Visual WebShawn: Herramienta visual para el análisis de simulaciones de redes de sensores inalámbricos en la web

Visual WebShawn: Visual tool for the analysis of wireless sensor networks simulations on the web

Diego Alberto Godoy, Santiago H. Bareiro, Fabián E. Favret[1]

Ingeniería en Informática/ artículo científico

Citar: Godoy, D. A., Bareiro, S.H., y Favret, F. E. (2020). Visual WebShawn: Herramienta visual para el análisis de simulaciones de redes de sensores inalámbricos en la web. *Cuadernos de Ingeniería* (13). Recuperado de: http://revistas.ucasal.edu.ar

Recibido: mayo/2021
Aceptado: octubre/2021

Resumen

En este artículo se presentan los avances realizados con relación al diseño de una herramienta de visualización gráfica para el análisis de resultados de simulación de WSN denominada Visual WebShawn. Esta permite simular redes de sensores inalámbricos desde una interfaz *web*. Al utilizarla como motor de simulación, Shawn puede simular miles de sensores en un tiempo mínimo comparado con otros simuladores. Para su modelado e implementación se utilizó la metodología WebML, la herramienta WebRatio y contenedores Docker, respectivamente. Finalmente se presentan casos de prueba para verificar el funcionamiento de la herramienta.

Palabras clave: simulación, redes de sensores inalámbricos - WebML

Abstract

This article presents the progress made in the design of a graphical visualization tool for the analysis of WSN simulation results called Visual WebShawn. It allows simulating wireless sensor networks from a web interface. By using it as a simulation engine, Shawn can simulate thousands of sensors in minimal time compared to other simulators. For its modeling and implementation, the WebML methodology, the WebRatio tool and Docker containers were used, respectively. Finally, test cases are presented to verify the performance of the tool.

Keywords: simulation, wireless sensors networks - WebML

1 Universidad Gastón Dachary, Argentina.

1. Introducción

En la actualidad existen diferentes *softwares* de simulación que brindan interfaces visuales con herramientas que permiten establecer parámetros y variables de entorno, como así también gestionar la configuración de escenarios y realizar combinaciones de todos estos elementos con el objeto de producir diferentes resultados. Entre ellos se pueden nombrar a NS-2 (Information Sciences Institute, 2017), TOSSIM (Group, 2018), AVRORA (University of California, s.f), TRM-Sim-WSN (Mármol y Pérez, 2009) y Shawn (Fischer et al., 2007). Pero estas aplicaciones suelen ser para una plataforma específica, no incluyen una interfaz basada en la Web o bien consumen un alto nivel de recursos para realizar el procesamiento por realizar una simulación más detallada de capas inferiores, tales como la física y la capa de enlace de datos, mientras que Shawn simplemente envía los mensajes a través de un modelo simplificado y más abstracto.

Respecto a este punto de vista, las aplicaciones *web* que emplean una arquitectura cliente-servidor cuentan con una gran ventaja, ya que aligeran la carga computacional de los equipos de los usuarios finales; principalmente, al realizar procesamiento de datos destinados a la generación de resultados gráficos en la simulación.

Otra característica importante de las aplicaciones *web* es la usabilidad que presentan. Esta permite simplificar, a través de las interfaces gráficas, tareas como la carga de datos de configuración en la simulación. Inicialmente el proyecto WebShawn (Godoy et al., 2014) presentó un cambio radical en la forma de utilizar el simulador Shawn para WSN, ya que los usuarios pueden interactuar directamente con el simulador a través de la *web*.

Sin embargo, en el prototipo que se realizó no existe una forma de ingresar todos los parámetros de configuración de visualizaciones que ofrece Shawn. Estos se deben escribir directamente sobre un archivo de configuración (disponible en la interfaz *web*). Al tratarse de múltiples opciones con distintas incidencias en las salidas gráficas, el usuario debe conocer el orden de ingreso y todos los posibles valores válidos que se aceptan por cada parámetro, lo que genera dificultades al usuario que trata de ingresar este tipo de datos.

Otro aspecto que no fue desarrollado en el prototipo inicial de WebShawn es la gestión de escenarios de simulación; es decir, permitir a los usuarios exportar o importar las configuraciones de estos. En este sentido, el simulador Shawn posee también distintos parámetros que se pueden utilizar para permitir, en este caso, una mejora de WebShawn aprovechando la característica disponible de gestión de escenarios.

Es por ello que en este trabajo se pretende extender las capacidades faltantes de WebShawn y mejorar las que se pueden realizar desde Shawn, aprovechando el proceso para integrar con otras librerías de generación de gráficos. Todo esto tendiente mejorar, específicamente, tanto la interacción como la generación de gráficos de resultados que se puedan obtener desde el navegador. Se planteó como objetivo diseñar una herramienta de visualización gráfica que permita analizar los resultados de simulaciones de WSN en WebShawn.

1.1. Trabajos relacionados

En esta sección se presentan tres trabajos relacionados con el presente trabajo.

1.1.1. WebShawn, simulating wireless sensors networks from the web

Este trabajo presenta la adaptación del simulador Shawn por medio de una interfaz basada en *web* (Godoy et al., 2017) y su alojamiento en servidor con sistema operativo GNU/Linux, facilitando así su acceso y su utilización dado su gran potencial de simulación, y aislando a los usuarios finales de la complejidad de instalación y configuración que se requiere. Con dicha implementación, el simulador WebShawn se presenta como una alternativa viable para que cualquier tipo de usuario pueda crear y ejecutar simulaciones de WSN de una manera sencilla.

La salida que generan dichas simulaciones sigue siendo la salida estándar de consola del simulador Shawn, pero dispuesta sobre archivos de texto plano y, además, la salida de un archivo visual con la disposición e interconexiones entre los nodos de la red carece de mayor nivel de detalle. Tampoco es posible identificar de forma correcta los nodos del sistema, ya que no existen etiquetas diferenciadas para cada tipo de nodo, principalmente por la falta de un módulo de configuración de parámetros de visualización. Además, no es posible gestionar la configuración del escenario de simulación ni exportar o importar escenarios para reutilizarlos en diferentes proyectos de simulación.

1.1.2 A Web-based integrated environment for simulation and analysis with NS-2

En este trabajo se desarrolla un completo entorno basado en la *web* para la simulación y posproceso de las salidas generadas por Network Simulation version 2 (NS-2) un popular simulador de redes (Saha et al., 2013; Information Sciences Institute, 2017). La aplicación ns2web permite la ejecución remota de simulaciones de WSN, incluyendo redes cableadas. También ofrece un conjunto de herramientas para analizar los archivos de rastreo que el simulador genera como salida. Una implementación del simulador se encuentra *on line* para público acceso en su sitio *web* (Saha, s.f.).

Ns2web está compuesto por dos módulos, uno de ellos —ns2sim— se encarga de visualizar el *frontend* de la aplicación y la ejecución remota en el servidor de la simulación con NS-2. El otro módulo —ns2trace— gestiona todas las tareas relacionadas al análisis de los archivos de rastreo possimulación.

La interacción por parte del usuario es directa con el navegador en todo momento, tanto para enviar los *scripts* de simulación como para la visualización de las salidas que genere la plataforma, así como también el guardado de los archivos de rastreo generados por ella.

El tipo de salidas que genera esta aplicación están orientadas a bajo nivel, ofrecen una gran cantidad de detalles, pero sin ningún tipo de ayuda en su visualización. Al igual que en el trabajo donde se desarrolló WebShawn, no existe algún componente que permita la gestión de los resultados de visualización que generen las simulaciones.

1.1.3. VisualSense: visual editor and simulator for wireless sensor network systems

Si bien VisualSense (VisualSense, 2020) no se constituye en una interfaz *web*, en sí es una interfaz gráfica de usuario creada para mejorar la funcionalidad del *framework* Ptolemy II (Ptolomeo II), el cual es un *framework* de simulación que provee interacciones con un diseño orientado a los actores/objetos de un escenario de simulación (Ptolemy II, 2020). Los actores son componentes de *software* que se ejecutan simultáneamente y que se comunican por medio de mensajes enviados a través de puertos interconectados. Un modelo es una interconexión jerárquica de actores. En Ptolemy II, la semántica de un modelo no está determinada por el marco, sino más bien por un componente de *software* en el modelo llamado director, que implementa un modelo de computación.

El modelado de WSN requiere un sofisticado modelado de canales de comunicación, canales de sensores, protocolos de redes *ad hoc*, estrategias de localización, protocolos de control de acceso de medios, consumo de energía en nodos de sensores, etc. Este *framework* de modelado está diseñado para soportar una construcción de tales modelos. Su objetivo es permitir a la comunidad de investigación compartir modelos de aspectos disjuntos del problema del modelado de las redes de sensores y construir modelos que incluyan elementos sofisticados de varios aspectos. Esto se considera como una gran ventaja, ya que no solamente se incluye la posibilidad de crear modelos personalizados en conjunto con la utilización de modelos preexistentes, sino que además se brinda la posibilidad de compartir dichos modelos, poder personalizarlos y configurar parámetros sobre los aspectos de visualización.

2. Metodología

A continuación, se hace una breve descripción del funcionamiento general del simulador basado en la Web, WebShawn, con el cual se pretende luego cumplir el objetivo de diseñar una herramienta de visualización gráfica para el análisis de resultados de simulación de WSN de forma gráfica.

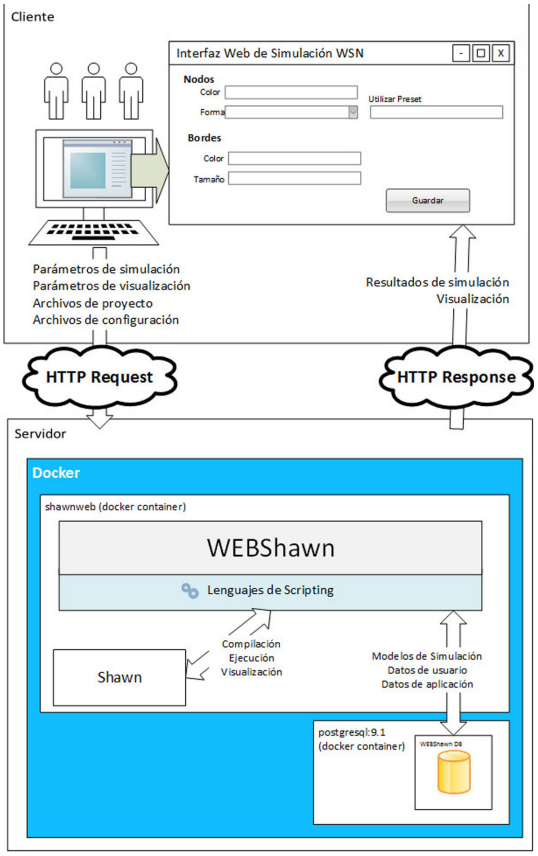

Figura 1. Esquema general del simulador basado en la Web.

En la Figura 1 se puede visualizar un esquema del funcionamiento general del simulador. A través de una interfaz de usuario brindada por el navegador *web* los usuarios podrán interactuar con el simulador e ingresar todas las entradas necesarias para realizar las corridas de simulación y configurar el aspecto de resultados visuales de esta por medio de sus correspondientes parámetros donde, además, contarán con una suite de *presets* personales que podrán reutilizar. También podrán utilizar las funcionalidades de importación y exportación de escenarios de simulación en sus proyectos. Las entradas son los parámetros de escenario de simulación y de visualización de resultados de simulación, archivos del proyecto y archivos de configuración. Como parámetros de entrada se mantiene la funcionalidad estándar de, por ejemplo, enviar la cantidad de nodos a simular, la semilla de la simulación, el modelo de comunicación, etc., y serán adicionadas las entradas de aspecto visual de nodos y bordes

de comunicaciones, además de la posibilidad de guardar y reutilizar *presets* de visualización, marcar escenarios para ser exportados y posteriormente reutilizados mediante importación en el mismo u otros proyectos de simulación, mientras que los archivos del proyecto y de configuración (los cuales son alojados dentro del servidor) darán el comportamiento específico a los nodos y al ambiente.

Como medio de comunicación entre el cliente y el servidor se podrá utilizar tanto una red local, o bien Internet. Además, en el mismo navegador, se podrá obtener, una vez finalizada la corrida de la simulación, los resultados obtenidos en forma de texto y visualizaciones generadas como salidas gráficas. Todas las dependencias de instalación y configuración del simulador Shawn están empaquetadas en una imagen de Docker (Docker Inc., 2020), la cual además contendrá el servidor *web* y las dependencias necesarias para que este pueda aceptar, interpretar y procesar todas las solicitudes del cliente. Asimismo, el servicio de bases de datos PostgreSQL utilizado por WebShawn será trasladado a otro contenedor Docker, mediante su correspondiente imagen, para aumentar la modularidad del entorno de simulación permitiendo una gestión sencilla de este facilitando, por ejemplo, tareas de respaldo y recuperación, como también posibilitando el escalado horizontal de recursos de servidor en caso de ser necesario; este mismo principio aplica para el contenedor que ejecute la imagen con las dependencias de WebShawn.

Para llevar a cabo el desarrollo del prototipo se seguirá la metodología WebML, la cual es una notación o lenguaje de alto nivel para el modelado, diseño e implementación de aplicaciones *web* que hacen uso intensivo de datos (Ceri et al., 2000).

La propuesta que realiza dicha metodología es combinar leguajes de modelado tradicionales o bien conocidos por los desarrolladores, como el diseño conceptual de datos, utilizando el modelo entidadrelación y la especificación de casos de uso con UML (*Unified Modeling Language*), incluyendo nuevos conceptos y métodos para el diseño de hipertexto, que son fundamentales para el desarrollo de aplicaciones *web* (Moreno et al., 2006).

El ciclo de vida del proyecto completo consiste en varias iteraciones, cada una produce un prototipo o una versión parcial de la aplicación. En cada iteración la versión actual de la aplicación se somete a un proceso de testeo y evaluación. Luego, si es necesario, se modifica o amplía para hacer frente a los requerimientos recolectados con anterioridad, así también como a los nuevos requisitos que puedan surgir (Brambilla et al., 2008). Este tipo de metodología con ciclos iterativos se adapta especialmente a los desarrollos *web*, dada la dinámica de cambios que pueden suceder durante el desarrollo y la rapidez con que se deben desplegar los sitios en Internet.

Además, es posible apoyarse en herramientas CASE (*Computer Aided Software Engineering*) para el modelado con WebML incluidas en WebRatio (WebRatio, 2020).

3. Construcción del simulador

3.1. Especificación de requerimientos

La fase de especificación de requerimientos se centra en la recopilación de información acerca del dominio de la aplicación y en la especificación de las funciones por medio de descripciones fáciles de entender. Esta fase se divide en la recolección de requerimientos y el análisis de requerimientos.

3.1.1. Identificación de grupos de usuarios

Inicialmente se determinó un único grupo de usuarios denominado «desarrollador de aplicaciones de simulación», dejándose abierta la posibilidad de añadir nuevos grupos que fueran necesarios.

3.1.2 Especificación de requerimientos funcionales

En esta sección se describirán a través de casos de uso los nuevos requerimientos funcionales del sistema, los cuales complementan a los requerimientos funcionales iniciales de este. En la especificación de casos de uso se detallan todas las acciones que el sistema realizará desde el punto de vista del usuario.

Tabla 1. Caso de uso: especificar parámetros de control para almacenarlos en archivo de configuración

Nombre	Especificar parámetros de control de simulación para almacenarlos en archivos de configuración.
Propósito	Que un usuario pueda especificar los parámetros de control de la simulación.
Precondición	El usuario debe tener una sesión activa en la aplicación.
Poscondición	La aplicación almacena temporalmente los parámetros de configuración y muestra la vista para configurar los parámetros de visualización.
Flujo de trabajo	1. El usuario selecciona un archivo de configuración de una lista desplegable que contiene todos los archivos del proyecto. 2. Ingresa los parámetros de control de simulación en el formulario de configuración de parámetros. 3. El usuario presiona el botón siguiente. 4. El sistema muestra la vista de configuración de parámetros de visualización.

3.1.3. Diccionario de datos

Como se puede visualizar en el diagrama de caso de uso de la sección anterior, los principales objetos de información que son gestionados por la aplicación WebShawn son: proyectos de simulación y usuarios. Un usuario puede tener múltiples proyectos, *presets* de visualización, y pertenecer a diferentes grupos. A su vez, un proyecto puede tener múltiples salidas de simulación y archivos de configuración asociados.

3.1.4. Especificación de vistas del sitio

A continuación, se describe un mapa y el contenido de la vista *home page* del sitio. La vista *home page* tiene las áreas listadas en la Tabla 2.

Tabla 2. *Áreas de la vista home page*

Nombre del área	Descripción	Objetos gestionados/ consultados
Control de simulación	Dentro del área de control de simulación el usuario podrá realizar las funciones descriptas en los casos de uso: «Especificar parámetros de control para almacenarlos en archivo de configuración», «Importar escenario de visualización», «Guardar parámetros de configuración y visualización en archivos de configuración», «Obtener y mostrar parámetros de visualización de archivos de configuración», «Eliminar parámetros de visualización de archivos de configuración», «Guardar *presets* de visualización de usuario», «Utilizar *presets* de visualización de usuario», «Eliminar *presets* de visualización de usuario»,	ControlSimulacion Preset ProyectoConfig ProyectoMundo ProyectoPreset Snapshots

3.2. Diseño de datos

La entrada del diseño de datos son todos los requerimientos identificados durante la fase de análisis de requerimientos. Más específicamente, lo que se logra en la fase es la trasformación del diccionario de datos en un diagrama de entidadrelación.

Esta fase implementa una de las disciplinas más tradicionales y consolidadas en la tecnología de la información, por lo cual WebML no propone otro lenguaje de modelado de datos, sino que utiliza el modelo de datos entidadrelación o el equivalente subconjunto de primitivas en UML para el diagrama de clases.

3.2.1. Diagrama de entidad-relación

En la Figura 2 se puede visualizar el diagrama de entidadrelación resultante de las especificaciones obtenidas del diccionario de datos. Está compuesto por ocho entidades y nueve relaciones. Las entidades son: Usuario, Proyecto, Grupo, Preset, Config, ProyectoWorld (que representa el mundo virtual o ambiente en el cual se alojan todos los objetos de simulación como son los nodos y los procesadores. Además, se especifican valores para tamaño del mundo, rango de comunicación, valor de semilla, número de iteraciones y modelos de borde, comunicación y trasmisión para los nodos), ProyectoPreset (representa la configuración de colores que va ser utilizada en la configuración del proyecto actual) y Snapshot (donde se guardan las distintas salidas de corridas para

un proyecto dada la configuración del mundo). La relación entre Usuario y Proyecto se denomina Desarrolla. La relación entre Usuario y Grupo se denomina Pertenece. La relación entre Usuario y Preset se denomina Establece. La relación entre Usuario y Snapshot se denomina Exporta. La relación entre Usuario y Config se denomina Establece. La relación entre Snapshot y Proyecto se denomina Pertenece. Las relaciones entre Proyecto y Config, Config y ProyectoWorld, Config y ProyectoPreset se denominan Contiene. Una forma de interpretar estas relaciones desde el punto de vista de la entidad Usuario es de la siguiente manera: Un Usuario Desarrolla Proyectos y un Usuario Pertenece a uno o más Grupos, la misma regla de interpretación es aplicable a las demás relaciones entre entidades.

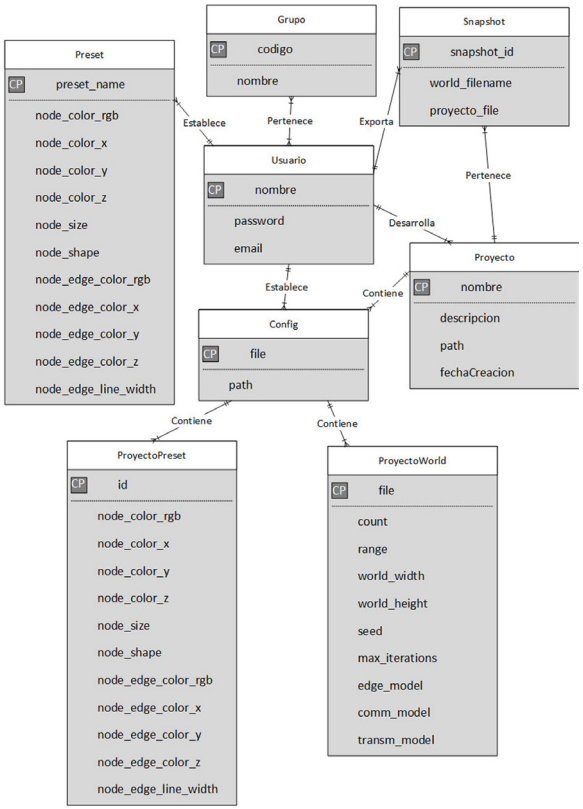

Figura 2. Diagrama de entidad-relación.

3.3 Diseño de hipertexto

El diseño de hipertexto especifica las visitas del sitio basado en el esquema de datos definido en el diseño de los datos con el objetivo de realizar la publicación de contenido y manipulación de servicios identificados durante el análisis de requerimientos.

Se inicia a partir de tres fuentes esenciales de entrada; el esquema conceptual de datos, que representa la estructura de los datos; los requisitos funcionales, que indican la funcionalidad a entregar por parte del sistema; y las vistas del sitio, que trazan la organización de los hipertextos que se ofrecen al usuario.

Luego estas entradas se convierten en una especificación WebML, que ofrece una visión de alto nivel de las interfaces de aplicación, independientemente de cualquier detalle de implementación, pero lo suficientemente precisos para ser utilizados como una guía u hoja de ruta en la fase de implementación.

3.3.1. Especificación WebML de páginas

Las especificaciones WebML del área de control de simulación (representado por la Figura 3) en la cual se muestra la página parámetros de control de simulación con acceso por defecto (*default*) dentro del área y la página de parámetros de visualización.

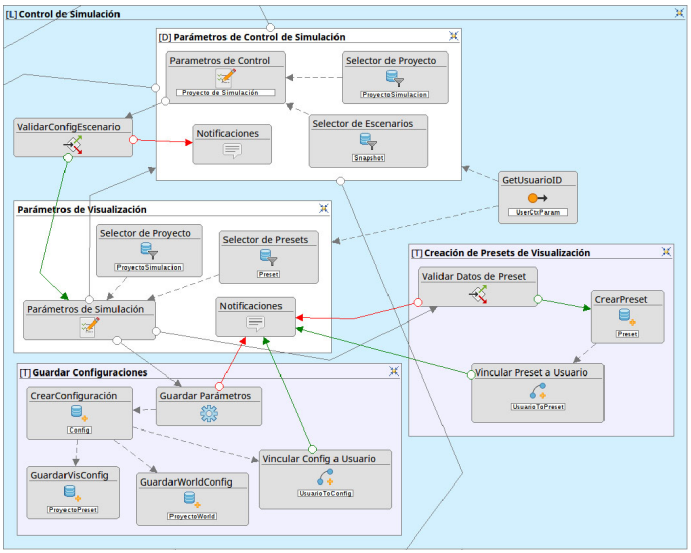

Figura 3. Especificación WebML de páginas del área control de simulación.

Como información global, se recupera el ID del usuario, que se representa en este caso por el componente *Get* GetUsuarioID. Luego de seleccionar un determinado proyecto, con el formulario que está representado por el componente *Form* Parámetros de Control se ingresan los diferentes parámetros que controlan la simulación y el escenario a generar o importar, la selección de un escenario a importar se realiza mediante un componente selector que obtiene todos los *Snapshots* generados por el Usuario para el proyecto indicado. Una vez ingresados los parámetros de control del escenario, estos se validan y si dicha validación falla se muestra un mensaje de notificación en el componente destinado a tal finalidad. Si la validación pasa con éxito se accede a la nueva página de parámetros de visualización, en donde el usuario podrá ingresar las configuraciones de visualización deseadas para el proyecto que haya indicado. Asimismo, podrá cargar dichos parámetros desde *presets* previamente almacenados, los cuales son indicados mediante el componente selector destinado a tal fin. Asimismo, el usuario podrá guardar las configuraciones de visualización en nuevos *presets*, para ello, el formulario de parámetros de visualización es validado previo a su envío, si esta validación falla se notifica al usuario mediante el componente de notificaciones. En cambio, si la validación se supera con éxito, se crea el *preset* y se lo vincula al usuario.

Una vez ingresados los parámetros en el formulario de parámetros de control y de parámetros de visualización, estos se guardan en el archivo de configuración correspondiente. Esto último queda representado por el componente *Operation* Guardar Parámetros. Si al guardar los parámetros de configuración en el archivo surge algún error se le notificará al usuario o, de lo contrario, si se guardan correctamente, se procede a almacenar la configuración del proyecto, de escenario y las configuraciones de visualización y finalmente se vincula dicha configuración al Usuario y se informa en la página de parámetros de control mediante el componente *Message* Notificaciones.

3.4. Implementación

Para realizar la implementación del prototipo se siguió la metodología WebML. Luego de finalizar esta sección, el prototipo estará listo para desplegarse en las diferentes tecnologías que aquí se van a especificar. En la implementación de datos se decidió continuar con el motor base de datos relacional PostgreSQL y el ORM RedBeanPHP, el cual fue actualizado a su última versión publicada. En la implementación de hipertexto se utilizaron los lenguajes HTML5, CSS3, JavaScript, la librería JQuery, el *framework* Bootstrap 4 y PHP en su versión 7.0.

3.4.1. Implementación de datos

Como primer paso para la implementación de datos se utilizó la herramienta para modelado de datos de WebRatio. El esquema de datos generado se puede utilizar luego para vincular con los elementos de la implementación de hipertexto utilizando la misma herramienta. Se consideraron las entidades Usuario, Grupo, Proyecto, *Preset, Snapshot, Config*, ProyectoPreset y ProyectoWorld.

La herramienta permite automatizar el pasaje del diagrama de entidadrelación a una implementación directa en código SQL y la posterior generación del esquema en una base de datos relacional.

3.4.2. Implementación de hipertexto

Para realizar esta fase se siguieron las primitivas de implementación de hipertexto y los lineamientos definidos por la metodología WebML. El lenguaje en que se implementaron los contenidos estáticos es HTML5. Para organizar y dar estilo uniforme a los elementos estáticos y dinámicos se utilizó CSS3 en conjunto con el *framework* Bootstrap 4. Para desplegar información dinámica y darle comportamiento específico a un componente dentro de las páginas se utilizó JavaScript y la librería JQuery, en tanto que para el despliegue de mensajes de notificaciones se ha empleado la biblioteca *SweetAlert* de JavaScript. Del lado del servidor para realizar la interacción de la información proveniente del hipertexto, se utilizó PHP 7.0. Además, se han empaquetado todas las dependencias y configuraciones tanto del servidor *web* Apache2 como de PHP y todas las dependencias y configuraciones de los simuladores Shawn y WebShawn utilizando Docker y gestionando el funcionamiento y la comunicación los contenedores de esta tecnología y la base de datos PostgreSQL (también empleada en un contenedor de Docker) mediante DockerCompose,

4. Pruebas

En esta sección se plantean dos escenarios para verificar que el prototipo es capaz de generar salidas gráficas de proyectos de simulación que reflejen el estado final indicado por la salida de texto brindada por Shawn. Seguidamente se muestra cómo gestionar *presets* de visualizaciones de usuario de manera tal de que se le permita reutilizar configuraciones entre sus proyectos de simulación.

4.1 Parametrización de visualizaciones de nodos y conexiones

Para el primer escenario se empleó como supuesto un usuario con un proyecto sencillo de simulación en el cual este define todos los parámetros concernientes a la configuración del escenario y a la configuración de simulación, es decir, la especificación de la cantidad de nodos, tipo y modelo de comunicación, rango de alcance, etc. Y, adicionalmente, el usuario establece el aspecto de un tipo de nodo en particular que intenta resaltar. Los parámetros empleados para la configuración del mundo y de la simulación se pueden observar de forma gráfica en la Figura 4. En esta figura se puede apreciar cómo fueron ingresados los parámetros desde la Web en la página control de simulación.

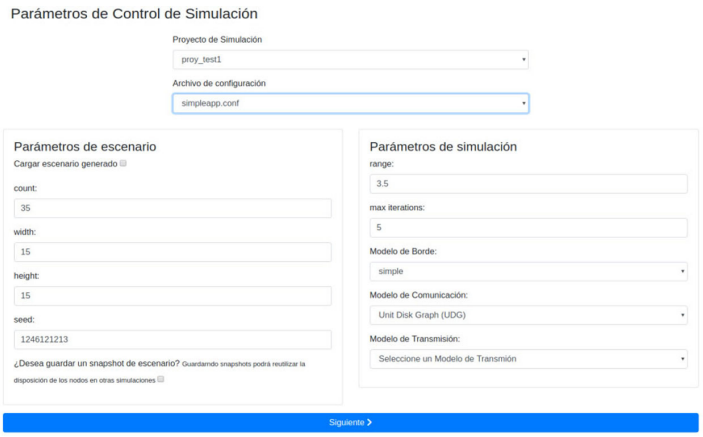

Figura 4. Parámetros ingresados desde el navegador para el primer escenario de simulación

Los parámetros empleados para la parametrización de resultados de visualización del proyecto pueden visualizarse en la Figura 5. En esta figura se puede apreciar cómo fueron ingresados los parámetros desde la Web en la página parámetros de visualización.

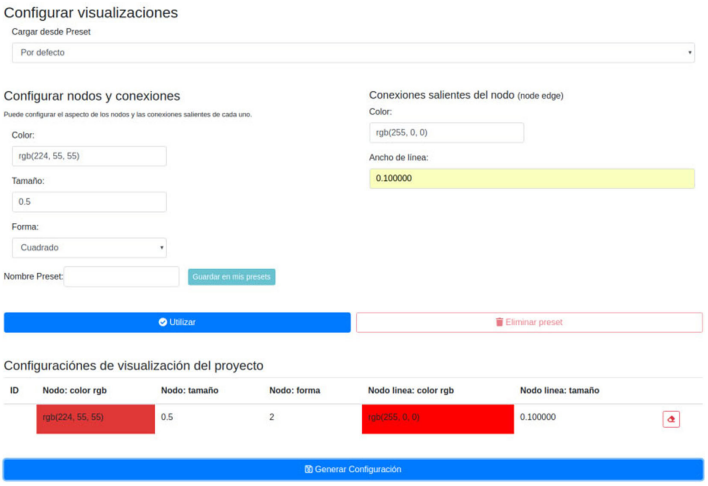

Figura 5. Parámetros de visualización ingresados desde el navegador para el primer escenario de simulación

Luego de que se ingresaron estos parámetros y se realizó la compilación y ejecución del proyecto desde el navegador *web*, las visualizaciones obtenidas se pueden observar en la Figura 6.

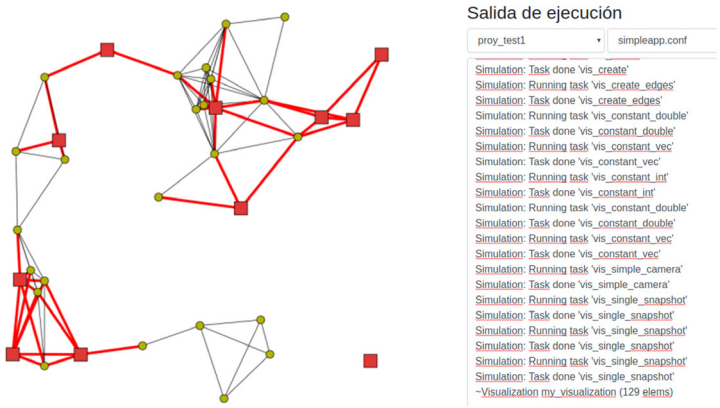

Figura 6. Visualización de resultados gráficos con nodos resaltados y captura parcial de salida en formato texto generada.

A simple vista puede verificarse que no todos los nodos de la red están interconectados dada la configuración del escenario y el alcance indicados por el usuario, al margen de esta situación han sido resaltados todos los nodos indicados por el usuario, así como además las comunicaciones de dichos nodos de acuerdo a los parámetros de visualización indicados por él.

Asimismo, mediante la salida de texto generada por el prototipo WebShawn puede verificarse la correcta ejecución de cada comando empleado por Shawn para indicarle a la biblioteca VIS qué y cómo debe graficar en la salida del proyecto de simulación. Para conocer en detalle la utilización de VIS en Shawn y los comandos que se utilizan se puede consultar Pfisterer et al. (2013).

4.2. Almacenamiento y reutilización de *presets* de visualización

Para el segundo escenario se usó como supuesto un usuario que, realizando un proyecto de simulación A desea, además, reutilizar los parámetros de configuración de visualizaciones en un proyecto B con posibles modificaciones mínimas inclusive respecto de la visualización de algunos dispositivos de la red de sensores inalámbricos. Para ello, primero debe completar los campos del formulario de la página parámetros de visualización. Esto es ejemplificado en la Figura 7.

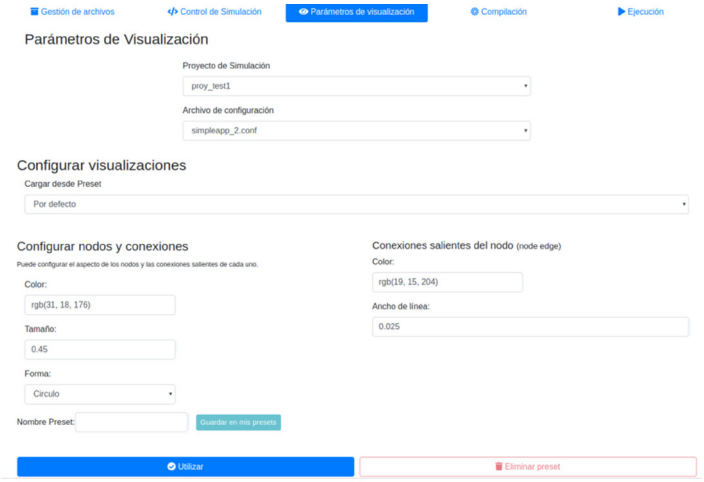

Figura 7. Parámetros de visualización ingresados desde el navegador para el segundo escenario de simulación.

Posteriormente debe indicar un nombre representativo para poder almacenar la configuración de visualización actual; hecho esto, se presionó el botón *guardar en mis presets*, el cual almacena la configuración indicada por medio de la base de datos a la cual se conecta el prototipo, relacionando al usuario conectado con dicho registro (Figura 8).

Figura 8. Mensaje de notificación al usuario de operación satisfactoria al almacenar su preset de visualización.

Luego, puede emplear la configuración en el proyecto actual haciendo clic en el botón *Utilizar*, el cual asociará dicha configuración de visualización a los parámetros de visualización del proyecto de simulación. Luego de indicar todas las configuraciones de visualización que se precise, y además de indicar la configuración del escenario, puede continuar con el proceso de simulación en el prototipo y generar las salidas de este.

Una vez ejecutado el proyecto, dentro de los archivos de configuración generados por el prototipo que fueron interpretados por Shawn, puede encontrarse un conjunto de comandos destinados a generar toda la configuración indicada por el *preset* de visualización empleado (Figura 9).

```
vis=my_visualization
vis_create
vis_create_edges source_regex=.* target_regex=.*
vis_constant_double value=0.450000
elem_regex=node.default.v1.* prop=size prio=1
vis_constant_vec x=0.121854 y=0.072450 z=0.690000
elem_regex=node.default.v1.* prop=background prio=1
vis_constant_int value=1 elem_regex=node.default.v1.*
prop=shape prio=1
vis_constant_double value=0.025000
elem_regex=edge.default.v1.* prop=line_width prio=1
vis_constant_vec x=0.074800 y=0.060000 z=0.800000
elem_regex=edge.default.v1.* prop=color prio=1
```

Figura 9. Configuración de visualización generada por módulo en WebShawn.

Cabe destacar dos aspectos importantes: el primero de ellos es que, de los ocho comandos expuestos, los primeros tres se utilizan para inicializar el motor de generación de gráficos de la biblioteca VIS empleada por Shawn, y los últimos cinco comandos especifican toda la configuración correspondiente al único *preset* empleado por el usuario en el proyecto de simulación, los demás nodos de la red emplearán la configuración por defecto de la biblioteca VIS. El segundo aspecto relevante se refiere a que fueron necesarios cinco comandos para establecer la configuración de visualización que se corresponde al *preset* indicado por el usuario mediante la interfaz *web* del prototipo, cada comando indica el tipo de variable de VIS a utilizar y seguidamente las propiedades del elemento en un orden establecido y crítico, ya que un error de sintaxis en cualquiera de dichos comandos puede ocasionar un error al momento de compilar y ejecutar el proyecto de simulación.

Por último, se creó el proyecto B en donde, indistintamente de la configuración de escenario que se indicó, es posible reutilizar la configuración de visualización creada y almacenada en el proyecto A; para ello en la pantalla parámetros de visualización se debe elegir de la lista desplegable la opción que contenga el nombre que este indicó anteriormente para almacenar dicha configuración (Figura 10).

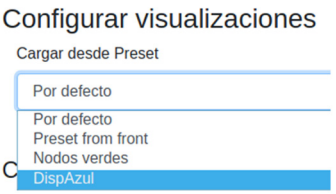

Figura 10. Lista de selección de presets de visualización creados por el usuario.

De manera análoga al procedimiento realizado en el proyecto A, se continuó con el proceso de simulación y se obtuvo la siguiente salida de simulación en donde puede apreciarse que se ha podido reutilizar la configuración de visualización con un esfuerzo relativamente bajo. La Figura 11 muestra las salidas generadas para los proyectos A y B las cuales, indistintamente de las configuraciones de escenario de simulación, reutilizan el mismo *preset* de visualización creado por el usuario.

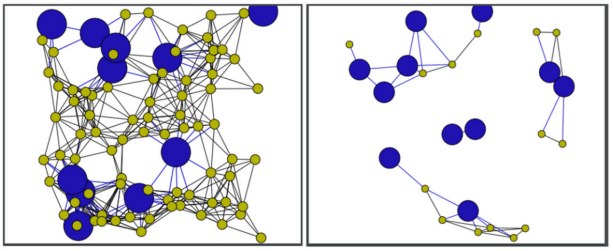

Figura 11. Resultados de visualización de los proyectos A y B reutilizando configuración de visualización mediante preset de usuario.

5. Conclusiones

Luego de realizar las modificaciones al prototipo WebShawn para poder emplear configuraciones de visualización mediante una interfaz *web*, se constató la facilidad que este brinda para que los usuarios especifiquen las configuraciones de visualización deseadas para sus proyectos de simulación con el objetivo de identificar de una manera ágil los diferentes dispositivos dentro de una red de sensores inalámbricos.

Asimismo, al brindar la posibilidad de almacenar dichas configuraciones mediante *presets*, se ha logrado un ahorro importante de tiempo al momento de reutilizar configuraciones visuales en nuevos proyectos de simulación sin la necesidad de reingresar todos los valores deseados. Además, es posible reutilizar las configuraciones almacenadas modificando ciertos parámetros de interés y guardando las mismas como nuevos *presets*. Esto da la posibilidad de volver a utilizar los *presets* en otros proyectos de simulación, permitiendo ahorrar tiempo y esfuerzo evitando escribir los comandos necesarios para que la biblioteca VIS pueda generar los resultados de visualización.

También se observó que con el prototipo se pudo lograr que la reutilización de escenarios sea mucho más simple y transparente para el usuario, pues le brinda la posibilidad de reutilizar por completo la disposición de los nodos en un escenario de un proyecto de simulación en cualquier otro proyecto en el que precise recrear el ambiente generado para verificar los efectos que pueden producirse dados otros factores implicados en la configuración de la simulación.

Por último, cabe destacar que, gracias al empaquetado de todas las dependencias del simulador Shawn y la inclusión del servidor *web* Apache con todas las configuraciones necesarias para

ejecutar el prototipo dentro de una imagen Docker, se elimina la necesidad de reconfigurar por completo todas las aplicaciones y dependencias en caso de falla del servidor y además se brinda la posibilidad de aumentar la potencia de procesamiento del simulador mediante el escalado horizontal de recursos soportados por dicha tecnología. Adicionalmente es posible compartir con otros usuarios la instalación del prototipo con facilidad, eliminando los requisitos de instalación por completo y permitiendo que estos puedan centrarse en crear proyectos de simulación.

6. Trabajos futuros

Como trabajos futuros se proponen los siguientes:

Incorporar la posibilidad de que los usuarios puedan establecer la posición de cada dispositivo dentro del escenario generado en forma manual, desarrollando para tal fin otro módulo específico con su respectiva interfaz gráfica de usuario. Esto permitirá generar ambientes de simulación con modelos de bordes poligonales.

Desarrollar un módulo que permita interactuar con el motor de simulación y visualización a través de Web Services. Esto permitirá crear sistemas de simulación con cualquier tipo de interfaz (no solo *web*). También permitirá generar salidas estandarizadas, por ejemplo en XML o JSON.

Construir un módulo que dé la posibilidad a los usuarios de interacción colaborativa, donde pueda compartirse no solamente el acceso a proyectos de simulación sino que puedan compartirse también los *presets* de visualizaciones gráficas deseados y los *snapshots* de escenarios exportados.

Refactorizar el módulo de compilación de proyectos para que este proceso sea ejecutado de manera automática por el prototipo una vez que el usuario termine de definir los parámetros de control de simulación y de visualización.

Referencias

Brambilla, M.; Comai, S.; Fraternali, P.; y Matera, M. (2008). Designing web applications with WebML and WebRatio. *Web engineering: Springer.* Londres.

Ceri, S.; Fraternali, P.; y Bongio, A. (2000). *Web Modeling Language (WebML): a modeling language for designing Web sites.* Computer Networks.

Docker Inc. (2020). *PostreSQL - Docker Hub*. https://hub.docker.com/_/postgres/

Fischer, S.; Pfisterer, D.; y Fekete, S. P. (2007). Shawn: The fast, highly customizable sensor network simulator. *Braunschweig University of Technology and University of Lubeck* (ISBN 1-4244-1231-5).

Godoy, D. A.; Sosa, E. O.; Bareiro, H.; y Díaz Redondo, R. (2014). Redes de sensores inalámbricos: interfaz web para Shawn. *XX Congreso Argentino de Ciencias de la Computación.* Buenos Aires.

Godoy, D. A.; Sosa, E. O.; Redondo, R. P.; y Bareiro, H. (2017). WebShawn, simulating wireless sensors networks from the web. *Wireless and Mobile Computing, Networking and Communications (WiMob) ISBN 978-1-5386-3839-2.*

Group, S. I. (2018). *TinyOS TOSSIM*. http://tinyos.stanford.edu/tinyos-wiki/index.php/TOSSIM

Information Sciences Institute. (2017). *The Network Simulator 2*. https://www.isi.edu/nsnam/ns/

Mármol, F. G., y Pérez, G. M. (2009). *TRMSim-WSN, Trust and Reputation Models.* University of Murcia, Facultad de Informatica.

Moreno, N.; Fraternalli, P.; y Vallecillo, A. (2006). A UML 2.0 profile for WebML modeling. *Workshop proceedings of the sixth international conference on Web engineering.*

Pfisterer, D. y Frick, M. (2013). *Shawn Wiki Visualization GitHub*. https://github.com/itm/shawn/wiki/Visualization

Ptolemy II. (2020). http://ptolemy.berkeley.edu/ptolemyII/

Saha, B. K. (s.f.). *NS2Web*. http://vlssit.iitkgp.ernet.in/

Saha, B. K.; Misra, S.; y Obaidat, M. S. (2013). A web-based integrated environment for simulation and analysis with NS-2 ISSN: 1536-1284. *IEEE Wireless Communications, 20*(4), 20.

University of California (s.f.). *AVRORA*. http://compilers.cs.ucla.edu/avrora/

VisualSense. (2020). https://ptolemy.berkeley.edu/visualsense/

WebRatio. (2020). *WebRatio*. http://www.webratio.com/

Diego Alberto Godoy
Perfil académico y profesional: Doctor en Tecnologías de la Información y Comunicación (Universidad de Vigo, España). Magister en Ingeniería de Software (Universidad Nacional de la Plata). Ingeniero en Informática (Universidad Gastón Dachary). Profesor Titular Regular. Director de la Especialización en Gestión de Tecnologías de la Información y Comunicación (Universidad Gastón Dachary).

Correo electrónico: diegodoy@citic.ugd.edu.ar
Identificador ORCID [0000-0002-7445-7375]

Santiago H. Bareiro
Perfil académico y profesional: Doctorando en Informática (UNAM). Ingeniero en Informática (Universidad Gastón Dachary). Profesor Adjunto.

Correo electrónico: hbareiro@citic.ugd.edu.ar
Identificador ORCID [0000-0002-3060-5217]

Fabián E. Favret
Perfil académico y profesional: Doctorando en Informática (UNAM). Ingeniero en Informática (Universidad Gastón Dachary). Profesor Adjunto.

Correo electrónico: efabianfavret@citic.ugd.edu.ar
Identificador ORCID [0000-0002-3774-8982]