio Institucional, cuyo contenido, ideas y criterios son de nuestra responsabilidad.

Sangolquí, 10 de febrero del 2022

.....	.....
Guerrero Tamayo, Alexander Maximiliano	**Samaniego Chamba Daniel Alexander**
C.C.: 1726779372	C.C.: 1723109573

Dedicatoria

Este trabajo lo dedico a mis padres y hermano quienes siempre me han apoyado a lo largo de mi vida, por alentarme a seguir adelante ante cualquier dificultad y por estar conmigo en los buenos y malos momentos.

Alexander Maximiliano Guerrero Tamayo

Dedicatoria

Este trabajo lo dedico a mis padres y hermanos por su amor y por hacerme la persona que soy, por ser un apoyo constante en cada etapa de mi vida, esto es por y para ellos.

A mis abuelitos Segundo, Lorgio, Carmen y Amada, por su infinito cariño y bendiciones.

A mis sobrinos que sepan que todo esfuerzo tiene su recompensa y que nunca negocien su esencia ni sus valores ante las adversidades de la vida.

Daniel Alexander Samaniego Chamba

Agradecimiento

Agradezco a mis padres Patricio Guerrero y Margarita Tamayo por su sacrificio para que no me falte nada, por su preocupación y afecto que me han brindado siempre y por ayudarme a culminar una meta más.

A mi hermano Joshua por sus consejos, por ser la persona con quien siempre puedo contar y por enseñarme el significado de perseverancia. A mis sobrinas Isabela y Renata por su cariño y buenos momentos.

A mis amigos y personas que he conocido en el transcurso de la universidad porque gracias a ellos he podido aprender muchas cosas. A mi compañero Daniel por su ayuda y constancia para culminar este trabajo.

Al ingeniero Derlin Morocho por el apoyo que nos brindó a mí y compañero de tesis para poder presentar el tema ante el consejo de la carrera.

Finalmente agradezco a nuestro tutor de tesis, el ingeniero Darwin Alulema, por permitirme ser parte de este proyecto, por su paciencia y todo el apoyo brindado en las revisiones para que el proyecto pueda salir adelante.

Alexander Maximiliano Guerrero Tamayo

Agradecimiento

Agradezco a Dios y a la vida por cada enseñanza y experiencia, por el continuo aprendizaje que es vivir.

A mis padres Eduardo y Haydee por su apoyo incondicional, por ser el pilar fundamental en mi vida, por estar siempre ante cualquier adversidad, por los consejos y por su amor absoluto.

A mis hermanos Yayo, Gaby, Carol y William con sus respectivos hogares por todas sus palabras de aliento, por su apoyo continuo, por las enseñanzas y por brindarme siempre la confianza de poder contar con ellos, por compartir momentos en familia que siempre será lo más preciado y recordado.

A mis sobrinos, por su alegría y cariño, por ser una razón para querer volver cada fin de semana, por ser inspiración y motivación para no decaer.

A mis amigos que siempre estuvieron ahí para cualquier circunstancia, por las experiencias, los consejos, los asados, la música, el fútbol y las bielas que nunca faltaron.

A mi compañero de tesis y amigo Alex por su compromiso y dedicación para sacar adelante este proyecto conjuntamente.

Finalmente agradezco a la Academia, al Ingeniero Derlin Morocho por el apoyo que nos brindó para poder presentar el tema ante el consejo de carrera y al Ingeniero Darwin Alulema por su tiempo, dedicación y paciencia para el desarrollo de este trabajo.

Daniel Alexander Samaniego Chamba

Contenido

Índice de Tablas

Índice de Figuras

Resumen

En la actualidad la convergencia de hardware y software está dirigida a las futuras ciudades inteligentes mediante el uso del internet de las cosas, las cuales necesitarán agrupar varias tecnologías, y donde la televisión tiene mucho potencial para controlar remotamente objetos domésticos e indicar información detallada de los mismos. Esto conduce hacia un mejor desarrollo tecnológico de todos los sectores productivos, especialmente dentro del país, donde la agricultura puede ser llevada a un mayor nivel mediante la automatización de sus procesos con la ayuda de la utilización de un enfoque Smart Agro, el cual permite generar información detallada en tiempo real y propiciar toma de decisiones más eficientes, a partir de datos objetivos. El presente trabajo, propone el desarrollo de aplicaciones interactivas para TDT, enfocadas en el dominio del Smart Agro mediante la utilización de un metamodelo, DSL gráfico, automatización de generación de código fuente, utilizando herramientas de software como: Eclipse Modeling Framework, Sirius y Acceleo; orquestación de servicios, bases de datos y servicios web, utilización de ordenadores de placa, sensores, actuadores y módulos Wifi que permitirán controlar y establecer el entorno de comunicación de Smart Agro y las aplicaciones IoTV. Para la comprobación del funcionamiento se ha implementado un escenario de prueba del sistema a partir del monitoreo de un huerto casero, pruebas de rendimiento y carga, usabilidad del editor gráfico y factibilidad en la generación de código.

PALABRAS CLAVE:

- **TELEVISIÓN DIGITAL TERRESTRE**
- **INTERNET DE LAS COSAS**
- **SMART AGRO**
- **LENGUAJE ESPECÍFICO DE DOMINIO**
- **METAMODELO**

Abstract

Currently, the convergence of hardware and software is aimed at future smart cities through the use of the internet of things, which will need to bring together various technologies, and where television has a lot of potential to remotely control household objects and indicate detailed information about them. This leads to a better technological development of all productive sectors, especially within the country, where agriculture can be taken to a higher level by automating its processes with the help of the use of a Smart Agro approach, which allows generating detailed information in real time and enabling more efficient decision making, based on objective data. The present work, proposes the development of interactive applications for DTT, focused on the Smart Agro domain through the use of a metamodel, graphical DSL, source code generation automation, using software tools such as: Eclipse Modeling Framework, Sirius and Acceleo; orchestration of services, databases and web services, use of board computers, sensors, actuators and Wi-Fi modules that will allow to control and establish the communication environment of Smart Agro and IoTV applications. For the verification of the operation, a system test scenario has been implemented based on the monitoring of a home orchard, performance and load tests, usability of the graphic editor and feasibility in code generation.

KEY WORDS:

- **DIGITAL TERRESTRIAL TELEVISION**
- **INTERNET OF THINGS**
- **SMART AGRO**
- **DOMAIN SPECIFIC LANGUAGE**
- **METAMODEL**

Capítulo I: Marco Metodológico

Antecedentes

En los últimos años como parte del proceso de transición hacia la Televisión Digital Terrestre (TDT) en el Ecuador, 5 de cada 10 habitantes (54%) están cubiertos por el servicio de TDT; sin embargo, tan solo el 5,6% de las estaciones han digitalizado su señal, lo que representa una oportunidad de inversión a futuro por parte de las industrias televisivas y de entretenimiento, previo al cese de señales analógicas a nivel nacional. Adicionalmente, el nivel de acceso a la TDT, a junio de 2017, el 16% de los hogares que visualizan la televisión abierta se encuentran equipados con receptores capaces de operar con el estándar ISDB-Tb para sintonizar las señales de TDT (MINTEL, 2018).

Completar la transición hacia la Televisión Digital Terrestre representa un papel importante para asegurar una mayor calidad de video, aumentar la factibilidad de las personas al acceso a la información y habilitar una amplia gama de contenidos para la población ecuatoriana (Viceministerio de Tecnologías de la Información y Comunicación , 2018). El mismo Plan detalla los siguientes datos con respecto a la línea base del año 2017 y las metas que se establecieron para el año 2021:

- La cobertura poblacional con señales de TDT debe tener un aumento del 54% al 66%.
- Se debe aumentar la cantidad de entidades que brindan el servicio de la TDT, se plantea aumentar está cantidad de un 5.70% a un 15%.
- Además, el porcentaje de hogares que conocen sobre el proceso de implementación de la TDT debe tener un incremento del 26% al 50%.

Esta investigación se alinea también con varios de los objetivos de desarrollo sostenible en la Agenda 2030 aprobados por la Asamblea General de las Naciones

Unidas en el 2015, los cuales deberán ser alcanzados en los próximos 15 años desde su aprobación y donde las Telecomunicaciones y la Sociedad de la Información juegan un rol primordial, entre otros, los objetivos 9,11 y 12 detallan lo siguiente (Naciones Unidas, 2018):

- Industria, innovación e infraestructura: Establecer esquemas de estructuras internas sólidas, alentar la inclusión, sostenibilidad e innovación de nuevas tecnologías.
- Comunidades rigurosas: Lograr una comunidad segura y capaz de auto sustentarse.
- Producción y consumo con responsabilidad: Establecer nuevos esquemas para mejorar los métodos actuales de producción y consumo.

Motivación

El Departamento de Asuntos Económicos y Sociales de las Naciones Unidas ha lanzado un documento que prevé que el 68 % de la población rural mundial habrá migrado a las ciudades de cara al 2050 (ONU, 2018), este proceso de migración producirá un desbalance en el incremento poblacional e inseguridad alimentaria de varias familias. Para minimizar el impacto de esta transición han surgido varias propuestas como huertos domésticos o la agricultura urbana, sin embargo, estas alternativas resultan difícil de implementar debido al alto grado de desconocimiento de las familias o ciudadanos para cuidar, mantener y desarrollar dichos cultivos.

Una de las propuestas por parte del Ministerio de Agricultura, Ganadería, Acuacultura y Pesca (MAGAP), ha sido crear medios televisivos como: El noticiero Ecuador Rural (2014) o la comunidad Agrotubers (2019) a fin de brindar mayor información a los ciudadanos, sin embargo, estos programas no aprovechan al máximo

las capacidades de la televisión, dado que se limitan únicamente a informar y no permiten a los espectadores interactuar con el contenido, por lo cual suelen pasar desapercibidos (MAGAP, 2019). Es así que, como mejora a las alternativas planteadas del MAGAP, se propone la adición de contenido interactivo a la TDT para permitir a los televidentes obtener información adicional y poder interactuar remotamente con los cultivos mediante la inclusión del Smart Agro (iniciativa propuesta por Telefónica Movistar y la FAO) (TELEFONICA, 2021).

Esta iniciativa busca impulsar el uso de nuevas tecnologías como IoT para lograr modernizar y mejorar la explotación del campo dentro de países como Colombia, Perú Ecuador y El Salvador, gracias a que IoT permite que la información obtenida de los sensores que se integran en los cultivos, pueda generar recomendaciones sobre el riego y sobre el uso de los insumos productivos para que el agricultor tome mejores decisiones y esto tenga un efecto directo sobre la producción. Finalmente, la integración de la TDT con el IoT permitirá que el concepto de Smart Agro pueda tener un mayor impulso dentro del Ecuador, gracias al entorno inalámbrico que se puede proporcionar, además de la gran cobertura y capacidad de informar de la TDT (EFEAGRO, 2020).

Sin embargo, dicha integración implica la participación de personal técnico dentro del desarrollo del contenido televisivo, esta práctica no es muy común debido a que generalmente las estaciones encomiendan este contenido a desarrolladores gráficos, audiovisuales o asesores externos, los cuales generalmente no poseen conocimiento acerca del desarrollo técnico para generar contenido interactivo para la TDT, por lo cual es necesario de herramientas que permitan generar este contenido sin la necesidad de usar código fuente directamente, como solución para la generación de dicha herramienta se tiene MDE.

El uso más habitual de MDE es la generación automática de código fuente a partir de modelos definidos, proporcionando fundamentalmente mejoras en la productividad: ahorro en tiempos de desarrollo y reducción del número de errores (Quinteros & Troya, 2018), además, un mayor uso de modelos tiene como finalidad disminuir el grado de dificultad tanto del desarrollo como de la gestión de sistemas de software modernos. Los DSL y la MDE se encuentran en el centro de atención de las industria, debido a que se las considera como mecanismos para combatir al gran índice de calidad que deben tener las industrias, mejorar los tiempos de entrega al mercado, mejoras en el rendimiento y la mantenibilidad (Ruiz Rube, 2012).

Importancia

Con el uso de MDE en la presente propuesta se podrá realizar la automatización del proceso de desarrollo mediante la abstracción de las características particulares de las tecnologías para la implementación de aplicaciones interactivas para la TDT, expandiendo así la creación de contenidos por diversos entes como: estudiantes, docentes, desarrolladores audiovisuales y en especial los programadores de contenido de los diferentes canales de televisión abierta del país, los cuales podrán difundir ampliamente información sobre varias áreas de interés como la medicina, agricultura, educación, ganadería, economía, etc., mediante programas interactivos, dando así el impulso al desarrollo de la TDT dentro del Ecuador, el cual en los últimos años se ha estancado respecto al cumplimiento del Plan Maestro de Transición a la TDT.

A través de MDE también se busca realizar la convergencia entre la TDT e IoT, debido a que IoT proporciona las herramientas que buscan facilitar la vida cotidiana con la interconexión de dispositivos y objetos que interaccionan a través de Internet, a su vez, se encuentra en constante expansión, por lo cual requiere de entornos robustos y amigables con el usuario, como alternativa a este requerimiento se plantea el uso de la

TDT, debido a que es una de las plataformas de mayor cobertura en el Ecuador que ha ido incorporando nuevas funciones que la hacen ideal para aplicaciones que desean llegar a muchas personas en poco tiempo, además tiene el potencial para controlar y otorgar información de dispositivos remotos.

IoT ha permitido la integración de software y hardware para ofrecer nuevos servicios como casas, ciudades o agricultura inteligente, su inclusión dentro de varios sectores industriales ha permitido que, países como Holanda, proporcionen un control eficiente a sus cultivos y se posicionen entre los principales exportadores de alimentos a nivel mundial. La agricultura es el segundo sector que más se exporta dentro del Ecuador, por lo cual, siguiendo el ejemplo de Holanda, se pretende otorgar nuevas herramientas a los agricultores para mejorar la producción de sus cultivos, además, dichas herramientas serán tanto informativas como interactivas, logrando así el establecimiento de la agricultura inteligente dentro del país (INFOAGRO, 2013).

Bajo el concepto de Smart Agro, varias aplicaciones y equipos ya se encuentran disponibles alrededor del mundo, sin embargo, la adquisición en el Ecuador resulta compleja y costosa, por lo cual, para lograr implementar esta tecnología dentro de la agricultura del país, se propone el uso de dispositivos de bajo costo y consumo de energía como son los módulos ESP8266, MKR WiFi 1010, Arduino Opla IoT. Otro problema que conllevan los equipos ya existentes es que poseen sus propias aplicaciones y estándares de funcionamiento, por esta razón se propone también la integración de la Web of Things (WoT) para la estandarización del envió y adquisición de información entre sensores y actuadores con los servicios web basados en microservicios, obteniendo así un sistema robusto, escalable y compatible con otros proyectos.

Estado del arte

Se han realizado investigaciones correspondientes a las diversas temáticas que aborda la propuesta presentada, diferentes publicaciones como tesis y artículos de revistas científicas han diseñado e implementado herramientas para generación de plantillas para TDT, las mismas que permiten desarrollar aplicaciones en su mayoría con interactividad local, en este mismo contexto se ha evidenciado la utilización también de otras herramientas como es NCL-Composer de donde han partido para la realización de aplicaciones específicas para distintos campos en los cuales enfocan su implementación, entre estos se encuentran: plataformas educativas como juegos de aprendizaje para niños, informativas como datos deportivos, clima e incluso la realización de noticieros con programaciones e interacción remota, otros campos que se explotan bastante son la salud y la seguridad que conllevan a la aparición de interacción entre IoT y TDT, esto debido a que se comienza a aprovechar de mejor manera el canal de retorno vía Internet, es así que varias de las investigaciones integran arquitecturas como SOA que rige la forma de comunicación vía Web entre las aplicaciones y los servidores con los cuáles se interactúa tanto para almacenar información como para otorgar datos y respuestas al usuario.

Una de las principales características del presente proyecto de investigación es la utilización de MDE para crear un DSL, con respecto a estas temáticas se ha encontrado importante información teórica y de aplicación especialmente con IoT, mas no conjuntamente con TDT, bajo contadas excepciones que, además, no utilizan microservicios como arquitectura principal.

Por tanto, se establece un resumen en la Tabla 1 donde se puede visualizar que existe la utilización de tecnologías que caracterizan la propuesta planteada en el proyecto, con el cual se tiene como objetivo de la investigación, integrar cada una de ellas y otorgar

un ámbito de aplicación hacia el principal sector industrial de nuestro país después del petróleo, que es la Agroindustria.

Tabla 1

Temas de investigación previos

DOCUMENTO	TEMAS	IoT	MDE	GUI	NLC-COMPOSER	SOA	WEB REST	TV DIGITAL
TESIS	TEMPLATE GENERATOR, SOFTWARE PARA LA GENERACIÓN DE PLANTILLAS INTERACTIVAS GINGA - NCL (Carlos Pillajo, ESPE, 2015)	X	X	✓	X	X	X	✓
	DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE AUTOMATIZACIÓN PARA HOGARES, CON APLICACION INTERACTIVA PARA LA TELEVISIÓN DIGITAL BASADA EN LA PLATAFORMA GINGA. (Belén Enríquez, ESPE, 2018)	✓	X	X	✓	✓	X	✓
	DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN INTERACTIVA BASADA EN GINGA-NCL PARA TELEVISIÓN DIGITAL ENFOCADA EN LA INFORMACIÓN DEL CLIMA (AILED VELÍN, EPN,2015)	X	X	X	✓	✓	X	✓
	DESARROLLO DE UNA APLICACIÓN INTERACTIVA EDUCATIVA MEDIANTE LA PLATAFORMA GINGA NCL/LUA PARA EL ESTÁNDAR ISDB-TB DE TELEVISIÓN DIGITAL (Saúl Araujo, U ISRAEL, 2019)	X	X	X	✓	✓	X	✓
	SOLUCIÓN DE INTERACTIVIDAD PARA TV DIGITAL USANDO CANAL DE RETORNO VIA INTERNET (Wanessa de Sousa,UNICEUB, 2010)	X	X	X	✓	✓	X	✓
	TV DIGITAL FIJA UTILIZANDO MIDDLEWARE GINGA-NCL APLICADO A UN NOTICIERO DIGITAL(T. Muñoz, L.Siguenza. Universidad de Cuenca 2012)	X	X	X	✓	✓	X	✓
PUBLICACIONES	UNA ARQUITECTURA DE INTEGRACIÓN TECNOLÓGICA DE INTERNET DE LAS COSAS Y COMPUTACIÓN EN LA NUBE (Piedra C, Tenezaca P. Universidad de Cuenca, 2018)	X	✓	X	X	X	✓	X
	METAMODELO PARA LA INTEGRACIÓN DEL INTERNET DE LAS COSAS Y REDES SOCIALES (Rodriguez J. Universidad de Oviedo, 2017)	✓	✓	X	X	✓	X	X
	A DSL FOR THE DEVELOPMENT OF HETEROGENEOUS APPLICATIONS(D.Alulema, J. Criado, L. Iribarme).2017	✓	✓	X	X	✓	X	✓
	IoTV: Merging DTV and MDE Technologies on the Internet of Things(D.Alulema, J. Criado, L. Iribarme.2019)	✓	✓	✓	X	✓	X	✓
	MDE4IoT: Supporting the Internet of Things with Model-Driven Engineering (F. Ciccozzi, R. Spalazzese.2017)	✓	✓	X	X	✓	X	X
	Midgar: Domain-Specific Language to generate Smart Objects for an Internet of Things platform(C.Gonzáles, J. Pascual, E.Núñez. 2014)	✓	✓	X	X	✓	X	X
	Cyber-Physical Microservices (K.Thramboulidis,C. Vachtsevanou, A. Solanos. 2018)	✓	✓	X	X	X	✓	X
PROPUESTA	Diseño e implementación de un Lenguaje Específico de Dominio para TV digital empleando ingeniería de modelos aplicado a un entorno Smart Agro.	✓	✓	✓	X	X	✓	✓

Nota: Comparativa entre trabajos previos y propuesta de la investigación.

Análisis Bibliométrico

El análisis bibliométrico es utilizado para descubrir tendencias o concurrencias que se manifiestan en el desempeño de artículos científicos, revistas, publicaciones de investigación, y para indagar en la composición intelectual de un dominio específico en la literatura existente (Donthu, Kumar , Mukherjee, Pandey, & Marc Lim, 2021). Los datos de mayor importancia en el análisis bibliométrico tienden a ser masivos y de naturaleza objetiva (cantidad de citas bibliográficas, apariciones de palabras clave y temas), aunque sus interpretaciones se basan tanto en objetivos y evaluaciones subjetivas establecidas a través de técnicas y procedimientos para descifrar y mapear la información acumulativa y matices evolutivas de campos específicos al otorgar una interpretación comprensible a grandes cantidades de datos no estructurados de manera rigurosa (Donthu, Kumar , Mukherjee, Pandey, & Marc Lim, 2021).

Es válido recalcar que el surgimiento de bases de datos científicas entre las que se encuentran Scopus y Web of Science ha permitido que la obtención de grandes volúmenes de datos bibliométricos sea más simple, además con la aparición de software bibliométrico tales como Gephi, Leximancer y VOSviewer posibilitan el análisis de los datos de una forma mucho más útil y funcional, incrementando el interés de los investigadores en utilizar este tipo de análisis en la actualidad (Donthu, Kumar , Mukherjee, Pandey, & Marc Lim, 2021).

Web of Science

Web of Science es una red online que facilita el acceso a múltiples bases de datos que otorgan datos completos de bibliografías para diferentes ámbitos académicos, en el sector de la investigación es conocida como una herramienta capaz de otorgar al usuario la realización de análisis de la información para sus investigaciones. Web of Science

utiliza una gran cantidad de opciones que se describe a continuación (CLARIVATE, 2020):

- Utiliza la indexación de bibliografías para determinar la influencia e impacto de una idea desde sus primeras menciones hasta el presente.
- Hace notar las tendencias de los patrones de gran importancia para los investigadores, dichas tendencias muestran los temas relevantes en la actualidad.
- Finalmente facilita la visualización mediante gráficos estadísticos.

La Web of Science Core Collection alberga seis bases de datos en línea:

- Emerging Sources Citation Index
- Science Citation Index Expanded
- Social Sciences Citation Index
- Arts & Humanities Citation Index
- Book Citation Index
- Conference Proceedings Citation Index

Desde 2008, la Web of Science alberga una serie de índices de citas regionales:

- Chinese Science Citation Database
- SciELO Citation Index
- Korea Citation
- Russian Science Citation Index
- Arabic Regional Citation Index

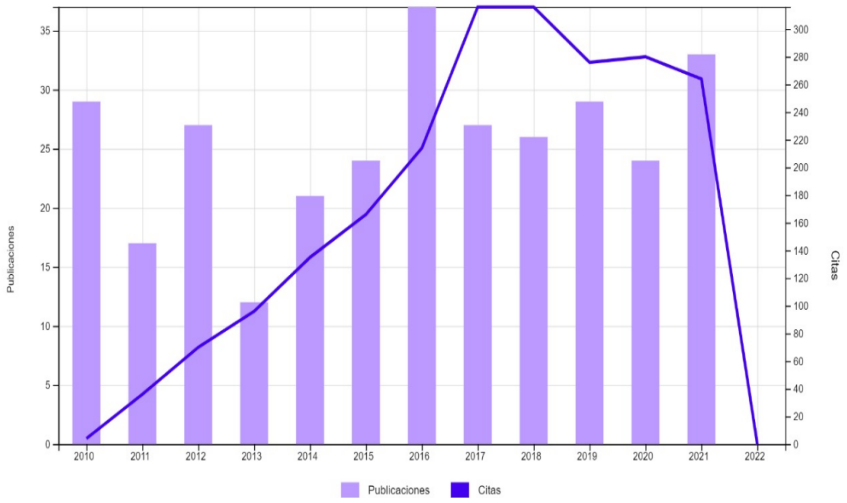
Gracias al uso de las herramientas citadas se han identificado las tendencias de los tópicos en publicaciones recientes, es así que se establecieron como parámetros

iniciales de búsqueda términos como: Televisión Digital Terrestre, IOT, SMART AGRO, MQTT y REST para el diseño de mapas en los cuáles se relacionan dichos términos bajo el concepto de incidencia y ocurrencia encontrados como más relevantes en los archivos de bases de datos.

En la Figura 1 se ha filtrado la información obtenida de la Web of Science respecto al término “Televisión Digital Terrestre”, donde se observa que actualmente existe una tendencia ascendente de investigaciones y artículos científicos, asemejándose al pico obtenido en el año 2016.

Figura 1

Publicaciones por año con el tema: “Televisión Digital Terrestre”

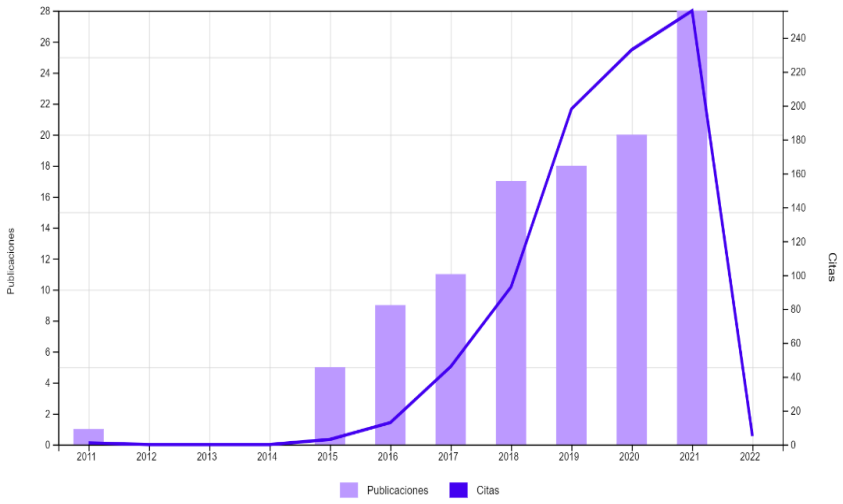


Nota: Cantidad de publicaciones y citas concernientes a Televisión Digital Terrestre en los últimos años, datos obtenidos de Web of Science.

La Figura 2 detalla la relación que existe entre las publicaciones y citas obtenidas para los términos TV e IOT, se encontraron un total de 109 publicaciones, de las cuáles se muestra una tendencia en ascenso hacia la investigación, relación y aplicación de estos temas.

Figura 2

Publicaciones e investigaciones relacionadas los términos TV e IOT.



Nota: Cantidad de publicaciones y citas que relacionan los términos TV e IOT en la plataforma Web of Science.

VOSviewer

VOSviewer es una herramienta que sirve para construir mapas de redes bibliográficas. Estas redes incluyen, por ejemplo, revistas, publicaciones de artículos individuales, y pueden construirse a partir de las relaciones de citación y acoplamiento bibliográfico para así visualizar co-ocurrencia de recursos extraídos de un cuerpo de literatura científica. A continuación, se resumen algunos aspectos destacados de VOSviewer (CWTS, Highlights, 2022).

Información

Web of Science, Lens y PubMed. Se pueden crear redes de coautoría, redes basadas en citas y redes de co-ocurrencia basadas en datos descargados de Web of Science, Scopus, Dimensions y Lens. Las redes de coautoría y las redes de co-ocurrencia también pueden crearse a partir de los datos de PubMed.

Visualización

- Zoom y desplazamiento
- Visualizaciones de densidad y superposición
- Capturas de pantalla

Técnicas

- Técnicas avanzadas de diseño y agrupación.
- Técnicas de procesamiento del lenguaje natural.
- Creación de redes bibliométricas (por ejemplo, redes de co-autoría, de acoplamiento bibliográfico y de co-citación).

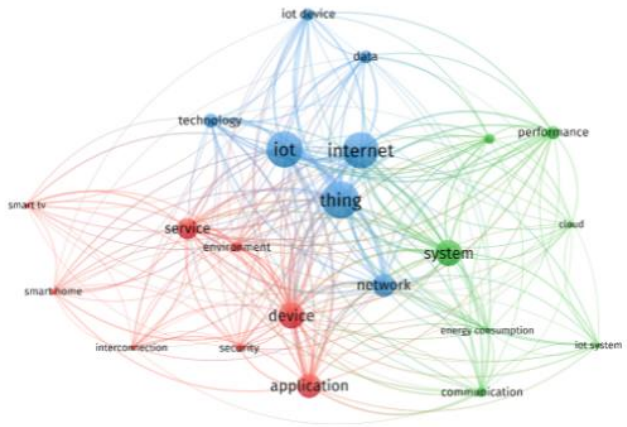
La disposición de la Figura 3 se refiere a la ventana de visualización tipo red en VOSviewer, en esta ventana de visualización se muestra la manera en que se enlazan las ocurrencias de los términos abordados; al aplicar lo expuesto en la Web of Science anteriormente, en VOSviewer se ha determinado 22 términos divididos en 3 grupos o clústeres los cuáles se caracterizan mediante colores, en este caso para el primer clúster (color rojo) se han agrupado 8 elementos: *application*, *device*, *environment*, *interconnection*, *security*, *service*, *smart home* y *smart tv*, para el segundo grupo (color verde) se tienen 7 términos: *cloud*, *communication*, *energy consumption*, *information*, *iot system*, *performance* y *system*, mientras que para el tercer clúster de color azul se han establecido los términos: *data*, *internet*, *iot*, *iot device*, *network*, *technology* y *thing*, esta caracterización se ha normalizado bajo el método de fuerza de asociación, con valores de atracción igual a 2 y repulsión igual a 1 integrados por defecto en la herramienta, esta fuerza de asociación dada por la aplicación detalla los grupos con características similares entre las 109 publicaciones encontradas; de la misma manera la dimensión de cada término detallado por el tamaño del círculo que lo contiene establece una mayor

ocurrencia y cantidad de relaciones con los demás términos, esto se refleja en los términos *iot*, *internet* y *thing* de color azul.

Basándose en el mapa expuesto, se verifica que los términos relacionados a IOT son los más destacados, adicionalmente se observan varias relaciones entre términos como Smart TV, sin embargo, no se ha encontrado un término que cuya concurrencia entre todas las publicaciones con los términos TDT, MDE o Smart Agro sea mayor a 5, denotando la posibilidad de abordar conjuntamente estos temas.

Figura 3

Coincidencias encontradas en Web of Science con los términos TV e IOT



Nota: Mapa relacional realizado de los términos repetidos dentro de las 109 publicaciones entre los términos TV e IOT.

Alcance

Con el siguiente trabajo se estima recolectar información de trabajos previos como tesis, artículos científicos sobre técnicas MDE, TDT, WoT, IoT, servicios web, hardware y software utilizado a fin de obtener un mayor grado de compresión de los parámetros que deben ser considerados para el diseño del DSL gráfico. Este DSL gráfico contendrá tres elementos: un metamodelo que cumpla con la sintaxis abstracta de acuerdo a las

características de interactividad de la TDT conjuntamente con IoT/WoT, un editor gráfico que conste de un lienzo y paleta de herramientas con secciones como botones, video principal, recursos multimedia o conectores para representar la relación entre todos los elementos, finalmente un motor de transformación de modelo a texto a fin de generar el código y obtener los archivos NCL-LUA correspondientes.

Mediante la utilización del DSL gráfico se diseñan las aplicaciones (local y remota) con enfoque al Smart Agro para la TDT, las cuales tendrán una interfaz de usuario amigable y permitirán que el usuario pueda acceder a información de sensores y actuadores en tiempo real, desplegados en huertos caseros inteligentes mediante el uso de servicios web e IoT. Una vez implementadas las aplicaciones de TDT y desplegados los sensores y actuadores, se realizarán pruebas de carga, usabilidad y de funcionamiento a fin de corregir errores que se puedan producir debido a sincronización de los sensores o actuadores con la interfaz de usuario desarrollada, esta evaluación permitirá obtener un sistema en óptimas condiciones y desarrollar las respectivas conclusiones del trabajo.

Objetivos

General

Desarrollar e implementar un DSL gráfico para la generación automática de código, empleando técnicas de MDE, con el propósito de crear componentes para interactividad local y remota en el dominio de Smart Agro y TDT.

Específicos

- Investigar los trabajos previos relacionados a la Ingeniería Dirigida por Modelos, Televisión Digital, IoT y Cloud Computing.
- Investigar el marco conceptual de las tecnologías relacionadas a la Ingeniería Dirigida por Modelos, Televisión Digital, IoT/WoT y Cloud Computing.

- Diseñar un DSL constituido por un metamodelo, un editor gráfico y una transformación de modelo a texto (M2T) para la integración de la TDT e IoT en el dominio del Smart Agro.
- Implementar el editor gráfico y la transformación M2T para la generación de los artefactos de software del sistema.
- Implementar un escenario de prueba para validar el funcionamiento del sistema.
- Evaluar el desempeño del sistema por medio de pruebas de usabilidad y rendimiento.

Capítulo II: Marco Conceptual

Televisión Digital Terrestre

La evolución correspondiente a la televisión analógica, abre paso al aparecimiento de la televisión digital de manera innata. Hoy en día, la información se genera digitalmente en el estudio. Estas señales se convierten en señales analógicas y se transmiten a los receptores de televisión analógicos. Con la televisión digital, los procesos de imagen, sonido y toda la información adicional se generan, transmiten y reciben como señales digitales, así se consigue la mejor definición de imagen y sonido (Sampaio de Alencar, 2009).

Un sistema de televisión digital está formado por un conjunto de normas que identifica los componentes básicos: video y audio representan los servicios esenciales para la difusión de la televisión digital; la interactividad y los nuevos servicios se añaden al sistema mediante el middleware. Estos nuevos servicios pueden utilizarse para ofrecer nuevos conceptos en la emisión de programas de televisión a los usuarios, o incluso para enviar datos para aplicaciones que no tienen una conexión directa con la programación televisiva (Crinon, y otros, 2006).

El estándar ISDTV

El proyecto para el desarrollo del Sistema Brasileño de Televisión Digital (SBTVD), que más tarde se conoció como Sistema Internacional de Televisión Digital (ISDTV o ISDB-Tb), se lanzó en noviembre de 2003. Más de cien instituciones participaron en el proyecto ISDTV, incluyendo la industria, las universidades, los centros de investigación y las empresas de radiodifusión. En febrero de 2006 se publicó el informe con las recomendaciones para el estándar ISDTV (Rodrigues de Carvalho, y otros, 2007).

La norma ISDTV utiliza una tecnología similar a la norma japonesa ISDB-T para la codificación y modulación de las señales de televisión digital. Las señales se transmiten

mediante la técnica de transmisión segmentada por banda (BST). El comité de la ISDTV adoptó H.264 como norma de compresión de vídeo. H.264 se utiliza para codificar tanto video estándar como de alta definición, así como videos de resolución reducida para receptores móviles o portátiles. La adopción de H.264 es una innovación clave en relación con todos los demás estándares de televisión digital (Sampaio de Alencar, 2009).

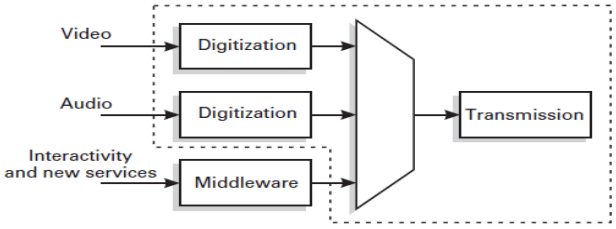
Middleware Ginga

Un middleware, por definición, es un software que une o interconecta diferentes sistemas. Un middleware también permite la interoperabilidad entre diferentes aplicaciones, permitiendo compartir información y funcionalidades entre sus sistemas. Por ejemplo, desde la perspectiva de la televisión digital, el middleware permite la interactividad entre los usuarios y el emisor o el proveedor de contenidos. Esto significa que un sistema que interactúa con los usuarios interoperará con el sistema de televisión digital a través de una vía de retorno (León Ruiz, 2011).

La Figura 4 muestra cómo las señales de video, audio y datos interactivos se multiplexan en un único flujo, donde el middleware actúa como interfaz entre la interactividad y el multiplexor.

Figura 4

Multiplexación de Televisión Digital

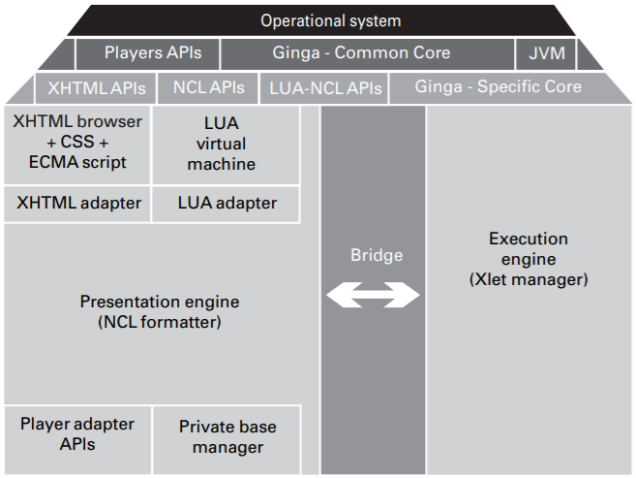


Nota: Obtenido de (León Ruiz, 2011).

Ginga-NCL es el entorno de aplicaciones declarativas del middleware Ginga que tiene como núcleo un lenguaje de contexto anidado (NCL), su arquitectura se muestra en la Figura 5. Un componente importante de Ginga es el motor para la ejecución de contenido procedimental, el cual contiene una máquina virtual Java. Los decodificadores de contenido comunes pueden utilizarse tanto para aplicaciones procedimentales para decodificar y presentar tipos de contenido comunes, como PNG, JPEG, MPEG, H.264 y otros tipos de formatos (Gomes Soares, Ferreira Rodrigues, & Ferreira Moreno, 2007).

Figura 5

Arquitectura de middleware Ginga



Nota: Obtenido de (Sampaio de Alencar, 2009).

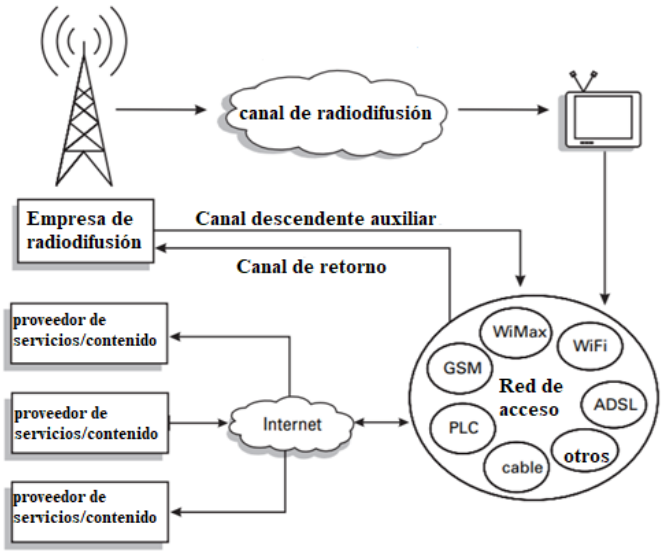
Canal de interactividad

Es responsable de todos los intercambios de información entre las aplicaciones interactivas que se ejecutan en los receptores de los usuarios y los servidores de aplicaciones situados en las emisoras de televisión y los proveedores de contenidos o servicios que están distribuidos por toda la red.

En la Figura 6 se presenta un diagrama más detallado que muestra la arquitectura básica del canal de interactividad. Se puede observar que hay dos tipos de rutas de comunicación. La primera establece el flujo de datos bidireccional entre las múltiples redes de comunicación y los servidores de aplicaciones. La segunda establece el flujo de datos bidireccional entre el servidor de aplicaciones y las empresas de radiodifusión o programadores. Se pueden adoptar múltiples tecnologías de acceso para el canal de interactividad, lo que confiere al sistema una característica muy interesante, con capilaridad, cobertura, número de usuarios y velocidad de transmisión.

Figura 6

Subsistema del canal de interactividad.



Nota: Obtenido de (Sampaio de Alencar, 2009).

Los componentes del canal de interactividad son los siguientes:

- Canal de bajada
- Canal de retorno
- Gateway

- Servidor de aplicaciones
- Módem
- Unidad de adaptación de la emisión
- Redes de acceso (redes de comunicación)

Servicios Remotos

Para la inclusión de los servicios remotos, es necesaria la implementación de un canal de retorno, así se establece la interacción del televidente para el envío de datos a un servidor. Básicamente, las consultas son enviadas a una base de datos remotos, navegación en línea, mensajería con servidores, entre otros. Tomando esto en cuenta, para el desarrollo de este tipo de aplicaciones en Ginga NCL, es indispensable el uso del Lua y en específico la extensión NCLua.

Los documentos NCL se escriben para una aplicación declarativa, con scripts Lua para codificar en un paradigma imperativo, como ocurre con HTML y JavaScript. En Lua, la comunicación con NCL y los elementos externos se logra con eventos, que pueden ser enviados y recibidos. De hecho, ya existe la comunicación de red, que proporciona medios para enviar y recibir datos a través de conexiones TCP, con un mecanismo de comunicación asíncrona en donde se debe registrar una función, para proporcionar el código de manejo.

TCP Lua dispone de funciones para el desarrollo de aplicaciones para la TDT que hacen el uso del canal de retorno. Estás aplicaciones suelen estar relacionadas a ámbitos educativos, de comercio y gubernamentales (Paucar Curasma, Velásquez Díaz, Mendoza Villaizán, Diaz Ataucuri, & Villanueva, 2009).

Televisión Digital Terrestre en Ecuador

El servicio de televisión digital terrestre (TDT) se encuentra en proceso de implementación en varios países de latinoamérica y Ecuador no es ajeno a esta tecnológica ya que desde el año 2010 ha iniciado el proceso de migración hacia la Televisión Digital Terrestre adoptando el estándar ISDB-Tb Internacional, el cual ya puede acceder a las señales de televisión digital en algunas de las principales ciudades como Quito, Guayaquil, Cuenca, Ambato, entre otras, otorgando una señal con gran área de cobertura, comparable o incluso mejor con relación a otros estándares (Viceministerio de Tecnologías de la Información y Comunicación , 2018).

Paulatinamente los canales de televisión regulares han adoptado el proceso de digitalización, comenzando con señales de prueba. Muchos de ellos, como Ecuavisa, Teleamazonas, TC Televisión, Gama TV, RTS, muestran en algunas de sus transmisiones una transformación en cuanto a la mejora de la calidad de la imagen y el sonido; sin embargo, no hay avances en la implementación de la interactividad. El 100% de los canales no ofrecen ninguna interactividad para el usuario (Martinez, Lucano, & Pazmiño, 2017). Este aspecto se irá implementando paulatinamente en el momento en que los canales incluyan contenidos propios con lenguajes de programación como Ginga.

(Martinez, Lucano, & Pazmiño, 2017) establecen el siguiente análisis con respecto a la situación de la TDT en el país:

- Los contenidos digitales en las transmisiones de prueba de la TDT en Ecuador son limitados porque sólo replican la señal analógica sin aportar valor añadido. Además, se carece de una plataforma para aplicar la interactividad para los televidentes.
- La baja tasa de televisores con IDTV en el país resulta ser uno de los problemas más complejos porque los usuarios deben adquirir nuevos televisores,

decodificadores o un sistema de televisión de pago para ver los nuevos contenidos digitales.

- Los canales de televisión ecuatorianos deben trabajar en contenidos multimedia, con la posibilidad de ofrecer metadatos complementarios y de apoyo a la información transmitida, sin dejar de lado las posibilidades de interactividad de los contenidos para los usuarios de la televisión no paga en el país.

Modelo de contexto anidado (NCM)

El Lenguaje de Contexto Anidado que se despliega a partir del Modelo de Contexto Anidado (NCM). Fue elegido por el grupo del SBTVD para ser el lenguaje declarativo del sistema de televisión digital brasileño.

Elementos de NCM

NCM se basa en el contexto de nodos y enlaces. Los nodos contienen información (como objetos multimedia) y los enlaces establecen las relaciones entre ellos. Cada aplicación basada en el NCM es una Entidad, y cada Entidad tiene propiedades y elementos. La esencia de la aplicación basada en NCM es la Entidad. Se representa a sí misma y a todo el contexto donde se ejecuta, los elementos básicos de la Entidad son el Descriptor, el Conector, el Enlace y el Nodo.

El Nodo es el elemento más importante, pueden existir muchos nodos al mismo tiempo, y cada uno de ellos contiene algún tipo de información. También se denomina objeto NCM, y está compuesto por un identificador, su contenido (la información propiamente dicha) y un conjunto de anclas. NCM es un modelo basado en la estructura y no en los medios. Esto significa que la programación será independiente del contenido de los nodos, ya que el modelo opera sobre relaciones entre nodos y no entre medios, clasifica cada nodo de medios en subclases dependiendo del tipo de medios que contenga el Nodo (es decir, texto, imagen, audio, video, objetos Lua, objetos Java, objetos

HTML, etc.). Los nodos se clasifican por su especialización: nodos de medios o contenido, compuestos y de contexto (León Ruiz, 2011).

Los anclajes, que forman parte del Nodo, son propiedades que describen cómo el Nodo mostrará su contenido, son completamente independientes del contenido, y se definen también por separado. Un Descriptor es una propiedad de la entidad NCM que rige cómo se mostrará un nodo (dónde y cuándo). Tiene la capacidad de asignar cierta región de la pantalla, y puede decidir cuándo se mostrará (sincronización espacio-temporal) (Gomes Soares, Ferreira Rodrigues, & Ferreira Moreno, 2007). Después del Descriptor, el Conector es un elemento clave también, ya que crea roles y enlaces para gobernar el flujo del programa. Diseña la máquina de estados definiendo cada estado y cómo el programa pasará de un estado a otro. Por último, el elemento Enlace pondrá en marcha las reglas del Conector. Se encarga de asignar esos roles y enlaces a los nodos, para que obedezcan a la máquina de estados diseñada.

Eventos NCM

Los eventos en NCM son la parte de comportamiento del entorno declarativo. Son la base de la máquina de estados de NCM, y permiten que los lenguajes basados en NCM (como NCL) definan ciertos estados para diferentes nodos, y que la presentación se adapte en función de los estados actuales de los nodos.

Cada evento tiene su propia máquina de estados definida por NCM, los eventos básicos definidos por NCM son los siguientes:

- Evento de presentación: es un evento que representa la exposición de un nodo de contenido en función de un descriptor específico y una situación concreta. Esto significa que diferentes descriptores y diferentes situaciones pueden aplicarse para eventos de presentación únicos.

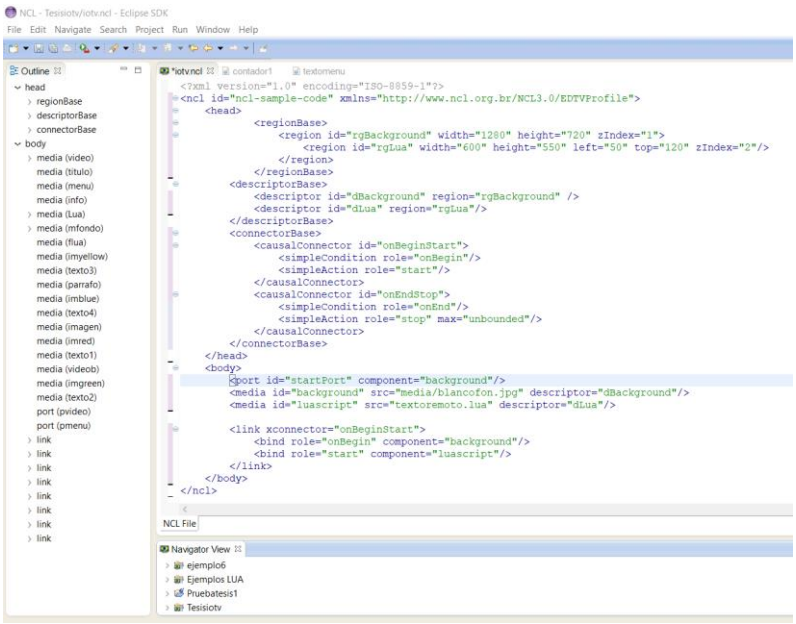
- Evento de Composición: es el evento que presenta el mapa compuesto.
- Evento de Selección: es un evento similar al Evento de Presentación, que se activa por la entrada del usuario. En el contexto de la televisión digital, podría ser un mando a distancia.
- Evento de Atribución: es un evento que se dispara con los anclajes específicos de un nodo.

Lenguaje de contexto anidado (NCL)

La Figura 7 muestra un ejemplo básico de un programa desarrollado en NCL. El aspecto más llamativo del código NCL es el hecho de que se trata de un documento basado en XML. NCL es un lenguaje declarativo basado en el modelo NCM, pero siguiendo el esquema XML para la compatibilidad entre sistemas e Internet (León Ruiz, 2011).

Figura 7

Código NCL desarrollado en Eclipse.



La estructura básica de una aplicación en NCL se compone de elementos representados con las siguientes etiquetas:

- Etiqueta <ncl>: Esta etiqueta representa la raíz del documento NCL, además es la contenedora de todos los demás elementos a implementarse en la aplicación.
- Etiqueta <head>: Esta etiqueta es la contenedora de los elementos de la cabecera, estos elementos promueven las funciones de reutilización en NCL o a su vez definen la información para los agentes de las aplicaciones.
- Etiqueta <body>: Esta etiqueta es la encargada de contener a todos los elementos multimedia con sus respectivas relaciones.

Dentro de la cabecera se definen elementos de regiones y descriptores como:

- Etiqueta <regionBase>: La cual agrupa un conjunto de elementos de tipo <region> que definen valores iniciales como las posiciones (top, left, down, right), tamaño (width, height) o profundidad (zIndex) de los elementos a utilizarse en la aplicación de la TDT.
- Etiqueta <descriptorBase>: La cual agrupa un conjunto de elementos <descriptor> los cuales definen los valores iniciales de las propiedades de los archivos multimedia.

Dentro del cuerpo del programa se definen elementos multimedia o conectores para relacionar funciones a los distintos elementos:

- Etiqueta <media>: Define el elemento multimedia a través de los atributos source e id y los asigna a un descriptor creado en la sección de la cabecera.

- Etiqueta <port>: Define un puerto por el cual se puede acceder al elemento multimedia, sus principales atributos son el id y el componente <media id> a mostrarse.
- Etiqueta <connectorBase>: Define un grupo de elementos denominados <connector> los cuales son los responsables de las causas y efectos en la interacción de las aplicaciones.
- Etiqueta <importBase>: Agrupa relaciones definidas mediante conectores, estos pueden ser importados mediante bibliotecas de base de conectores creados previamente como por ejemplo el archivo ConnectorBase.ncl que alberga una gran cantidad de acciones causa y efecto listas a ser utilizadas por el usuario.
- Etiqueta <link>: Asigna las acciones de interactividad declaradas en <connectorBase> a los elementos multimedia, generalmente se declaran acciones de causa y efecto entre los botones y los archivos multimedia.

NCLua

Lua es un lenguaje de programación enfocado para la creación de secuencias de comandos (scripting) para ser integrados en otros programas, los componentes principales de Lua son:

- El intérprete de Lua: Compila un archivo Lua en código de bytes, está escrito en ANSI C para ser un lenguaje portátil y compatible con varios dispositivos.
- La máquina virtual Lua: Ejecuta el código de bytes compilado, aumenta el rendimiento general de la aplicación.

Lua se relaciona a NCL mediante el uso de las librerías de NCLua, estas librerías se basan en la biblioteca estándar de Lua, y proporcionan al programador otros 5 módulos para la presentación de contenidos (la API NCLua) (León Ruiz, 2011).

La API de NCLua incluye los siguientes módulos:

- módulo ncledit: permite a las aplicaciones NCLua modificar el documento NCL.
- módulo canvas: es el módulo NCLua para manipular imágenes y dibujar en pantalla.
- módulo event: permite a las aplicaciones NCLua comunicarse con el reproductor NCL a través de eventos NCL.
- módulo settings: exporta una tabla con variables para que la aplicación NCLua las recupere.
- módulo persistent: presenta la API NCLua con variables que persisten después de que la aplicación termina.

Si bien los 5 módulos expuestos son la base de la NCLua, es importante hablar de las funciones del módulo canvas o librerías utilizadas para la creación de contenido interactivo, puesto que el usuario tiene acceso directo al uso de dichas funciones.

Módulo Canvas

El módulo canvas permite crear objetos canvas (lienzos) que almacena en su estado los atributos con los cuales operan las primitivas gráficas (imágenes, cuadros de texto). Las principales funciones de este módulo son las siguientes:

- Canvas:new (): Crea nuevos objetos gráficos a partir de un lienzo.
- Canvas:attrSize(): Devuelve las dimensiones del lienzo en píxeles.
- Canvas:attrColor(): Accede o modifica al atributo color del lienzo.

- Canvas:drawRect(): Dibuja un rectángulo en el lienzo, sus parámetros son modo(fill o frame),x,y,ancho,largo.
- Canvas:drawText(): Dibuja un texto, sus parámetros son x,y, “texto”
- Canvas:compose(): Permite mostrar imágenes entre lienzos.
- Canvas:flush(): Actualiza el lienzo tras las operaciones de dibujo o composición.
- Canvas:clear(): Elimina los elementos del lienzo.

Librería I/O

La librería de entradas y salidas (input and output) de Lua permite realizar la manipulación de archivos basada en dos modelos, uno simple y uno compuesto. El modelo I/O simple opera en archivos actuales, es decir, inicializa el archivo de entrada actual y el archivo de salida es el mismo, mientras que en el modelo compuesto se usan identificadores de archivo para brindar una operatividad sobre varios archivos.

Entre las principales funciones de la librería I/O se encuentran:

- io.open(): Esta función define el archivo con el cual operarán las demás funciones, también permite establecer si el archivo es de tipo lectura o modificable.
- File:write(): Permite introducir información dentro del archivo previamente abierto.
- File:close(): Permite guardar y cerrar el archivo actual.
- File:read(): Permite leer la información del archivo.
- String.find(): Devuelve la posición inicial y final de una cadena de caracteres específicos dentro del archivo.
- String.sub(): Permite desplazarse a través de posiciones del archivo.

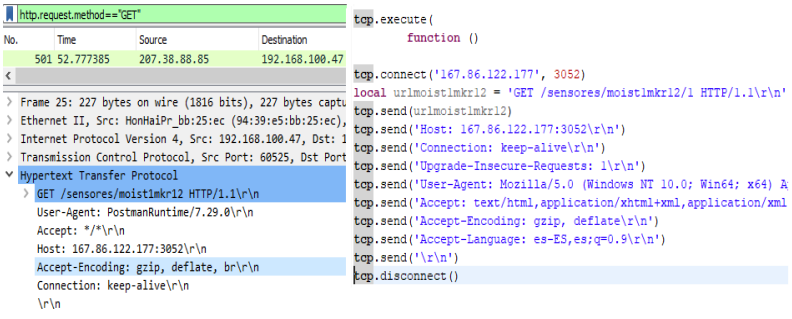
Librería tcp.lua

Esta librería permite realizar conexiones tcp con servidores web a través de peticiones de tipo GET y POST, su composición se basa de los siguientes elementos:

- require ‘tcp’: Esta línea de comandos permite llamar a la librería para hacer el uso de sus funciones.
- tcp.execute(): Es la función principal de la librería, en su contenido se debe definir los argumentos para que se realice la conexión TCP con el servidor remoto, en la Figura 8 se puede observar la estructura que debe llevar la función tcp.execute.
- tcp.connect(): Permite la conexión hacia un servidor remoto, se debe definir el host y número de puerto del servicio como ejemplo *tcp.connect('167.86.122.177', 3052).*
- tcp.send(): Permite enviar las peticiones de los métodos de publicación o recepción de datos HTTP (POST y GET) como ejemplo *tcp.send('GET /sensores/moist1mkr12/1 HTTP/1.1\r\n')*
- tcp.disconnect(): Permite cerrar la comunicación hacia el servidor remoto, para evitar colas.

Figura 8

Estructura de la función tcp.lua



Nota: En la parte izquierda se visualiza la estructura obtenida desde la herramienta Wireshark y en la derecha el código en Lua.

Entornos de desarrollo

Eclipse

El IDE (Entorno de Desarrollo Integrado) Java de Eclipse es un entorno de desarrollo de software de código abierto con una gran capacidad de plug-in. Originalmente comenzó como un IDE de Java, pero tiene varios plug-ins que permiten programar en PHP, C/C++, HTML y muchos más lenguajes.

Eclipse está basado en Java, por lo que es multiplataforma y tiene versiones para Microsoft Windows, Linux y Mac OS. El IDE de Eclipse permite por sí mismo la gestión de proyectos, el resaltado de sintaxis y la finalización de código (para Java), y tiene un registro de mensajes para advertencias y errores de compilación. Los desarrolladores de complementos han aprovechado estas características para crear complementos adecuados para sus lenguajes específicos. Este es el caso del plug-in NCL Eclipse, desarrollado por una asociación entre la Universidade Federal do Maranhão y la PUC-RIO.

NCL Eclipse

NCL Eclipse es un plug-in para el IDE Eclipse destinado a facilitar la creación de documentos NCL para la televisión digital. Entre sus principales características se encuentran el resaltado de sintaxis NCL, el plegado de código (para ocultar o mostrar partes del código fuente), los asistentes, el autoformato, la validación de errores y la finalización de código (Azevedo, Soares Neto, Meireles Teixeira, & Mesquita Santos, 2011).

Tras las mejoras introducidas en las versiones de NCL Eclipse, se implementaron más funciones relativas a la visualización y a las vistas previas de los medios. Hasta 2010, basándose en las cifras de descargas, se podría argumentar que NCL Eclipse es, quizás,

la herramienta más utilizada para el desarrollo de NCL. Es un plug-in muy robusto, y tiene una comunidad muy grande que proporciona retroalimentación (León Ruiz, 2011).

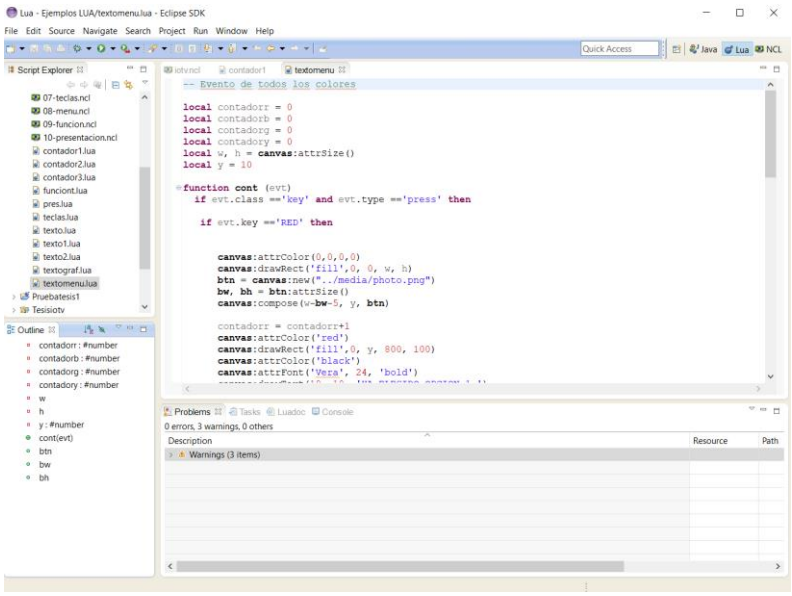
Lua Eclipse

El otro complemento más relevante para el IDE de Eclipse es el plug-in Lua Eclipse. Esta herramienta proporciona un entorno con completado de código, resaltado de sintaxis, plegado de código como NCL Eclipse también. Además, proporciona una interfaz para la depuración rápida utilizando un intérprete preconfigurado (dado que Lua no es un lenguaje compilado, sino de análisis, no tiene un compilador, sino un intérprete).

Proporciona completado de código para la sintaxis estándar de Lua, es realmente útil para escribir código NCLua, y el intérprete puede ayudar a depurar y probar las partes de código Lua estándar. Para que el complemento del intérprete funcione, es necesario instalar el intérprete de Lua (también gratuito). La Figura 9 ilustra un ejemplo del plug-in interactuando con el intérprete preconfigurado.

Figura 9

Código desarrollado en LUA Eclipse



Arquitectura Dirigida por Modelos MDA

OMG (Object Management Group) propone este enfoque Dirigido por Modelos, disponible desde principios de 2000, y centrado principalmente en la definición de modelos y sus transformaciones. MDA permite establecer varios modelos con diferentes niveles de abstracción, tales como: CIM (Computational Independent Models), PIM (Platform Independent Models) y PSM (Platform Specific Models) (OMG, 2001).

Las plataformas computacionales corresponden a implementaciones concretas de servidores de aplicaciones, servidores de bases de datos, sistemas de gestión de contenidos, frameworks y arquitecturas de software; estas plataformas pueden describirse por sí mismas mediante PDM (Platform Description Models). MDA también considera diferentes tipos de transformaciones de modelo a modelo, a saber: CIM-CIM, CIM-PIM, PIM-PIM, PIM-PSM y PSM-PSM. Además, considera la transformación de modelos PSM en código fuente y otros tipos de artefactos textuales (PSM-Text). En teoría, una aplicación desarrollada bajo el enfoque MDA es independiente de la plataforma, lo que permite instalarla en diferentes plataformas informáticas y soportar adecuadamente diferentes tecnologías gracias a estas transformaciones (Rodrigues da Silva, 2015).

Enfoques Dirigidos por Modelos de Alto Nivel

MDE se encuentra en la cúspide de la abstracción de los enfoques Dirigidos por Modelos. Es una perspectiva de ingeniería de software la cual considera que los modelos pueden utilizarse en múltiples áreas de la tecnología y en cualquier ámbito aplicativo. En este mismo nivel de abstracción se encuentran MDD y MBT, el enfoque MDD (Model-Driven Development) se centra principalmente en las disciplinas de requisitos, análisis y diseño e implementación. Los enfoques MDD concretos tienden a definir lenguajes de modelado para especificar el SUS (System Under Study) a diferentes niveles de

abstracción, para proporcionar transformaciones M2M y M2T con el fin de mejorar la productividad y la calidad del proceso y del sistema de software final (Stahl & Völter, 2006).

Por otro lado, MBT (Model-Based Testing) se especializa en el ámbito de automatizar las pruebas. Los modelos de pruebas se utilizan para representar el comportamiento deseado del SUT (System Under Test), para representar las estrategias de pruebas y el entorno de pruebas. Un modelo de pruebas que describe el SUT suele ser una representación abstracta y parcial del comportamiento deseado del SUT (Rodrigues da Silva, 2015).

Ingeniería Dirigida Por Modelos (MDE)

MDE promueve el uso activo de modelos a lo largo del proceso de desarrollo de software, lo que conduce a una generación automatizada de la aplicación final. Estos modelos se definen a veces utilizando lenguajes de modelado de propósito general como UML (Unified Modeling Language), pero para dominios restringidos y conocidos, también es frecuente el uso de lenguajes de modelado de dominio específico (DSL) adaptados al dominio de aplicación y a los objetivos del proyecto (de Lara, Guerra, & Sánchez Cuadrado, 2015).

En MDE, los modelos representan el concepto central y se consideran una abstracción del sistema en desarrollo. Las reglas y restricciones para la construcción de modelos se describen a través de la correspondiente definición del lenguaje de modelado, además de la abstracción, un pilar fundamental del MDE es la provisión de automatización en términos de manipulación y refinamiento de modelos, que se realiza a través de transformaciones de modelos (Czarnecki & Helsen, 2003). La heterogeneidad del software y el hardware es al mismo tiempo un punto fuerte y un gran reto dentro de IoT. Gracias a los lenguajes de modelado, y más concretamente a los lenguajes de

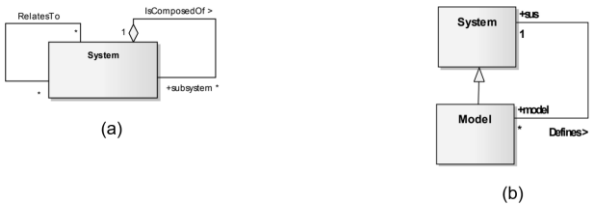
modelado de dominio específico (DSL), el MDE puede proporcionar medios únicos para representar los numerosos aspectos de los sistemas heterogéneos en un solo lugar. Los modelos definidos a través de estos lenguajes están pensados para estar mucho más orientados al ser humano que los artefactos de código comunes, que están naturalmente orientados a la máquina. Esto significa que, por ejemplo, el software puede definirse con conceptos que no dependen necesariamente de la plataforma o tecnología subyacente (Ciccozzi & Spalazzese, 2016).

Sistemas y Modelos

En el contexto del MDE, se define sistema como: “Un concepto genérico para designar una aplicación de software, una plataforma de software o cualquier otro artefacto de software". Además, como se sugiere en la Figura 10 (a), un sistema puede estar compuesto por otros subsistemas y un sistema puede tener relaciones con otros sistemas. Un modelo es una abstracción de un sistema que puede existir o que se pretende que exista en el futuro. En particular, cuando se piensa en un modelo de un modelo, se tiene que considerar que uno de ellos desempeña el papel del sistema en estudio y, en consecuencia, es en sí mismo un sistema. En resumen, y como se sugiere en la Figura 10 (b), se define modelo como “un sistema que ayuda a definir y dar respuestas del sistema en estudio sin necesidad de considerarlo directamente" (Rodrigues da Silva, 2015).

Figura 10

Relaciones entre modelo y sistema



Nota: Obtenido de (Rodrigues da Silva, 2015).

La utilidad descrita por los modelos en ingeniería se refiere a proporcionar distintos puntos de vista, de la misma manera, es necesario determinar qué información se destaca en busca de satisfacer la finalidad de cada punto de vista. La representación simplificada de los modelos pierde su aplicación práctica al incluir en ellos la información completa del sistema. Suelen señalarse cinco características de un modelo útil (Selic, 2003):

- Abstracto
- Comprensible
- Preciso
- Predictivo
- Barato

Metamodelo

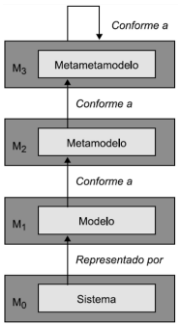
Se determina como un modelo que caracteriza a un conjunto de modelos, en el contexto de MDE, es común aplicar un modelado conceptual orientado a objetos, donde los metamodelos se crean a partir de conceptos como: clase, atributo, asociación y generalización. Si se utilizan los conceptos orientados a objetos, el modelado de un lenguaje se puede realizar teniendo en cuenta que:

- Las declaraciones del lenguaje se declaran como clases.
- Los atributos de las declaraciones, como parámetros de las clases.
- Las relaciones entre declaraciones como referencias, agregación o composiciones.
- El concepto del lenguaje se comprime en un solo paquete.

Según la propuesta de la OMG (Object Management Group), el procedimiento recursivo de definir modelos conforme a otros modelos de mayor grado de abstracción acaba cuando se alcanza el nivel de meta metamodelo, ya que los meta metamodelos se dice que son conformes a ellos mismos, tal y como se ilustra en la Figura 11 (Durán Muñoz, Troya Castilla, & Vallecillo Moreno, 2015).

Figura 11

Organización en cuatro niveles de OMG



Nota: Obtenido de (Durán Muñoz, Troya Castilla, & Vallecillo Moreno, 2015).

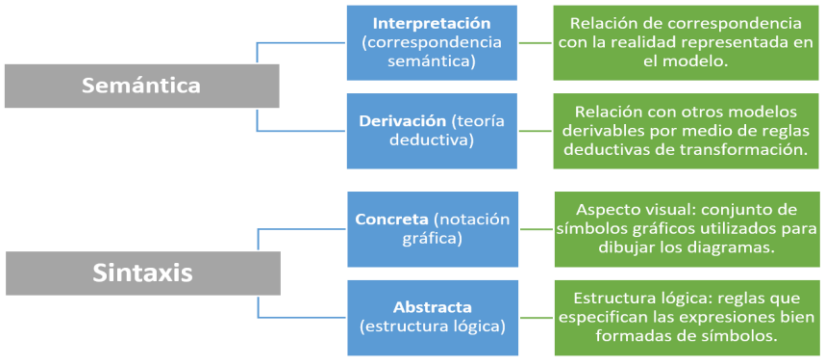
Semántica y Sintaxis

La definición completa de un lenguaje de modelado necesariamente incluye los aspectos detallados en la Figura 12. Estos cuatro aspectos no son completamente independientes. Por un lado, la sintaxis concreta se define en función de la sintaxis abstracta, referente a que se define un símbolo gráfico para cada concepto establecido en la sintaxis abstracta, este símbolo pretende expresar de manera adecuada los

atributos de cada concepto mediante representaciones de nodos y arcos(García Molina, y otros, 2012).

Figura 12

Aspectos del lenguaje de modelos.



Nota: Obtenido de (García Molina, y otros, 2012).

DSL

Un Lenguaje Específico de Dominio (DSL) es una solución específica para resolver un problema específico de un dominio concreto. Un DSL aumenta la productividad, reduce los errores, es portable, tiene un fácil mantenimiento y permite su reutilización para diferentes propósitos. Sin embargo, un DSL tiene una peor eficiencia que la codificación nativa y tiene una mayor dificultad para crearlo porque los desarrolladores necesitan una abstracción del problema específico (González García, Pascual Espada, Núñez-Valdez, & García-Díaz, 2014).

Un DSL sirve para hacer que los aspectos claves de un dominio sean formalmente expresables y modelables. Para ello, posee un metamodelo, que incluye su semántica estática, y una sintaxis concreta correspondiente. La semántica de un DSL es relevante en varios aspectos: por un lado, el modelador debe conocer el significado de los elementos del lenguaje a su disposición para poder crear modelos razonables, mientras

que, por otro, las transformaciones automáticas de los modelos deben ejecutar exactamente esta semántica. La semántica de un DSL debe estar bien documentada o ser intuitivamente clara para el modelador. Esto es más fácil si el DSL adopta conceptos del espacio del problema, de modo que un experto en el dominio reconocerá su "lenguaje de dominio" (Stahl & Völter, 2006).

Productos de software, plataformas y transformaciones

El enfoque MDE afirma que el uso de lenguajes de modelado ayuda a especificar modelos en un determinado nivel de abstracción, y también que esos modelos se utilizan para apoyar el desarrollo de aplicaciones de software (Atkinson & Kühne, 2003). Una aplicación de software (o producto de software) se define como:

> "un sistema compuesto por una integración no trivial de plataformas de software, artefactos generados a través de transformaciones M2T, artefactos directamente escritos por los desarrolladores y, eventualmente, modelos directamente ejecutables en el contexto de una plataforma de software concreta" (Rodrigues da Silva, 2015).

Para esto se detallan algunas características principales:

- Las plataformas de software significan un conjunto integrado de elementos computacionales que permiten el desarrollo y la ejecución de una clase de productos de software, se refieren a tecnologías como middleware, bibliotecas de software, marcos de aplicación y componentes de software, pero también sistemas de gestión de bases de datos, servidores web, sistemas de gestión de contenidos y de documentos, sistemas de gestión de flujos de trabajo, etc.
- Los artefactos generados y no generados también son elementos de la aplicación de software. Se pueden considerar muchos ejemplos de estos artefactos: archivos de código fuente y binario, scripts de configuración y despliegue, scripts de bases

de datos e incluso archivos de documentación, incluyendo los propios modelos (Rodrigues da Silva, 2015).

- En el enfoque MDE se suelen considerar dos tipos principales de transformaciones. Por un lado, las transformaciones de modelo a texto (M2T) que generan o producen artefactos de software, normalmente código fuente, XML y otros archivos de texto, a partir de modelos (Czarnecki & Helsen, 2003). Por otro lado, las transformaciones de modelo a modelo (M2M) permiten traducir los modelos en otro conjunto de modelos, normalmente más cercanos al dominio de la solución o que satisfacen necesidades específicas para diferentes partes interesadas. Estas transformaciones se especifican a través de distintos lenguajes, como los principales lenguajes de programación, pero también a través de lenguajes especializados de transformación de modelos, con diferentes propósitos y con diferentes paradigmas de modelado como QVT1, Acceleo2, ATL3, VIATRA4, DSLTrans5 (Czarnecki & Helsen, 2003).

Herramientas MDE

La mayoría de estas herramientas proporcionan una colección de características para ayudar a los usuarios a definir DSL's, con editores específicos, validación de modelos, transformación de modelos, etc. A continuación, se presenta brevemente Eclipse EMF, Microsoft Software Factories, JetBrains MPS, pero también podrían considerarse algunas otras como: MetaEdit+, SDF/Stratego/Spoofax, xText, Obeo Designer/Sirius o algunas propuestas académicas como MIC (Model Integrated Computing) con la herramienta GME (Generic Modeling Environment), VMTS, MetaSketch o AtomPM por citar varios nombres (Rodrigues da Silva, 2015).

Eclipse Modeling Project

El proyecto Eclipse Modeling Project se centra en la evolución y promoción de las tecnologías de desarrollo basadas en modelos dentro de la comunidad Eclipse, proporcionando un conjunto unificado de marcos de modelado, herramientas e implementaciones de estándares (ECLIPSE, 2010).

EMF (Eclipse Modeling Framework) es un marco de modelado y herramienta de generación de código para la construcción de herramientas y demás aplicaciones centradas en un modelo estructurado de datos. Partiendo de la especificación de modelo descrita en XMI, EMF otorga herramientas y soporte de tiempo de ejecución para establecer un conjunto de clases Java para el modelo, unido a un conjunto de clases adaptadoras que faculta la visualización y edición referidos a comandos del modelo, y un editor básico (ECLIPSE, 2009).

Hay varias herramientas y marcos desarrollados sobre EMF, como GMF, Sirius, etc. En general, la mayoría de estas herramientas son populares, relativamente fáciles de usar y mantener, y cuentan con un apoyo abierto y fuerte de la comunidad (Rodrigues da Silva, 2015).

Internet de las cosas

El internet de las cosas (IOT) es un concepto habitual en los últimos años debido a los beneficios tecnológicos y comodidades que ofrece a las industrias y población. En términos generales el IOT es considerado como una red digital de objetos inteligentes, todos ellos conectados, comunicándose, compartiendo información y reorganizándose ante distintos escenarios constantemente (Keyur K Patel, 2016). Entre las principales características del IOT se encuentran las siguiente:

- Interconectividad: Para que cualquier objeto inteligente pueda ser conectado a la infraestructura global de la información y comunicación del sistema.
- Heterogeneidad: Para que los objetos inteligentes basados en distintos hardware o software, puedan interactuar con otros dispositivos a través de diferentes redes.
- Conectividad: Permite la accesibilidad para entrar en una red y la compatibilidad para consumir y producir datos.

Protocolo MQTT

El término de transporte telemétrico de mensajes queuing o mejor conocido como MQTT es un protocolo de mensajería liviano que se basa en publicación y suscripción, es utilizado comúnmente entre dispositivos o máquinas IOT. La ventaja del uso de este protocolo radica en que tanto el publicador como suscriptor (cliente MQTT) no necesitan conocerse, debido a que existe un broker (servidor MQTT) como mediador para el intercambio de mensajes. (Pongnapat Jutadhamakorn, 2017).

El procedimiento de conexión mediante el este protocolo empieza cuando se envía un mensaje de conexión al broker, una vez obtenida la respuesta por parte del broker y que la conexión haya sido realizada, el cliente puede publicar o suscribir mensajes mediante la utilización de un mensaje publish o subscribe, estos mensajes constan de un tópico (cadena de caracteres como MQTT, por ejemplo) para que el broker pueda enviar el mensaje.

Sensores

Los sensores son dispositivos capaces de identificar y convertir alteraciones ambientales (fenómenos físicos como luz, temperatura, movimiento) en señales eléctricas, Existen varios tipos de sensores según el fenómeno físico que se desee

analizar, sin embargo, para el presente proyecto de investigación se han definido los siguientes:

- Temperatura: Sensor que permite obtener una señal eléctrica a partir de la variación de la temperatura del ambiente.
- Luz: Sensor que produce una señal eléctrica a partir de la variación de la intensidad de luz del ambiente.
- Humedad Ambiental y Suelo: Sensor capaz de detectar variaciones en los niveles de humedad del aire o suelo y convertir estos cambios en una señal eléctrica.

Además, actualmente existen sensores de tipo analógico y digitales, la principal diferencia radica en el tipo de señal entregada, siendo así, los sensores analógicos los que requieren de un procesamiento previo para obtener el valor a medir del fenómeno físico.

Actuadores

Los actuadores son dispositivos (mecánicos o electrónicos), capaces de activar diferentes procesos mediante la aplicación de una fuerza o movimiento generado a partir de energía eléctrica, neumática o hidráulica. Para el diseño del sistema propuesto se definieron los siguientes actuadores:

- Bomba de Agua: Actuador capaz de convertir la energía eléctrica en energía mecánica a través de un motor DC.
- Led: Actuadores mejores conocidos como diodo led, son capaces de transformar la energía eléctrica en luz.
- Ventilador: Actuador capaz de generar energía mecánica a través de la aplicación de energía eléctrica en una bobina.

- Relé: Actuador de tipo eléctrico usado generalmente como interruptor por su capacidad de permitir la continuidad de un circuito eléctrico a partir de la aplicación de una energía eléctrica en una bobina interna.

Controladores

Estos dispositivos generalmente son conformados por microcontroladores que se pueden programar de forma que interactúen con los sensores y controladores, actualmente existen placas PCB desarrolladas que incluyen el microcontrolador con varios elementos, módulos adicionales, lenguajes de programación propios y una cantidad sustancial de librerías para mejorar la experiencia del usuario. Para integrar los sensores y actuadores al sistema, en el presente proyecto se ha hecho el uso de controladores que integran un módulo WiFi y son de bajo consumo energético como: ESP8266, MKR WIFI 1010 y OPLA Arduino Kit.

Smart Agro

El Smart Agro es una técnica emergente que nace ante la necesidad de obtener una agricultura sostenible mediante la integración de un sistema de monitorización basado en IOT, donde se incorporan sensores capaces de proporcionan información del estado de varios factores ambientales como la humedad, temperatura, luminosidad, riego, pH con la finalidad de aumentar la productividad a través de recomendaciones, alertas de valores fuera de rangos óptimos y el estado climático o condiciones ambientales a las cuales el cultivo está expuesto (TELEFONICA, 2021).

Servicios Web

Los servicios web son considerados como vías para la comunicación entre máquinas conectadas en una red, esta comunicación se basa en envió de peticiones y respuestas en una arquitectura de tipo cliente-servidor.

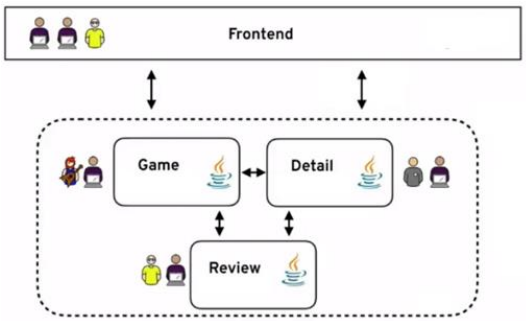
En un principio estos servicios web se ligaban estrechamente con un protocolo simple de acceso a objetos (SOAP), el cual usa un lenguaje de marcado extensible (XML) para el intercambio de información, sin embargo, en la actualidad bajo el mismo concepto de SOAP se identifica un nuevo termino denominado como “REST” el cual se basa en los protocolos HTTP para la comunicación de las máquinas (García, 2015).

Arquitectura orientada a servicios

Varios autores han tratado de definir a la arquitectura orientada a servicios (SOA) basándose en diferencias puntos de vistas como son la tecnología o los negocios, razón por la cual actualmente no existe una definición ya establecida dentro de las tecnologías de la información, sin embargo en la mayoría de definiciones los autores coinciden con la idea de que SOA puede considerarse como un concepto arquitectónico que fomenta la interoperabilidad, agilidad y eficiencia, enfocándose en dividir cada proceso de negocio en bloques más pequeños de funciones denominados servicios. (Naghmeh Niknejad, 2020), SOA también está ligada al concepto de arquitectura monolítica, debido a que todos sus servicios se incorporan en una sola estructura o programa como se observa en la Figura 13.

Figura 13

Arquitectura monolítica



Nota: Se puede observar cómo los servicios se incorporan dentro de una misma estructura (Soto, 2018).

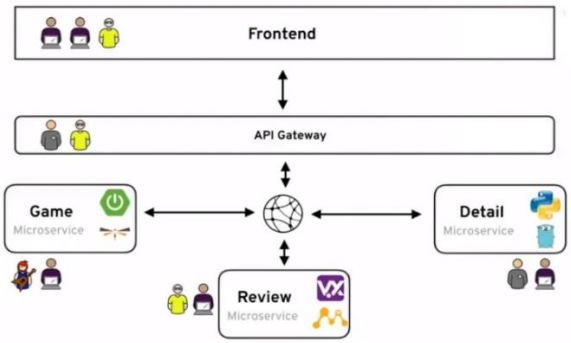
Por lo cual en caso de que un servicio deje de funcionar o se llegue al máximo de peticiones admitidas por las capacidades de servidor, todo el sistema colisionaría.

Arquitectura basada en Microservicios

Para superar el inconveniente presentado en SOA respecto a la arquitectura monolítica, se ha planteado el concepto de microservicio el cual sugiere que una aplicación grande y compleja, puede ser dividida en pequeños servicios autónomos que se comunican entre sí a través de protocolos. Cada microservicio se dedica a una única tarea por lo cual pueden ser fácilmente desplegados mediante el uso de diferentes herramientas o lenguajes como se observa en la Figura 14.

Figura 14

Arquitectura basada en microservicios.



Nota: Se puede observar como una aplicación grande y compleja, puede ser dividida en pequeños sistemas autónomos denominados microservicios (Soto, 2018).

Gracias a la división en microservicios, esta arquitectura adquiere factores como mayor tolerancia a fallos (debido a que, si un microservicio fallará, los demás mantendrían su funcionamiento), escalabilidad (los elementos son reutilizables y pueden hacer mayor uso del hardware) por lo cual facilitará el desarrollo de aplicaciones IOT a gran escala (Sun, Yan, & Memon, 2017).

Servicios de transferencia de estado representacional (REST)

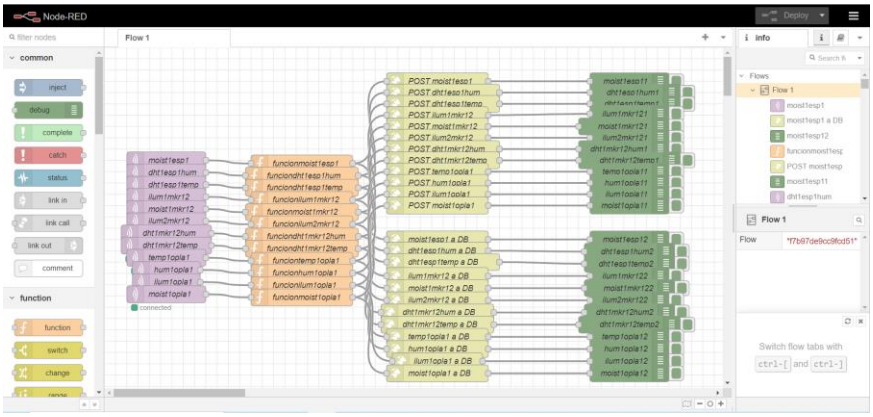
En los servicios REST, tanto las peticiones como las respuestas, se construyen dependiendo del cambio de estado y transferencia de representaciones de información, es importante aclarar que este método no guarda los estados anteriores, por lo cual es necesario que el cliente envía toda la información de estado en cada petición. Al ser un servicio basado en los protocolos de HTTP (GET, POST, PUT y DELETE) devuelve la información solicitada en formatos como XML y JSON, siendo este último el más utilizado debido a que estructura los datos de manera que sean más comprensibles para el usuario (Guido, Simone,Murzilli, & Cudini, 2016).

Node-RED

Es una herramienta de desarrollo de código abierto, donde se utiliza la programación basada en flujo para la integración de dispositivos IOT, API's o servicios web. Está basado en JavaScript y construida sobre la plataforma Node.js que proporciona un editor que se puede abrir desde cualquier navegador web (Milica Lekić, 2018) como se observa en la Figura 15.

Figura 15

Interfaz gráfica de Node-RED



Nota: Interfaz gráfica de Node-RED abierta en navegador web.

Su uso es intuitivo porque se compone de nodos representados por íconos de diferentes clases y funciones que deben ser conectados según el requerimiento del usuario, además consta de la posibilidad de especificar que flujos deben ejecutarse y de una ventana de mensajes de depuración para visualizar su ejecución.

Los nodos se dividen según sus características de funcionamiento, entre los principales se encuentran (OpenJS Foundation, 2020):

Nodos comunes

- inyector: Inyecta flujos de información en intervalos de tiempos definidos por el usuario
- debug: Muestra las propiedades de los mensajes seleccionados en la pestaña de la barra lateral de depuración. Puede configurarse para definir el tipo de mensaje a mostrarse como ejemplo el resultado de una expresión JSON.

Nodos de función

- function: Es un nodo que permite introducir funciones manualmente para modificar el mensaje. Se estima que la función devuelva uno o varios objetos de mensaje, aunque para parar un flujo puede elegir no devolver nada. La pestaña *On Start* contiene el código que se ejecutará cada vez que se inicie el nodo. La pestaña *On Stop* contiene el código que se ejecutará cuando el nodo se detenga.
- switch: Dirige los mensajes en función de los valores de sus propiedades o de su posición en la secuencia. Cuando llega un mensaje, el nodo evaluará cada una de las reglas definidas y reenviará el mensaje a las

salidas correspondientes de cualquier regla que coincida. Existen cuatro tipos de reglas:

  - Valor: Se evalúan para la propiedad configurada.
  - Secuencia: Se utilizan en secuencias de mensajes, como las generadas por el nodo Split.
  - Expresión JSON: se evaluará con respecto a todo el mensaje y coincidirá si la expresión devuelve un valor verdadero.
  - Otherwise: Se utiliza si ninguna de las reglas anteriores ha coincidido.

- change: Permite modificar los atributos de un mensaje. El nodo permite detallar varias reglas que se van a aplicar acorde se vayan definiendo.

Nodos de red

- mqtt in: Se conecta a un broker MQTT y se suscribe a los mensajes del tópico especificado. Varios nodos MQTT (de entrada, o de salida) pueden compartir la misma conexión al broker si es necesario. Sus principales propiedades de salida son:
  - payloadstring | buffer: una cadena a menos que se detecte como un buffer binario.
  - topicstring: Es el tópico MQTT, utiliza “/” como separador de jerarquía.
  - qosnumber: 0 dispara y olvida, - 1 al menos una vez, - 2 una vez y sólo una vez.
- mqtt out: Se conecta a un broker MQTT y publica los mensajes. Varios nodos MQTT (de entrada, o de salida) pueden compartir la misma conexión con el broker si es necesario. *msg.payload* se utiliza como carga

útil del mensaje publicado. Si contiene un objeto, se convertirá en una cadena JSON antes de ser enviada. Si contiene un Buffer binario el mensaje se publicará tal cual. El tópico utilizado puede ser configurado en el nodo o, si se deja en blanco, puede ser establecido por *msg.topic*. Sus principales propiedades de entrada son:

  - payloadstring | buffer: La carga útil a publicar. Si esta propiedad no está establecida, no se enviará ningún mensaje.
  - topicstring: el tópico MQTT a publicar.
  - qosnumber: 0 dispara y olvida, - 1 al menos una vez, - 2 una vez y sólo una vez. Por defecto 0.

- http request: Envía peticiones HTTP y devuelve la respuesta. Si la URL se configura como *example.com/{{topic}}*, tendrá el valor de *msg.topic* insertado automáticamente. Contiene opciones para configurar parámetros de seguridad, autenticación, proxy, entre otras; sus principales propiedades de entrada son:
  - Urlstring: Si no está configurada en el nodo, esta propiedad opcional establece la URL de la petición.
  - Methodstring: Si no está configurada en el nodo, esta propiedad opcional establece el método HTTP de la petición. Debe ser uno de los siguientes: GET, PUT, POST, PATCH o DELETE.

  Las principales propiedades de salida son:
  - payloadstring | object | buffer: El cuerpo de la respuesta. El nodo puede ser configurado para devolver el cuerpo como una cadena, intentar parsearlo como una cadena JSON o dejarlo como un buffer binario.

Nodos de secuencia

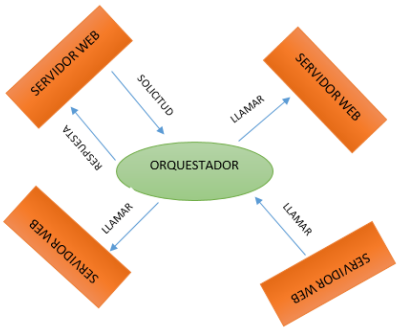
- join: Interconecta cadenas de mensajes en uno solo. Existen tres modos disponibles:
    - Automático: El modo automático utiliza la propiedad de las partes de los mensajes entrantes para determinar cómo debe unirse la secuencia. Esto le permite invertir automáticamente la acción de un nodo dividido.
    - Manual: Cuando se configura para unir en modo manual, el nodo es capaz de unir secuencias de mensajes en un número de resultados diferentes: cadena o buffer, arreglo, objeto clave o valor, objeto fusionado
    - Secuencia reducida: Cuando se configura para unirse en modo de reducción, se aplica una expresión a cada mensaje de una secuencia y el resultado se acumula para producir un único mensaje.

Orquestación de servicios

El orquestador de servicios define un proceso central encargado de coordinar la ejecución de diferentes operaciones sobre los servicios web ligados como se observa en la Figura 16.

Figura 16

Diagrama de Representación de un Orquestador de servicios



Nota: Representación de varios servicios web ligados a un único orquestador.

En comparación a otros procesos, la orquestación permite que los servicios web puedan incorporarse sin ser conscientes de que están participando en un proceso de negociación de mayor nivel, además es posible implementar escenarios alternativos en caso de que se produzcan fallas (ALBRESHNE, FUHRER, & PASQUIER, 2009).

Pruebas de evaluación del sistema

Prueba de carga

Este tipo de prueba es utilizada para medir el rendimiento del sistema bajo una gran cantidad de tráfico creciente, además se obtienen como resultados métricos de rendimiento como el tiempo de respuesta o nivel de procesamiento de la CPU con la finalidad de comprender el comportamiento esperado del sistema ante picos de carga provistos o improvistos (Brajesh, 2017).

Prueba de puntos de función

Los puntos de función se basan en el análisis de la medida del software, tratan de determinar la funcionalidad que una aplicación o software proporciona a los usuarios

finales, los aspectos que hay que tomar en cuenta para realizar este análisis (Rodríguez, 1999) son los siguientes:

- Planificación del conteo de los Puntos de Función.
- Recolección de información del sistema actuales.
- Cálculo del factor de ajuste y factor de peso
- Realizar el conteo de las transacciones y ficheros lógicos.
- Organización y clasificación de los componentes.
- Revisión de las características generales del sistema.
- Tabulación y validación de los resultados.

El procedimiento para realizar estas pruebas, comienza con la determinación de los tipos de conteo y componentes funcionales, la IFPUG-FPA define 5 elementos: archivo lógico interno, archivo de interfaz externo, entrada externa, salida externa y consulta externa, estos dos últimos difieren en la salida producen valores agregados, mientras que las consultas solo hacen referencia al contenido de los archivos y los presentan en listados.

Posteriormente se asigna un nivel de complejidad de cada componente, estos niveles dependen de factores como repetición de elementos, números de archivos creados, números de salidas externas, entre otras. A mayor número de elementos se debe asignar un mayor número de complejidad (PMOinformatica, 2015) como se observa en la Figura 17.

Figura 17

Ejemplo de asignación de complejidad IFPUG-FPA

Tipo de componente	Complejidad bajo	Complejidad medio	Complejidad alto
Entrada externa	3	4	6
Entrada externa	4	5	7
Consulta externa	3	4	6
Salida externa	7	10	15
Archivo lógico interno	5	7	10

Nota: Imagen obtenida de (PMOinformatica, 2015).

Según la tabla anteriormente definida y a modo de ejemplo se presenta la Figura 18, en donde se asignan los puntos a cada componente.

Figura 18

Ejemplo Puntos de función asignados

Componente	Tipo de componente	Nivel de complejidad
Ingreso de cliente	Entrada externa	Bajo
Modificación de cliente	Entrada externa	Medio
Listado de clientes	Consulta externa	Bajo
Reporte de clientes por país	Salida externa	Medio
Tabla de clientes	Archivo lógico interno	Medio

Nota: Imagen obtenida de (PMOinformatica, 2015)

Una vez asignados los puntos de función se realiza la sumatoria y se obtiene el número de puntos de función no ajustado (FPAS). Finalmente se aplica un factor de ajuste definido por la IFPUG-FPA, obtenido a partir de todos los componentes del sistema y representados mediante una tabla de parámetros, los puntos de función ajustados (FPA) serán:

$$FPA = FPAS * [0.65 + (0.01 * Factor\ de\ ajuste)]$$

Para estimar las horas o días hombre a partir del resultado de los FPA se debe aplicar un factor de conversión, el cual se obtiene a partir de datos históricos de productividad del equipo respecto a cada funcionalidad descritas en las tablas anteriores.

Pruebas de usabilidad

Las pruebas de usabilidad (SUS) son utilizadas para evaluar que tan fácil es el uso de un producto para los usuarios finales, permite conocer al desarrollador el grado de satisfacción de los usuarios en un escenario específico. Las principales características de estas pruebas son las siguiente:

- Se basan en un cuestionario corto que no requiere muchos recursos para ser administrado.
- Es un método rápido de implementar y analizar, solamente se debe copiar, pegar y modificar según el escenario.
- Es un método que requiere entre 8 a 12 personas para realizar el análisis.

Una vez realizado el cuestionario, el proceso de evaluación de las pruebas SUS consiste en los siguientes pasos:

- Restar 1 al puntaje de cada pregunta impar.
- Restar de 5 el puntaje de cada pregunta par.
- Realizar la sumatoria de los puntajes obtenidos.
- Al total multiplicar por 2.5.

El puntaje total de las pruebas SUS se puede comparar con las siguientes métricas:

- Puntajes superiores al 80.3 corresponde a una calificación de “A”, la cual significa que las personas se encuentran satisfechas con la aplicación y la recomendaran a sus amigos

- Puntajes mayores, igual o menores a 68 corresponde a una calificación de “C”, la cual significa que la aplicación fue realizada correctamente, pero se podría mejorar.
- Puntajes menores al 51 corresponde a una calificación de “F”, la cual significa que la aplicación debe corregirse inmediatamente y tomando como prioridad a la usabilidad.

Capitulo III: Diseño

Requisitos de diseño

Estos requisitos permiten la comunicación entre cliente y desarrollador, para este propósito es necesario tener en claro los requerimientos del proyecto, por tanto, es indispensable que la arquitectura a generar cumpla con los objetivos planteados y que los resultados obtenidos se acoplen a los alcances establecidos inicialmente. A continuación, se muestra el diseño diferenciado para cada una de las capas utilizadas por medio de información de requisitos funcionales y no funcionales, permitiendo identificar sus parámetros, funciones y clases para comprender en primer lugar de manera diferenciada su estructura y luego realizando la unión de los mismos.

Requisitos funcionales

- Requerimientos del cliente
    - Al estar enfocado en usuarios que no dominan los diversos lenguajes de programación, el código y la información proporcionada para el uso de los implementos generados debe ser accesible, fácil de utilizar, asimilable y confiable.
    - Es necesario el conocimiento básico de electrónica enfocada en la utilización de dispositivos como controladores, sensores y actuadores, además de su programación.
    - Se debe tener creadas dos bases de datos, una para sensores y otra para actuadores.
    - El usuario debe tener a disposición herramientas de programación las cuáles permitirán establecer el código fuente generado para cada una de ellas, entre estas se encuentran: Arduino, Visual Studio Code, Eclipse (con extensiones para EMF, LUA y NCL), Node-Red, Ginga NCL y MySQL (se

puede utilizar phpMyAdmin o las extensiones de MySQL como Shell, Workbench, etc.).

- Necesidades y requisitos del software que debe cumplir de manera satisfactoria
    - La publicación y obtención de los datos se debe establecer acorde a la interacción del usuario con la aplicación.
    - El usuario debe tener la capacidad de acceso a las bases de datos y los servicios web desde cualquier dispositivo y en cualquier locación.
    - Las bases de datos deben ser incrementales para datos que varían constantemente y no incrementales para datos que cambian bajo dos estados como encendido y apagado.
- Funciones que el software será capaz de realizar
    - Diseñar aplicaciones locales y remotas para la Televisión Digital enfocado en Smart Agro.
    - Crear un editor gráfico funcional con la interacción de los elementos dispuestos en la barra de herramientas.
    - Otorgar al usuario código fuente generado funcional para cada programa utilizado y acorde a los datos ingresados en el editor gráfico.
- Detalles técnicos
    - Es indispensable poseer el acceso a la red de internet para la implementación de los programas generados en aplicaciones remotas.
    - Sistema operativo Windows 7 o superior; procesador Intel Core i3 o superior; memoria RAM mayor o igual a 4GB, tarjeta de video opcional.
- Manipulación de datos de entrada y salida
    - El usuario está en la facultad de establecer los datos de entrada y salida para los elementos del sistema en el editor gráfico, mismos valores que

son manejados por el orquestador, API REST, base de datos, Node-RED y demás componentes descritos en la arquitectura.

Requisitos no funcionales

- Propiedades o cualidades que el software debe tener
    - Interfaz gráfica amigable y asimilable para la utilización de los usuarios, los campos de elección y llenado deben tener opciones preestablecidas, así también debe existir la libertad del usuario de interponer los elementos a través de la interfaz de la manera que mejor disponga.
    - La aplicación TDT debe otorgar en relación de aspecto, dimensiones y posiciones una visualización que integre todos los botones y recursos multimedia de manera que no interfiera con la programación de fondo.
- Restricción del sistema operativo
    - Funcionar sin complicaciones en sistemas operativos iguales o superiores a Windows 7.
- Restricción del ambiente
    - Para la integración de los artefactos de hardware, se debe procurar implementar en un espacio cerrado, con nula exposición a lluvia, demás restricciones se establecen por valores máximos y mínimos de condiciones de tarjetas controladoras, sensores y actuadores.
    - Sistema enfocado en arquitectura basada en microservicios.
- Restricciones o condiciones del producto
    - La parte de hardware está diseñada para su funcionamiento con dispositivos Arduino, con las tarjetas controladoras ESP8266 y MKR WiFi 1010, la inclusión del MKR IoT Carrier es opcional. Sensores y

actuadores a utilizar van a depender de la tarjeta controladora y su admisión con respecto a la distribución y capacidad de pines analógicos y digitales.

    - Se deben tener 2 bases de datos, una para sensores y otra para actuadores.

- Restricción de rapidez
    - Obtención de datos en tiempo real.
    - Correcta sincronización en implementación dependerá de la capacidad de la conexión a internet del usuario.
- Restricción de seguridad
    - El usuario tiene la capacidad de ingresar información acorde a sus servicios web, orquestador, bróker y bases de datos con restricciones de usuario y contraseña, el ejemplo de aplicación no cuenta con restricciones respecto a seguridad.
    - Valores en bases de datos no son manipulables por el usuario desde el editor gráfico.
- Restricción de usabilidad
    - Conocimiento básico de herramientas de software citadas y de utilización de elementos de hardware.

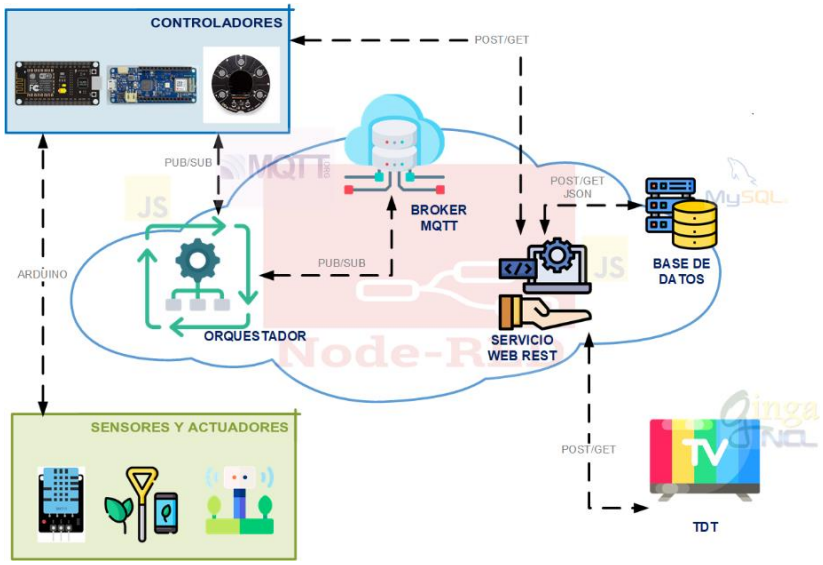
Arquitectura

La creación del Lenguaje Específico de Dominio para el presente proyecto de titulación se ha implementado gracias a la unión de varias fases, en este sentido, el metamodelo, editor gráfico y la generación de código automático se basaron en la arquitectura del sistema que se expone a continuación: Para el sistema denominado IoTV enfocado al Smart Agro, se ha dispuesto como eje central (por el cual se establecen las

condiciones de funcionamiento y la distribución de los demás componentes del sistema) al broker MQTT; por un lado, da lugar al orquestador el cual maneja las peticiones y solicitudes de los sensores y actuadores como suscriptores y publicadores de los mensajes MQTT respectivamente, los mismos que son dirigidos por el controlador; por otro lado, también permite el funcionamiento del servicio web creado basado en servicios REST, esto se realiza mediante la inclusión de la herramienta Node-RED que sirve de puente entre la información emitida por MQTT de parte de los sensores y actuadores hacia las bases de datos creadas, el servicio web al estar basado en REST utiliza protocolo HTTP, con sus respectivos métodos GET y POST. Con esta arquitectura, se abre paso a la inclusión del dominio de la Televisión Digital Terrestre especialmente a la creación de las aplicaciones interactivas remotas, gracias a la función *tcp.lua* que permite obtener los valores directamente de las bases de datos mediante los servicios REST. Lo descrito anteriormente se resume en la Figura 19.

Figura 19

Arquitectura del sistema propuesto



Abstracción de los atributos del sistema

Para la creación del metamodelo se instauran las clases de manera separada, para evidenciar el funcionamiento de cada una acorde a la arquitectura planteada y así corregir errores que puedan producirse; a su vez, se determinan los atributos necesarios que permitan describir las sintaxis concreta y abstracta para cada fase del proyecto y el correcto sentido con respecto a la semántica planteada, tomando en cuenta dichos atributos para la posterior creación del editor gráfico y la generación de código fuente.

TDT

Para la abstracción de las características correspondientes a la TDT, se toma en cuenta programas de prueba realizados en Ginga NCL, de los cuales se define un ejemplo de plantilla para el usuario, en este caso se proporcionará la libertad de diseñar las aplicaciones interactivas permitiendo establecer distintos recursos multimedia y modificando sus parámetros de dimensión y posición.

API REST

Los atributos esenciales que se establecen para un servicio REST son host y puerto, su función es permitir realizar peticiones HTTP desde los recursos que se requieran, para el caso propuesto es necesario la publicación con el método POST en las bases de datos de los parámetros dados por los sensores y el estado de los actuadores mediante MQTT, a su vez, es necesario que estos datos almacenados en las bases de datos se puedan visualizar en las aplicaciones de Televisión Digital, haciendo uso del método GET.

Base de datos

Para el sistema presentado se establece utilizar una base de datos relacional, esta base de datos permite archivar datos en tablas separadas, además, es conveniente al ser de código abierto y fácil de usar por medio de la web. En este contexto, es necesaria la

creación de dos bases de datos, una para sensores de manera incremental con parámetros establecidos de medición para Smart Agro como: *temperatura*, *humedad* (ambiental), *iluminación* y *moisture* que es la humedad del suelo; y otra para actuadores, que en este caso debe constar de tablas correspondientes a *led*, *riego* y *ventilador,* las cuales deberán almacenar los estados de los mismos, siendo estos: encendido (ON) o apagado (OFF), estos estados se van actualizando y no se almacenan de manera incremental.

Servidor MQTT

Para la implementación del broker MQTT se debe instalar la herramienta correspondiente en la VPS, el broker permite la interacción entre el orquestador y el hardware utilizado, en este caso los mensajes enviados por los sensores y actuadores son mensajes MQTT *publish* y *subscribe*. Al igual que para el servicio web REST, los atributos abstraídos para esta clase son el host y el número de puerto a utilizar.

Node-RED

La inclusión de Node-RED en el presente proyecto otorga una gran cantidad de soluciones para la unión de varias de las fases requeridas, en primera instancia permite la recepción de los valores otorgados por los sensores y actuadores por medio del controlador y basado en mensajes MQTT, estos valores se procesan y se transforman a lenguaje JSON por medio de funciones, posteriormente se realiza el envío de estos valores hacia las tablas dentro de las bases de datos tanto de sensores como de actuadores por medio de HTTP, por otro lado establece la inclusión del orquestador conjuntamente con la TDT, además de gestionar la interacción del usuario de la TDT para la activación y desactivación de los actuadores.

Orquestador

El orquestador debe establecer la conexión con el bróker MQTT, para esto, es necesario conocer el host y puerto MQTT utilizado, estos parámetros se dispondrán como atributos para la respectiva creación de la clase dentro del metamodelo; el orquestador también controla la interacción entre la TDT y los elementos de hardware bajo valores umbrales definidos por el usuario.

Hardware

El hardware debe incorporar controladores con la capacidad de conectarse por medio de Wifi a internet, además deben tener la capacidad de leer datos analógicos como digitales y ser dispositivos de bajo consumo eléctrico para que puedan ser alimentados mediante una fuente portátil.

También se debe incorporar sensores acordes a las magnitudes físicas que influencian el crecimiento de los cultivos como luz solar, humedad ambiental y de suelo, temperatura y actuares para poder controlar dichas magnitudes como ventilación, riego e iluminación

Estructura del metamodelo

Una vez realizada la abstracción de las principales características de los elementos del sistema, se debe realizar la construcción del metamodelo, para lo cual se ha procedido a dividir el metamodelo en distintos módulos que describan a detalle los parámetros de cada elemento.

Controlador

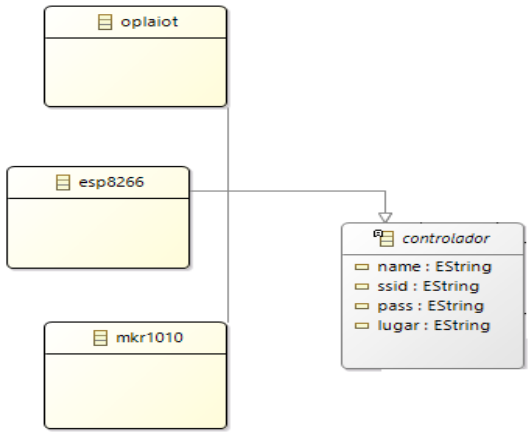
En este módulo se ha procedido a crear una clase padre denominada *Controlador* que posee los siguientes atributos para establecer la comunicación hacia internet y todo el sistema:

- *Name*: Este atributo de tipo *EString* identificará el nombre del controlador.
- *Ssid*: Atributo de tipo *EString* que define el nombre de la red para la conexión WiFi.
- *Pass*: Atributo de tipo *EString* que define la contraseña para la conexión Wifi.
- Lugar: Atributo de tipo EString que define la ubicación del controlador.

Además, se han establecido tres clases hijas (*oplaiot*, *esp8266*, *mkr1010*) que corresponden a las diferentes opciones de controlador a escoger, cabe recalcar que al ser clases hijas, estas heredan los atributos de la clase padre *controlador* como se observa en la Figura 20.

Figura 20

Módulo del esquema de los controladores



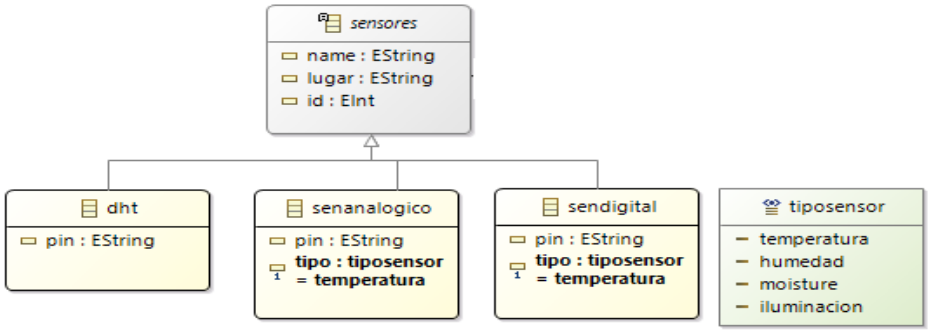
Nota: Módulo que representa a los controladores a utilizarse dentro del proyecto.

Sensores

En este módulo se ha creado la clase padre *sensores* compuesta de un atributo denominado *name* de tipo *EString* para identificar a los sensores que se utilizarán en los sistemas, además se han establecido tres clases hijas dependiendo de los tipos de sensores compatibles con los controladores utilizados en el sistema como se observa en la Figura 21.

Figura 21

Módulo del esquema de sensores



Nota: Esquema que representa los diferentes sensores a utilizarse dentro del sistema.

La primera clase hija se denomina *dht* y corresponde al sensor digital dht11 que es capaz de entregar tanto la medida de la temperatura y humedad ambiental, además se detalla un atributo denominado *pin* de tipo *EString* el cual define el pin por el cual se conectará al controlador.

La segunda clase hija denominada *senanalogico* corresponde a los sensores de tipo analógico, además se han establecido los siguientes atributos:

- *Pin*: Atributo de tipo *EString que* define el pin analógico a utilizarse en el controlador.
- *Tipo*: Atributo de tipo *tiposensor* que define el fenómeno físico a medirse.
- Lugar: Atributo de tipo EString que define la ubicación del sensor.
- Id: Atributo de tipo Eint para identificar el cultivo del sensor.

La tercera clase hija denominada *sendigital* corresponde a los sensores de tipo digital, posee atributos similares a los de la clase *senanalogico* con la diferencia que el pin a utilizarse debe ser de tipo digital.

Actuadores

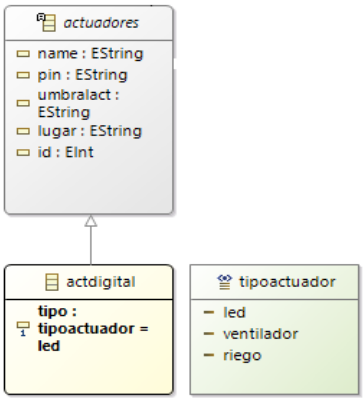
Este módulo posee una clase padre denominada *actuadores* con atributos principales como:

- *Name*: Atributo de tipo *EString* para identificar el actuador.
- *Pin*: Atributo de tipo *EString* para identificar la conexión al pin del controlador.
- *Umbralact*: Atributo de tipo *EString* para identificar el umbral de activación de los actuadores.
- Lugar: Atributo de tipo *EString* para identificar el lugar del actuador.
- Id: Atributo de tipo *Eint* para identificar el cultivo del actuador.

Además, posee una clase hija denominada *actdigital* con atributo de nombre *tipo* de opción *tipoactuador* para identificar a los actuadores (*led*, *ventilador*, *riego*) que se pueden utilizar dentro del sistema como se observa en la Figura 22.

Figura 22

Módulo del diagrama de los actuadores



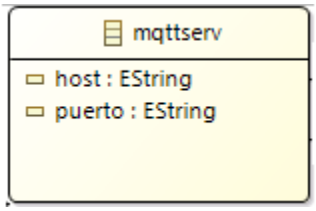
Nota: Diagrama que representa la estructura de los actuadores a utilizarse dentro del sistema.

MQTT

Para el broker MQTT se ha creado la clase *mqttserv* que contiene como principales parámetros el *host* y *puerto* de tipo *EString* necesarios para publicar los datos obtenidos de los sensores o los estados de los actuadores, el esquema se observa en la Figura 23.

Figura 23

Diagrama del broker MQTT



Nota: Clase que representa al broker MQTT.

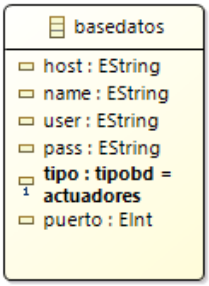
Base de Datos

Para la base de datos se ha establecido la clase *basedatos* que se muestra en la Figura 24 y contiene los siguientes parámetros:

- *Host*: Atributo necesario para realizar la conexión hacía el servidor que contiene a la base de datos.
- *Name*: Atributo para identificar la base de datos
- *User*: Atributo que define el usuario para acceder a la base de datos.
- *Pass*: Atributo para definir la contraseña para acceder a la base de datos.
- *Tipo*: Atributo para diferenciar si la base de datos corresponde al estado de los actuadores o sensores.
- Puerto: Atributo para definir el puerto por el cual acceder al servicio.

Figura 24

Diagrama de la base de datos



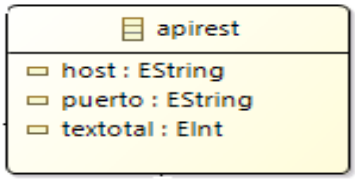
Nota: Clase que representa a la base de datos a utilizarse en el sistema.

API REST

Como se observa en la Figura 25, para los servicios REST se ha definido la clase *apirest* con atributos *host*, *puerto* para realizar la conexión al servidor web que contiene a este servicio y el parámetro textotal para identificar el número de máximo de cultivos a interactuar dentro de la aplicación de TDT.

Figura 25

Diagrama de la clase apirest



Nota: Clase que representa al servicio de API REST creado para el sistema.

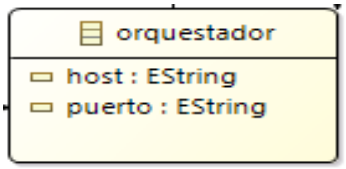
Orquestador

Para este servicio se ha establecido la clase *orquestador* con atributos similares a la clase *apirest* para establecer la conexión al servidor que contiene este servicio y que se relacionará directamente con el servidor MQTT y con los controladores para ser capaz

analizar y tomar las decisiones respecto al cambio en las medidas de los sensores y estados de los actuadores, lo descrito se visualiza en la Figura 26.

Figura 26

Diagrama del Orquestador



Nota: Clase que representa al servicio del Orquestador que se relacionará directamente con el servidor MQTT y los actuadores.

Televisión Digital Terrestre.

Debido a las necesidades del sistema se ha creado una clase padre denominada *Tdt* en la cual se definirán el nombre del dispositivo y la programación (video) a reproducirse mediante los atributos *name* y *programa* correspondientemente.

Una vez definida la clase padre, se define la subclase denominada *App* para establecer el fondo que alberga los demás elementos de la TDT con atributos como *name* para identificar a la aplicación, *fondo* para establecer una imagen que se sobrepondrá a la programación actual, se debe establecer la extensión de la imagen (jpg o png), los atributos restantes son para posicionar y delimitar al fondo.

También se obtienen clases hijas que corresponden a los distintos botones y elementos (texto, imagen y video) que se pueden visualizar a través de la acción de ellos, se define una clase abstracta denominada *Botones* la cual contiene atributos para la configuración de los botones. Entre las principales se tienen *label* para definir las etiquetas, *labelsizebo*t para definir el tamaño de las etiquetas y los parámetros restantes determinan la posición y tamaño de los mismos.

Entre las clases hijas correspondientes a los elementos, se tiene la clase *Texto* local y remoto con atributos como:

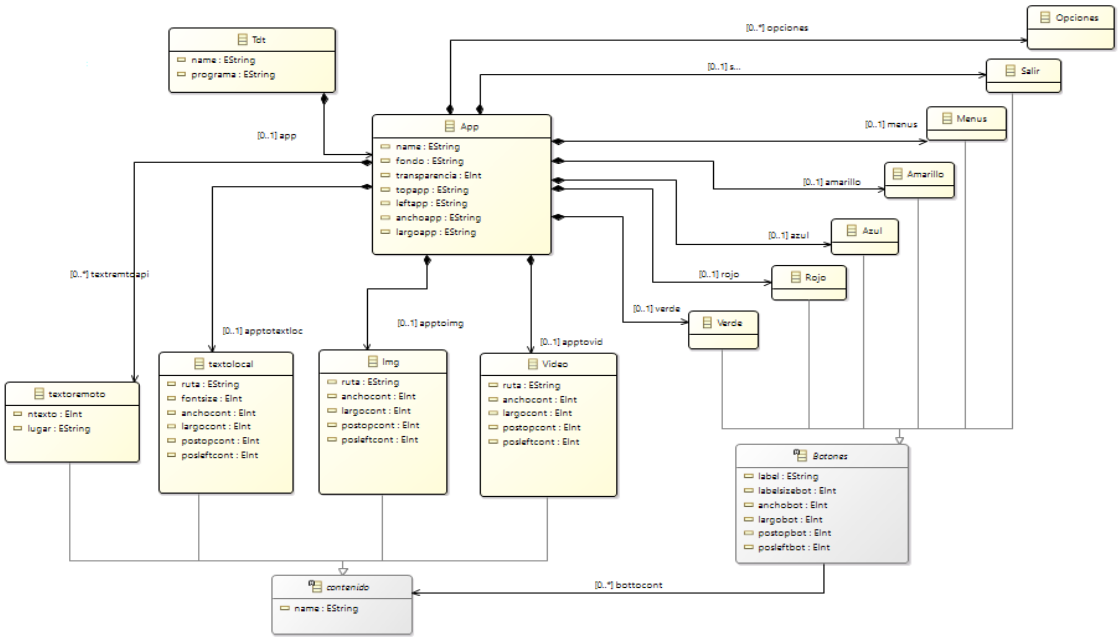
- *Ruta*: Atributo que define la dirección del contenido del texto.
- *nTexto*: Atributo que define el identificador del texto.
- *Fontsize*: Determina el tamaño de letra del contenido del texto.
- *Lugar*: Atributo que define el lugar del cultivo a mostrarse en la TDT.

La clase *Img* facilita la posibilidad de insertar imágenes, el atributo *ruta* define el nombre de la imagen que se desea mostrar. Mientras que la clase *Video* permite insertar videos de formato .mp4 solamente definiendo el nombre del archivo dentro del atributo *ruta*.

Estas últimas clases mencionadas son parte de la clase abstracta *contenido* y poseen parámetros generales como la posición y tamaño de los elementos (expresados en porcentaje), el diagrama correspondiente a la TDT se puede observar en la Figura 27.

Figura 27

Diagrama correspondiente a la TDT



Nota: En el diagrama de la figura se presentan las clases creadas para definir una *aplicación dentro de la TDT.*

Relaciones entre los esquemas

Una vez presentados los diferentes módulos de los esquemas que conforman al metamodelo, es necesario establecer una visión del funcionamiento del sistema cuando es sometido a un escenario real basado en la arquitectura propuesta.

Como primera instancia se debe plantear los tipos de controladores a utilizarse, debido a que, a partir de sus pines, se conectarán los sensores y actuadores, por lo cual se establece una relación bidireccional en la cual se podrán conectar varios sensores o actuadores, pero estos elementos podrán estar conectados solamente a uno de los pines de cada controlador.

Una vez seleccionados los sensores, actuadores y controladores, se establece una relación directa entre la clase *controlador* y *mqttserver* de cero a uno debido a que el controlador deberá publicar y suscribir los tópicos del sistema en un único servicio MQTT para poder conocer el estado actual la información de los sensores y el estado de los actuadores.

También se ha realizado una relación desde 0 a varios desde la clase *apirest* hacia *controlador* para conocer los dispositivos conectados al controlador y a su vez desde *apirest* hacia *basedatos* para almacenar o conocer la información o estados pertinente de los sensores o actuadores.

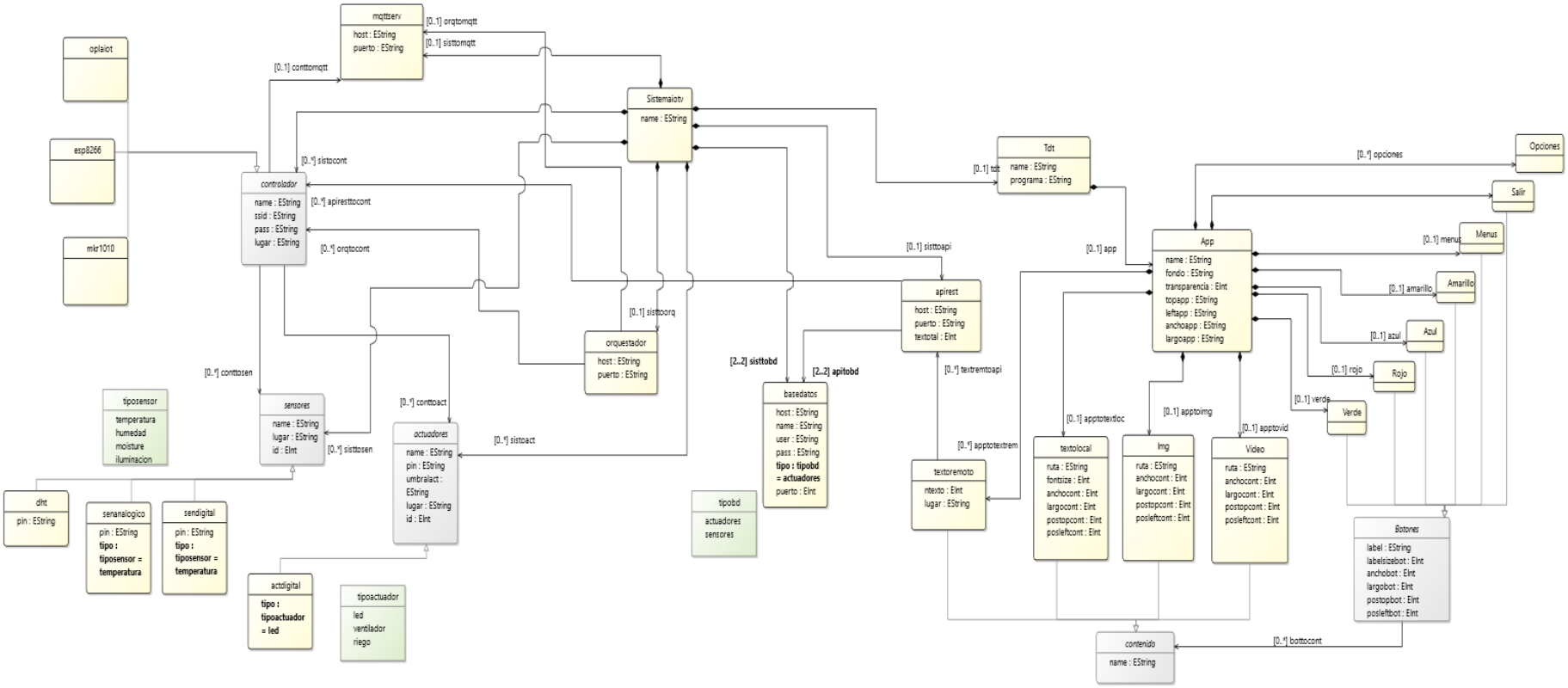
La clase *orquestador* tiene una relación directa de uno a varios hacia la clase *mqttserv* y *controlador* para consultar los valores y estado de los tópicos publicados o suscritos y realizar las acciones pertinentes para regular el estado del sistema.

La clase *botones* tiene una relación de 0 a 1 con la clase *contenido* debido a que un botón puede o no accionar un único elemento. Dentro de los posibles elementos de la TDT también se tiene una relación de cero a varios desde la clase *texto remoto* hacia la clase *apirest* debido a que se tiene como opción realizar (mediante el uso del protocolo TCP que proporciona LUA) la consulta al servidor web.

Todas las clases planteadas son composición de la clase *Sistemaiot,* la cual es la raíz del metamodel. En la Figura 28 se visualiza el metamodelo final obtenido a partir de la arquitectura planteada.

Figura 28

Diagrama del Metamodelo del proyecto



Nota: En el presente diagrama se observa el metamodelo diseñado para el sistema de Smart Agro.

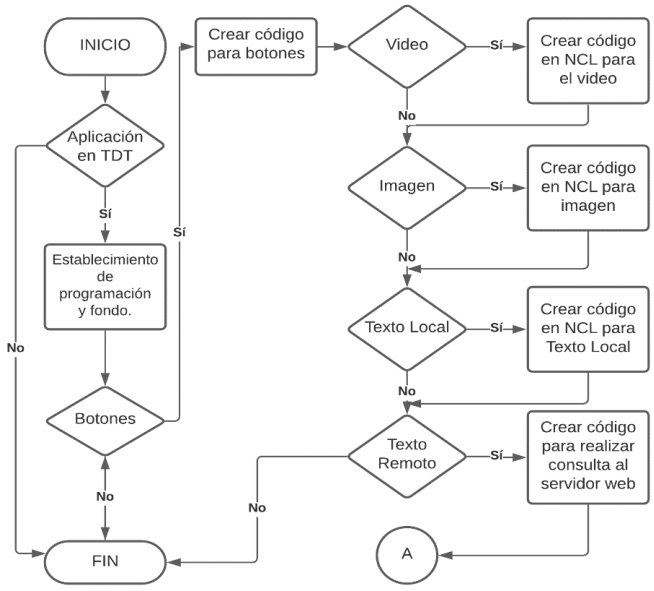
Diagramas de flujo para la generación M2T

Como motor de generación de código se ha optado por utilizar la herramienta Acceleo debido a que permite establecer un lenguaje de propósito general para describir el comportamiento y funcionalidad de cada componente del sistema, además, porque permite acceder a las características del modelo gráfico diseñado con Sirius.

Para cada elemento del sistema, se ha procedido a realizar distintos diagramas de flujo para describir la lógica de la generación de código. En la Figura 29 se observa el diagrama de flujo para la TDT, el cual permitirá generar código de aplicaciones tanto locales como remotas según las necesidades del usuario, el código final se entrega en formato NCL y Lua.

Figura 29

Diagrama de Flujo de la TDT

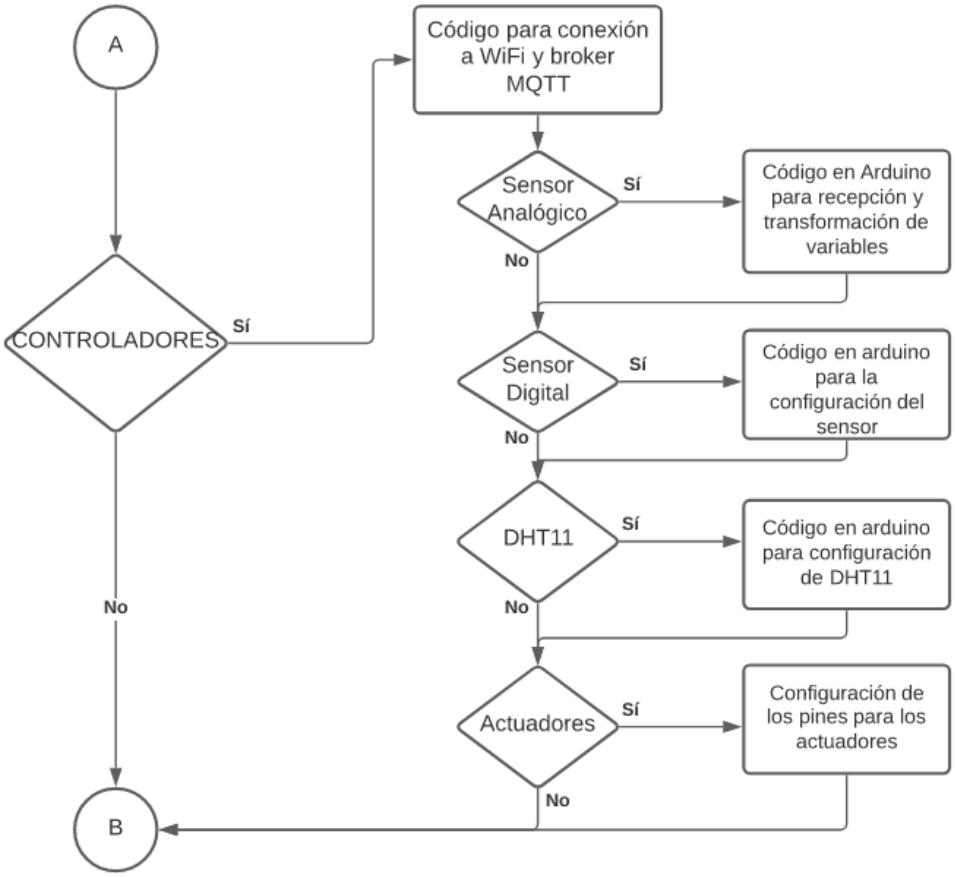


Si el usuario opta por la generación de una aplicación interactiva remota se procederá con la generación del código del controlador, como se observa en la Figura 30,

se puede optar por los controladores definidos en secciones anteriores. A partir de estos controladores se generará el código en Arduino (.ino) correspondiente a la conexión a la red WiFi, servidor MQTT y sensores y actuadores que el usuario haya definido en el editor gráfico.

Figura 30

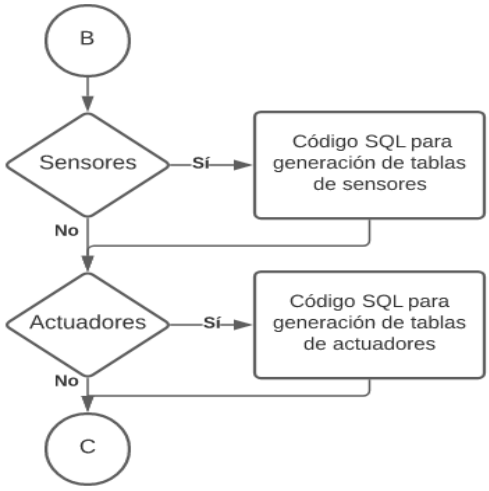
Diagrama de flujo de los controladores



En la Figura 31 se detalla la lógica para la generación de código SQL correspondiente a las bases de datos que almacenarán la información correspondiente a los sensores y actuadores.

Figura 31

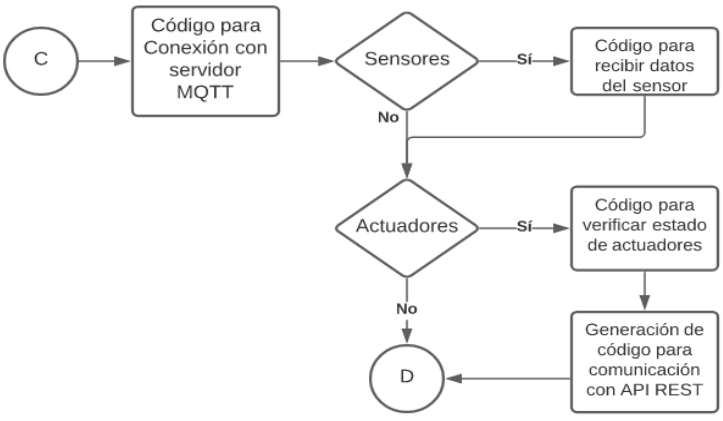
Diagrama de flujo de la Base de Datos



En la Figura 32 se observa la lógica para la creación de código de Node-RED, el cual servirá como puente de comunicación entre la API REST y el broker MQTT.

Figura 32

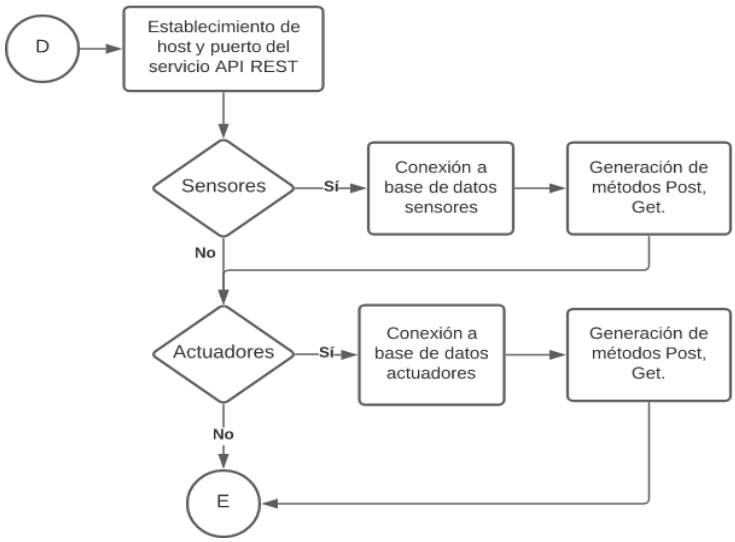
Diagrama de Flujo de Node-RED



En la Figura 33 se establece la lógica para la generación del código de la API REST para almacenar los datos de los sensores y actuadores en las bases de datos y para acceder a esta información a través de la TDT.

Figura 33

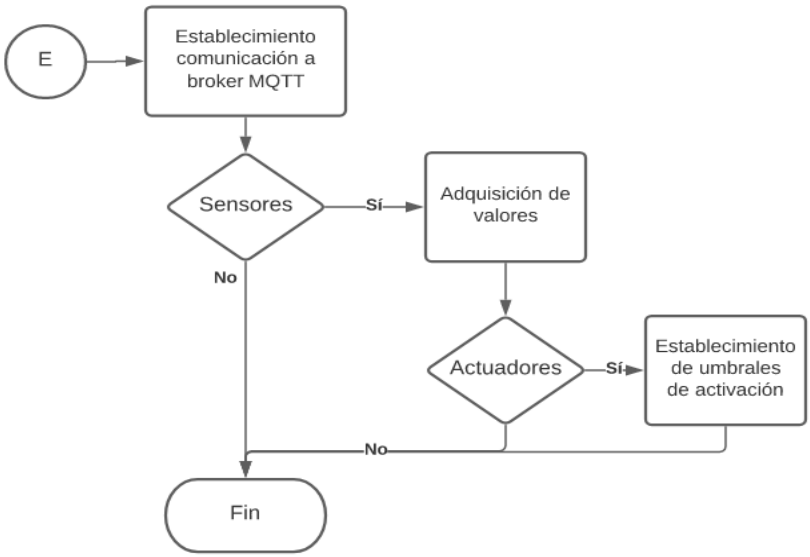
Diagrama de flujo de la API REST



En la Figura 34 se detalla el diagrama de flujo correspondiente a la generación de código del orquestador, el cual proporcionará al sistema la capacidad de activar los actuadores ante umbrales definidos por el usuario.

Figura 34

Diagrama de flujo del Orquestador



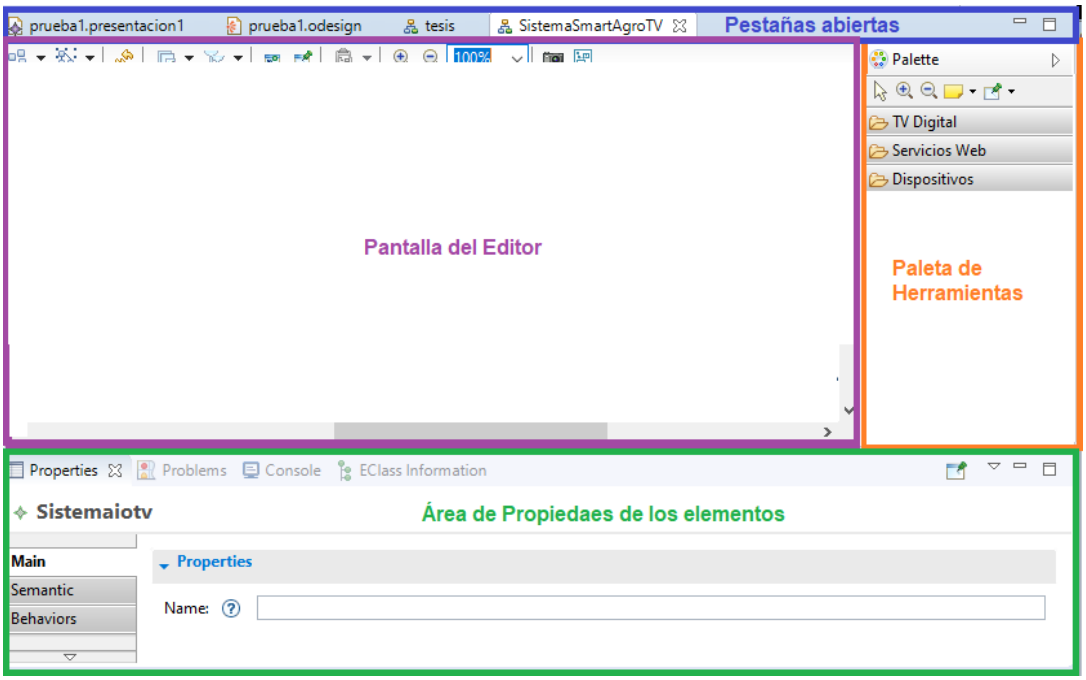
Capitulo IV: Implementación

Editor Gráfico

Para realizar el DSL gráfico, se ha procedido a utilizar una extensión del software Eclipse Modeling Framework (EMF) denominada Sirius, mediante esta herramienta se ha incorporado imágenes que representen a cada clase creada en el metamodelo, una paleta de herramientas en forma de menú desplegable donde el usuario tendrá que seleccionar los elementos y relaciones del sistema, una pantalla del editor para que el usuario ubique los elementos del sistema y una ventana de propiedades para delimitar los atributos de cada elemento como se observa en la Figura 35.

Figura 35

Editor Gráfico del proyecto

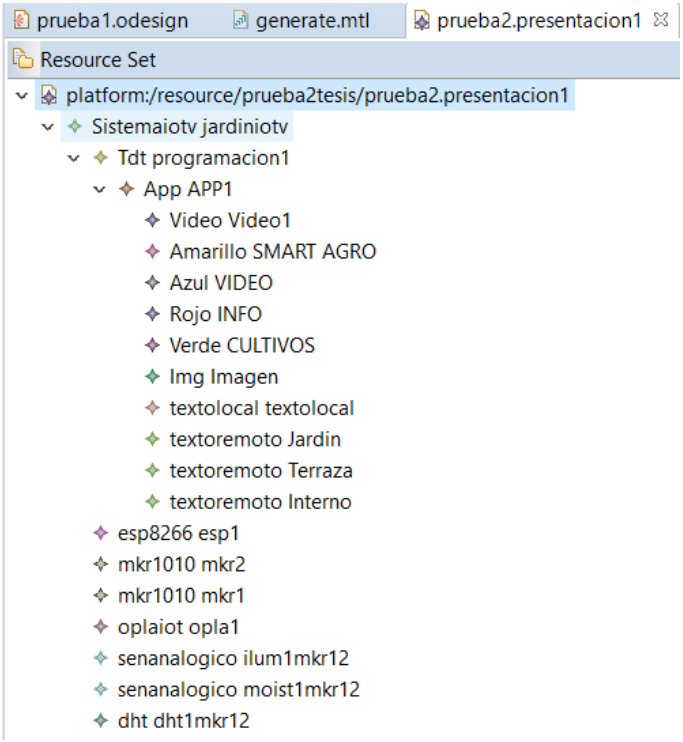


Nota: En la figura se observa el editor con sus principales componentes.

Adicionalmente existen dos archivos aparte del editor gráfico, en el primero se puede observar en la Figura 36 y corresponde al diagrama de árbol de los elementos creados en el editor gráfico.

Figura 36

Diagrama de árbol de las clases

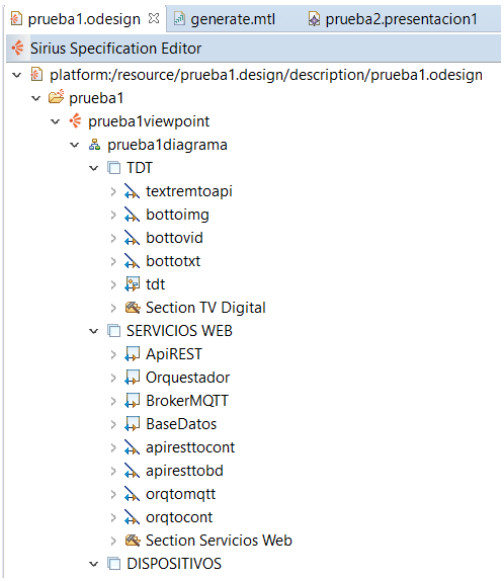


Nota: En la figura se indica a modo de ejemplo el diagrama de árbol de una aplicación de TDT.

Mientras que el segundo se observa en la Figura 37 y corresponde a la configuración de diseño, en donde se edita la forma en la que se observan los elementos, la paleta y también se crean las relaciones que permiten interactuar a los elementos dentro del editor y que a su vez sean editables.

Figura 37

Archivo de configuración del editor gráfico



Nota: En la figura se aprecia el archivo que corresponde a la configuración de los elementos de la interfaz a presentarse dentro del editor gráfico.

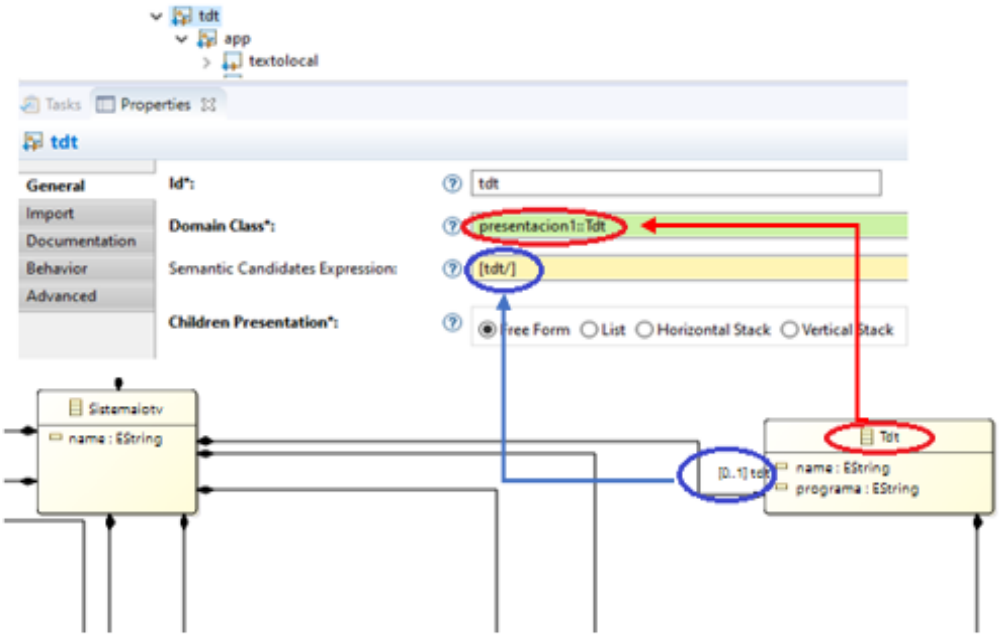
Los elementos de la paleta o relaciones mencionadas en el archivo de diseño, se basan en el metamodelo previamente establecido y el procedimiento de creación inicia con el establecimiento de capas compuestas por nodos, contenedores y bordes de relaciones.

Para la creación de contenedores se define un *id* (nombre) para identificar al contenedor, posteriormente se añade la clase padre que contendrá a todas las demás clases consecuentes y finalmente se introduce el camino por el cual se llega a dicha clase padre, en la Figura 38 se observa la creación del contenedor *tdt* para almacenar a todos los nodos (elementos multimedia, botones, etc) respecto a la aplicación en TDT, este contenedor presenta como clase padre *Tdt* y el camino por el cual se llega desde la clase

raíz *Sistemaiotv* a la clase padre *Tdt* está representado con el nombre *tdt*, estas relaciones se pueden observar en la Figura 38.

Figura 38

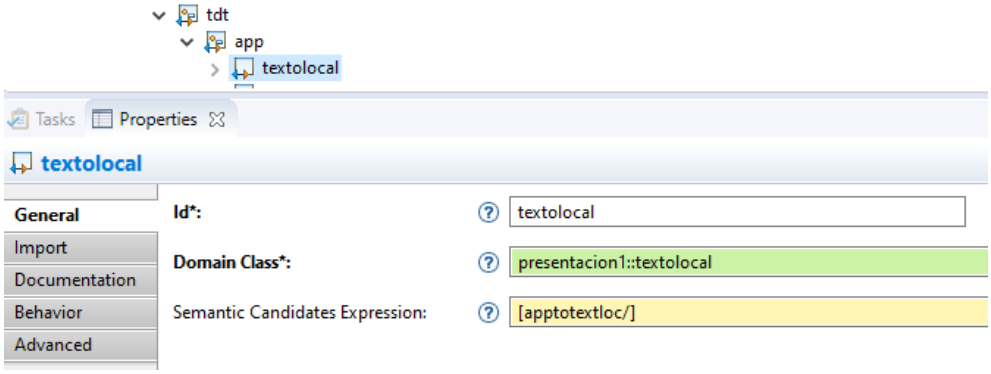
Creación de contenedores para el editor gráfico



La definición de un nodo es muy similar a la de un contenedor, sin embargo, hay que tomar en cuenta que este representará a los elementos disponibles únicamente dentro de cada contenedor, en la Figura 39 se observa la creación del nodo *textolocal* en la cual se define la clase con el mismo nombre y el camino previo definido por la relación *apptotextloc*.

Figura 39

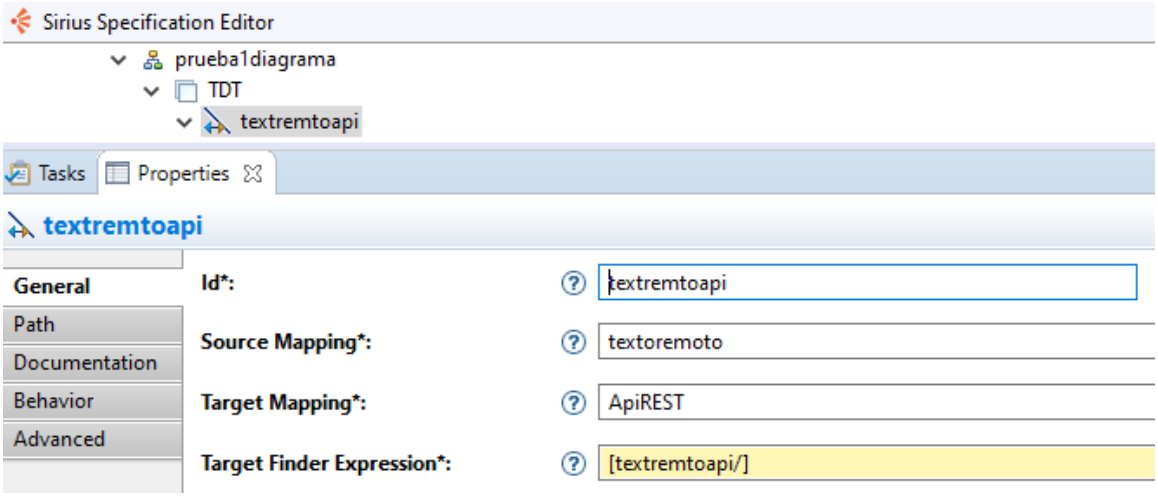
Creación de nodos para el editor gráfico



Para la creación de las relaciones entre los distintos nodos se debe definir un elemento de borde de relación, en el cual hay que definir parámetros como el identificador (*id*), el nodo fuente donde inicia la relación, el nodo destino donde finaliza la relación y el camino por el cual se interconectan, en la Figura 40 se visualiza la creación del borde de relación denominado *textremtoapi* la cual relaciona los nodos *textoremoto* y *ApiREST* mediante el camino *textremtoapi* definido en el metamodelo.

Figura 40

Creación de bordes de relación en editor gráfico

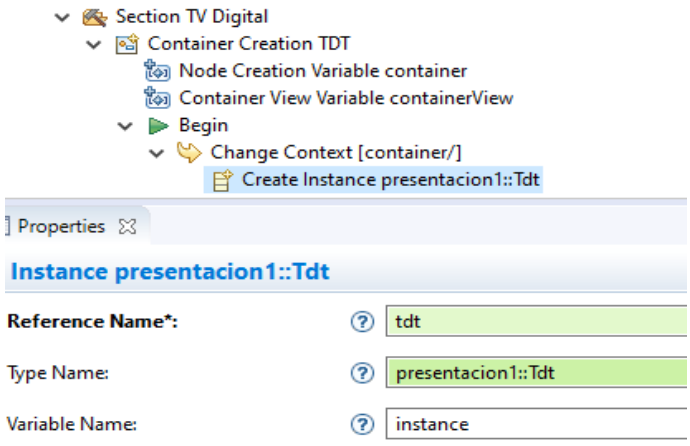


Una vez definidos los elementos contenedores, nodos y bordes de relación, se procede con la creación de la paleta de herramientas a través de la opción *sección* del archivo de diseño, dicha paleta permitirá al usuario visualizar de mejor manera los elementos que conformarán el sistema.

Para crear la representación en la paleta de los contenedores se debe definir un elemento denominado creación de contenedor, en el cual se añadirán tres operaciones (comenzar, cambiar contexto e instanciar) para poder visualizarlo dentro de la paleta. En la Figura 41 se define la creación el elemento contenedor dentro de la paleta y el proceso de instanciación donde se define la clase que se desea mostrar, el nombre de referencia y la clase de variable que debe tomar.

Figura 41

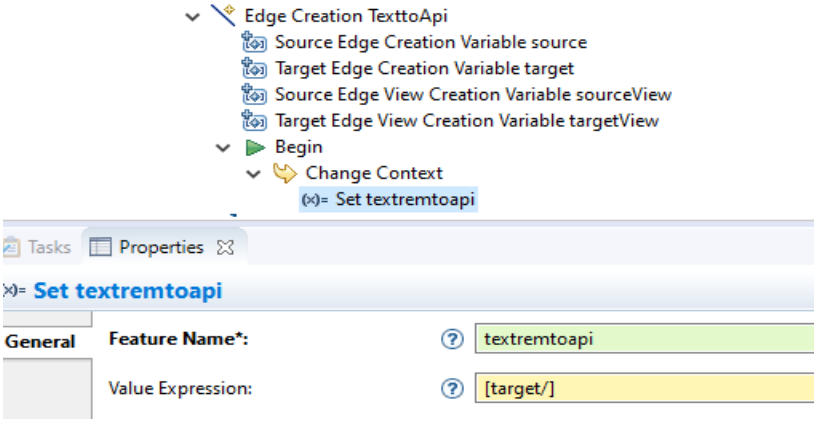
Creación de los elementos contenedores en la paleta de herramientas



El proceso de creación anterior es igual para la instanciación de los nodos y relaciones dentro de la paleta de herramientas, con la diferencia que para las relaciones se establece un proceso de colocación en lugar del de instanciación como se observa en la Figura 42.

Figura 42

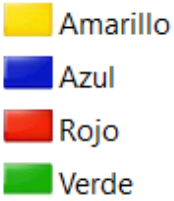
Creación de relaciones para la paleta de herramientas

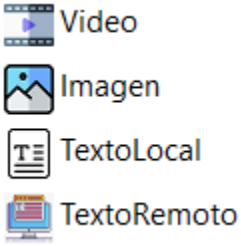


Para definir visualmente cada clase del metamodelo se han propuesto las representaciones gráficas que se observan en la Tabla 2.

Tabla 2

Representación gráfica de las clases del metamodelo

TELEVISIÓN DIGITAL TERRESTRE	
	Icono para representar la interfaz de la TDT.
	Icono para representar la aplicación de la TDT.
	Iconos para representar a los botones de la TDT.

	Iconos para representar los elementos multimedia (video, imagen y texto) dentro de la TDT.
HARDWARE	
	Icono que representa el módulo ESP8266.
	Icono que representa el módulo MKR 1010 WiFi.
	Icono que representa el módulo Arduino OPLA IoT.
	Icono que representa al sensor digital DHT11.
	Icono que representa a los sensores analógicos.
	Icono que representa los sensores digitales.

	Icono que representa a los actuadores.
	SERVICIOS WEB
	Icono que representa a las bases de datos
	Icono que representa al broker MQTT
	Icono que representa al orquestador
	Icono que representa a los servicios de API REST

Nota: En la tabla se describe los iconos que representan a las clases principales del metamodelo.

Transformación M2T

Mediante el uso de la herramienta Acceleo se ha implementado el motor generador de código que hace posible obtener los archivos de ejecución de cada uno de los elementos establecidos en la interfaz gráfica y definidos por el metamodelo.

En la Figura 43 se observa la creación de los archivos que corresponden a los controladores, la función “for” facilita realizar la creación de código de los n elementos que pertenezcan a la clase controlador mientras que la función “if” permite que el código

se genere a partir de la existencia de un controlador en específico (MKR 1010, ESP8266 u OPLA IoT).

En la Figura 44 se establecen las variables que ocuparan cada sensor y actuador relacionado directamente con cada tipo de controlador, mediante la sentencia “for” se accede las clases definidas para sensores o actuadores y según su tipo se asigna como variable el nombre correspondiente al atributo name y el pin definidos como atributos en el metamodelo. Bajo la misma lógica se crean las subscripciones de dichas variables al servidor MQTT.

Figura 43

Generación de Código de los controladores.

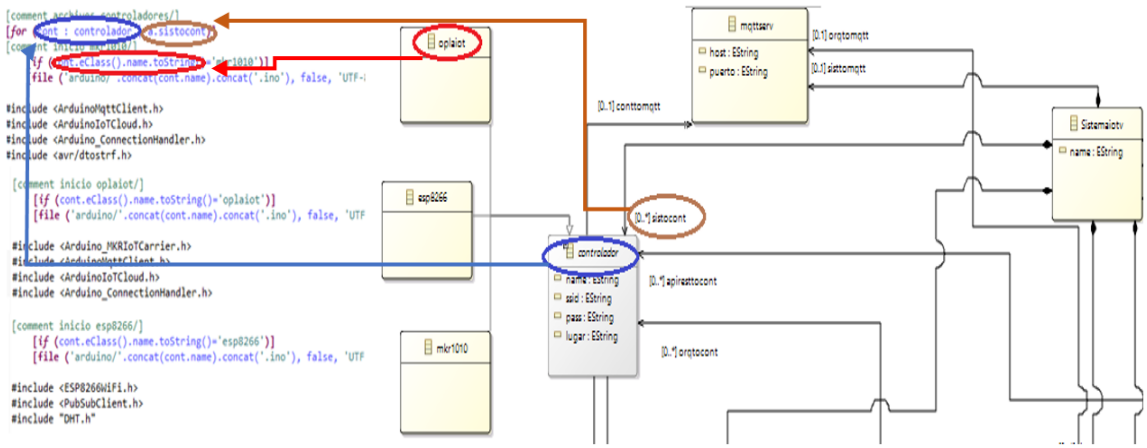


Figura 44

Generación de variables dentro de los controladores

```
[for (tiposen : senanalogico | cont.conttosen)]
[if (tiposen.tipo.toString()='iluminacion')]
int [tiposen.name/];
int const [tiposen.name/]pin=[tiposen.pin/];

[for (tipoact : actdigital | cont.conttoact)]
[if (tipoact.tipo.toString()='led')]
int [tipoact.name/];
const int [tipoact.name/]pin=[tipoact.pin/];

[for (tipoact : actdigital | cont.conttoact)]
[if (tipoact.tipo.toString()='led')]
client.subscribe("jardin/[tipoact.name/]");
[/if]
[if (tipoact.tipo.toString()='ventilador')]
client.subscribe("jardin/[tipoact.name/]");
[/if]
[if (tipoact.tipo.toString()='riego')]
client.subscribe("jardin/[tipoact.name/]");
[/if]
[/for]
```

Definiendo la lógica anterior que se basa en el barrido de elementos y comparación con cada tipo de sensor o actuadores, se observa en la Figura 45 los códigos en general para la lectura de datos de las magnitudes físicas y cambio de estado en los pines de los actuadores.

Figura 45

Lectura de los sensores y activación de actuadores

```
[for (tiposen : senanalogico | cont.conttosen)]
[if (tiposen.tipo.toString()='humedad')]
//read humidity value
int [tiposen.name/]aux = analogRead([tiposen.name/]pin);
[tiposen.name/]aux1 = [tiposen.name/]aux * ((3.3 / 1023.0)*100);
client.publish("jardin/[tiposen.name/]", [tiposen.name/]aux1);
[/if]
[/for]

[for (tipoact : actdigital | cont.conttoact)]
[if (tipoact.tipo.toString()='led')]
if(topic=="jardin/[tipoact.name/]"){
      Serial.print("[tipoact.name/]: ");
      if(messageTemp == "true"){
        digitalWrite([tipoact.name/], HIGH);
        Serial.print("On");
      }
      else if(messageTemp == "false"){
        digitalWrite([tipoact.name/], LOW);
        Serial.print("Off");
      }
  }
```

Para la generación de los archivos de las bases de datos como se muestra en la Figura 46, se verifica la existencia de dicho elemento creado en el editor gráfico y se crean las bases de datos con el nombre según la clase asignada (sensores o actuadores), cada tabla dentro de las bases de datos se creará con el llamado del atributo *nombre* de los sensores o actuadores y se definirá dos columnas con datos iniciales 1 (id) y 0 (valor de la magnitud del sensor o estado del actuador).

Figura 46

Generación de código de las bases de datos

```
[for (auxbd : basedatos | a.sisttobd)]
    [if (auxbd.tipo.toString()='sensores')]
    [file ('basedatos/'.concat('sensores.sql'), false)]
CREATE TABLE `[tiposen.name/]hum` (
  `id` int(11) NOT NULL,
  `dato` float DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

-- --------------------------------------------------------
--
-- Volcado de datos para la tabla `[tiposen.name/]hum`
--

INSERT INTO `[tiposen.name/]hum` (`id`, `dato`) VALUES
(1, 0);
```

Para la generación de código de Node-RED se toma como referencia la Figura 47, donde se observan funciones que relacionan varios elementos como el servidor MQTT, controlador, tipos de sensores y tipos de actuadores, debido a que Node-RED es el puente de comunicación entre todos los servicios web y por ende ocupa los atributos de la mayoría de las clases del metamodelo.

Figura 47

Generación código Node-RED

```
[comment incio node-red/]
[for (auxmqtt : mqttserv | a.sisttomqtt)]
[file ('nodered/'.concat('flows.json'), false, 'UTF-8')]
['['/]
[for (contbd : controlador | a.sistocont)]
[comment incio node-red sensores/]
[for (tiposen : sendigital | contbd.conttosen)]
[if (tiposen.tipo.toString()='iluminacion')]
    {
        "id": "[tiposen.name/]69ce711ff99384ea20",
        "type": "mqtt in",
        "z": "[tiposen.name/]cf186ff0cf7f6577",
        "name": "[tiposen.name/]",
        "topic": "jardin/[tiposen.name/]",
        "qos": "2",
        "datatype": "json",
        "broker": "[tiposen.name/]90af34975437fdf3",
        "nl": false,
        "rap": true,
        "rh": 0,
        "x": 90,
        "y": 460,
        "wires": ['['/]
            ['['/]
                "[tiposen.name/]0d8f2636b0084350"
            ]
        ]
    },
{
    "id": "[tiposen.name/]08e8037fa723930b",
    "type": "http request",
    "z": "[tiposen.name/]cf186ff0cf7f6577",
    "name": "[tiposen.name/] a DB",
    "method": "POST",
    "ret": "obj",
    "paytoqs": "ignore",
    "url": "http://[auxmqtt.host/]:[a.sisttoapi.puerto/]/sensores/[tiposen.name/]/add",
    "tls": "",
    "persist": false,
    "proxy": "",
    "authType": "",
    "credentials": {},
    "x": 570,
    "y": 460,
    "wires": ['['/]
        ['['/]
            "[tiposen.name/]1d731ec923a783d6"
        ]
    ]
},
```

Para la creación del Orquestador se implementan la lógica que se observa en la Figura 48, primero se debe acceder a los atributos host y puerto de la clase orquestador del metamodelo, posteriormente se debe definir la subscripción a cada tópico del servidor MQTT y finalmente se tiene que establecer la condiciones para que este responda según las relaciones entre los sensores y actuadores

Figura 48

Generación código orquestador

```
[comment inicio orquestador/]
[for (orqaux : orquestador | a.sisttoorq)]
[file ('orquestador/'.concat(a.name).concat('orq.js'), false, 'UTF-8')]

//index
const express = require('express');
const router = express.Router();
const app =express();
const delay = require('delay');

//Settings
app.set('port', process.env.PORT || [orqaux.puerto/]);
```

```
[for (tiposen : senanalogico | a.sisttosen)]

[if (tiposen.tipo.toString()='iluminacion')]
    client.subscribe('jardin/[tiposen.name/]')
[/if]
[if (tiposen.tipo.toString()='temperatura')]
    client.subscribe('jardin/[tiposen.name/]')
[/if]
[if (tiposen.tipo.toString()='humedad')]
    client.subscribe('jardin/[tiposen.name/]')
[/if]
[if (tiposen.tipo.toString()='moisture')]
    client.subscribe('jardin/[tiposen.name/]')
[/if]
[for (tiposen : senanalogico | a.sisttosen)]
[if (tiposen.tipo.toString()='iluminacion')]
          case 'jardin/[tiposen.name/]':
          var aux = message.toString();
          global.[tiposen.name/] = aux;
          console.log(aux)
        return tomadatos(message)
```

La base de la generación de código para la TDT comienza con el establecimiento del archivo con extensión “. ncl” por lo cual se verifica la existencia del elemento TDT en el editor gráfico, posteriormente se definen las regiones, descriptores y los elementos media que toman los parámetros definidos en las clases multimedia (video, Img, texto local o texto remoto) dentro del metamodelo como se observa en la Figura 49.

Figura 49

Generación de código para NCL

```
[comment inicio NCL/]
[for (tdtaux : Tdt | a.tdt)]
[if (tdtaux.eClass().name.toString()='Tdt')]

[file ('tdt/'.concat(tdtaux.name).concat('.ncl'), false)]
<?xml version="1.0" encoding="ISO-8859-1"?>
<!-- Generated by NCL Eclipse -->
<ncl id="[tdtaux.name.toString()/]" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">

<head>
       <regionBase>
          <region id="rvideo" height="100%" width="100%" zIndex="1"/>
          <region id="rlua" height="100%" width="100%" zIndex="3"/>
          <region id="rgtitulo" width="[app.anchoapp/]%" height="[app.largoapp/]%" left="[app.leftapp/]%" top="[app.topapp/]%" zIndex="2"/>
          [if (tdtaux.app.apptoimg.eClass().name='Img')]
          <region id="rimg" height="[app.apptoimg.largocont/]%" width="[app.apptoimg.anchocont/]%" left="[app.apptoimg.posleftcont/]%" top="[app.apptoimg.postopcont
          [/if]
```

```
<descriptorBase>
    <descriptor id="dvideo" region="rvideo"/>
    <descriptor id="dtitulo" region="rgtitulo">
    <descriptorParam name="fontSize" value="25"/>
    <descriptorParam name="fontColor" value="green"/>
    </descriptor>

    <descriptor id="dlua" region="rlua" focusIndex="appFocus">
        <descriptorParam name="transparency" value="1%"/>
    </descriptor>
    [if (app.apptoimg.eClass().name.toString()='Img')]
</descriptor>
[/if]
```

```
[for (auxcontenido : App | tdtaux.app)]
<media id="video" src="media/[tdtaux.programa/].mp4" descriptor="dvideo">
    <property name="bounds"/>
</media>
<media id="titulo" src="media/titulo.txt" descriptor="dtitulo">
</media>
<media id="menu" src="media/menu.png" descriptor="dmenu"/>
<media id="info" src="media/exit.png" descriptor="dinfo"/>
<media id="Lua" type="application/x-ginga-settings">
    <property name="service.currentKeyMaster" value="appFocus"/>
</media>
<media id="mfondo" src="media/[tdtaux.app.fondo/]" descriptor="dfondo">
    <property name="transparency" value="[auxcontenido.transparencia/]%"/>
</media>
```

Posteriormente se definen las relaciones de activación de los elementos multimedia a través de los botones, por lo cual se define un camino a través de la clase abstracta *botones* hacia *contenido* con el fin de definir únicamente un tipo de elemento multimedia para cada botón como se observa en la Figura 50.

Figura 50

Generación de código para los elementos multimedia

```
[for (auxcontenido2 : Video | auxcontenido.rojo.bottocont)]
 [if (auxcontenido.eClass().name.toString()='Video')]

    <link xconnector="conector#onKeySelectionStartNStopN">
    <bind role="onSelection" component="imred"/>
    <linkParam name="keyCode" value="RED"/>
    <bind role="start" component="videob"/>

        [for (auxncl : Img | tdtaux.app.apptoimg)]
            [if (auxncl.eClass().name.toString()='Img')]
                <bind role="stop" component="imagen"/>
            [/if]
        [/for]
        [for (auxncl : textolocal | tdtaux.app.apptotextloc)]
            [if (auxncl.eClass().name.toString()='textolocal')]
                <bind role="stop" component="parrafo"/>
            [/if]
        [/for]
        [for (auxncl : textoremoto | tdtaux.app.apptotextrem)]
            [if (auxncl.ntexto=1)]
                <bind role="stop" component="flua"/>
            [/if]
        [/for]

    /link>

    [/if]
[/for]
```

Posteriormente a la creación del archivo “.ncl”, es necesario generar también el archivo “.lua” que contendrá el script para realizar la conexión remota con el servidor web, para ello se verifica en primera instancia que el elemento *texto remoto* se encuentre definido dentro de la *app* de TDT, posteriormente se emplean los atributos de las clases *APIREST*, *sensores* y *actuadores* con el fin de realizar peticiones de tipo GET y POST para acceder y modificar la información dentro de la base de datos para que los actuadores puedan funcionar manualmente o para la activación del orquestador que permitirá que el sistema funcione automáticamente, este procedimiento se lo puede visualizar en el ejemplo mostrado en la Figura 51.

Figura 51

Generación de código Lua

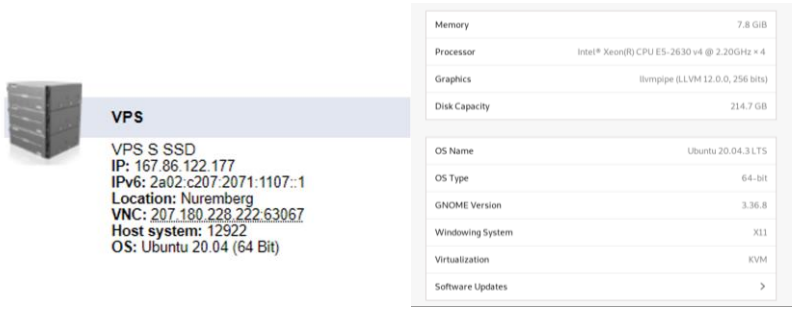
```
[comment inicio textoremoto lua ----------------------------------------------------------------------------------/]
[for (auxlua : apirest | a.sisttoapi)]

[file ('tdt/'.concat('textoremoto').concat('.lua'), false, 'UTF-8')]

[comment humedad/]

local v[auxsen.name/]hum = humvertical -- valor para el primero
humvertical = humvertical+0.05 -- valor para el siguiente

tcp.execute(
        function ()

tcp.connect('[auxlua.host/]', [auxlua.puerto/])
local url[auxsen.name/]hum = 'GET /sensores/[auxsen.name/]hum/1 HTTP/1.1\r\n'
tcp.send(url[auxsen.name/]hum)
tcp.send('Host: [auxlua.host/]:[auxlua.puerto/]\r\n')
tcp.send('Connection: keep-alive\r\n')
tcp.send('Upgrade-Insecure-Requests: 1\r\n')
tcp.send('User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/95.0.4638.69 Safari/537.36\r\n')
tcp.send('Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n')
tcp.send('Accept-Encoding: gzip, deflate\r\n')
tcp.send('Accept-Language: es-ES,es;q=0.9\r\n')
tcp.send('\r\n')


             result[auxsen.name/]hum = tcp.receive()

-- Escritura de archivos
             file[auxsen.name/]hum = io.open("TEXTO2.txt","w")
             file[auxsen.name/]hum:write(result[auxsen.name/]hum)
             file[auxsen.name/]hum:close()

           file[auxsen.name/]hum=io.open("TEXTO2.txt","r")
          a[auxsen.name/]hum= file[auxsen.name/]hum:read ("*all")
          i, j = string.find(a[auxsen.name/]hum, "dato") --buscar indice
          [auxsen.name/]hum= string.upper(string.sub(a[auxsen.name/]hum,j+3,string.len(a[auxsen.name/]hum)-2))   --palabra desde el indice

         tcp.disconnect()
```

Servidores Web

Cabe recalcar que para el funcionamiento de cada uno de los servicios se utilizó una VPS (Virtual Private Server), gracias a esto se obtuvo una dirección IP por la cual se realiza la interacción de cada uno de los servicios con sus diferentes puertos definidos en una única máquina, en la Figura 52 se detallan las características del VPS utilizado.

Figura *52*

Características principales del Servidor Virtual Privado utilizado



API REST

Se desarrolló un único servicio REST que incluye las bases de datos de sensores y actuadores, en primera instancia, se determina la creación del puerto bajo el cuál va a funcionar el servicio, posteriormente se realiza la conexión a las bases de datos, estableciendo los parámetros de *host*, *user*, *password* y *database*, a continuación, se crean las peticiones GET y POST por medio de funciones de enrutamiento, estableciendo las respectivas URL direccionadas hacia las bases de datos creadas, para las peticiones GET las URL establecidas tienen la siguiente estructura:

Para obtener valores de toda la tabla:

host:puerto/base de datos/tabla(nombre de sensor o actuador)

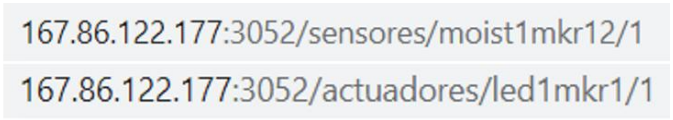
Para obtener un valor específico dentro de la tabla:

host:puerto/base de datos/tabla(nombre de sensor o actuador)/id

En la Figura 53 se muestra un ejemplo para las bases de datos creadas.

Figura 53

Ejemplo de URL para peticiones GET hacia bases de datos



Con respecto a las peticiones POST, se establecen dos tipos de URL tanto para datos que se ingresan de manera incremental en la tabla como para datos que van a actualizar el valor de un id específico dentro de la tabla, estas peticiones se rigen bajo las siguientes estructuras:

Insertar: *host:puerto/base de datos/ tabla(nombre de sensor o actuador)/add*

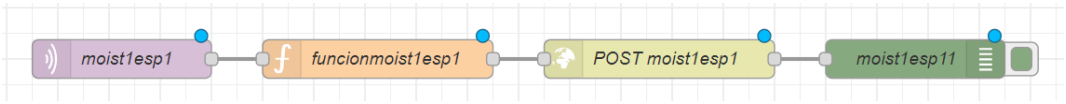
Actualizar: *host:puerto/base de datos/ tabla(nombre de sensor o actuador)/update/id*

Node-RED

En la Figura 54 se visualiza el flujo correspondiente a la publicación de valores provenientes de los sensores en las bases de datos creadas. Para esto, se utilizó el nodo *mqtt in* como suscriptor de los tópicos establecidos, el cuál abstrae los valores booleanos provenientes de los sensores, para posteriormente mediante el nodo de función realizar la conversión a valores JSON acorde a las tablas diseñadas, estos valores se van a publicar con métodos POST en nodos de requerimiento HTTP en las tablas por medio de dos tipos de flujos, uno de los flujos ubica los datos de manera incremental, mientras que el otro actualiza la primera fila de cada tabla acorde a lo establecido en la API REST.

Figura 54

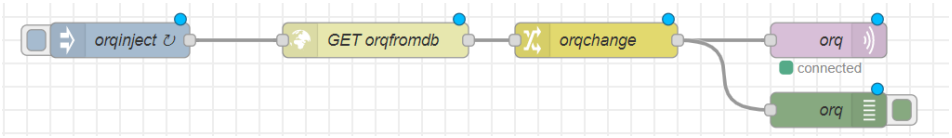
Flujos desplegado para sensores en Node-RED



La Figura 55 muestra el flujo de obtención del estado del orquestador en Node-RED, esto se obtiene del valor enviado por la aplicación TDT al momento de elegir un tipo de control tanto manual como automático, el dato obtenido de la tabla se envía por MQTT para su utilización tanto en el orquestador como en la aplicación TDT; este flujo se establece de manera recurrente con la inyección de peticiones bajo un tiempo determinado.

Figura 55

Obtención y publicación de estado de orquestador en Node-RED



Para la obtención y publicación de los valores correspondientes a actuadores, en la Figura 56 se detalla el proceso que inicia con la verificación del estado del orquestador desde la tabla en la base de datos, si este valor correspondiente al estado es igual a 0, se determina como control manual, mientras que, si es 1, se traduce como control automático en la aplicación TDT; en este caso, el nodo *switch* toma el dato y realiza una comparación para guiar el flujo a diferentes procesos que se detallan a continuación.

Al obtener un 0 como estado, el flujo se conduce hacia otra consulta, en este caso para obtener el valor directamente de la variable establecida en el editor gráfico que corresponde a la interacción de uno de los actuadores predefinidos, la consulta se la realiza al último valor detectado en la tabla del actuador, posteriormente se realiza la publicación mediante MQTT; este flujo se establece de manera recurrente con la inyección de peticiones bajo un tiempo determinado.

Si se obtiene el valor de 1 como estado, el flujo establece un proceso de combinación con el valor de la variable obtenida (true o false) de la suscripción previa al tópico MQTT del actuador utilizado, si existen estos dos valores, entonces se permite el paso del flujo hacia el siguiente nodo *switch*, en este nodo se dirige el valor booleano que llega por MQTT a dos salidas determinadas por los valores *true* o *false*, una vez dado este caso se continua con la conversión del dato a formato JSON por medio del nodo *función* y posteriormente se utiliza el método POST para la publicación del valor en la base de datos en la tabla del actuador correspondiente.

Figura 56

Flujo desplegado para actuadores en Node-RED

Orquestador

Esta herramienta establece en primera instancia la conexión con el broker MQTT, para esto, es necesario conocer el host y puerto MQTT utilizado. Una vez realizada la conexión, se procede a realizar la suscripción a los tópicos establecidos para cada variable dada por los sensores y actuadores, a continuación, mediante una declaración switch con instancias case para cada tópico suscrito, se asigna una variable que contiene el valor dado por los sensores, la cual permitirá posteriormente establecer las comparaciones con los valores umbrales, para poder realizar estas comparaciones se establece en primer lugar una función que permite verificar el valor del estado del orquestador que es enviado por la aplicación TDT, y que se traduce en el tipo de control que va a ser desplegado.

Si el valor obtenido es 0, se desactiva la interacción de las variables con los valores umbrales para la activación o desactivación de los actuadores, es decir, el usuario de la aplicación TDT está determinando un control manual del cultivo, mientras que si el valor obtenido es 1, el usuario determina un control automático del cultivo, esto quiere decir que se activa la utilización de las variables para la comparación con los valores umbrales definidos por el usuario con el fin de activar o desactivar los actuadores desplegados en el escenario de funcionamiento, esto se realiza mediante la publicación de valores booleanos por medio de MQTT para el tópico requerido, lo detallado anteriormente posibilita un monitoreo constante del escenario de aplicación en el presente proyecto.

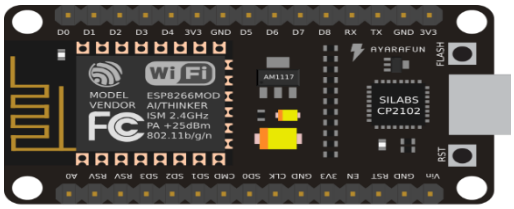
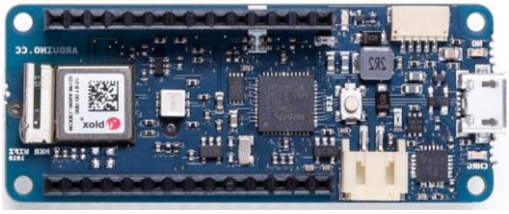
Hardware

Para el desarrollo del sistema aplicado al Smart Agro, se utilizan controladores Arduino como: ESP8266 y el MKR WiFi 1010 tanto solo, como con el uso del Carrier MKR IoT, gracias a sus características se puede incluir los sensores y actuadores necesarios para el despliegue del sistema, las Tablas 3,4 y 5 detallan los dispositivos a utilizar, entre

estos constan: controladores, carrier, sensores y actuadores que permiten ejemplificar el sistema dentro del dominio del Smart Agro.

Tabla 3

Controladores a utilizar

Dispositivo	Características principales
ESP8266 	• V. Entrada: 3.3V • Protocolos: 802.11 b,g,n • P. salida: 0.15 W, +19.5 dBm en 802.11b • Consumo en baja energía: <10 uA • Incluye procesador de 32 bits • Dimensiones: (24 x 16) mm
Arduino MKR WiFi 1010 	• Microcontrolador: MCU ARM de baja potencia SAMD21 • V. entrada: 5V • V. funcionamiento circuito: 3.3 V • 8 pines E/S digitales, 13 Pines PWM (0-8, 10, 12, 18 / a3, 19 / a4); 1 pin UART; 1 pin SPI; 1 pin I2C. • 7 pines analógicos • 1 pin de salida analógica de 10 bits • Corriente D/C por pin: 7ma • Memoria flash interna CPU de 256 kB • SRAM de 32 kB • 32,768 kHz/48 MHz de velocidad de reloj • USB de alta velocidad y host integrado • Dimensiones: (61.5 x 25) mm

Nota: Controladores a utilizar para el sistema Smart Agro propuesto.

Tabla 4

Carrier IoT

Dispositivo	Características principales

Arduino MKR IoT Carrier 	• Sensor LPS22HB de presión • IMU: LSM6DS3 • Sensor de humedad y temperatura: HTS221 • Sensor de luz ambiental, gestos y proximidad: APDS-9960 • 5 botones capacitivos • Zumbador • 2 relés V23079 de 24 V • 5 LEDs RGB • Monitor: KD013QVFMD002-01 • Ranura para tarjeta SD • Soporte para batería recargable de iones de litio 18650

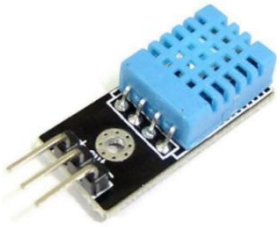
Nota: Descripción del Carrier a utilizar en las implementaciones aplicadas al Smart Agro para el presente proyecto.

Se determinan sensores tanto analógicos como digitales para su uso, los cuáles miden las variables de humedad del ambiente, temperatura, humedad del suelo e intensidad de luz, a su vez, entre los actuadores con el uso de relés se puede utilizar dispositivos que se acoplen a los parámetros de salida, entre estos están luces LED, ventiladores y para el riego el uso de bombas de agua de 12 V.

Tabla 5

Sensores y actuadores

Dispositivo	Características principales
Sensor capacitivo de humedad del suelo 	• Admite interfaz de sensor de 3 pines • Salida analógica • V. de funcionamiento CC: 3.3-5.5 V • Voltaje de salida CC: 0-3 V • Interfaz: PH2.0-3P • Dimensiones: 99x16 mm

DHT11 	• Voltaje de operación: de 3,5 V a 5 V • Consumo de corriente: 2,5 mA • Salida: Digital Temperatura • Rango: 0 a 50ºC • Precisión: 25ºC ± 2ºC • Resolución: 1ºC (8-bits) Humedad • Rango: de 20 a 90% RH • Precisión: entre 0 y 50ºC ± 5% RH • Resolución: 1% RH
HTS221 (en MKR IoT Carrier) 	• Humedad relativa: 0 a 100 % • Voltaje de alimentación: 1.7 a 3.6 V • Consumo de corriente: 2 µA a 1 Hz ODR • ODR seleccionable de 1 Hz a 12,5 Hz • Alta sensibilidad de rH: 0,004% RH /LSB • Precisión humedad: ± 3,5% RH, 20 a +80% RH • Precisión temperatura: ± 0,5 °C, de 15 a +40 °C • Integra ADC de 16 bits • Salida de temperatura y humedad de 16 bits • Interfaces SPI e I2C
LM35 	• Voltaje de alimentación: 4 a 30V • Rango de trabajo: -55 a +150℃ • Precisión de rango: -10 a +85°C: ±0.5°C • Pendiente: 10mV / ºC • Bajo consumo de energía: 60uA • Pines: +VCC, Voltaje salida, GND • Impedancia de salida baja
LDR 	• Resistencia en luz: 200 Ohm • Resistencia en oscuridad: 15 kOhm • Máximo voltaje DC: 150 V • Máximo consumo: 100mW

Relés V23079 (en MKR IoT Carrier) 	• Tensión de la Bobina: 12VDC • Configuración de Contacto: DPDT • Corriente de Contacto: 2A • Rango de Producto: P2/V23079 • Tensión de Contacto VAC: 250V • Tensión de Contacto VDC: 220V • Material del Contacto: Plata-Níquel • Resistencia de Bobina: 1.029kohm
Módulo Relé 12V 	• 2 canales independientes protegidos con Optoacopladores. • Voltaje de Operación: 12V • Corriente de salida: 10A • Corriente de activación por relé: 15 a 20 mA • Aplica aislamiento • Alerta tipo jumper de señal de disparo (bajo o alto) • Cada canal contiene un led indicador • Dimensiones: 40mm x 50mm
Relé 5V 	• Tensión de bobina: 5V DC • Corriente de contacto: 10 A • Tipo de bobina: Sin enclavamiento • Voltaje de contacto VAC máximo: 250 VAC / 10 A • 5 terminales

Nota: Descripción de las características principales de los sensores y relés para el presente proyecto.

Capítulo V: Pruebas de Validación

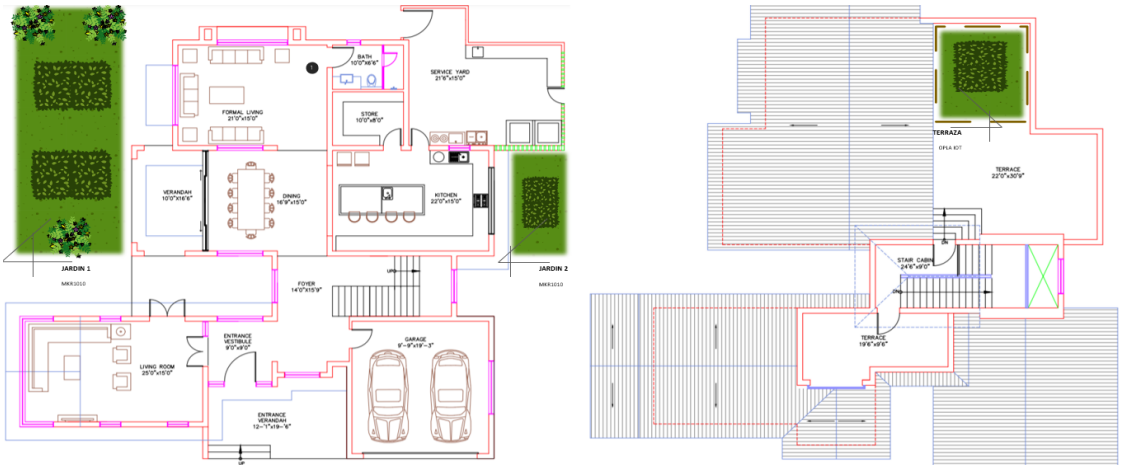
Pruebas de funcionamiento

Planteamiento del escenario del sistema

Se ha planteado un escenario que corresponden a cultivos caseros en diferentes zonas de una vivienda como se visualiza en la Figura 57, el primer y segundo cultivo se establecen como jardines al exterior de la casa y un tercer cultivo en la terraza.

Figura 57

Planteamiento de escenario de prueba.



Aplicación interactiva TDT con enfoque al Smart Agro

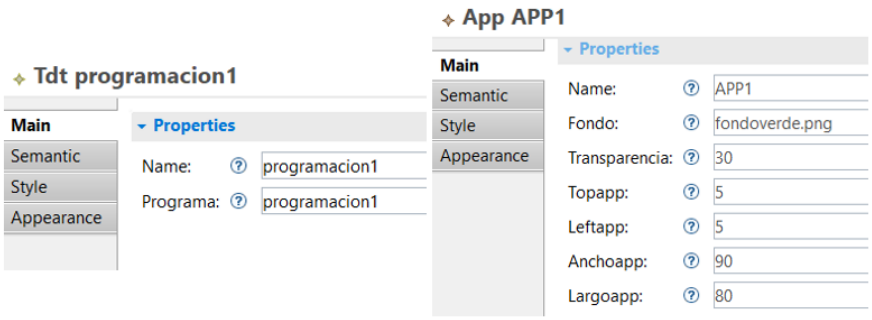
Se dispuso utilizar el editor gráfico diseñado con la herramienta Sirius como se detalló en el Capítulo anterior, para el diseño de las aplicaciones se realizan las acciones de arrastrar y conectar los recursos dispuestos en la paleta de herramientas descritos en la Tabla 2; para la creación de la plantilla de la aplicación interactiva, se ha dispuesto realizar a modo de ejemplo una plantilla con varios textos remotos, para esto, se siguieron los pasos descritos a continuación.

Se eligió la interfaz denominada *TDT* en la pestaña *TV* Digital de la paleta de herramientas, en la cual se establece un nombre y una programación definida en la

pestaña propiedades, en este caso *programacion1*; posteriormente, se definen los valores correspondientes al recurso *APP,* se detalla el nombre así como la ruta del archivo con su extensión que se impondrá como fondo, donde aparecerá la información concerniente a los recursos multimedia, en este caso *fondoverde.png*, además se define la transparencia, posiciones y las dimensiones, tal como se muestra en la Figura 58.

Figura 58

Configuraciones para elementos TDT y APP en editor gráfico



Nota: Descripción de parámetros configurados, TDT (Izquierda) y APP(Derecha).

Una vez agregados tales elementos, dentro de *APP* se insertan los recursos multimedia, para el ejemplo expuesto se han utilizado todos los recursos disponibles: *Imagen, Video*, *Texto Local* y *Texto Remoto*, además de sus propiedades como rutas, dimensiones y posiciones, se ha asignado un botón a cada uno, dentro de los cuales se permite configurar las mismas propiedades. Al utilizar *Texto Remoto* es necesario detallar también el número de texto remoto que se utiliza, así como el lugar, además es necesario que exista la conexión con el elemento *apirest*, lo que conlleva a la inserción de los servicios web, lo expuesto se detalla en la Figura 59.

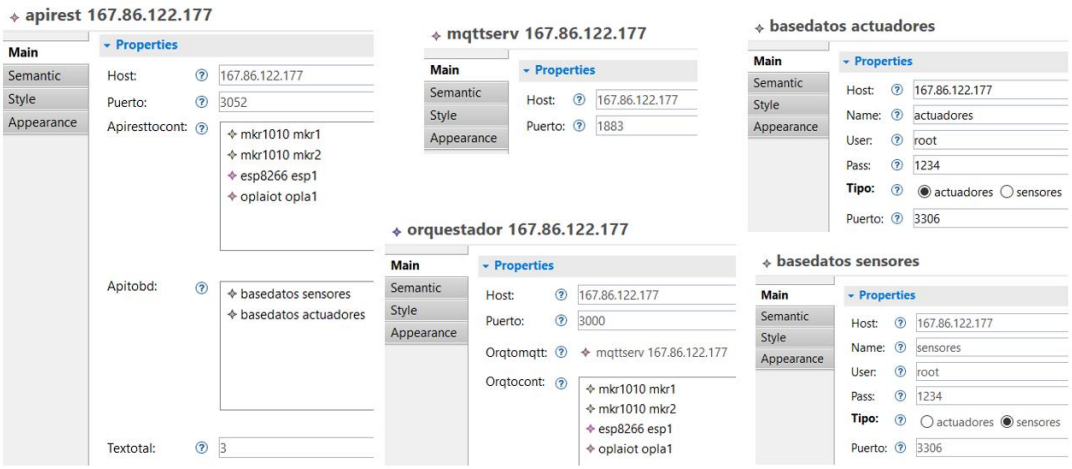
Figura 59

Configuración de elementos multimedia dentro de editor gráfico

Para el uso de los servicios web, se configuran cada uno de los elementos que representan a los servicios web, en este caso: *apirest*, *mqttserv*, *orquestador* y las dos *basedatos*, estableciendo la información que se muestra en la Figura 60. Como se puede apreciar acorde a los atributos establecidos en el metamodelo, se configura la información específica de cada elemento web, además se realiza la asignación de las relaciones como: agregar las bases de datos y controlador a la apirest; asignar el broker MQTT y el controlador al orquestador y finalmente la elección del tipo de base de datos sea esta sensores o actuadores.

Figura 60

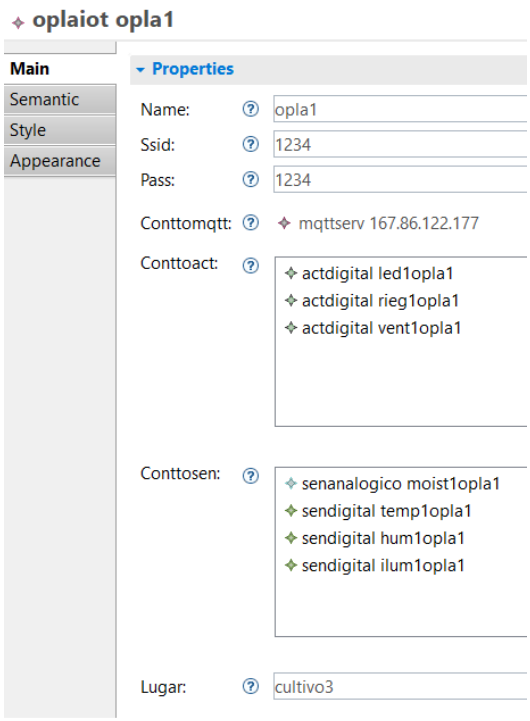
Configuraciones para el *apirest*, *mqttserv*, *orquestador* y *basedatos*



Para la sección de Dispositivos se consideró establecer la mayoría de los recursos utilizables en la paleta de herramientas, en el ejemplo que se muestra a continuación se utilizan los controladores OPLA, ESP8266 y MKR 1010, los cuales tienen wifi, pines analógicos y digitales. En los controladores se configuran el nombre y la conexión a la red de Internet, para esto, se solicita ingresar los datos de *SSID* y *pass* (contraseña), el *id* y *lugar* que permiten la identificación y asignación de los mismos a los cultivos pertenecientes, además, se asignan los sensores y actuadores a usar, así como el broker al que se conecta y el orquestador por el cual va a ser administrado, esta información se puede apreciar en la Figura 61.

Figura 61

Configuración de parámetros de controlador en editor gráfico

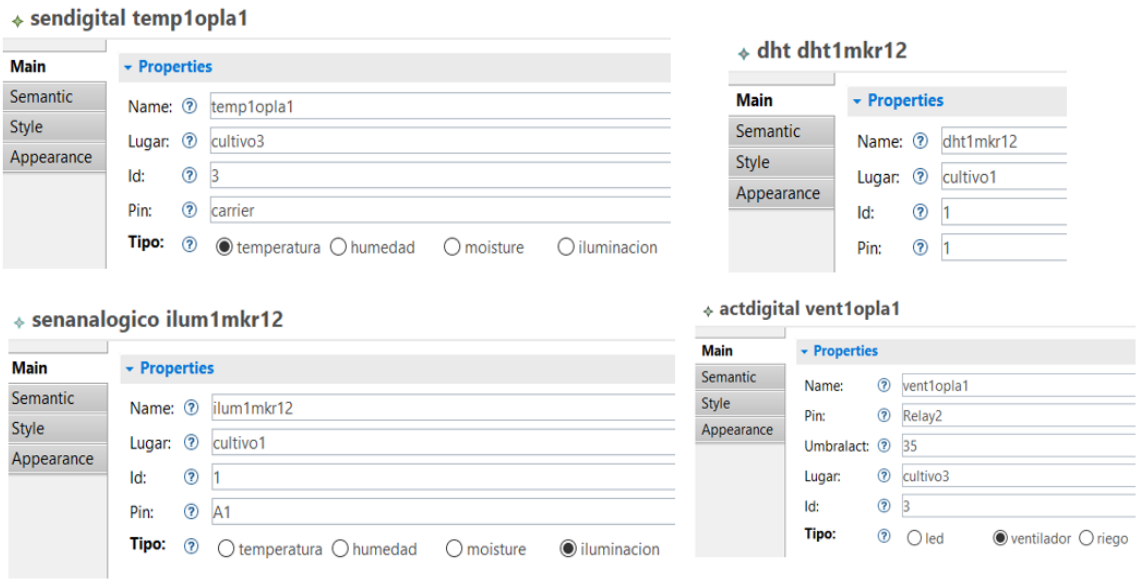


Para los sensores analógicos y digitales se configuran los mismos valores como son: *nombre* y *pin*, se asigna el controlador, *id* y *lugar* permiten la identificación y

asignación de los mismos a los cultivos pertenecientes, además, se elige el *tipo* de variable que mide de entre las opciones preestablecidas; para los actuadores existe una configuración similar, con la diferencia que se debe establecer el *umbral de activación* de la variable elegida, estos parámetros de sensores y actuadores se muestran en la Figura 62. Para el caso del DHT11 se asigna *nombre, id, lugar* y el *pin* por el cual se va a conectar al controlador asignado.

Figura 62

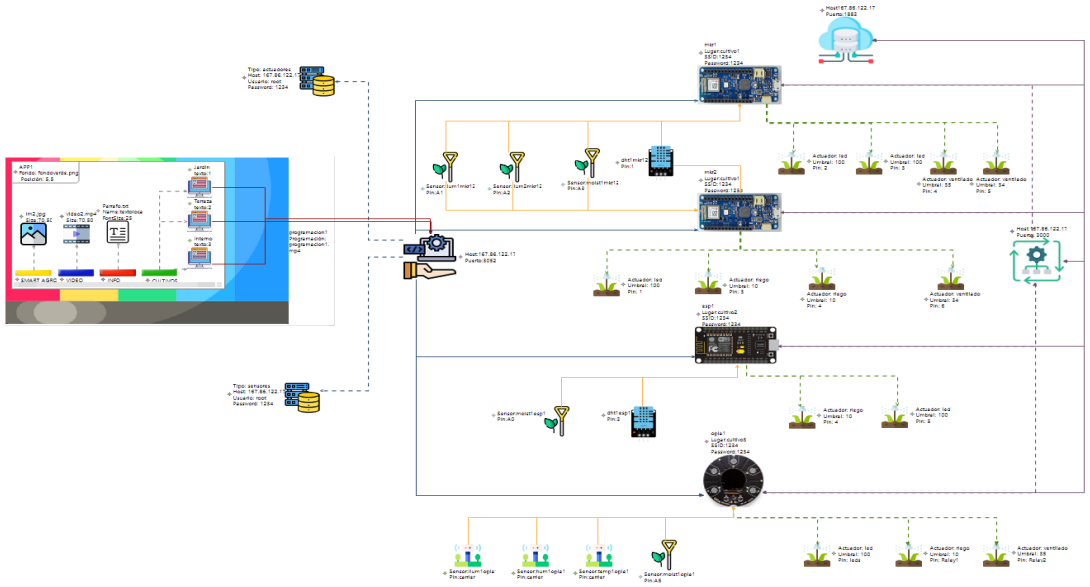
Configuración de parámetros sensores y actuadores en editor gráfico



Luego de haber realizado las configuraciones para los elementos, se obtiene el diseño gráfico del sistema de TDT enfocada al Smart Agro de la Figura 63, en la cual se observan de mejor manera la creación de una aplicación para TDT compuesta de archivos multimedia como (imagen, video y texto) y que es capaz de interactuar remotamente con un servidor web para obtener los valores almacenados en las bases de datos mediante el APIREST.

Figura 63

Diseño de Aplicación TDT interactiva enfocada al Smart Agro



La Figura 63 también muestra la interacción de cada controlador con los respectivos servicios web como el APIREST, el orquestador y el servidor MQTT, debido a que es elemento principal para el monitoreo de estado de los cultivos, adicionalmente se observan que existe la posibilidad de combinar sensores o actuadores entre uno o más controladores con la finalidad de solventar las necesidades de escalamiento mediante los pines de los controladores que el usuario necesite , además se ha incluido en la vista de la interfaz gráfica del editor, la información configurada en cada elemento a fin de que el usuario pueda comprobar los parámetros introducidos más fácilmente.

Sin embargo, con el fin de reforzar la comprensión del sistema, se han añadido en la Tabla 6 los sensores y actuadores con sus respectivos pines de forma que puedan ser identificados por el lugar en el que se encuentran implementados o por el tipo de controlador al que están relacionados.

Tabla 6

Disposición de elementos de hardware acorde a DSL gráfico

	Sensor Analógico	Sensor Digital	DHT11	Actuador	Pin
Lugar	***Jardín 1***				
Controlador 1	***MKR1010***				
Temperatura			*dht1mkr12*		*1*
Humedad			*dht1mkr12*		*1*
Iluminación	*ilum1mkr12*				*A1*
	ilum2mkr12				*A2*
Moisture	*moist1mkr12*				*A3*
Led				*led1mkr1*	*2*
				led2mkr1	*3*
Ventilador				*vent1mkr1*	*4*
Riego					
Controlador 2	***MKR1010***				
Temperatura			*dht1mkr12*		*1*
Humedad			*dht1mkr12*		*1*
Iluminación	*ilum1mkr12*				*A1*
	ilum2mkr12				*A2*
Moisture	*moist1mkr12*				*A3*
Led				*led1mkr2*	*2*
Ventilador				*vent1mkr2*	*3*
Riego				*rieg1mkr2*	*4*
				rieg2mkr2	*5*
Lugar	***Jardín 2***				
Controlador	*ESP8266*				
Temperatura			*dht1esp1*		*3*
Humedad			*dht1esp1*		*3*
Iluminación					
Moisture	*moist1esp1*				*A0*
Led				*led1esp1*	*5*
Ventilador					
Riego				*rieg1esp1*	*4*
Lugar	***Terraza***				
Controlador	*OPLA IOT*				

Temperatura	*temp1opla1*		*Carrier*
Humedad	*hum1opla1*		*Carrier*
Iluminación	*ilum1opla1*		*Carrier*
Moisture	*moist1opla1*		*A5*
Led		*led1opla1*	*Leds*
Ventilador		*vent1opla1*	*Relay2*
Riego		*rieg1opla1*	*Relay1*

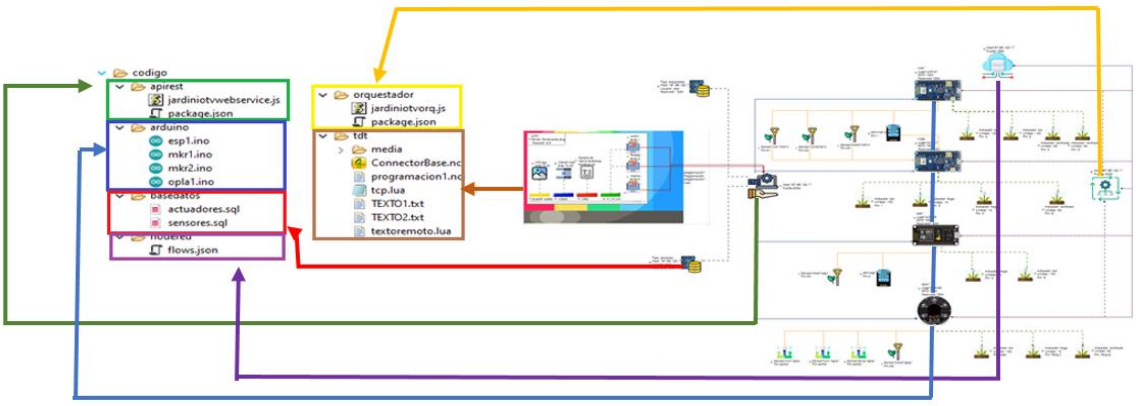
Nota: En la tabla se describe la disposición de los elementos de hardware para cada uno de los cultivos establecidos en el editor gráfico.

Código generado

Una vez finalizado la estructura del diseño gráfico del sistema, se procede a ejecutar el motor de generación automática de código, para obtener así los archivos que se observan en la Figura 64 correspondientes a los servidores web, hardware y aplicación de TDT.

Figura 64

Código generado del sistema diseñado



Los archivos generados respecto a los servicios de APIREST y el Orquestador son basados en el lenguaje de JavaScript, por lo cual son subidos a la plataforma Visual Studio Code dentro de la máquina virtual. De manera similar para las bases de datos se

han creado archivos con extensión .sql y los flujos para la comunicación entre los distintos servicios en archivos de extensión JSON, los cuales tienen que ser importados dentro de las plataformas de administración de base de datos y de Node-RED correspondientemente.

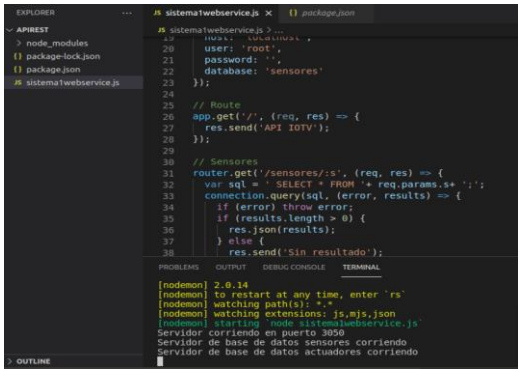
Los archivos correspondientes a los controladores tienen una extensión. ino por lo cual deben ser abiertos desde el IDE de Arduino para ser ejecutados, finalmente se obtiene el código correspondiente a NCL dentro de la TDT, el cual deberá ser abierto mediante un emulador de Ginga para ver la operatividad de la aplicación. Los archivos textoremoto.lua y tcp.lua son los scripts que permiten establecer la comunicación remota y no deben ser modificados, ni ser cambiados de posición, deben encontrarse siempre en la raíz.

API REST

Se desarrolló un único servicio REST que incluye las bases de datos de sensores y actuadores, en la Figura 65 se puede visualizar a modo de ejemplo la configuración del método GET para responder ante las peticiones con un archivo JSON que contenga los valores del sensor solicitado.

Figura 65

Servicio Web desplegado para las bases de datos utilizadas

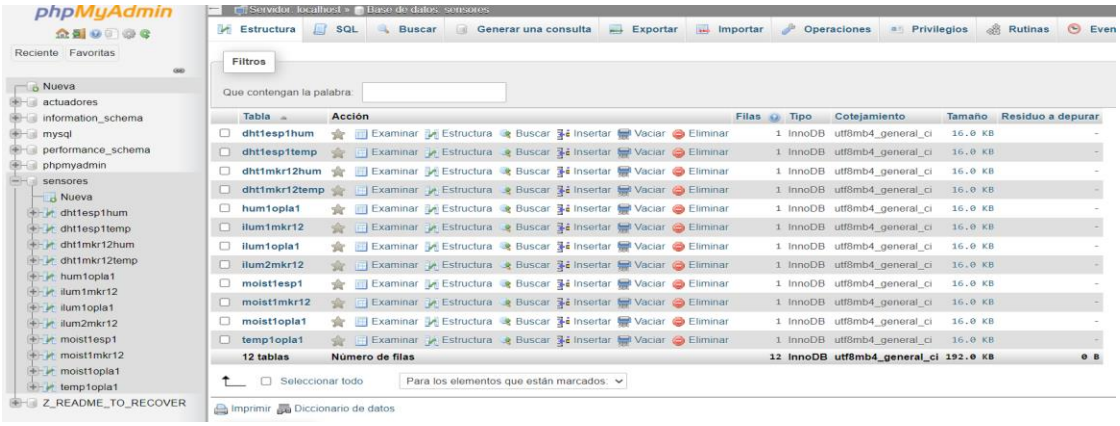


Base de datos

Las Figura 66 y Figura 67 muestran las bases de datos creadas para los sensores y actuadores utilizados en el escenario de pruebas, también muestra el uso de la herramienta phpMyAdmin para la administración de las mismas.

Figura 66

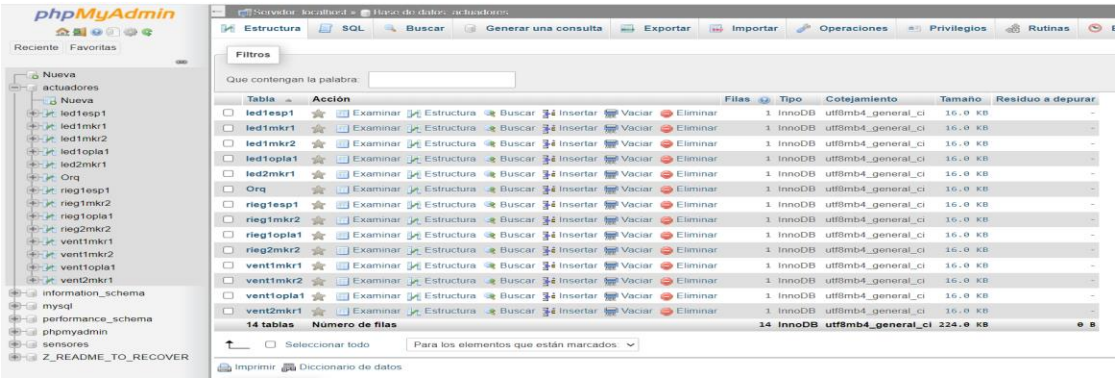
Base de datos sensores



Las tablas creadas para los sensores fueron diseñadas para ser auto incrementables, para así poder visualizar el histórico de los datos mediante una petición GET por parte del usuario al APIREST, además los datos son de tipo entero debido a que muestran magnitudes relacionadas a fenómenos físicos.

Figura 67

Base de datos actuadores



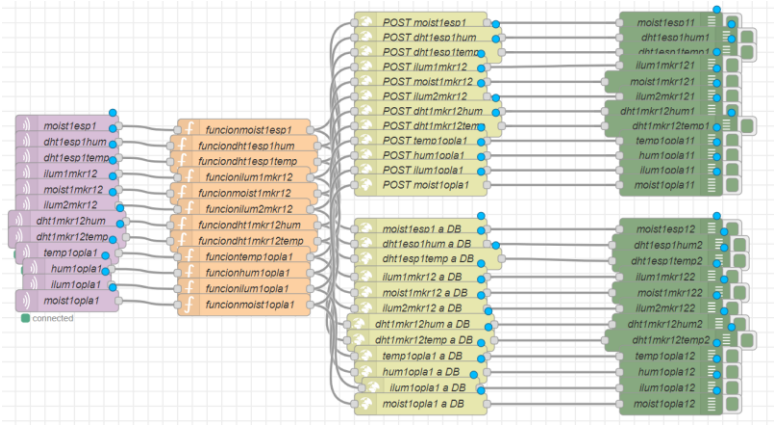
Las tablas correspondientes a los controladores fueron diseñadas para no ser auto incrementales y se definieron tipos de datos *string*, debido a que los actuadores se basan en una estructura de *on* y *off* para su funcionamiento.

Node-RED

En la Figura 68 se visualiza el flujo correspondiente a la publicación de valores provenientes de los sensores en las bases de datos creadas, este flujo comienza por escuchar las publicaciones de valores con sus respectivos tópicos en MQTT, posteriormente estos datos son transformados mediante una función a la estructura JSON para ser enviados a las bases de datos mediante una solicitud tipo POST al APIREST.

Figura 68

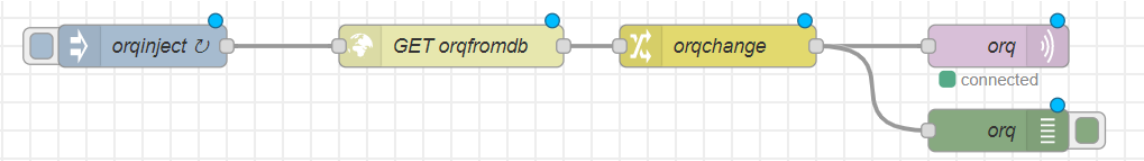
Flujos desplegado para sensores en Node-RED



La Figura 69 muestra el flujo de obtención del estado del orquestador en Node-RED, este flujo se caracteriza debido a que mediante un inyector se realizan peticiones de tipo GET para consultar el estado del orquestador en la base de datos en caso que el usuario lo haya activado mediante la aplicación de TDT, posteriormente mediante una conversión JSON a entero se publica el dato por medio de MQTT para que el orquestador también reciba este valor y pueda activarse o desactivarse.

Figura 69

Obtención y publicación de estado de orquestador en Node-RED

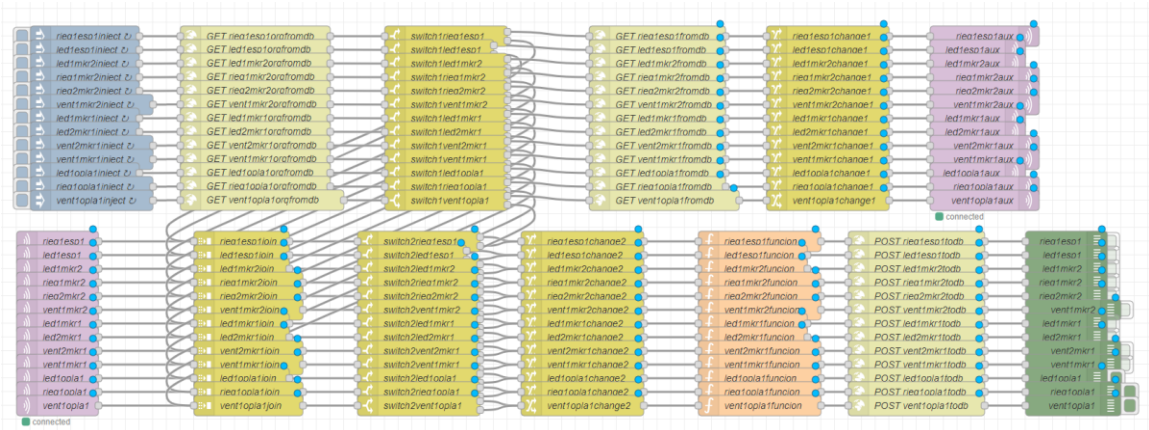


En la Figura 70 se detalla el proceso para la obtención y publicación de los valores correspondientes a actuadores, la parte superior de la imagen representa el caso en el cual el orquestador se encuentra apagado (Manual en la TDT), la publicación del estado de los actuadores se lo realizará desde la base de datos (por medio de peticiones GET) hacia los controladores (por medio de MQTT).

La parte inferior de la imagen representa el caso en que el orquestador se encuentre prendido (Automático en TDT), la publicación del estado de los actuadores dependerá únicamente del orquestador, por lo cual esté será el encargado de publicar directamente el estado por medio de MQTT a los controladores y también de actualizar el valor dentro de la base de datos mediante el método POST definido en la APIREST.

Figura 70

Obtención y publicación de datos de actuadores en Node-RED

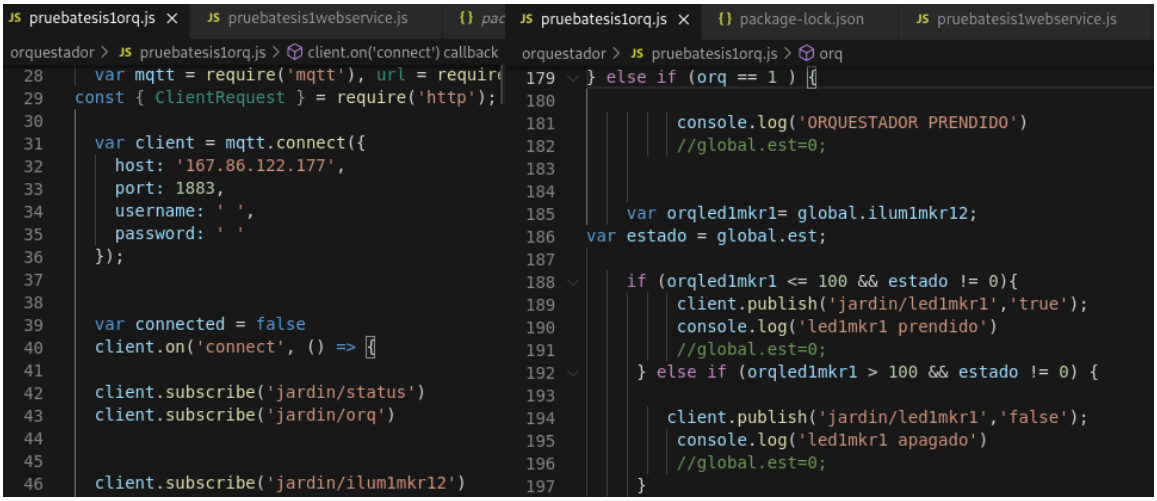


Orquestador

En la Figura 71 se muestra el despliegue del código del orquestador, en la parte derecha de la figura se aprecia la conexión al servidor MQTT mediante el establecimiento del host y puerto en donde está implementado dicho servicio, también se visualiza la suscripción a cada tópico publicado en MQTT.

Figura 71

Orquestador desplegado en la VPS utilizada



Por otra parte, en el lado derecho de la figura se aprecia el establecimiento de las condiciones a partir de los valores de las magnitudes de los sensores y los umbrales definidos para publicar a través de MQTT valores booleanos y que los controladores a través de dichos valores puedan cambiar el estado de los pines (low o high).

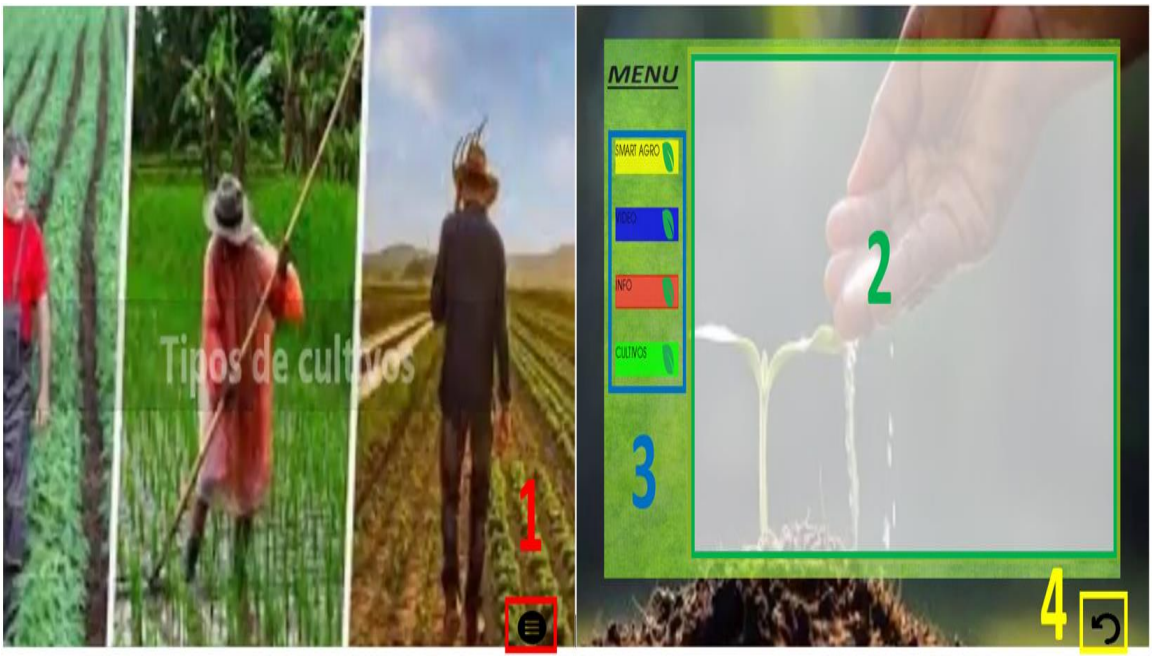
TDT

Para su visualización dentro de la TDT, se utiliza la herramienta Ginga NCL, al ingresar a la aplicación, es necesario presionar el botón *Menú* (1), cuyo ícono se muestra en la esquina inferior derecha de la programación, posteriormente, se muestra la aplicación acorde a los parámetros configurados en el editor gráfico. Para el diseño realizado a modo de ejemplo del presente proyecto, la interfaz se encuentra con

transparencia y sobrepuesta en la pantalla, por lo cual no incide en la programación principal (2) y sobre este fondo definido se muestran los recursos multimedia configurados, estas características se muestran en la Figura 72.

Figura 72

Interfaces desplegadas en aplicación TDT



Acorde a lo implementado en la aplicación TDT, se establecen los botones correspondientes a las opciones de recursos multimedia a elegir (3), además del botón *Salir* (4), ubicado en la esquina inferior derecha, el mismo que permite retornar a la programación sin la interfaz del menú desplegado anteriormente, a continuación, en la Figura 73 se muestran los diferentes recursos multimedia que se incluyen dentro de la interfaz dispuesta.

Figura 73

Recursos multimedia integrados en aplicación TDT

Implementación del sistema

Para presentar el escenario propuesto se ha realizado una maqueta que contiene un cultivo a monitorear acorde a la aplicación creada anteriormente, conjuntamente se ha dispuesto ubicar los sensores de temperatura, humedad e iluminación alrededor de la misma para obtener las medidas óptimas de las variables, en el caso del sensor capacitivo de humedad del suelo, se ha procedido a ubicarlo dentro de la tierra del cultivo. Para los actuadores, la luz utilizada se procuró ubicarla en la parte superior de la maqueta para una mayor exposición hacia el cultivo, de la misma manera el ventilador se ubica en la esquina superior de la maqueta facilitando la asimilación del aire hacia la planta, en el caso que se active dada una alta temperatura; para el caso de la bomba de agua, se ha elaborado un pequeño sistema de riego por el cual desde un recipiente al activarse el actuador se bombea el agua contenida para recorrer la manguera que determina el riego

al cultivo, el agua regresa al recipiente estableciendo la reutilización de la misma, en la Tabla 7 se detallan los materiales utilizados.

Tabla 7

Lista de materiales del escenario utilizado

Materiales y Dispositivos	
• Maqueta de madera de 20x40 cm • Recipiente de plástico tipo maceta • Planta de prueba	
• Fuente de 12V, 0.5 A	
• Ventilador de 5V, 0.24 A	
• Bomba de 12 V, 5 W y • Manguera de 2 metros de largo con 1.5 cm de diámetro	
• Foco de 12 V	

- Controladores, sensores y actuadores detallados en las Tablas 2,3 y 4.

- Cables para Arduino

Nota: En la tabla se describe los materiales y dispositivos utilizados para el monitoreo del cultivo de prueba.

Una vez obtenida la aplicación creada en Sirius y el código generado por Acceleo subido a las respectivas plataformas, se dispone a comprobar el funcionamiento del sistema aplicado al escenario creado, para este caso con el uso del OPLA IoT Kit de Arduino, se muestra la aplicación con los valores obtenidos por los sensores del carrier, además del estado de los actuadores acorde a los valores umbral predefinidos, la Figura 75 muestra el escenario de implementación de la aplicación interactiva.

Figura 74

Aplicación TDT con OPLA IoT

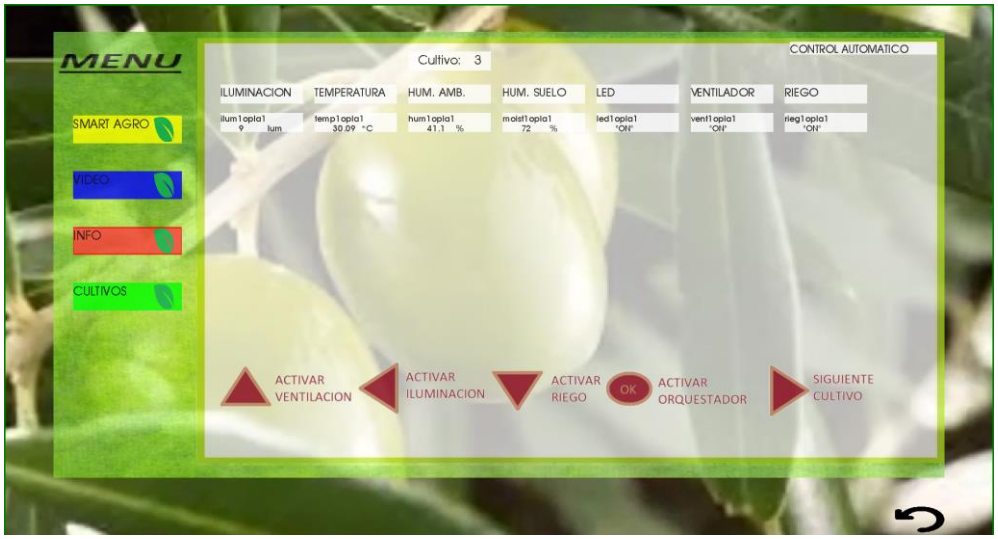


Figura 75

Escenario de implementación con Opla IoT Kit

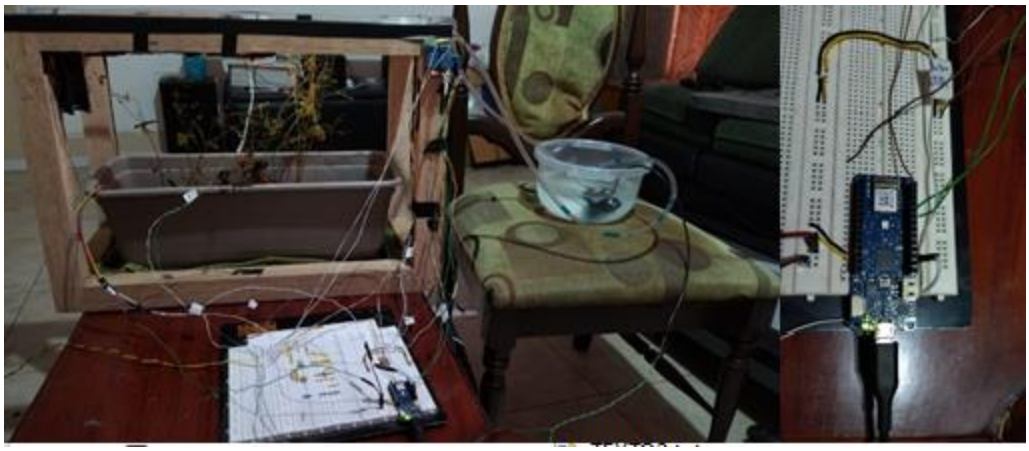
Así también se comprueba el funcionamiento para las demás placas citadas anteriormente, en la Figura 76 y Figura 77 se puede la aplicación TDT y el entorno físico de prueba para el caso del controlador MKR WiFi 1010.

Figura 76

Aplicación TDT con MKR WiFi 1010

Figura 77

Escenario de implementación con MKR WiFi 1010



En la Figura 78 se observa la aplicación diseñada para el controlador ESP8266 en la cual se configuró el sensor de iluminación en la única entrada analógica del controlador, en la Figura 79 se observa el escenario físico de implementación del sistema.

Figura 78

Aplicación TDT con ESP8266

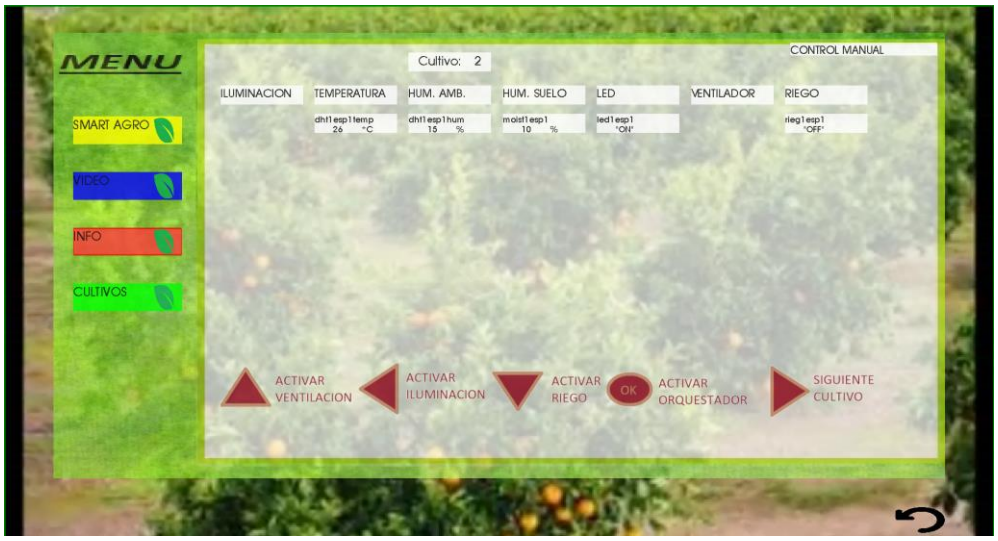


Figura 79

Escenario de implementación con ESP8266

Pruebas de carga

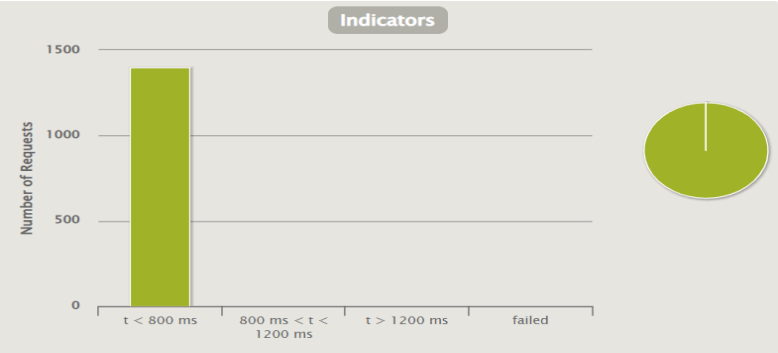
Para la comprobación del comportamiento del servicio web planteado que establece el funcionamiento de las aplicaciones, se ha dispuesto utilizar la herramienta Gatling por medio del software IntelliJ IDEA Educational Edition. Para este propósito, se establece el escenario por el cual se realizan tanto las peticiones POST y GET, que describen la interacción entre los valores enviados por los sensores hacia las bases de datos y también los valores obtenidos por las aplicaciones TDT.

Dadas las consideraciones, se determinó realizar las peticiones de manera simultánea para los dos métodos descritos y para cada tabla establecida en las bases de datos (actuadores y sensores) que correspondan a los elementos del controlador OPLA IoT, al ser 7 tablas, se realiza el análisis con 14 peticiones (7 POST, 7 GET) por usuario con distintas cantidades de usuarios como se describe a continuación:

En la Figura 80 se muestra el resultado de cargar al sistema con 100 usuarios realizando un total de 1400 peticiones, de las cuales tiene un porcentaje de falla del 0% y un tiempo de respuesta promedio correspondiente a 240ms.

Figura 80

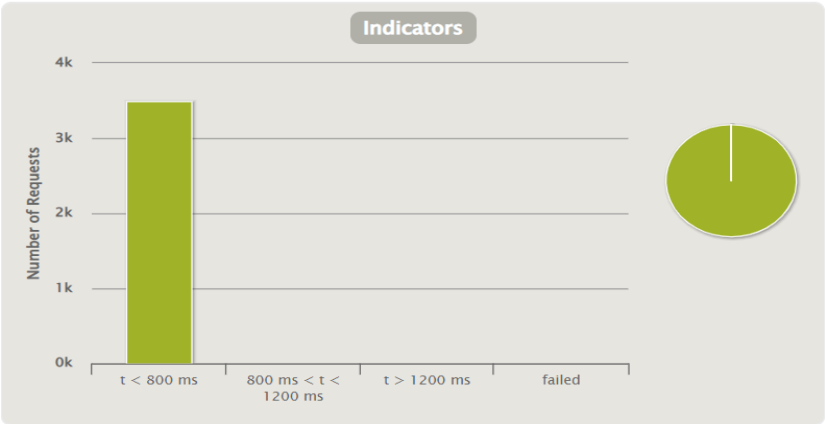
Pruebas de carga con 100 usuarios



En la Figura 81 se observa el resultado de cargar al sistema con 250 usuarios y un total de 3500 peticiones simultaneas, de las cuales se tiene un porcentaje de fallo del 0% y un tiempo promedio de respuesta de 247ms.

Figura 81

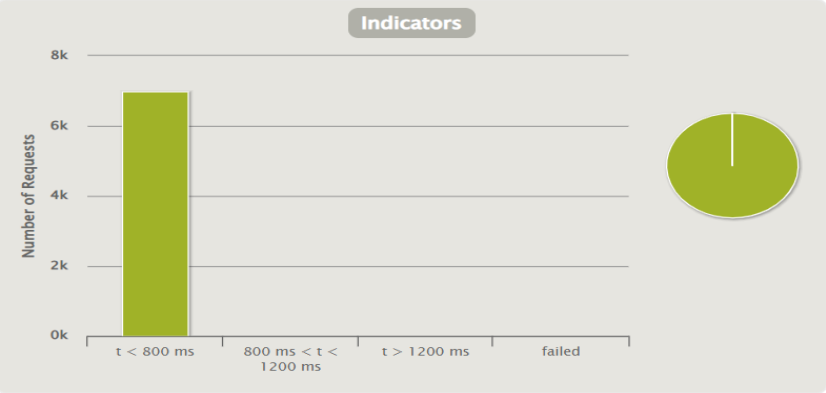
Prueba de carga con 250 usuarios.



En la Figura 82 se observa el resultado de cargar el sistema con 500 usuarios y un total de 7000 peticiones, de las cuales se obtiene un tiempo promedio de respuesta de 247ms y un porcentaje de error del 0.014% correspondiente a una petición fallida.

Figura 82

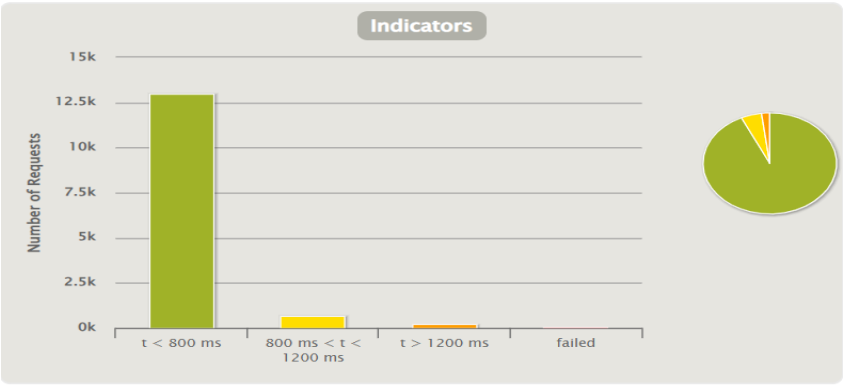
Prueba de carga con 500 usuarios



En la Figura 83 se observa el resultado de cargar el sistema con 1000 usuarios y un total de 14000 peticiones, de las cuales se tiene que un error de falla correspondiente al 0.41% el cual corresponde a 58 peticiones, además se observa peticiones que sobrepasan el valor de envió a 800 ms, sin embargo, el tiempo promedio de respuesta es de 338 ms.

Figura 83

Prueba de carga con 1000 usuarios

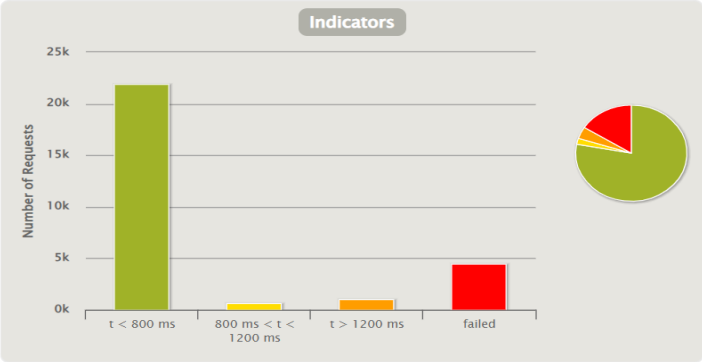


En la Figura 84 se observa el resultado de haber cargado al sistema con 2000 usuarios y un total de 28000 peticiones de las cuales, se han obtenido un error de

peticiones fallidas del 15.86% correspondiente a 4443 peticiones, con un tiempo promedio de respuesta de 1.99 segundos debido a los tiempos correspondientes a las peticiones fallidas.

Figura 84

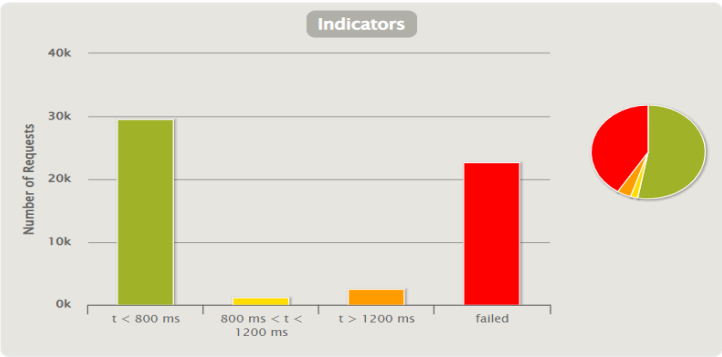
Prueba de carga con 2000 usuarios



En la Figura 85 se observa el resultado de cargar el sistema con 4000 usuarios y un total de 56000 peticiones, de las cuales se tiene un porcentaje de fallo del 40.64% correspondiente a 22758 peticiones y un tiempo promedio de respuesta de 4.39s.

Figura 85

Pruebas de carga del sistema con 4000 usuarios

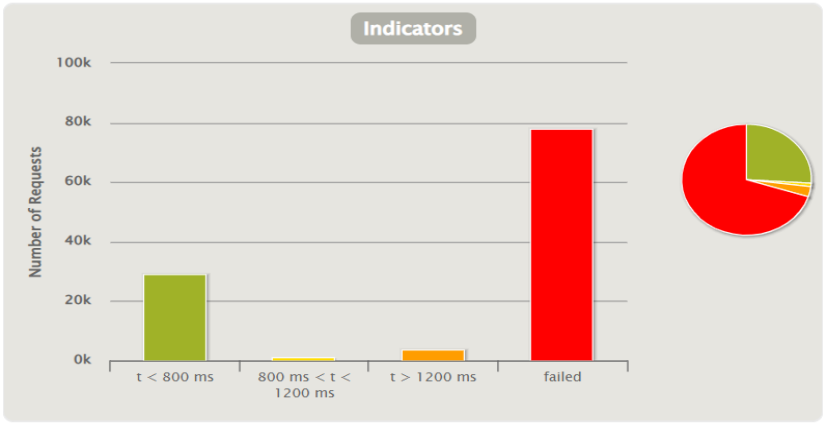


En la Figura 86 se observa el resultado de someter al sistema a una carga de 8000 usuarios y un total de 112000, de las cuales se tiene un porcentaje de fallo del

69.43% correspondiente a 777989 peticiones y un tiempo promedio de respuesta de 7.19 segundos.

Figura 86

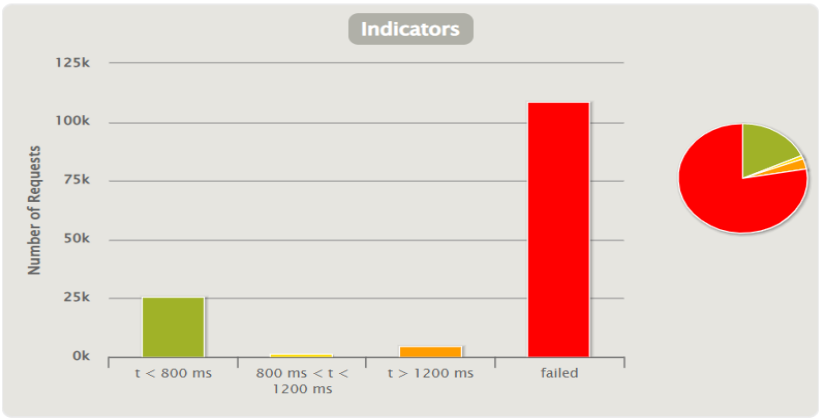
Prueba de carga con 8000 usuarios



Como se observa en la Figura 87 se ha sometido al sistema a una carga de 100000 usuarios y un total de 140000 peticiones, de las cuales se tiene un porcentaje de falla del 77.46% correspondiente a 108450 peticiones y un tiempo promedio de respuesta de 7.98 segundos

Figura 87

Prueba de carga con 10000 usuarios



En la Tabla 8 se observa el resumen de los datos adquiridos de las pruebas de carga realizadas por medio de la herramienta Gatling, en donde se aprecia que al aumentar el número de usuarios y peticiones hacia el servidor web existe una mayor tasa de peticiones fallidas y por ende un mayor tiempo de respuesta.

Tabla 8

Estadísticas de rendimiento del servicio web

N° usuarios	N° de peticiones	N° de peticiones fallidas	% de error	Tiempo promedio de respuesta
100	1400	0	0.00	240 ms
250	3500	0	0.00	247 ms
500	7000	1	0.01	247 ms
1000	14000	58	0.41	330 ms
2000	28000	4443	15.87	1.99 s
4000	56000	22758	40.64	4.39 s
8000	112000	77989	69.63	7.19 s
10000	140000	108450	77.46	7.98 s

Nota: Se presentan los valores de rendimiento obtenidos acorde a la cantidad de usuarios y número de peticiones con la herramienta Gatling.

Puntos de función

Como herramienta de estimación del esfuerzo y duración del proyecto de software se ha dispuesto utilizar el método por puntos de función, para este propósito, el diseño del proyecto presentado se establece como producto de software final las herramientas de editor gráfico y generación de código, las mismas que se visualizan por medio de las

aplicaciones que se permiten crear a través de las mismas. En la Tabla 9 se detallan los valores determinados para la primera fase del análisis, la cual corresponde a los puntos de función sin ajustar, como primer parámetro se ha elegido un nivel de complejidad bajo y acorde a la tabla estandarizada se dispuso asignar puntajes que al sumarlos reflejan el factor de peso, valor necesario para el posterior análisis de estimación del esfuerzo.

Tabla 9

Puntajes para factor de peso

Factor de Peso					
	Parámetros de Medida (1)			Contador	Total Multiplicador
Factores Funcionales de Peso	Baja	Media	Alta	(2)	$(1) * (2)$
N° Entradas externas (EI)	3	4	6	1	3
N° Salidas externas (EO)	4	5	7	6	24
N° Consultas externas (EQ)	3	4	6	1	3
N° Archivos Lógicos Internos (tablas) (ILF)	7	10	15	7	49
N° Interfaces externas (EIF)	5	7	10	0	0
					Factor de Peso = 79

Nota: Se determina el factor de peso acorde a los puntajes asignados para un proyecto de software con complejidad baja.

Para la siguiente fase de estimación correspondiente a los puntos de función de ajuste, en la Tabla 10 se describen 14 características establecidas como niveles de influencia para la determinación del factor de ajuste.

Tabla 10

Niveles de influencia para factor de ajuste

Niveles de influencia	Puntaje
Comunicación de Datos	5
Procesamiento Distribuido	4
Objetivos de Rendimiento	1
Configuración del equipamiento	1
Tasa de transacciones	1
Entrada de Datos en Línea	4
Interfaces con el usuario	3
Actualizaciones en Línea	2
Procesamiento Complejo	2
Reusabilidad del Código	5
Facilidad de Implementación	4
Facilidad de Operación	1
Instalaciones Múltiples	2
Facilidad de Cambios	2
Factor de Ajuste	37

Una vez obtenido el factor de ajuste se aplica la función descrita a continuación para adquirir el valor de punto de función ajustado:

$$PFA = PFSA * [0.65 + (0.01 * factor\ de\ ajuste)]$$

$$PFA = 79 * [0.65 + (0.01 * 37)]$$

$$PFA = 80.58 \cong 81$$

Para la estimación del esfuerzo y duración, se debe elegir uno de los lenguajes detallados en la Tabla 11, donde se describen las horas promedio y el número de líneas

de código por punto de función; para el presente proyecto se elige la opción *Lenguajes de 4ta Generación*, obteniendo un valor de 8 horas promedio de punto de función y 20 líneas de código por punto de función.

Tabla 11

Parámetros para estimación de esfuerzo

Lenguaje	Horas de punto de función promedio	Líneas de código por punto de función
Ensamblador	25	300
COBOL	15	100
Lenguajes 4ta Generación	8	20

Luego de haber obtenido los valores de punto de función ajustado y horas de punto de función promedio, se procede a calcular el esfuerzo en horas/hombre para el proyecto de software expuesto y la duración en meses del mismo:

$$Horas\ hombre = PFA * Horas\ de\ punto\ de\ función\ promedio$$

$$Horas\ hombre = 81 * 8$$

$$Horas\ hombre = 648$$

Estimando 7 horas diarias de trabajo, se tienen 92.57 días de trabajo, lo que en meses se traduce a 4.63 meses para desarrollar el software, en tanto que, al utilizar las herramientas desplegadas en el presente proyecto para el desarrollo de aplicaciones, toma un tiempo aproximado de 4 horas.

Pruebas de usabilidad

Para el presente proyecto se establece utilizar la herramienta de Escala de Usabilidad de un Sistema (SUS) que permite medir de la usabilidad de las herramientas y aplicaciones desplegadas a partir del análisis de 10 preguntas. Para este propósito se

contó con la ayuda de estudiantes, egresados y profesionales de áreas como ingeniería electrónica, software, sistemas y ciencias relacionadas a la agricultura, quienes realizaron el uso de la herramienta y posteriormente una encuesta de la cual se ha procedido a dividir sus resultados en preguntas impares y pares como se observa en la Figura 88 y Figura 89 correspondientemente.

Figura 88

Resultados de preguntas impares de la encuesta

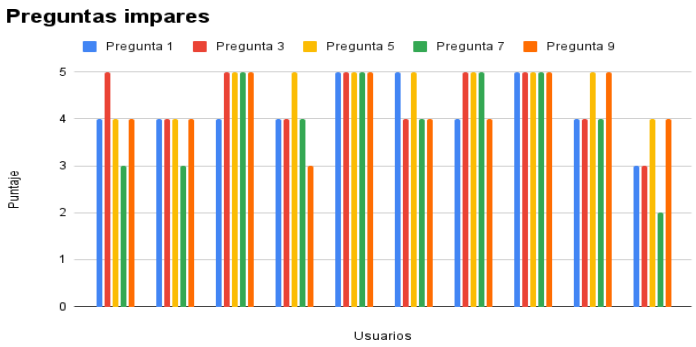
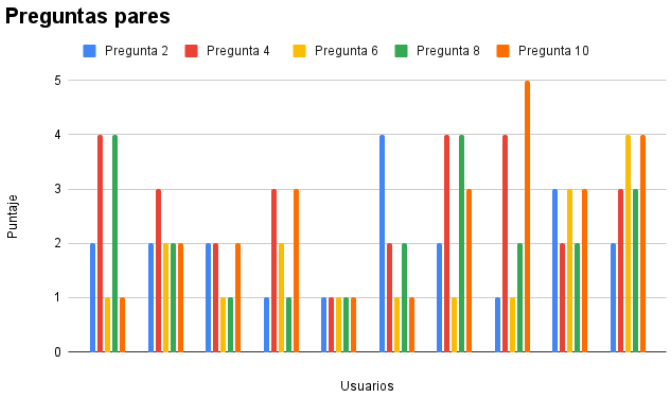


Figura 89

Resultados de preguntas pares de la encuesta



Con los resultados obtenidos se procedió a realizar el cálculo del puntaje de usabilidad mediante la aplicación de su algoritmo, tal como se observa en la Tabla 12.

Tabla 12

Resumen de resultados obtenidos aplicando escala SUS

Usuarios	Resultados Escala de SUS										Sumatoria	Factor (x2.5)
	P1	*P2*	*P3*	*P4*	*P5*	*P6*	*P7*	*P8*	*P9*	*P10*		
1	3	3	4	1	3	4	2	1	3	4	28	70
2	3	3	3	2	3	3	2	3	3	3	28	70
3	3	3	4	3	4	4	4	4	4	3	36	90
4	3	4	3	2	4	3	3	4	2	2	30	75
5	4	4	4	4	4	4	4	4	4	4	40	100
6	4	1	3	3	4	4	3	3	3	4	32	80
7	3	3	4	1	4	4	4	1	3	2	29	72,5
8	4	4	4	1	4	4	4	3	4	0	32	80
9	3	2	3	3	4	2	3	3	4	2	29	72,5
10	2	3	2	2	3	1	1	2	3	1	20	50
											Promedio	76

Nota: En la tabla se puede observar el procedimiento y resultado del puntaje de usabilidad del sistema.

El resultado obtenido corresponde al valor de 76, el cual representa que el sistema diseñado está en condiciones óptimas para ser utilizado por los usuarios.

Capítulo VI: Conclusiones y Recomendaciones

Conclusiones

Se desarrolló un DSL en base a Ingeniería Dirigida por Modelos, el cual está constituido por un metamodelo definido por clases padres e hijas, se utilizó las relaciones correspondientes para elementos de composición, referencia y supertipos, estas relaciones describen la convergencia entre la TDT, IOT, Smart Agro y diferentes herramientas de software (API REST, Orquestador, MQTT, Node-RED) , además, se dispuso el uso de clasificadores por numeración para las clases que determinaron límites de ejecución en cuanto a los tipos de sensores, actuadores y bases de datos; los atributos dentro de cada clase fueron obtenidos mediante un alto nivel de abstracción, analizando la constitución de cada uno de los elementos de hardware y software mediante pruebas de funcionamiento previo.

Con el uso de la herramienta Sirius en Eclipse fue posible el desarrollo de un editor gráfico basado en las clases y relaciones definidas previamente, a partir de esto se creó en primera instancia la representación por la cual se anexa el editor con las propiedades dispuestas en el metamodelo, se resalta la importancia del uso de contenedores que permitieron la generación de nodos agrupados y conectores bajo características comunes como Televisión Digital, Servicios Web y Dispositivos, esto favoreció el establecimiento de la paleta de herramientas de manera ordenada y asimilable para el usuario, en este mismo sentido la herramienta Sirius otorga la facilidad del uso de restricciones con respecto a la cantidad de elementos, conexiones, acciones de borrado y reconexión, siempre bajo la lógica establecida por el metamodelo.

Se realizó la programación individual de cada uno de los elementos integrados en el editor gráfico dentro de los entornos de desarrollo de la siguiente manera: TDT en NCL y Lua desplegados en Eclipse; Orquestador y API REST en JavaScript desplegados en

Visual Studio Code; Bases de datos en MySQL administrados por phpMyAdmin; Hardware (controladores, sensores, actuadores) en Arduino IDE, para esto se establecieron programas base y escalables enfocados en Smart Agro, cuya relación fue posible con la ejecución de pruebas de funcionamiento previas.

Se desplegó un motor generador de código automático con la herramienta Acceleo integrado con EMF, a partir del cual se obtienen los archivos a ejecutar en los diferentes entornos de desarrollo, al ser un lenguaje intuitivo, facilitó la validación referente a los elementos iterativos desarrollados en el editor gráfico, para la generación de código bajo una misma lógica, ruta y lenguaje de programación; para esto fue imprescindible el uso de sentencias condicionales “for” e “if”, además de la sentencia “file” para diferenciar las plataformas a las cuáles se direccionó cada archivo generado.

Se estableció un sistema que permite al usuario la creación de aplicaciones interactivas por medio del editor gráfico, tanto locales como remotas con la capacidad de integrar servicios web con la TDT, esto se logró con la utilización de la librería tcp.lua con los parámetros *execute*, *connect*, *send* y *disconnect*, además de los métodos HTTP GET y POST, para esto fue esencial la obtención de las cabeceras generadas por cada una de las peticiones, se debió tomar en cuenta la longitud de cada dato dado por la estructura establecida en JSON y almacenada en las tablas correspondientes a cada sensor y actuador creado. Las aplicaciones que implementa el usuario disponen de la interacción de elementos multimedia como videos, imágenes y textos, como ejemplo se desplegó una aplicación remota que validó la integración descrita anteriormente.

Se realizó la implementación de un escenario de prueba con distintos controladores (ESP8266, MKR 1010 Wifi y OPLA), sensores (analógicos y digitales) y actuadores en donde se verificó el funcionamiento práctico de los códigos generados y la

visualización de los datos dentro de la aplicación interactiva de la TDT enfocada al Smart Agro.

El sistema fue sometido a pruebas de carga, en las cuales se simuló la interacción de varios usuarios realizando peticiones simultaneas de todas las variables desde la TDT hacia los servidores web, en las cuales se determinó un rendimiento óptimo de 1 a 1000 usuarios (14000 peticiones) obteniendo fallos menores al 1% con tiempos menores o iguales a 330 ms; mientras que al superar los 2000 usuarios (28000 peticiones) se obtuvieron errores mayores al 15% y con más de 4000 usuarios (56000 peticiones) se obtuvieron pérdidas de más del 40%. Sin embargo, para las características del servidor en donde se albergaron los servicios web se considera un valor aceptable, debido a que los escenarios de implementación están dirigidos hacia huertos caseros cuyo monitoreo está limitado a una cantidad reducida de usuarios.

Con la utilización de los puntos de función se verificó que diseñar e implementar el sistema conllevaría un tiempo aproximado de 648 horas o un equivalente a 4.67 meses, debido al número de herramientas utilizadas, mientras que con la utilización del DSL tomaría un aproximado de 4 horas, minimizando así el esfuerzo y tiempo de los desarrolladores.

Mediante la herramienta de medición de usabilidad (SUS) los desarrolladores encuestados evaluaron al sistema con una calificación de 76 (“C”), la cual representa que están satisfechos con el DSL, sin embargo, se verificó que se podrían implementar mejoras como capacitaciones o manuales dentro del sistema con la finalidad de mejorar la comprensión del mismo.

Recomendaciones

Se recomienda que al utilizar el canal de retorno en el desarrollo de aplicaciones interactivas TDT conjuntamente con los servicios web, se realice previamente un análisis de los métodos HTTP en un analizador de tráfico como Wireshark, para establecer los encabezados, direcciones e información necesaria dentro de la configuración de la librería tcp de lua.

Se recomienda la utilización de máquinas virtuales privadas (VPS) con características óptimas para cada servicio web a implementar, realizando como primera instancia una estimación de recursos a utilizar, cantidad de tráfico y carga que se someterá al sistema para que los servicios no colapsen.

Es importante la inclusión de información de ayuda como guías desplegables dentro de las aplicaciones desarrolladas o manuales para facilitar al usuario el desarrollo de las aplicaciones interactivas.

Al momento de implementar el hardware se recomienda tener en cuenta la corriente necesaria para la activación de cada actuador tomando en cuenta las características del controlador a utilizar o a su vez el uso de módulos integrados con circuitos amplificadores de corriente.

Trabajos futuros

La inclusión de opciones de visualización de valores acumulados en las bases de datos de sensores y actuadores con información espacial y temporal diaria, semanal o mensual, esto otorga al usuario la capacidad de mejora en la toma de decisiones y un control y monitoreo más óptimo.

Implementar opciones de alertas del estado del hardware en tiempo real en la utilización de TDT para otorgar a los usuarios un mejor control y toma de acciones correctivas en el funcionamiento del sistema.

Implementar mayor cantidad de opciones TDT como la inclusión de librerías para LUA como MQTT que faciliten el uso del canal de retorno para el desarrollo de aplicaciones interactivas remotas o permitiendo un mayor grado de libertad al momento configurar los elementos multimedia para desarrollar interfaces más complejas.

Desplegar soluciones Smart Agro conjuntamente con la TDT en sentido macro que permitan aportar su inclusión para agricultores y empresas agrícolas que desplieguen gran cantidad y variedad de cultivos, otorgando un enfoque más amplio de aplicación en la implementación de herramientas bajo el dominio del IOT y la TDT.

Implementar soluciones que hagan uso de la TDT e IOT enfocadas en diversas temáticas como Seguridad, Educación, Domótica, entre otras con envío y recepción de datos en tiempo real, para este propósito se ampliaría la inclusión de mayor número y tipos de controladores, sensores, actuadores e incluso asistentes virtuales, determinando nuevas soluciones para su introducción hacia las futuras ciudades inteligentes.

Acrónimos

- ATL. Atlas Transformation Language (Lenguaje de transformación de Atlas)
- CIM. Computational Independent Models (Modelos Computacionales Independientes)
- EMF. Eclipse Modeling Framework (Marco de Modelado Eclipse)
- GME. Generic Modeling Environment (Entorno Genérico de Modelado)
- HTTP. Protocolo de Transferencia de Hipertexto.
- IDE. Entorno de Desarrollo Integrado.
- HTML. Lenguaje de Marcado de Hipertexto
- JSON. JavaScript Object Notation (Notación de Objeto de JavaScript)
- M2M. Model to Model Transformation (Transformación Modelo a Modelo)
- MBT. Model Based Testing (Pruebas Basadas en Modelos)
- MDA. Model Driven Architecture (Arquitectura Basada en Modelos)
- MDD. Model Driven Development (Desarrollo Basado en Modelos)
- MPS. Meta Programing System (Sistema de Meta Programación)
- OMG. Object Management Group (Grupo de Gestión de Objetos)
- PCB. Printed Circuit Board (Placa de Circuito Impreso)
- PDM. Platform Description Models (Plataforma de Descripción de Modelos)
- PHP. Hypertext Preprocessor (Preprocesador de Hipertexto)
- PIM. Platform Independent Model (Modelo Independiente de la Plataforma)
- PSM. Platform Specific Models (Modelos Específicos de Plataforma)
- PUC-RIO. Pontificia Universidade católica do Rio de Janeiro (Pontificia Universidad Católica de Río de Janeiro)
- QVT. Query/Views/Transformation (Consulta/Vistas/Transformación)
- SOAP. Simple Object Access Protocol (Protocolo Simple de Acceso a Objetos)

- SSID. Service Set Identifier (Identificador de Conjunto de Servicios)
- SUT. System Under Test (Sistema Bajo Prueba)
- UML. Unified Modeling Language (Lenguaje de Modelado Unificado)
- VIATRA. Visual Automated Model Transformations (Transformaciones Visuales Automatizadas de Modelos)
- VPS. Virtual Private Server (Servidor Privado Virtual)
- XHTML. Extensible Hypertext Markup Language (Lenguaje de Marcado de Hipertexto Extensible)
- XMI. XML Metadata Interchange (Intercambio de Metadatos XML)
- XML. Extensible Markup Language (Lenguaje de Marcas Extensible)

Referencias Bibliográficas

ALBRESHNE, A., FUHRER, P., & PASQUIER, J. (Septiembre de 2009). *Web Services Orchestration and.* Obtenido de https://www.unifr.ch/inf/softeng/en/assets/public/files/research/publications/pdf/WP09-03.pdf

Asami, H., & Sasaki, M. (2006). Outline of ISDB Systems. *Proceedings of the IEEE, 94*(1), 248-250. doi:10.1109/JPROC.2005.859690

Atkinson, C., & Kühne, T. (2003). Model-driven development. *IEEE Software, 20*(5), 36-41. doi:10.1109/MS.2003.1231149

AYALA, M. G. (2016). *ESPACIALIZACIÓN DE LOS COMPONENTES DEL PROYECTO DE*. Recuperado el 4 de 2 de 2021, de http://www.dspace.uce.edu.ec/bitstream/25000/8365/1/T-UCE-0004-59.pdf

Azevedo, R. G., Soares Neto, C., Meireles Teixeira, M., & Mesquita Santos, R. C. (2011). Textual authoring of interactive digital TV applications. *Proceddings of the 9th international interactive conference on Interactive television*, 1-9. doi:10.1145/2000119.2000169

Brajesh, D. (2017). API Testing Strategy. En B. De. Berkeley, CA: Apress.

Carvalho de Farias, B., de Lima Filho, E. B., & Souto, E. (2020). Extensions to Middleware Ginga for Integration with IoT Environments. *2020 IEEE International (ICCE)*, 1-5. doi:10.1109/ICCE46568.2020.9043145

Ciccozzi, F., & Spalazzese, R. (2016). MDE4IoT: Supporting the Internet of Things with Model-Driven Engineering. *Studies in Computational Intelligence, 678*. doi:https://doi.org/10.1007/978-3-319-48829-5_7

CLARIVATE. (2 de 11 de 2020). *Web of Science.* Obtenido de www.clarivate.com: https://clarivate.com/scientific-and-academic-research/research-discovery/web-of-science/

Crinon, R. J., Bhat, D., Catapano, D., Thomas, G., Van Loo, J. T., & Bang, G. (2006). Data Broadcasting and Interactive Television. *Proceedings of the IEEE, 94*(1), 102-118. doi:10.1109/JPROC.2005.861020

CWTS. (2022). *Highlights*. Obtenido de www.vosviewer.com: https://www.vosviewer.com/features/highlights

CWTS. (2022). *Welcome to VOSviewer*. Obtenido de www.vosviewer.com: www.vosviewer.com

Czarnecki, K., & Helsen, S. (2003). Classification of Model Transformation Approaches. *OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture*, 1-17.

de Lara, J., Guerra, E., & Sánchez Cuadrado, J. (2015). Model-Driven Engineering with Domain-Specific Meta-Modelling Languages. *Software and Systems Modeling, 14*, 429-459. doi:10.1007/s10270-013-0367-z

Donthu, N., Kumar , S., Mukherjee, D., Pandey, N., & Marc Lim, W. (2021). How to conduct a bibliometric analysis: An overview and guidelines. *Journal of Business Research, 133*, 285-296.

Durán Muñoz, F., Troya Castilla, J., & Vallecillo Moreno, A. (2015). *Desarrollo de software dirigido por modelos.* Barcelona: Universitat Oberta de Catalunya.

ECLIPSE. (2009). *Eclipse Modeling Framework (EMF).* Obtenido de https://www.eclipse.org/modeling/emf/

ECLIPSE. (2010). *Eclipse Modeling Project.* Obtenido de https://www.eclipse.org/modeling/

EFEAGRO. (21 de 12 de 2020). *efeagro*. Recuperado el 10 de 1 de 2021, de https://www.efeagro.com/microsite/smart-agro-tecnologia-agric

Farias, M., Carvalho, M. M., & Alencar, M. S. (2008). Digital Television Broadcasting in Brazil. *IEEE MultiMedia, 15*(2), 64-70. doi:10.1109/MMUL.2008.25

García Molina, J., García Rubio, F. O., Pelechano, V., Vallecillo, A., Vara, J., & Vicente-Chicote, C. (2012). *Desarrollo de Software Dirigido por Modelos.* Madrid: RA-MA Editorial.

García, F. (3 de Agosto de 2015). *Arsys Blog*. Obtenido de https://www.arsys.es/blog/programacion/web-services-desarrollo

Gomes Soares, L. F., Ferreira Rodrigues, R., & Ferreira Moreno, M. (2007). Ginga-NCL: the declarative environment of the Brazilian digital TV system. *Journal of the Brazilian Computer Society, 12*(4), 37-46. doi:10.1590/S0104-65002007000100005

González García, C., Pascual Espada, J., Núñez-Valdez, E. R., & García-Díaz, V. (2014). Midgar: Domain-Specific Language to generate Smart Objects for an Internet of Things platform. *Eight International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing* (págs. 352-357). Oviedo: University of Oviedo. doi:10.1109/IMIS.2014.48

Greenfield, J., & Short, K. (2003). *Software Factories: Assembling Applications with Patterns, Frameworks, Models and Tools*, (págs. 1-13). doi:DOI:10.1145/949344.949348

Guido, B., Simone,Murzilli, & Cudini, S. (2016). Definition of REST web services with JSON schema. *Wiley Online Library*, 907-920.

INFOAGRO. (10 de Diciembre de 2013). *El secreto de la agricultura holandesa.* Recuperado el 05 de Febrero de 2021, de https://www.infoagro.com/noticias/2013/el_secreto_de_la_agricultura_holandesa.asp

ITU-T. (1996). A Guide to Digital Terrestrial Television Broadcasting in the VHF/UMF Bands.

Keyur K Patel, S. M. (2016). Internet of Things-IOT: Definition, Characteristics, Architecture, Enabling Technologies, Application & Future Challenges. *IJESC*.

León Ruiz, J. H. (26 de Agosto de 2011). *Contributions to the design and development process of interactive applications for digital TV based on Ginga-NCL.* Obtenido de www.researchgate.net: https://www.researchgate.net/publication/317720791_Contributions_to_the_design_and_development_process_of_interactive_applications_for_digital_TV_based_on_Ginga-NCL

MAGAP. (25 de Abril de 2019). *MAG crea la comunidad de AgroTubers para apoyar la difusión de innovaciones en el sector agropecuario.* Obtenido de www.agricultura.gob.ec: https://www.agricultura.gob.ec/mag-crea-la-comunidad-de-agrotubers-para-apoyar-la-difusion-de-innovaciones-en-el-sector-agropecuario/

Martinez, M., Lucano, S., & Pazmiño, A. (2017). The challenge of implementing interactive content in Digital Terrestrial Ecuadorian Televison. *Prisma Social: Ciudadanía Digital y Open Data Access, 18*, 572-578.

Milica Lekić, G. G. (2018). IoT sensor integration to Node-RED platform. *IEEE*.

MINTEL. (2018). *Libro Blanco de la Sociedad de la Información y del Conocimiento.* Quito: MINTEL.

Naciones Unidas. (2018). *La Agenda 2030 y los Objetivos de Desarrollo Sostenible: Una oportunidad para América Latina y el Caribe.* Santiago: Naciones Unidas.

Naghmeh Niknejad, W. I. (2020). Understanding Service-Oriented Architecture (SOA): A systematic. *ELSEVIER*.

OMG. (2001). *MDA (Model Driven Architecture).* Obtenido de Guide Version 1.0.1: www.omg.org/mda/

ONU. (16 de Mayo de 2018). *Las ciudades seguirán creciendo, sobre todo en los países en desarrollo.* Obtenido de Noticias ONU: https://www.un.org/development/desa/es/news/population/2018-world-urbanization-prospects.html

OpenJS Foundation. (25 de Agosto de 2020). *User Guide.* Obtenido de www.nodered.org: https://nodered.org/docs/user-guide/

Paucar Curasma, R., Velásquez Díaz, C., Mendoza Villaizán, E., Diaz Ataucuri, D., & Villanueva, J. M. (2009). Análisis del Canal de Retorno para la Televisión Digital Interactiva utilizando la Clase TCP-Lua. *Revista Ciência e Tecnologia, 12*(20/21), 49-54.

PMOinformatica. (6 de 04 de 2015). Obtenido de http://www.pmoinformatica.com/2015/04/estimacion-puntos-funcion-

introduccion.html: http://www.pmoinformatica.com/2015/04/estimacion-puntos-funcion-introduccion.html

Pongnapat Jutadhamakorn, T. P. (2017). A Scalable and Low-Cost MQTT Broker Clustering. *IEEE*.

Quinteros, & Troya. (2018). *Modelos en Ingeniería del Software: DSLs, MDE y simulación.* Obtenido de https://masteroficial.us.es/mis/asignaturas/complementos-modelos.html

Rodrigues da Silva, A. (2015). Model-driven engineering: A survey supported by the unified conceptual model. *Computer Languages, Systems and Structures, 43*, 139-155.

Rodrigues de Carvalho, E., de Paula Costa, L. }., Garcia de Barros, G., Alves Faria, R. R., Pernas Nunes, R., de Deus Lopes, R., & Knörich Zuffo, M. (2007). The Brazilian digital television system access device architecture. *Journal of the Brazilian Computer Society, 12*(4). doi:10.1590/S0104-65002007000100009

Rodríguez, F. S. (1999). Los Puntos de Funcionalidad (Function Points). En F. S. Rodríguez, *Planificación y Gestión de Sistemas de Información* (págs. 9-34). España: Universidad de Castilla-La Mancha.

Ruiz Rube, I. (2012). *Ingeniería Dirigida por Modelos y Calidad de Software.* Obtenido de https://rodin.uca.es/bitstream/handle/10498/15861/JORPRESI%20-%20Ruiz%20Rube%20et%20al%202010.pdf?sequence=1&isAllowed=y

Sampaio de Alencar, M. (2009). *Digital Television Systems.* New York: Cambridge University Press.

Selic, B. (2003). The Pragmatics of Model-Driven Development. *IEEE Software, 20*(5), 19-25. doi:10.1109/MS.2003.1231146

Soto, A. (12 de 11 de 2018). *OpenWebinars*. Obtenido de https://openwebinars.net/blog/diferencia-entre-arquitectura-monolitica-y-microservicios/

Stahl, T., & Völter, M. (2006). *Model-Driven Software Development.* Chichester: John Wiley & Sons, Ltd.

Sun, L., Y. L., & Memon, R. h. (2017). An Open IoT Framework Based on Microservices. *IEEE*, 154-162.

Takada, M., & Saito, M. (2006). Transmission System for ISDB-T. *Proceedings of the IEEE, 94*(1), 251-256. doi:10.1109/JPROC.2005.859692.

TELEFONICA. (25 de Junio de 2021). *Smart Agro y Vertical Green, las soluciones de Telefónica para reforzar la sostenibilidad del planeta.* Obtenido de www.telefonica.com: https://www.telefonica.com/es/sala-comunicacion/smart-agro-y-vertical-green-las-soluciones-de-telefonica-para-reforzar-la-sostenibilidad-del-planeta/

Thomson Reuters. (2008). *Capabilites.* Obtenido de ISI Web of Knowledge: https://web.archive.org/web/20101123014042/https://www.thomsonreuters.com/content/science/pdf/Web_of_Knowledge_factsheet.pdf

Viceministerio de Tecnologías de la Información y Comunicación . (19 de Septiembre de 2018). *Plan Maestro de Transición a la Televisión Digital Terrestre.* Obtenido de www.telecomunicaciones.gob.ec: https://www.telecomunicaciones.gob.ec/wp-content/uploads/2018/10/PLAN-MAESTRO-DE-TRANSICI%C3%93N-A-LA-TELEVISI%C3%93N-DIGITAL-TERRESTRE-2018-2021.pdf

Voelter, M., Benz, S., Dietrich, C., Engelmann , B., Helander, M., Kats, L., . . . Wachsmuth , G. (2013). *DSL engineering: Designing, implementing and using domain-specific languages.* dslbook.org.

Anexos