UNIVERSIDAD POLITÉCNICA DE CARTAGENA

Escuela Técnica Superior de Ingeniería de Telecomunicación

DESARROLLO DE UN ENTORNO VIRTUAL CON UNITY PARA LA SIMULACIÓN DE MECANISMOS DE SEGURIDAD EN REDES WI-FI

TRABAJO FIN DE GRADO

GRADO EN INGENIERÍA TELEMÁTICA

Autor: Magdalena Martos Núñez

Director: Fernando Losilla López

Cartagena, 14 de abril de 2021

A mi familia por apoyarme durante todo el camino.

A mi pareja por no dejarme caer nunca.

A mis amigos que siempre han estado para sacarme una sonrisa.

A mi tutor Fernando por todo su apoyo y tiempo.

“No tengas miedo de ir lento, ten miedo de quedarte quieto." - Proverbio chino

Resumen

Actualmente nos enfrentamos a un nuevo paradigma, la digitalización. Lo que se pretende es transformar tecnológicamente cualquier tarea. En este caso se va a recrear el entorno de un laboratorio en el que se tendrán que configurar mecanismos de seguridad para redes Wi-Fi, basándose en la configuración del mecanismo de seguridad WPA2 Enterprise. Para esto es necesario configurar un punto de acceso a una red inalámbrica y un servidor RADIUS, que va a ser el encargado de autenticar a los clientes que acceden a la red. Todo esto quedará virtualizado usando la herramienta Unity que es un motor de entornos virtuales. En ella nos apoyaremos para toda la implementación.

Palabras clave: Unity, interfaz de usuario, entorno virtual, seguridad en redes Wi-Fi

Abstract

Nowadays, we are facing a new paradigm, digitalization. Its aim is to use technology to transform any task that can be imagined. In our case, we will replicate an existing laboratory deployment by means of Graphical User Interfaces in a virtual world. Thanks to this user interfaces, the user can interact and configure the parameters to satisfy the requirements of a practice session where the parameters of the WPA2-Enterprise security system are set. To implement this system, it is necessary to configure an access point and a RADIUS server, which will be responsible for authenticating clients that access to the network. All these functionalities are going to be implemented with Unity, as it is the most used engine to generate virtual environments.

Keywords: Unity, user interface, virtual environment, Wi-Fi networks securit

Contenido

Ilustraciones

CAPÍTULO 1. INTRODUCCIÓN.

1.1 Breve introducción

Estamos viviendo una era de transformación digital que, junto con la llegada de la pandemia, han creado una mayor necesidad de digitalización. Esto está generando un nuevo paradigma en la sociedad. Particularizando, se va a acercar esta necesidad de digitalización al caso de nuestra propia facultad. El desarrollo del proyecto va a estar enfocado en la virtualización de las herramientas necesarias para el desarrollo de una práctica de seguridad en redes Wi-Fi en el laboratorio de Redes Inalámbricas para el Grado de Ingeniería Telemática. La implementación se realizará con una tecnología que está en completo auge, la virtualización de entornos.

Con el fin de realizar un proyecto que garantice el entendimiento por parte del alumno, se va a poner en marcha un entorno en Unity que permite configurar entornos virtuales, en este caso, una interfaz de usuario. La finalidad es lograr un entorno virtual en el que el usuario sea capaz de interaccionar y aprender.

El proyecto se ha llevado a cabo creando réplicas de los paneles de control para la distribución libre Zeroshell y para el router Linksys con el firmware DD-WRT, utilizados en la versión tradicional de la práctica. Estos paneles recogerán toda la lógica para poder configurar los mecanismos de seguridad en redes Wifi. De forma que se pueda interactuar con los dos paneles de control mediante una interfaz de usuario. La forma de representar estos paneles de control es mediante capturas de pantalla que se les han realizado a los mismos, para luego añadir al proyecto de Unity junto con elementos de como botones, campos de texto u otros elementos, que permitan interacción con los equipos. Las capturas permiten simular el funcionamiento de dichos dispositivos mediante una interfaz gráfica análoga a la que posee cada equipo, haciendo así más fácil familiarizarse con el entorno.

El principal problema que conlleva la realización de estas prácticas es que cada alumno deba ser provisto de un router Linksys y un equipo donde ejecutar Zeroshell. Puesto que no hay suficientes equipos en el laboratorio las prácticas se realizan en grupos para así poder desarrollarla. Además, con la llegada de la pandemia, los alumnos

no han podido realizar esta práctica, resultando inviable que cada alumno pueda realizar la completa configuración desde casa o incluso en el laboratorio. Por lo tanto, se ha optado por esta solución, para que, de este modo, todos los alumnos sean capaces de aprender a configurar los mecanismos de seguridad que se requieren en las prácticas. La solución expuesta consta de proveer a cada alumno de la aplicación para realizar la práctica. Con un coste altamente menor intentando mantener la calidad de aprendizaje para los alumnos.

1.2. Objetivos del proyecto

El principal objetivo es dar al alumno la posibilidad de poder configurar la red como en el laboratorio. Ya que debido a la situación generada por la COVID-19, se ha visto inhabilitada la forma de poder acercar este tipo de práctica a los alumnos. Para el desarrollo de esta práctica, no solo basta con el uso de un ordenador que nos permite acceder a la red como clientes, sino que también se configuran características de Zeroshell y del router. Llevaremos a cabo la digitalización de forma que se virtualiza la red, para que el alumno pueda configurar el protocolo de seguridad WPA2 Enterprise, el cual cuenta con un servidor de autenticación. Con esto se da la posibilidad a los alumnos de configurar todos los mecanismos para obtener un recorrido por todo lo implementado en la práctica.

La profundización en Unity es también un objetivo principal a la hora de desarrollar la aplicación por la capacidad que tiene pare recrear entornos.

1.3. Estructura del proyecto

En el capítulo 2 se van a explicar las tecnologías empleadas: Unity y el mecanismo WPA2 Enterprise. El capítulo 3 es el corazón del proyecto, donde se explica cada paso que se ha llevado a cabo para realizar la aplicación en Unity y el entorno simulado para realizar las capturas de pantalla con WPA2 Enterprise. En el capítulo 4 se expone un caso de uso, para implementar la práctica y hacer uso de todos los mecanismos incluidos de seguridad en redes Wifi. Por último, con el capítulo 5 se concluye el proyecto y se presentan posibles líneas futuras.

CAPÍTULO 2. TECNOLOGÍAS USADAS

Para llevar a cabo el proyecto se requiere la profundización en Unity y en el mecanismo de seguridad WPA2 Enterprise.

2.1 Unity

Unity es conocido como un motor de videojuegos multiplataforma que nos permite crear experiencias virtuales. El desarrollo del proyecto se ha implementado con Unity versión 2019.4, la cual era la última versión de soporte a largo plazo (LTS) cuando comencé con la realización del proyecto. Hoy en día ya hay una versión actualizada, la 2020.3 LTS que fue lanzada en marzo de 2021.

Unity está compuesto por una interfaz muy atractiva que permite configurar cualquier aplicación desde cero, pero antes es necesario comprender cómo funciona el entorno.

2.1.1 Ventajas e inconvenientes de Unity

La selección de Unity para desarrollar el proyecto está potenciada por la capacidad para crear entornos virtuales en 3D.

Algunas de las ventajas e inconvenientes que presenta:

- Es práctico el desarrollo, ya que está provisto de varios menús con los que se permite interactuar al usuario.
- La ejecución de código es un pilar esencial, ya que cuenta con la gran ventaja de que con unas pocas líneas de código se pueden llegar a construir aplicaciones muy potentes. Unity scripting como veremos es una rama principal en la implementación de una aplicación.
- Soporta aplicaciones multiplataforma, lo que quiere decir que podemos construir una aplicación para diferentes plataformas de ejecución como pueden ser Xbox, Android, PlayStation, entre otros.
- Es un entorno con el que se pueden desplegar juegos multijugador, en el que los jugadores puedan interaccionar en tiempo real.

- Testing es una tarea sencilla gracias a que Unity incorpora un debugger, esta nos facilita comprobar nuestra aplicación, sin necesidad de usar herramientas externas.

En contraposición:

- Tiene un sistema de iluminación muy complejo.
- No se dispone de código fuente, por lo que no se podrá customizar el rendimiento. Haciendo más compleja la tarea de encontrar fallos en el rendimiento.

2.1.2 Organización del proyecto

Un proyecto en Unity cuenta con una gran variedad de ficheros y recursos, que crecen de manera exponencial conforme aumenta la complejidad de la aplicación. Por lo tanto, resulta muy útil estructurar el proyecto. Éste se organiza en carpetas que permiten separar scripts, escenas, texturas, materiales, etc. Para que los recursos puedan ser encontrados de la manera más eficiente posible. Esto se podría volver una tarea muy ardua no tener el proyecto organizado.

2.1.3 Componentes de Unity

Para crear un proyecto, primero es necesario conocer de forma detallada los componentes de Unity que vamos a usar durante la realización de este. Unity cuenta con multitud de componentes que le permiten crear estos entornos. También cuenta con componentes como cámara o iluminación que permiten visualizar este entorno. Además, Unity cuenta con una interfaz distribuida en ventanas que permite que su implementación sea más ágil.

2.1.3.1 Ventanas de Unity

Para poder adentrarnos en el mundo de Unity es necesario conocer las ventanas en las que éste se distribuye. Mediante estas ventanas, va a ser posible configurar cualquier parámetro necesario en la aplicación, como la visualización del entorno y recursos. Las ventanas principales son *Project*, *Scene*, *Hierarchy*, *Game e Inspector*. En la ilustración 1 se pueden ver estas ventanas.

1) Project o ventana del proyecto. Nos muestra la estructura de carpetas del proyecto. Esta ventana permite navegar entre las diferentes carpetas que tengamos en el proyecto. La ventana del proyecto será dónde nos encontremos packages y assests del juego.

    - Asset. Es la representación de cualquier objeto que pueda ser utilizado en el proyecto. Un asset puede importarse desde Unity o incluso desde el exterior, como puede ser una imagen o un modelo 3D.
    - Package. Es un paquete que contiene características para satisfacer las necesidades del proyecto. Puede incluir cualquier característica del core de Unity. Para manejar los diferentes paquetes que puedan ser instalados o customizados, se emplea el Package Manager. Al igual que al generar software mediante cualquier aplicación.

2) Scene o vista de escena como su nombre indica, nos permite visualizar el contenido y añadir cualquier objeto a la escena. Lo que se ve en esta está controlado por la cámara de vista de escena, que nos permitirá rotarla y posicionarla de tal forma que veamos la vista en la que estamos interesados. La vista en escena cuenta con dos tipos de proyecciones en Perspectiva y Ortográfica. En nuestro caso haremos uso de la ortográfica, también conocida como isométrica. Este tipo de proyección implica la inexistencia de perspectiva en la escena y corresponderá con una vista en dos dimensiones.
3) Vista del juego (Game) es muy similar a la vista de escena. Es lo que se muestra es la representación de la aplicación. Esta ventana es renderizada por la cámara del juego, para controlar lo que el jugador puede ver cuando se ejecuta. Esta cámara es la ubicada en la jerarquía del juego. En la vista de juego existen tres botones para controlar el juego:
    - Play
    - Pause
    - Restart
4) Hierarchy o ventana de jerarquía se encuentran todos los objetos del juego, incluyendo los componentes que son añadidos por defecto. Cada vez que actualicemos un objeto en la escena, se actualizará de forma síncrona en la jerarquía y al revés. Los objetos aparecen según el orden en el que se introducen

en la jerarquía del proyecto. Es decir, el primero que se añade al proyecto está en la parte alta, mientras que los demás irán por debajo de este.

Esto es posible de modificar arrastrando cada objeto a la posición idónea.

- Parentesco. Unity hace uso del concepto para referirse a que cualquier objeto puede configurarse como padre de un objeto hijo. Un hijo pasará a compartir con su padre su componente, Transform. Para que un objeto quede configurado como hijo, solo hay que arrastrarlo en la jerarquía dentro del objeto padre.

5) Ventana Inspector. En ella se permite editar y verificar propiedades y componentes que tiene un objeto o un assest en concreto, pudiéndose configurar en esta ventana preferencias u otros ajustes característicos de Unity. Una vez que se añade un objeto al juego se podrán visualizar y editar sus componentes. Los valores asignados en la ventana de inspector también pueden configurarse mediante scripts de código.

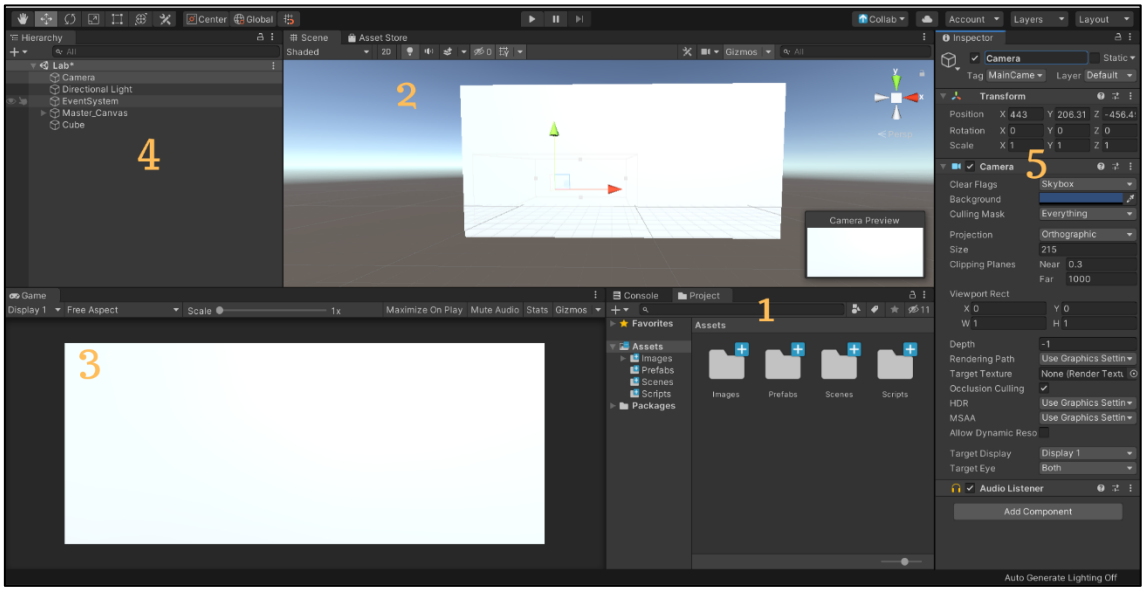

ILUSTRACIÓN 1. VENTANAS DE UNITY

2.1.3.2 Luces

Las luces son un componente esencial en la creación de una escena, puesto que nos permiten definir el color e iluminar del entorno, para así generar una mayor calidad del proyecto. Los tipos de iluminación en Unity son:

- Directional Light. Con esta luz se ilumina la escena y los objetos. Esta luz es similar a la irradiada por la luz del Sol o la Luna. Así que todos los objetos reciben la luz desde la misma dirección.
- Area Light consiste en determinar un rectángulo, la luz será emitida por uno de los lados del rectángulo.
- Point Light es muy similar a la iluminación de una lámpara. Esta iluminación permite situar un punto en el espacio emitiendo luz en todas las direcciones de forma simétrica. Conforme va aumentando la distancia se pierde intensidad lumínica.
- Spot Light enfocará en una sola dirección en forma de cono, como si fuese una linterna alumbrando. Está compuesta por una posición específica y un rango sobre el cual la luz incide, permitiendo iluminar sólo cierto ángulo respecto al foco emisor.

2.1.3.3 Cámara

Una vez que se ha entendido la iluminación, se necesita un dispositivo que permita capturar el entorno virtual. Éste será la cámara. Se pueden configurar múltiples cámaras en una escena para visualizarla desde distintos ángulos.

La cámara puede manipular la proyección, con dos configuraciones: Perspectiva y Ortográfica. Con perspectiva se renderizan los objetos creando una sensación de profundidad, mientras que con la cámara en proyección ortográfica se renderizan los objetos de forma uniforme, no existe profundidad, sino solo líneas paralelas.

En nuestro caso particular usaremos una cámara ortográfica para visualizar la interfaz de usuario.

2.1.3.4 Los imprescindibles de Unity

A la hora de familiarizarse con el entorno de Unity, es muy importante conocer a los objetos. En el mundo de la programación estamos muy acostumbrados a hablar sobre objetos, los cuales tienen asignadas funciones o parámetros. De forma análoga, en este entorno tendremos los conocidos como *game objects*.

Un game object se define como cualquier componente que se puede añadir dentro de la escena. Podría ser desde un componente de audio hasta un modelo 3D. Por sí solo un game object no tiene ninguna característica asignada por defecto, sino que es necesario el uso de componentes. Estos se añadirán para permitir cambiar el comportamiento de los game objects.

Los game objects pueden ser de diversos tipos, de hecho, nos encontramos con cientos de ellos. Todos ellos tienen en común un componente que es el *Transform*. Este componente se encarga de manejar la posición y orientación del game object en la escena. A parte del componente Transform, hay otros muchos como puede ser el componente Image, que permite añadir una foto al game object que se añada al juego. También podemos añadir contenido de audio, entre otras opciones. Es importante conocer el etiquetado de los game objects, permitiendo así agrupar a los game objects con alguna característica común. En caso de no usar ninguna etiqueta, los game objects quedarán como untagged, es decir sin etiquetar.

Los componentes de iluminación y cámara también serán game objects que se añaden al juego. También usaremos otros tipos de game objects que se van a emplear durante el proyecto, todos serán elementos UI.

- Canvas, es el game object esencial en cualquier proyecto en entornos de interfaz de usuario. Todos los elementos UI deben ser hijos del Canvas. Mediante el uso de su componente *Rect Transform*, representada como un rectángulo que define el área dónde se añadirán los elementos UI hijos. Canvas cuenta con el componente *EventSystem* para ayudarle al sistema a comunicar cualquier evento. Gracias a este objeto, podremos conseguir la estructura jerárquica. En caso de añadir cualquier elemento UI y que no esté instanciado un Canvas, éste se creará automáticamente como padre del elemento.

El Canvas cuenta con un componente de tipo *Canvas*, en el que podremos configurar el espacio del Canvas en la pantalla de juego. Existen tres modos de renderizado: *Overlay*, *Camera y Ward Space*. Para nuestro Canvas usaremos el modo Overlay que consiste en colocar los elementos en la parte superior de la escena. Esto permite que se reajuste el tamaño de los hijos de éste. Además, cuenta con el componente *Canvas Scaler*.

- Panel nos permite definir un área, que se ajustará a las dimensiones del Rect Transform de su objeto padre, que será un Canvas. Se podría entender al panel como un contenedor que permite agrupar game objects que tienen configuraciones similares.
- Text es un game object visual, el cual nos permite ilustrar un texto. En este podremos agregar en su componente *Text* el texto que se quiera introducir. Al texto se podrán configurar una fuente, estilo y color. También podremos configurar la alineación del texto. El campo de texto entonces no será más que un espacio de nuestra aplicación donde poder escribir "Hola, bienvenidos".
- Image es un game object que a su vez contará con su componente Image. En este componente se añadirá la imagen que queramos asignar al game object. La opción *Image Type* nos permite definir el tipo de Sprite que se va a aplicar. Las imágenes añadidas a este componente se importan como imágenes *UI Sprite*, en el campo *Texture Type* de la imagen importada. Esto permite que las imágenes no se distorsionen.
- Button es un game object que nos permite interaccionar con el entorno virtual. Este tiene un componente *Selectable*, que permite según el estado del botón asignarle funcionalidades. Los estados del botón son: normal, subrayado, presionado o deshabilitado. Cualquier botón tiene asignado un *Unity Event onClick*, que será un evento que se activará al pulsar sobre el botón.
- Toggle funciona de forma similar a un conmutador. Tiene dos estados que los caracterizan: activo e inactivo. Mediante una casilla se permitirá intercambiar entre estos dos estados. Está compuesto por un componente *Toggle*, que permite asignar el objeto como interactivo y también existe una casilla *Is On* para controlar su estado. Este cuenta con el componente

Selectable como el botón. Para permitir la transición entre estados, existe un evento asignado a Toggle que se activará cuando cambie su valore, este es un *Unity Event OnValueChanged.*

- Dropdown que permite crear un desplegable con diferentes opciones (A, B,C…), las cuales pueden ser customizadas. Tiene por tanto un componente *Dropdown* donde podremos definir las distintas opciones. Este game object contará con las funciones de interactivo y seleccionable. El evento que asignamos a Dropdown, es el mismo que asignamos al Toggle, que es de tipo *Unity Event OnValueChanged.* Nos encontraremos toggle cuando estemos en un entorno en el que necesitemos que el usuario elija entre diferentes opciones, como la cantidad de artículos que quieres de un producto, en valores de 1, 2, 3…
- Input Field es un campo de texto donde el usuario que interactúa puede asignar caracteres de texto en él. Se comporta de forma muy similar al game object Text. A diferencia de Input Field tiene asignados Unity Events para detectar cualquier modificación en el campo. Está compuesto por los eventos *OnValueChanged y OnEndEdit.* Este último nos va a permitir detectar cuándo se ha terminado de editar el campo de texto. Por ejemplo, podría ser un campo en el que el usuario introduzca su nombre.

2.1.3.5 Event System

El sistema de eventos Unity es capaz de capturar los eventos generados por el usuario. Nos permite manejar las comunicaciones entre módulos del EventSystem. La principal funcionalidad reside en su capacidad de detectar qué game object ha sido seleccionado. También maneja el modelo de entrada.

Los modelos de entrada no son más que el lugar donde se organiza la lógica, para definir cómo se va a comportar el Event System. Este se encargará de manejar qué se selecciona y enviará eventos a la escena de juego.

Una cualidad principal de Event System es el manejo de Graphic Raycaster, que es un emisor de rayos de luz sobre una escena compuesta por elementos UI. Gracias a él se podrá detectar qué elemento ha sido seleccionado. También se podrá configurar qué objetos dentro de la escena van a generar eventos.

2.1.3.6 Prefab

Se puede definir como plantilla. Consiste en almacenar un game object al cuál se le añaden componentes u objetos hijos. Un prefab se almacena, con el fin de volver a reutilizar esa porción de la escena que hayamos creado, para usar posteriormente. Si se modifica el prefab, se pueden modificar todos los objetos en los que se haya instanciado al prefab.

2.1.3.7 Transform

Este componente se añade a cada game object. Lo que nos permite es determinar la posición, rotación y escalado del objeto. Estas propiedades del componente quedarán representadas en el espacio con las componentes X,Y y Z.

Cuando nos encontramos con entornos 2D, como es nuestro caso, se utiliza **Rect Transform.** Queda representado como un rectángulo donde podremos añadir un elemento UI. Si el objeto añadido tiene un padre con el componente Rect Transform, el hijo podrá posicionarse y escalarse en la escena con las propiedades del Rect Transform del padre.

2.1.3.8 Scripts

La programación es un pilar esencial en la implementación de este tipo de entornos. Mediante los scripts seremos capaces de controlar y crear nuevos objetos, asignar componentes o incluso se podría implementar un sistema de inteligencia artificial. El uso de scripts convierte la labor de crear un proyecto en una tarea mucho más eficiente. Debido a que se programa de forma más rápida que con la interfaz de usuario distribuida en las ventanas de Unity.

2.1.4 Implementación de un proyecto en Unity

Procedemos a la creación del proyecto en Unity. Para lo que se inicia Unity y se crea un proyecto desde cero. Escogeremos el template de 3D, que será la opción para crear entorno de interfaces de usuario, como se muestra en la ilustración 2.

.

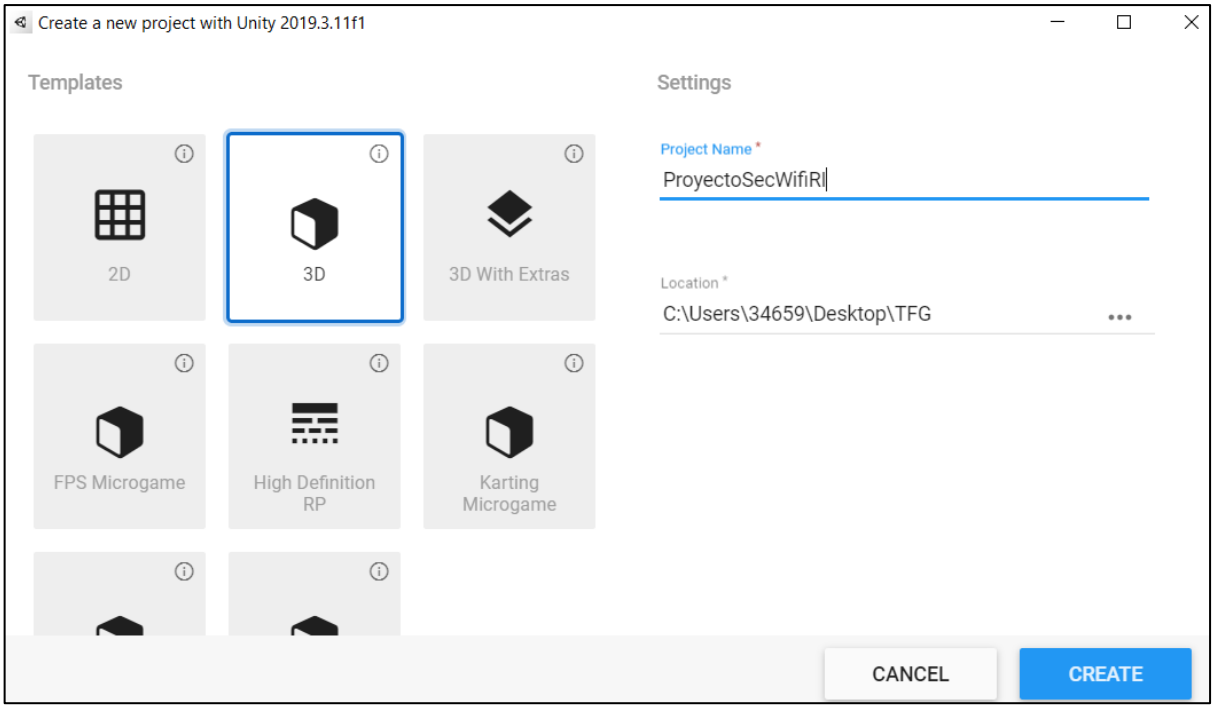

ILUSTRACIÓN 2. CREACIÓN DE UN PROYECTO.

Con el proyecto creado tendremos cargada una escena por defecto, donde se incluirán los componentes del proyecto que son la cámara principal y la luz direccional. Dentro del proyecto podemos tener varias escenas y dentro de estas escenas podremos tener tantos game objects como deseemos.

Cualquier recurso externo a Unity o Prefabs se agruparán en la carpeta de Assets. Esta carpeta es donde importaremos cualquier nuevo activo dentro de nuestra aplicación, como puede ser un script o una nueva imagen.

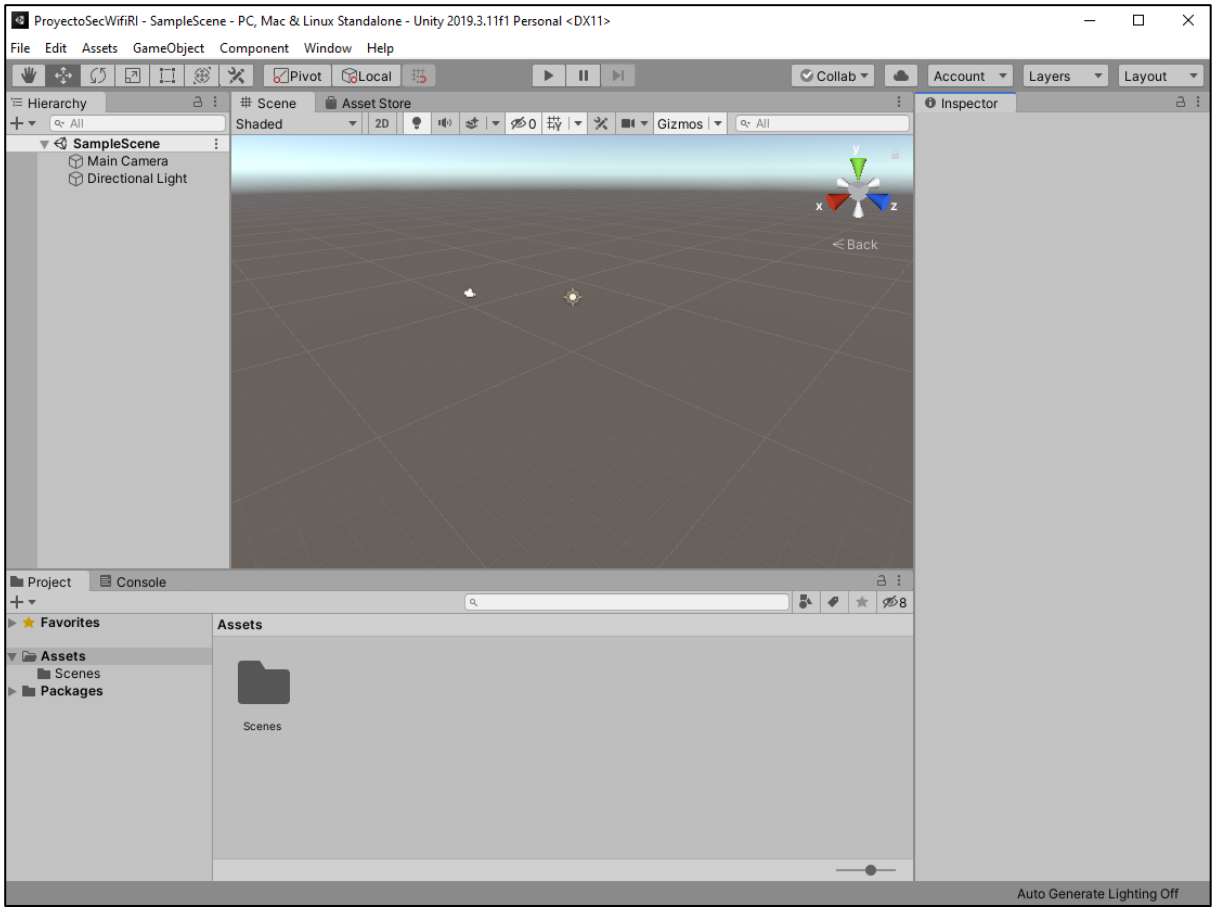

ILUSTRACIÓN 3. PROYECTO INICIAL

Como ya hemos indicado anteriormente, nuestra aplicación se va a basar en una interfaz de usuario para configurar mecanismos de seguridad Wifi. Vamos a realizar la interfaz de usuario, para lo que se hará uso de elementos de tipo UI. Así que el elemento principal que añadiremos será un Canvas, del cuál colgarán todos los game objects del proyecto.

La forma de añadir un nuevo game object es pulsando botón derecho del ratón sobre la ventana Hierarchy, o arrastrando desde la ventana de Project un Prefab almacenado. Así que pulsando botón derecho en Hierarchy accedemos a UI → Canvas. En este menú de UI estarán todos los game objects que utilizaremos.

Si queremos consultar información sobre cualquier componente lo haremos sobre la ventana del Inspector.

De forma que quede clara la organización del proyecto lo he separado en carpetas, donde cada una alberga diferentes elementos con el fin de poder encontrar de forma más rápida cualquier recurso. En la figura 3 se muestran las carpetas que componen el proyecto. La carpeta Image, contiene todas las imágenes que hemos importado al proyecto, las cuales tendrán todas textura Sprite. En Prefab, estarán los game objects que decidamos crear para poder usarlos de forma iterativa durante el proyecto. En mi caso configuraré un esqueleto de la estructura que seguirán las interfaces de Zeroshell y DD-WRT. La escena quedará guardada en la carpeta Scenes. Por último, los scripts de código en C# se almacenarán en Scripts.

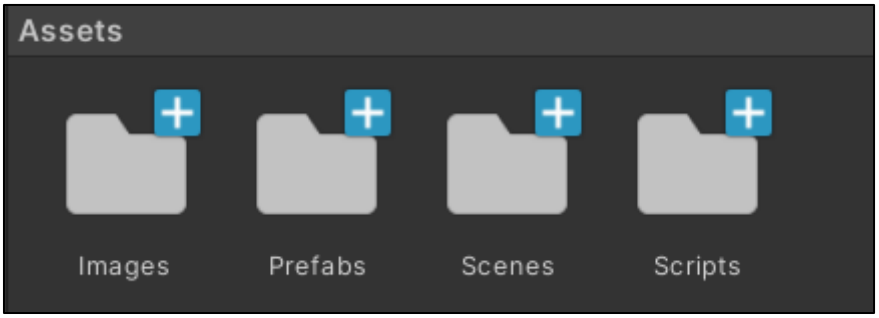

ILUSTRACIÓN 4. ESTRUCTURA CARPETAS PROYECTO.

2.2 Seguridad en redes Wi-Fi

Wi-Fi es una tecnología que permite la interconexión inalámbrica de dispositivos. Gracias a la tecnología Wi-Fi el mundo ha cambiado en la forma de comunicarse. Este pertenece a la familia de los estándares del IEEE (Institute of Electrical and Electronics Engineers). El IEEE definió una nueva familia de estándares para las redes inalámbricas definido como 802.11.

Si queremos ampliar las técnicas de seguridad en la red Wi-Fi se podría usar el filtrado de direcciones MAC. Cada red tiene configurados unos requisitos de asociación que deben satisfacerse para poder conectarse a la red.

2.2.1 Mecanismo de seguridad WPA2-Enterprise

En este caso, nos vamos a centrar en el mecanismo de seguridad WPA2 Enterprise, que se basa en el protocolo 802.1X para la autenticación de usuarios.

Previamente a WPA2, se diseñó WPA (Wi-Fi Protected Access), que protege el acceso a la red mediante un protocolo temporal de intercambio de claves TKIP (Temporal Key Integrity Protocolo). La diferencia entre WPA2 y WPA es el protocolo de encriptación puesto que en WPA2 se mejora TKIP con el uso de CCMP (basado en el estándar de cifrado AES).

Con WPA2-Enterprise, se crea un sistema más robusto. Este implementa autenticación a nivel de usuario, con el uso del estándar 802.1X, que se basa en el protocolo EAP para transportar los mensajes de autenticación. Los usuarios tienen que autenticarse mediante un usuario y contraseña, en lugar de tener una clave compartida puesto que esta clave es vulnerable al reutilizarse. Así que, reforzando la autenticación con este protocolo, los usuarios se deben identificar frente a un servidor de autenticación RADIUS. Con 802.1X se crea un nuevo paradigma en la seguridad, puesto que provee de autenticación y encriptación en la comunicación wireless.

Si nos fijamos en la siguiente figura, podemos ver que dado un cliente que quiere acceder a la red, debe conectarse a ella solicitando conexión a un AP (Access Point). La autenticación por parte del cliente será mediante usuario y contraseña o con un certificado. El AP, que es el autenticador, reenviará la información que le llega del cliente hacia un servidor de autenticación donde se usa un servidor RADIUS para esta tarea. Para comunicarse entre el AP y el servidor RADIUS comparten una clave. El servidor debe tener un certificado, que enviará al cliente que comprobará si el certificado es válido y confía en él. En caso afirmativo, el servidor le enviará una clave al cliente para poder cifrar los datos durante la conexión a la red, esta clave va cambiando lo que la hace más resiliente. Por último, el AP recibirá la respuesta del servidor con la información sobre el estado de la autenticación y la clave con la que cifrará las comunicaciones el cliente.

ILUSTRACIÓN 5. CONFIGURACIÓN 802.1X

2.2.1.1 Uso de WPA2-Enterprise

Posteriormente para poder crear el entorno es necesario tener 3 componentes: cliente, autenticador y servidor de autenticación. El cliente queda identificado como el usuario, que intentará acceder a la red y tendrá que conocer la clave expedida por el servidor para poder cifrar las comunicaciones. El cliente debe autenticarse frente al servidor mediante un usuario y contraseña para proceder a la autenticación. Por último, el autenticador, que dará acceso al cliente de forma transparente a la red, siempre que las credenciales del cliente sean válidas. Este también debe conocer la clave de cifrado que usará el cliente.

En nuestro caso, como autenticador usaremos un router Linksys que nos dará acceso una red con SSID: DD-WRT. Una vez que nos conectamos a esa red podremos adaptar todos los parámetros de configuración necesarios, para que el AP use como mecanismo de seguridad WPA2 Enterprise. En el router la configuración es WPA2 RADIUS Mixed, que satisface las mismas características.

La pieza principal es el servidor RADIUS o servidor de autenticación, el cuál será el encargado de comprobar las credenciales para cada usuario que intente acceder de nuevo a la red. Para poner en marcha el servidor RADIUS, vamos a hacer uso de una distribución de Linux, Zeroshell. Esta permite configurar firewalls y servicios de red. El cliente es un equipo que quiere acceder a una red, por tanto, no se va a detallar. Para hacernos a la idea de lo que vamos a implementar, podemos fijarnos en la siguiente imagen.

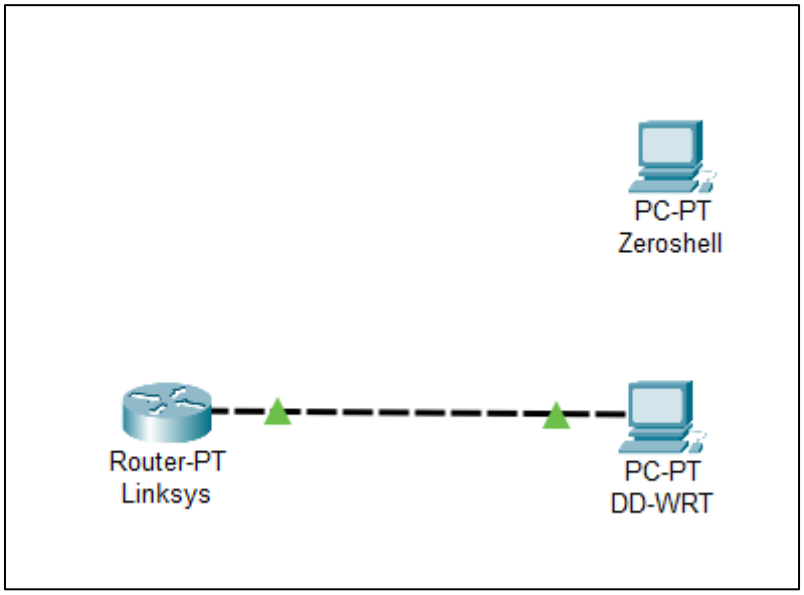

ILUSTRACIÓN 6. ESQUEMA DE WPA2-ENTERPRISE

2.3 Estado del arte

La virtualización está tomando cada vez más importancia en nuestras vidas. Existen varios casos en los que se hace uso de la creación de entornos virtuales en la docencia. De hecho, hay muchos proyectos compartiendo conocimientos con el resto del mundo, en los que se han creado entornos virtuales para que los alumnos puedan aprender de forma más didáctica y visual.

Un proyecto que se ha llevado a cabo es la creación de laboratorios virtuales simulando salas de laboratorio de ciencias, donde pueden realizar experimentos sobre física, química y biología. Un ejemplo, es un experimento de biología que consiste en apreciar cada capa que compone la piel. Esta puede ser explorada a través de una muestra epitelial que se ponga en un microscopio y ahí observarlos, estos pueden ir acompañados del nombre de cada capa.

Para la virtualización de laboratorios de redes o de configuración de distintos equipos, ya existen herramientas como Packet Tracer o GNS3. Pero la finalidad del proyecto es distinta, ya que se pretende aportar a los alumnos de una experiencia lo más similar al laboratorio. Aportando también la posibilidad de guiar a los alumnos, cosa que difiere de lo que Packet Tracer o GNS3 puedan proveer.

CAPÍTULO 3. DESARROLLO.

El proyecto se va a estructurar en carpetas que es la forma de organizarse en Unity. Ya conocemos el objetivo y cómo implementar una aplicación en Unity con el uso de sus componentes y utilidades.

La idea principal es permitir al alumno interactuar con el equipamiento del laboratorio para configurar los parámetros de seguridad, que consisten en la configuración de una red inalámbrica con WPA2-Enterprise. Previamente al desarrollo de la aplicación que usarán los alumnos, se ha procedido a realizar un montaje que consiste en conectar los dispositivos físicos que se encuentran en el laboratorio, que son un router y un equipo para simular la configuración del servidor de autenticación. Lo que se pretende de este montaje es poder hacer capturas de pantalla a cada elemento de las interfaces de usuario de estos equipos físicos y recrear gracias a estas el entorno del laboratorio en Unity. A estas capturas se le superpondrán botones, campos de texto, entre otros elementos de Unity que permitan al alumno interactuar con el entorno.

Para poner en marcha el montaje es necesario disponer de una red inalámbrica y configurar cada uno de los mecanismos para la práctica, ofreciendo al alumno una visión lo más completa y similar posible.

3.1 Preparamos el entorno para el desarrollo

Como ya hemos explicado, antes de simular en Unity tenemos que llevar a cabo la puesta en marcha. Así que debemos configurar los componentes necesarios en una red con WPA2-Enterprise. Los componentes que lo forman son el AP autenticador y el servidor de autenticación.

Una vez que, con el montaje completo, se han explorado los diferentes menús por los que los alumnos deberán navegar, comprobando cada una de las posibilidades de configuraciones que puedan darse. Una vez obtenida la visión general, se realizaron capturas de pantalla para cada menú y cada botón que los alumnos puedan configurar.

3.1.1 AP Autenticador

Para implementar el AP y hacer las pruebas, se hace uso del router que se conecta a Internet. Ahora se conectará un PC a la red DD-WRT, para poder hacer las pruebas y seleccionar la configuración adecuada. La configuración del AP se realiza mediante el panel de control DD-WRT, que es un firmware para routers que permite configurarlos. Para poder acceder al panel, es necesario navegar hacia la dirección 192.168.1.1, que coincide con la IP asignada al AP. Una vez dentro, ya podremos configurar el modo de seguridad wireless, WPA2 Mixed RADIUS.

3.1.2 Servidor autenticación

El servidor RADIUS, puede ser implementado en cualquier equipo que tenga conexión a Internet. En mi caso se utilizó un PC. Para poder realizar la configuración es necesario Zeroshell, que es una distribución libre que permite la configuración de servidores, permitiendo ofrecer servicios extra a una red local. Nuestro objetivo es habilitar el servidor RADIUS, pero pueden configurarse diversos servidores y servicios. Veamos entonces cómo arrancar Zeroshell en un equipo.

El primer paso, fue descargar e instalar VirtualBox, que permite virtualizar equipos. A su vez, se descargó el CDlive para la instalación de Zeroshell en la máquina. Una vez instalado VirtualBox, se configuró una nueva máquina virtual con 128 MB de memoria RAM y 1 GB de disco duro, en el almacenamiento se añadió el CDlive de Zeroshell. Una vez creada la nueva máquina, la arrancamos y ya podremos ver Zeroshell ejecutándose. Si consultamos la IP vemos que por defecto tiene asignada la 192.168.0.75. Navegamos a la IP indicada y ya podremos llevar a cabo la configuración para administrar el servidor RADIUS mediante el panel de control de Zeroshell.

3.1.3 Realización de capturas

Para llevar a cabo la implementación en Unity, primero es necesario realizar capturas de pantalla de cada uno de los menús. A estas capturas, se superpondrán los distintos botones, elementos de activación o campos de texto para completar datos, permitiendo de esta forma interaccionar con el entorno simulado. Los elementos superpuestos serán game objects de Unity. Cada uno tendrá los atributos necesarios para configurar la funcionalidad pedida.

Estas capturas van a ser la pieza principal de este puzle. Las capturas se realizan en cada configuración para Zeroshell o DD-WRT. El primer paso a la hora de crear el proyecto en Unity es tener las capturas para poder aplicarlas. El proyecto es una implementación en 3D, donde se alojará nuestra interfaz de usuario.

Imaginamos una captura, que representa el panel donde un usuario puede autenticarse. A esta captura se superpondrá 2 game objects, uno de tipo Input field y otro de tipo Button. Al botón se le añadirá una captura del botón que tiene el panel de autenticación para permitir hacer el login.

3.1.4 Cómo se hizo

Para facilitar cuando el alumno quiera hacer uso de la aplicación. Se ha decidido crear una aplicación, que tendrá un menú de inicio mostrado al ejecutarla. Una vez en el menú inicial se nos permitirá abrir un navegador web, el punto de partida dentro de la configuración.

El alumno aparénteme configurará todo dentro de un navegador web, que le permitirá acceder a configurar cualquiera de los 2 equipos. El usuario tendrá dos posibilidades, configurar DD-WRT o Zeroshell. Para simplificar el uso de direcciones, se ha decidido que la IP 192.168.0.1 sea la usada para acceder al AP de DD-WRT y la 192.168.0.2 para Zeroshell. A continuación, se puede ver el cómo se realiza una conexión con el navegador web a los equipos, en el que antes se tenían que ver involucrados hasta 3 equipos. Ahora todo queda encapsulado en un único equipo que nos permitirá simular las configuraciones.

.

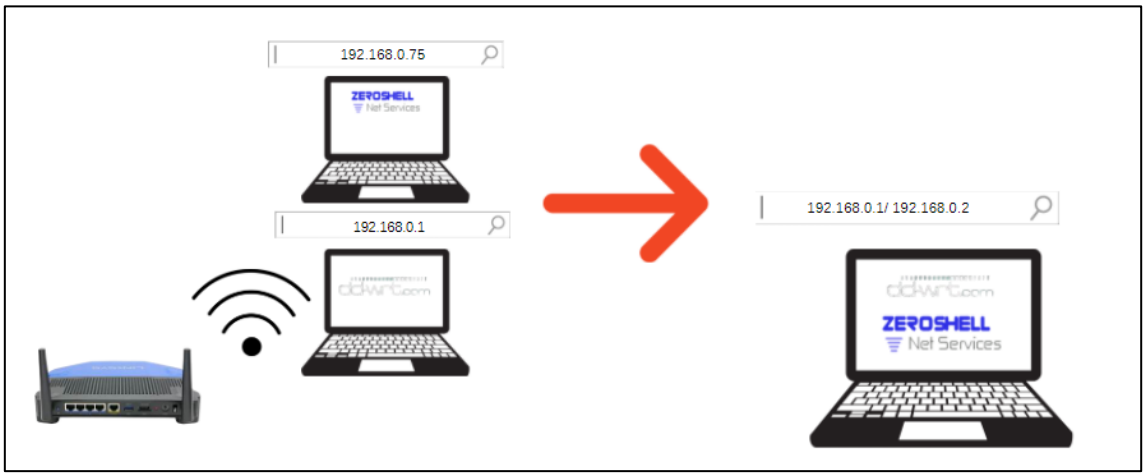

ILUSTRACIÓN 7. NAVEGADOR WEB PARA ACCEDER A LOS EQUIPOS

Una vez solicitado el acceso a cualquiera de los 2 equipos, llega la hora de configurar los mecanismos requeridos. En las siguientes ilustraciones se mostrarán las posibles configuraciones para cada uno de ellos, de forma enumerada. En caso de DD-WRT, como ya hemos dicho es el firmware para asociar un AP donde podremos configurar el protocolo, el nombre del punto de acceso y el filtrado MAC seleccionado. En Zeroshell se configurará el servidor RADIUS, un AP asociado y se añadirá un nuevo usuario.

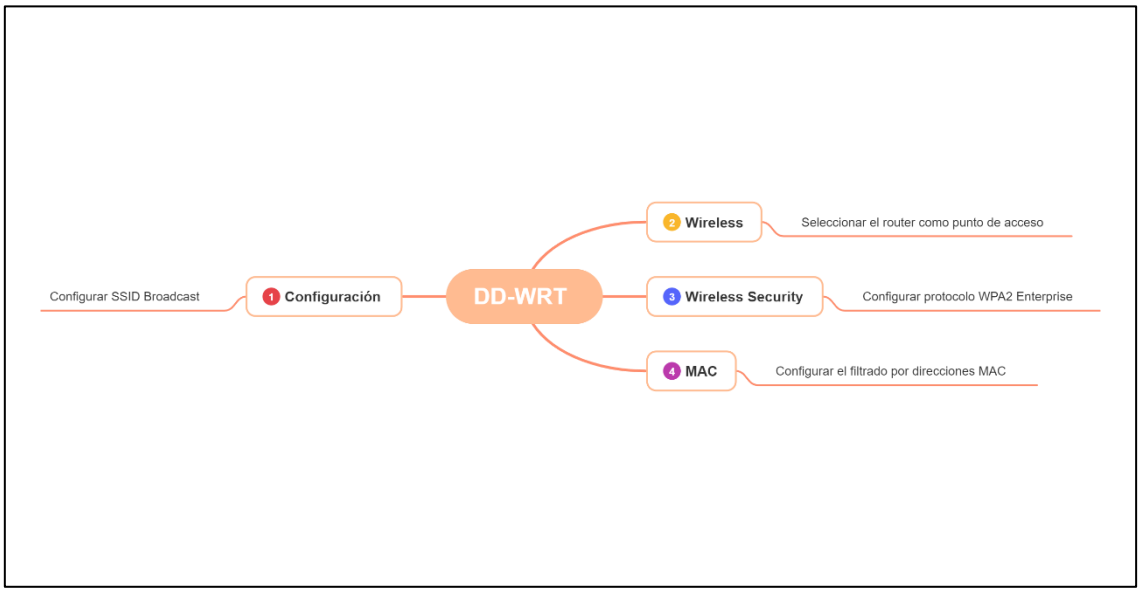

ILUSTRACIÓN 8. CÓMO CONFIGURAR DD-WRT COMO AP

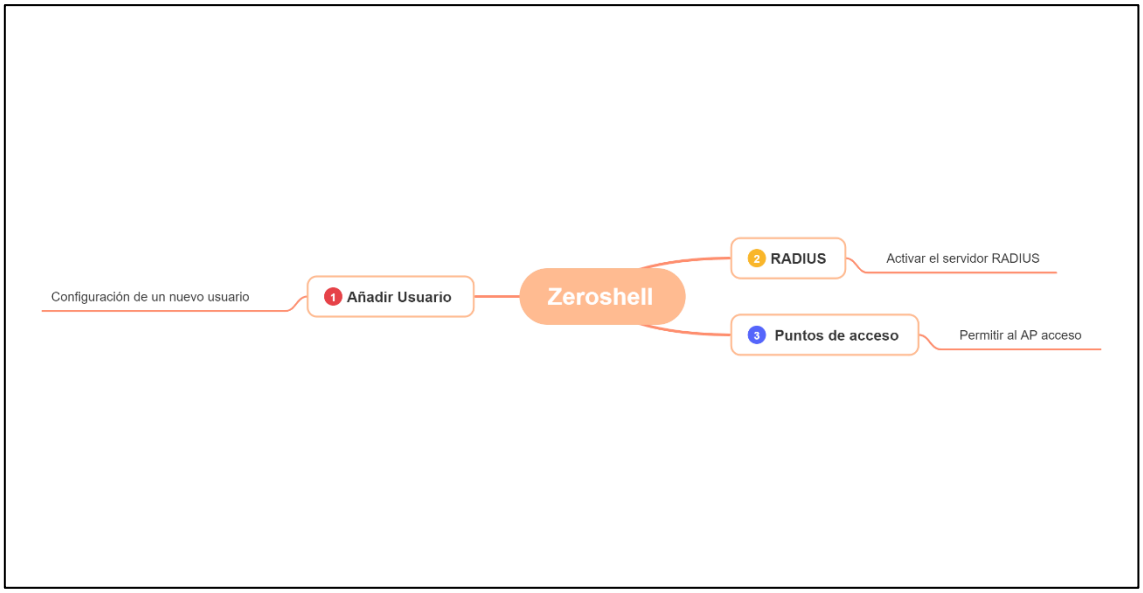

ILUSTRACIÓN 9. CÓMO CONFIGURAR ZEROSHELL PARA ACTIVAR RADIUS Y ASOCIAR UN AP

3.2 Desarrollo en Unity

3.2.1 Estructura jerárquica de interfaz de usuario

En esta parte vamos a entrar en detalle, en cómo se ha estructurado y desarrollado la aplicación, para lograr la funcionalidad requerida.

Primero nos centraremos en la jerarquía de Unity, que como hemos explicado previamente se organiza en game objects. Mediante la creación de estos objetos y añadiendo a cada uno los componentes necesarios, se podrá ejecutar la aplicación de la forma más análoga al laboratorio.

Para comenzar necesitamos un componente Scene para mostrar la escena del juego, será el padre de la jerarquía de nuestro proyecto.

Esta escena tiene por defecto configurados tres hijos: un game object Camera, un game object Light y un game object Event System. El objecto Camera se encarga de enfocar a la parte de la escena que queremos mostrar, el objecto Light que se encarga de la iluminación a la escena que queremos visualizar. Event System servirá para generar los eventos y poder capturarlos. Por último, se añade otro hijo que será un game object

Canvas, su finalidad es básicamente agrupar todos los componentes de la interfaz de usuario, puesto que el Canvas es un requisito a la hora de desplegar cualquier tipo de interfaz de usuario en Unity. Todos los objetos que compondrán esa interfaz serán hijos de este objeto canvas los cuales serán explicados con más detalle después.

En la ilustración 10 vemos a continuación se muestra el esqueleto básico de nuestra aplicación.

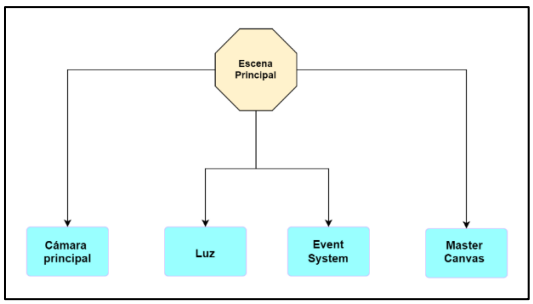

ILUSTRACIÓN 10. ESQUEMA PRINCIPAL.

3.2.2 Master Canvas

Para satisfacer los requisitos a la hora de incorporar game objects del tipo interfaz de usuario a la escena, es necesario por lo tanto tener predefinido un Canvas. Este será el objeto padre, del que colgarán todos los game objects que se configuran para la interacción con la interfaz, definidos como hijos de Master Canvas. Este game object, por sí solo no nos aporta ninguna funcionalidad. Pero es un prerrequisito para poder instanciar cualquier tipo de game object para interfaces de usuario.

La importancia de Master Canvas en nuestra aplicación reside en su capacidad para poder interactuar con los diferentes game objects que se vayan añadiendo a la jerarquía como descendientes, pudiendo controlar cómo intercambiar información entre estos objetos.

Tendremos 4 hijos para este Master Canvas, que serán game objects vacíos por el momento. Se instancian como *Inicio, Navegador, DD-WRT* y *Zeroshell*, que van a ser definidos con más detalle en las siguientes secciones. Esta estructura jerárquica se muestra en la ilustración 11.

Como se ha expuesto, este Master Canvas nos va a proporcionar la capacidad de poder manejar a todos los objetos que cuelguen de él. Para eso es necesario una configuración previa de un controlador que pueda delegar funciones para cada game object adjunto. La forma de implementarlo es añadiendo a Master Canvas un nuevo componente Script, denominado *Controlador principal*, en el que se implementará toda la lógica que nos permita controlar estos 4 objetos.

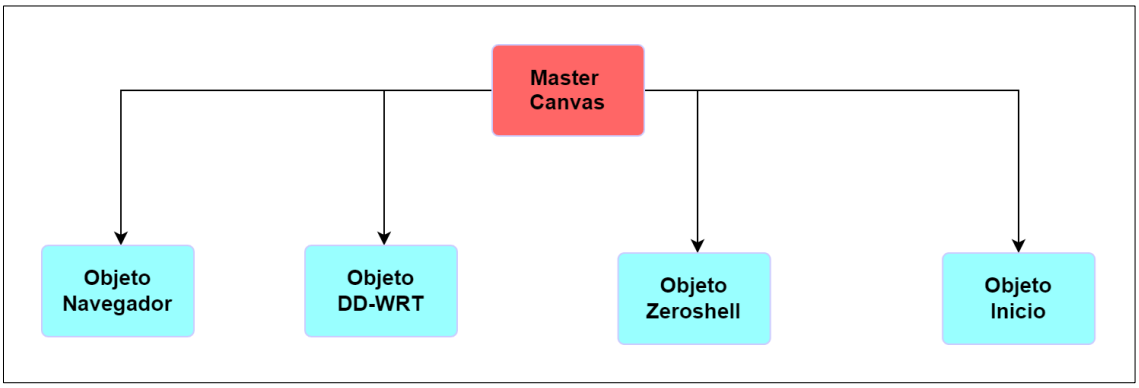

ILUSTRACIÓN 11. ESQUEMA MASTER CANVAS.

3.2.2.1 Inicio

Al arrancar la aplicación se activará el game object Inicio. Este contendrá todos los componentes necesarios para crear el primer menú, a su vez será la primera toma de contacto del usuario con la experiencia virtual.

Anteriormente, se ha expuesto en la definición de Master Canvas que los hijos de este son creados, por defecto como game objects vacíos. En este caso tenemos el game object Inicio, el cual no tiene adjuntado más que un Transform. Con el fin de implementar este menú se añaden otros game objects al game object *Inicio.*

El primer paso, es añadir un game object UI (User Interface) de tipo Panel, con el fin de agrupar toda la información bajo ese panel, referenciado como *panel Inicio.* A su vez estará compuesto por 3 game objects, que nos permitirán referenciar los diferentes suplementos que tiene nuestro panel Inicio; estos son *Bienvenida, Ayuda* y *Encendido.* El game object Bienvenida, tendrá un descendente de tipo Text, en el que se incluirá un texto que se imprimirá al arrancar la aplicación. Por otro lado, contamos con el game

object Encendido, al que se ha adjuntado un game object de tipo Button. Esto nos permitirá simular el arranque de nuestra aplicación. El botón de encendido contará con un componente de tipo Image donde se adjunta una imagen que hace referencia a un icono de un botón típico de encendido. Como hemos expuesto antes, los botones tienen eventos asociados que nos permiten interactuar con ellos y ejecutar determinadas funciones al detectar un tipo de evento determinado. En nuestro caso, evaluaremos los eventos onClick, que se ejecutarán cuando se pulse el botón, permitiendo así a los usuarios poder controlar el entorno simulado. También existe, un game object de Ayuda, el cual será explicado con más detalle posteriormente. En la ilustración 12 y 13 se muestra el funcionamiento del panel y su visualización en la ventana de juego de Unity.

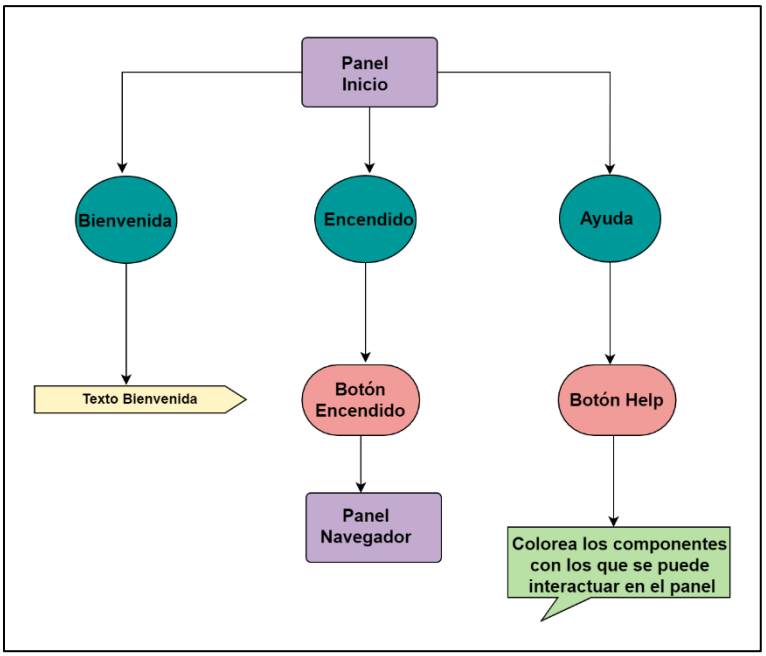

ILUSTRACIÓN 12. ESQUEMA PANEL INICIO

ILUSTRACIÓN 13. PANEL INICIO

3.2.2.2 Navegador

Una vez entramos en la vista de juego y simulamos el arranque, nos encontraremos con el panel de navegación web. Previamente debemos haber salido del panel de inicio. Lo que se quiere lograr con este panel es conseguir simular un explorador web para así acceder a los routers mediante una URL. Este navegador web, no tiene salida a Internet como tal, sino que está configurado para aceptar las configuraciones que se requieren en la práctica. Así que nos encontramos que la URL tendrá dos posibles valores, o bien 192.168.0.1 o 192.168.0.2, que nos redirigirán a los paneles de DD-WRT o Zeroshell, respectivamente.

Es necesario que se configuren game objects con determinada funcionalidad con el fin de poder hacer un uso interactivo del panel Navegador. Por lo que se le ha añadido en la jerarquía un game object, que es de tipo Image. Esta imagen no será más que una captura de pantalla de un navegador web. Toda la lógica de este navegador colgará del game object Imagen, al cual se añadirán sus descendientes. A esta imagen se adjuntan 3 game objects vacíos: *Ayuda*, *Seleccionar IP*, y *Volver*.

Para seleccionar la IP se ha añadido un game object de tipo Input Field, que nos permite añadir la url del navegador. Así que se configura este game object, con el fin de escribir en él la IP del dispositivo que se vaya a querer configurar en ese momento. También contamos con Volver, su implementación está pensada para poder salir del panel en el que nos encontramos y volver al panel de Inicio. Esto se consigue, añadiendo un game object de tipo Button. Este botón tiene una particularidad, que difiere del resto, ya

que añade en él una imagen como fondo del botón. Que consta de una flecha. La configuración de Ayuda y Volver se explicará más tarde.

En la ilustración 14 y 15 se muestra el funcionamiento del panel y su visualización en la ventana de juego de Unity.

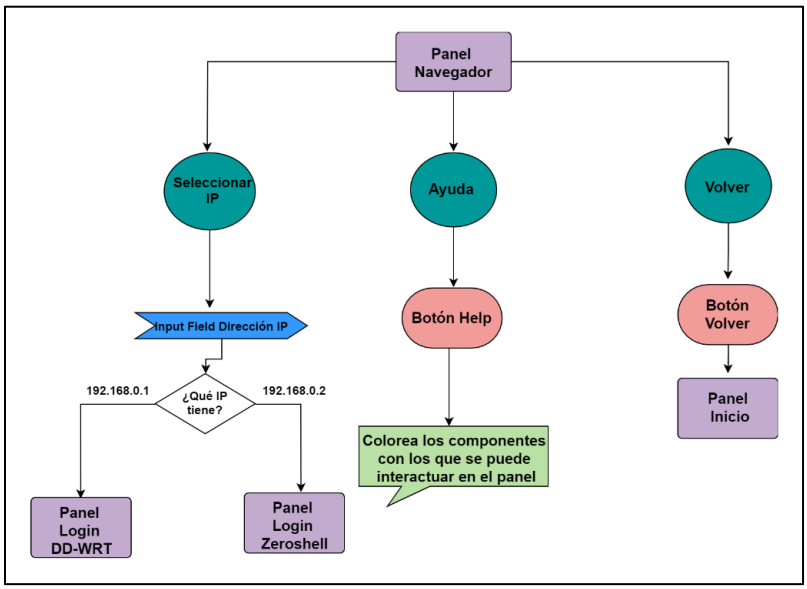

ILUSTRACIÓN 14. ESQUEMA PANEL NAVEGADOR

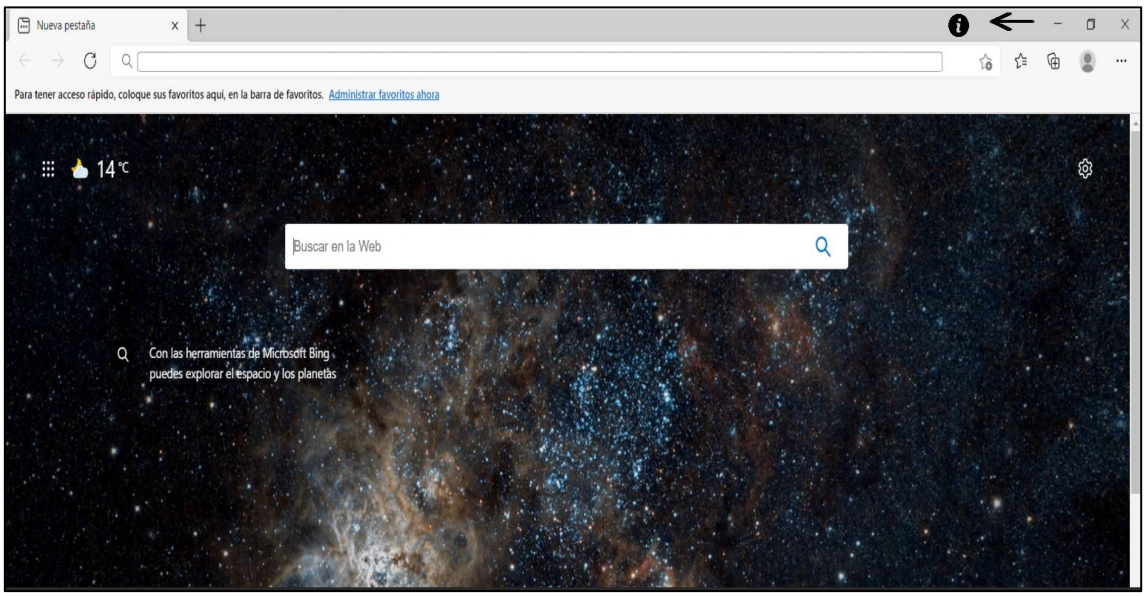

ILUSTRACIÓN 15. PANEL NAVEGADOR

3.2.2.3 Zeroshell

El objeto Zeroshell es donde queda alojada toda la lógica para configurar Zeroshell. Este va a permitir al usuario poder interaccionar con cualquiera de las posibles configuraciones que hay dentro de este software que ejecuta funciones de red.

El game object padre que se denomina Zeroshell en Unity. De él cuelgan 9 game objects de tipo Panel. Se denominan: *panel Login, panel Main, panel Usuarios, panel Usuarios modificado, panel Añadir usuario, panel RADIUS ON, panel RADIUS OFF, panel Clientes autorizados y panel Clientes autorizados modificado*. Cada uno de estos objetos corresponderá con los paneles que se podrán visualizar en la escena.

El fin de mostrar cada panel es poder ofrecer al usuario, la capacidad de poder interaccionar con cada uno de los posibles menús con los que se deben enfrentar al realizar la práctica. Por ejemplo, el alumno debe ser capaz de configurar un nuevo usuario, añadir un servidor RADIUS, entre otros. Cada uno de estos paneles se explicará con más detalle a continuación.

Para poder hacer un correcto uso e implementar los mecanismos para cada panel, se ha implementado un script Controlador Zeroshell, que se encargará de cubrir toda la lógica que hay en nuestro objeto Zeroshell, permitiendo al usuario realizar las tareas que deben llevarse a cabo a través de este software.

Cabe remarcar que todos estos paneles tendrán un mismo esqueleto, puesto que se ha diseñado un Prefab, llamado panel Zeroshell, con el contenido básico común entre todos los paneles de Zeroshell. Todos contarán con un game object tipo Panel, al cual se adjuntará una Imagen. Esta imagen será la que engloba todas las funcionalidades quedaban tener nuestro panel. En la siguiente ilustración se muestran todos los paneles que cuelgan de Zeroshell y en la ilustración 17 la configuración en Unity de Añadir usuario.

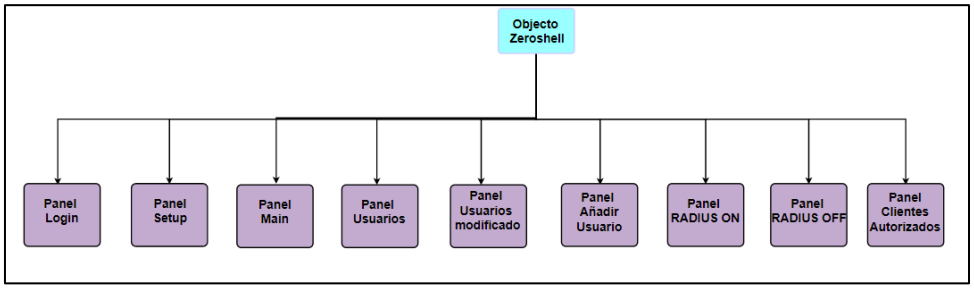

ILUSTRACIÓN 16. ESQUEMA OBJETO ZEROSHELL.

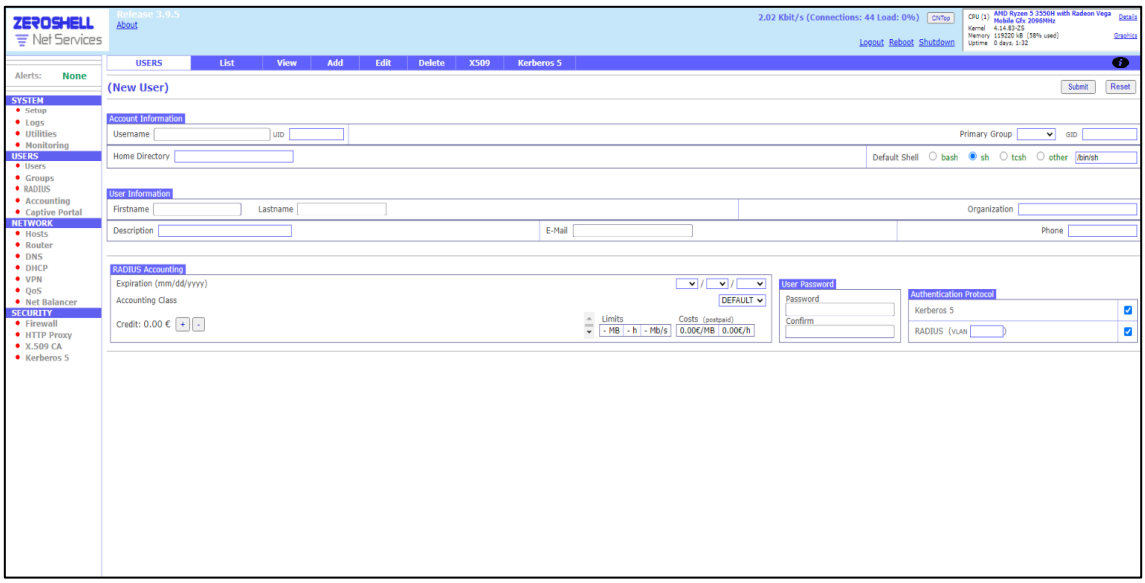

ILUSTRACIÓN 17. PANEL AÑADIR USUARIO

3.2.2.3.1 Panel Prefab Zeroshell

Con el fin de optimizar la realización del proyecto se va a hacer uso de un panel que se guardará como un prefab. Este prefab podrá ser instanciado en cualquier momento para crear a partir de estos nuevos paneles. Lo interesante de este panel, es que podremos agrupar componentes que son comunes entre todos los paneles de la aplicación de Zeroshell. El panel principal de prefab, se debe encontrar en un entorno de interfaz de usuario, por lo que solo podrá añadirse cuando haya un canvas en la jerarquía ascendente de este panel. En primer lugar, contamos con un hijo de este panel que es un game object de tipo Image, en cada caso se añadirá la imagen adecuada con cada menú que vayamos a configurar. Los otros elementos que serán compartidos entre todos los paneles son 4 game objects: *Usuario, RADIUS, Setup y Ayuda*, los cuales serán objetos vacíos. El fin de estos objetos será contener un game object botón adjuntado.

Los botones principales de la aplicación son: Usuarios, RADIUS y Setup. Estos serán los posibles menús que podremos visitar y configurar dentro de Zeroshell. Estos 3 game objects serán comunes en todos los paneles de Zeroshell, excepto en el panel Login. Estos game objects, tienen en común que de todos ellos colgará un game object de tipo Botón, que permitirá capturar cualquier clic sobre estos botones y nos redirigirán al panel apropiado con cada botón. Por lo tanto, la correspondencia entre botones y paneles quedará de tal forma que: el panel Usuarios se alcanza pulsando el botón Usuarios, el panel RADIUS mediante el botón RADIUS y por último el botón Setup que nos llevará hacia el panel Main. Esta configuración se lleva a cabo para crear una funcionalidad lo más similar a la del software real.

El game object Ayuda estará configurado con el botón Ayuda, que será explicando posteriormente.

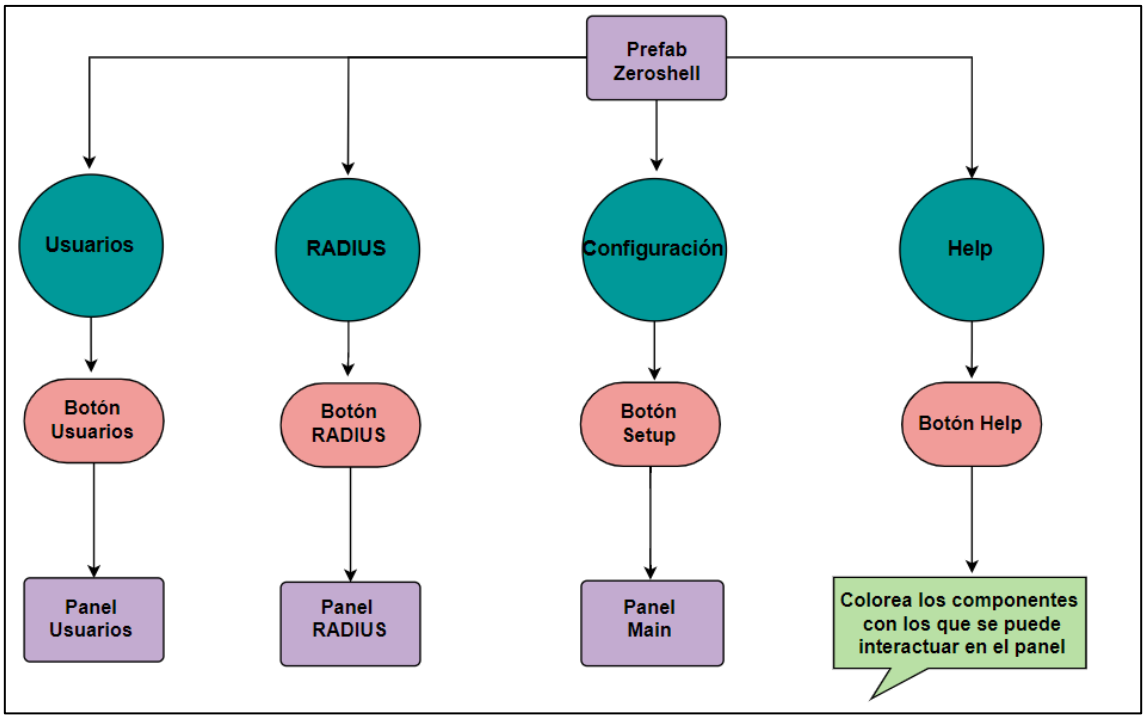

ILUSTRACIÓN 18. ESQUEMA PANEL PREFAB ZEROSHELL.

3.2.2.3.1 Login

Para poder autenticarse frente al software de Zeroshell se implementa el panel Login para que mediante las credenciales se pueda acceder a la configuración.

Para crear este panel se va a utilizar un game object de tipo Panel, el cual tendrá un hijo de tipo Image. En este se añade una captura del login que nos aparece al arrancar Zeroshell. La imagen contará 5 game objects hijos: *Contraseña*, *Datos Login*, *Login*, *Ayuda y Volver* como se puede ver referenciado en la figura 5.

El game object Datos Login, será el padre de 2 input fields, uno que servirá para almacenar el nombre y otro para la contraseña. Como sabemos una contraseña debe quedar cifrada, para evitar cualquier problema por el robo de datos. Por lo tanto, se debe configurar el input field Contraseña seleccionando el tipo de contenido como password.

Contamos también con el game object Contraseña y Login, que tienen la misma funcionalidad, que no es más que comprobar que los valores introducidos en los inputs field del objeto Datos Login son correctos, mediante un botón. Los game objects restantes son Volver y Ayuda, que será explicada su lógica de forma separada posteriormente.

Imaginamos una captura que sea del panel donde un usuario puede autenticarse, a esta captura se superpondrán 2 game objects de tipos Input field y Button. A este botón se le añadirá una captura del botón que tiene el panel de autenticación para permitir hacer el login.

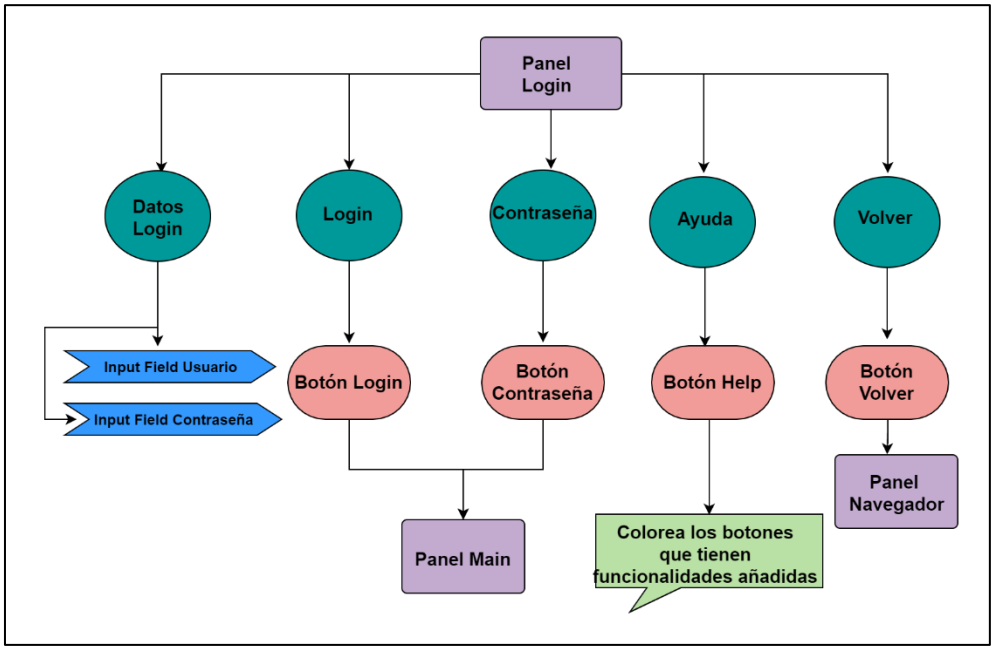

ILUSTRACIÓN 5. ESQUEMA PANEL LOGIN

3.2.2.3.2 Panel Main

El propósito de este panel es poder ver la configuración principal donde se pueden observar la lista de los servicios que pueden ejecutarse en Zeroshell. La lógica de este panel no será más que una transición. Básicamente nos permite interactuar con los demás menús configurables de Zeroshell.

Para crear el panel Main, primero añadimos el prefab Zeroshell. Modificaremos el game object Image adjuntado al panel, añadiendo la imagen de este nuevo panel. Esta contará con 5 game objects: *Usuarios, RADIUS, Setup, Help y Volver*. Los 4 primeros, que vienen descritos por el prefab y por último Volver. En este último se añadirá un game object tipo Button, el cuál al ser pulsado nos permite salir de la aplicación de Zeroshell y entrar en el Navegador.

3.2.2.3.3 Panel Usuarios

Para poder consultar información sobre los usuarios o añadir nuevos usuarios a la aplicación debemos entrar en el panel Usuarios.

Nos encontramos con una estructura en este panel muy similar al anterior. Partiendo de imagen del prefab, tiene como fondo el menú de usuarios de Zeroshell. De esta imagen colgarán 5 game objects: *Usuarios, RADIUS, Setup, Help y Añadir Usuario*.

Para permitir configurar nuevos usuarios se va a hacer uso del game Object Añadir Usuario, que cuenta con un hijo de tipo Button que nos permitirá acceder al menú para añadir nuevos usuarios.

3.2.2.3.4 Panel Añadir Usuarios

Si queremos agregar usuarios a la aplicación, tendremos que registrarnos con los datos que nos pide. Tendrán que quedar rellenados todos los datos para poder añadir este nuevo usuario.

Este panel cuenta la imagen de fondo y por debajo estarán los game objects: *Usuarios, RADIUS, Setup, Help y Datos Usuario.* La lógica de este panel está por lo tanto bajo el nuevo componente que nos encontramos denominado Datos Usuario. Mediante la ilustración del panel se expone con más detalle cómo quedaría la jerarquía. Hay que tener en cuenta que en la ilustración solo se va a añadir el nuevo game object Datos Usuario.

Este game object tiene el fin de recoger todos los valores necesarios para poder añadir un nuevo usuario a Zeroshell. Para ello se hace uso de game objects que permitan al usuario introducir texto en la aplicación mediante Input Fields. También es importante comprobar que todos los datos que se introduzcan sean válidos o no haya campos obligatorios incompletos. En caso de darse algún error se usarán avisos al usuario mediante game object tipo Text. Finalmente, existe un botón para poder añadir a este nuevo usuario. Los hijos de Datos Usuario son: *Nick, Nombre, Apellido, Email, Contraseña1, Contraseña2, Directorio, Descripción, Contraseña inválida, Precaución y Subir.*

Los 6 primeros game objects, que serán de tipo Input Field, son los campos que el usuario podrá introducir, simplemente tendrá que pulsar sobre la caja para introducir texto y escribirlo, de forma que el usuario deberá completar sus datos personales con nombre, usuario, apellido, email y contraseña. Hay que tener en cuenta que los Inputs field de Contraseña tendrán el tipo de contenido configurado como Password.

Por otra parte, los game objects Directorio y Descripción son de tipo text, la cualidad de estos campos para este caso es que se generarán de forma automática. El directorio nos especifica la ruta del usuario que vendrá definida por “/home/nick”, nick tendrá el valor del game object nick cuando este haya terminado de editarse por el usuario. De forma muy similar se actualizará Descripción, que una vez completados los game

objects Nombre y Apellido nos generará una cadena de texto compuesta por "NombreApellido", cuando ambos hayan sido editados.

Una vez se hayan completado todos estos game objects entra en juego el botón Subir, el cuál comprobará que todos los valores hayan sido introducidos permitiendo así actualizar la lista de usuarios. Si todos los datos son correctos, se accederá al panel Usuarios Modificados, que contendrá a este nuevo usuario. Pero en caso de que pulsemos el botón y alguno de los campos no se introduzca, saltará un mensaje de error. Este mensaje corresponde con el game object Precaución, que nos advertirá que debemos introducir todos los parámetros para poder subirlos.

Además, el game object Contraseña inválida de tipo text nos alertará siempre y cuando las contraseñas introducidas en los game objects Contraseña1 y Contraseña2 no coincidan.

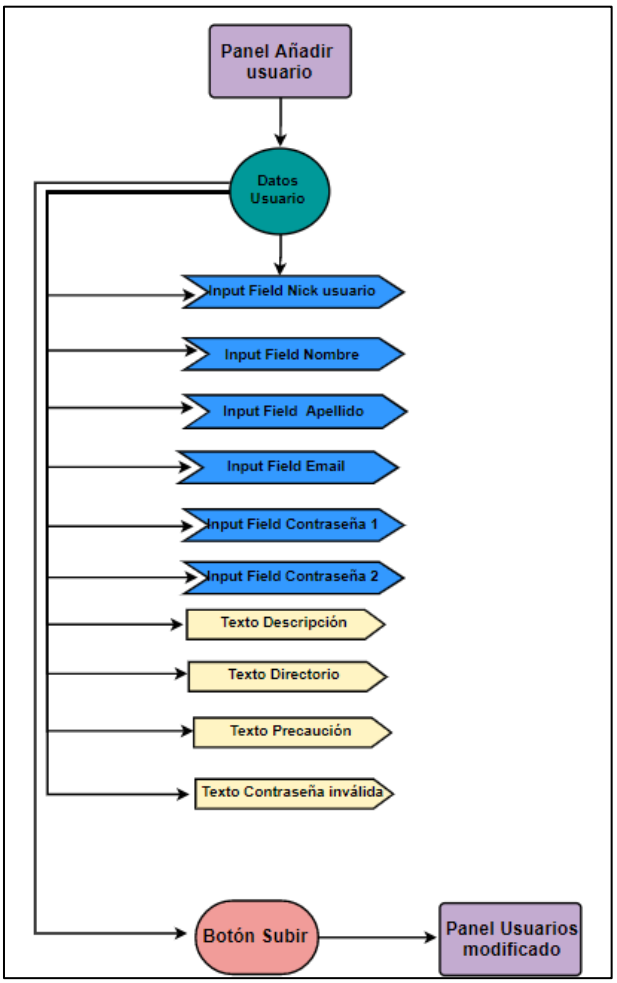

ILUSTRACIÓN 19. ESQUEMA DATOS USUARIO

3.2.2.3.5 Panel Usuarios Modificados

Para poder actualizar los nuevos usuarios crearemos este nuevo panel, cuya función principal es añadir los nuevos datos. La interfaz será igual que el panel Usuarios.

Al subir los datos desde el panel Añadir usuarios, pasamos a este panel. La implementación de este panel es muy similar a la del panel Usuarios, con la diferencia de que éste nos mostrará el nuevo usuario que haya sido añadido. El panel Usuarios modificados estará compuesto por los game objects: *Usuarios, RADIUS, Setup, Help y Datos Nuevo Usuario.*

El game object Datos Nuevo Usuario contendrá 4 game objects: *Grupo, Descripción, Nick y Email.* Todos serán de tipo Text, por lo tanto, será un texto que se autogenerará con los valores introducidos para el nuevo usuario en el panel Añadir Usuario. El campo de Grupo se rellenará con un 1 por defecto, mientras que los campos restantes contendrán los valores añadidos.

Se implementará el panel de tal forma que solo se pueda acceder a él una vez que se ha añadido un nuevo usuario. Cuando esta acción se ejecute, implicará que cada vez que se pulse el botón Usuarios, para cualquiera de los paneles, el panel en activarse sea el panel Usuarios Modificados. Puesto que contendrá la información actualizada de los usuarios que componen Zeroshell.

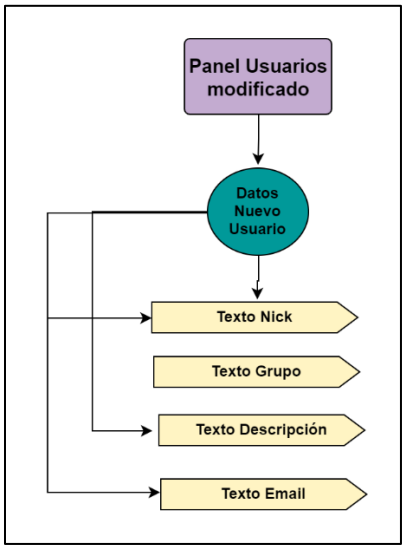

ILUSTRACIÓN 20. ESQUEMA OBJECTO DATOS NUEVO USUARIO

3.2.2.3.6 Panel RADIUS Desactivado

Para poder configurar el servicio RADIUS en nuestro router es necesario activarlo. Por lo tanto, se implementa este panel para llevarlo a cabo. En este podremos activar RADIUS y comprobar que el certificado no esté revocado. El panel está formado los game objects: *Usuarios, RADIUS, Setup, Help y RADIUS ON.*

Con el fin de activar RADIUS se ha configurado el game Object RADIUS ON, que cuenta con 4 game objects: *Comprobar CRL, Activar RADIUS, Precaución y Guardar.*

Tanto Comprobar CRL como Activar RADIUS son game objects de tipo toggle, lo que permitirá cambiar el estado de estos objetos. El primero, nos permite comprobar que el certificado adjuntado no está caducado y es válido. Mediante el toggle Activar RADIUS conseguiremos activar el servicio de RADIUS en nuestra red. En caso de que cualquiera de estos game objects se active aparecerá un texto que advertirá que los cambios no se han guardado, debido a que no se almacenan cambios de forma automática. Para que los cambios queden guardados es necesario pulsar sobre el botón Guardar, que será hijo del game object Guardar. Una vez se guarden los cambios, pasaremos al panel RADIUS Activado.

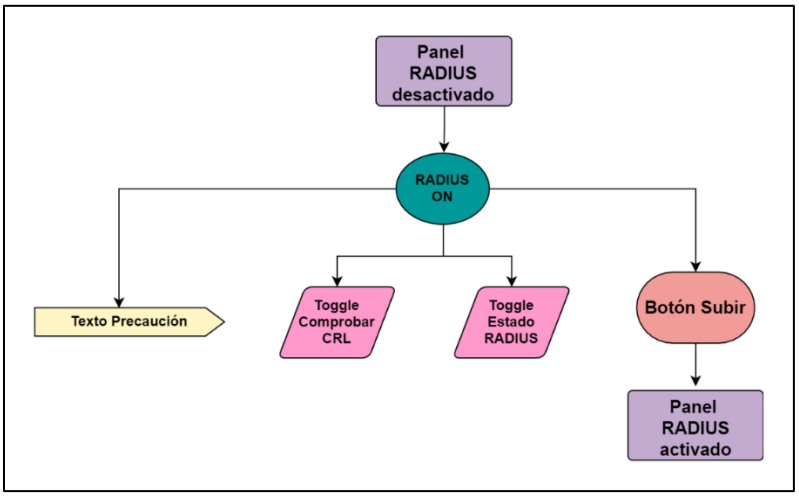

ILUSTRACIÓN 21. ESQUEMA OBJECTO RADIUS OFF

3.2.2.3.7 Panel RADIUS Activado

Una vez se ha activado RADIUS se pasa a este panel, en el que podremos ver que el servicio está funcionando y se podrán añadir clientes autorizados al mismo.

Este panel está compuesto por los game objects: *Usuarios, RADIUS, Setup, Help, RADIUS OFF y Clientes Autorizados.*

El game object RADIUS OFF permitirá desactivar RADIUS y volver a activar el panel de RADIUS desactivado. Una vez que RADIUS se encuentra activo, podemos añadir clientes que pueden configurarse. Esto se puede realizar en el panel Clientes Autorizados, que se explicará en el siguiente punto.

Antes se ha expuesto que por defecto la aplicación nos mostrará el panel de RADIUS Desactivado, pero una vez que se guardan los cambios en ese panel quedará RADIUS activado. Por lo tanto, de ese momento en adelante en cualquier momento que se pulse sobre el botón RADIUS se mostrará el panel de RADIUS Activado.

3.2.2.3.8 Panel Clientes Autorizados

Una vez que queramos añadir clientes, entraremos a añadirlos en el panel Clientes Autorizados, donde aparecerán todos los posibles AP que puedan conectarse. Este panel constará de 7 game objects: *Nombre Cliente, IP, Máscara, Contraseña, Añadir, Cerrar y Precaución.*

Los 5 primeros corresponden con los campos que el usuario podrá añadir de forma manual, por lo tanto, se usan game objects de tipo input field. El game object Nombre contendrá el nombre del cliente. La IP y Máscara identificarán a la IP correspondiente a ese dispositivo junto con su máscara de red. Por último, el usuario debe añadir la contraseña que permitirá la asociación del cliente a RADIUS.

En caso de querer guardar un cliente, será necesario pulsar el botón Añadir, que almacenará los cambios de este nuevo cliente. Si alguno de los campos no se ha completado obtendremos una alerta, que corresponde con el game object Precaución que será de tipo texto, en la que se remarcará al usuario de que debe completar todos los campos. Si se han completado todos de forma satisfactoria pasamos al panel Clientes Autorizados Modificado.

Si deseamos salir del menú para añadir clientes, bastará con pulsar sobre el botón Cerrar, que corresponde con el game object Cerrar. Esto nos llevará al panel que estábamos previamente; panel RADIUS Activado.

3.2.2.4 DD-WRT

En el router Linksys podremos configurar los parámetros de seguridad para la red DD-WRT. Mediante el navegador web podremos acceder a configurar todas las funcionalidades para este router.

El game object DD-WRT se maneja toda la lógica de este software para lograr el cumplimiento del desarrollo de la práctica. Básicamente distribuiremos los distintos paneles configurables de igual forma que para Zeroshell. En la ilustración 23 se puede ver una captura de la interfaz de usuario para este equipo.

Lo primero a tener en cuenta es que para poder manejar este nuevo game object es que para poder configurar de forma adecuada cada una de las posibles configuraciones. La lógica de cómo manejar estos nuevos requerimientos será a través de un script denominado Controlador DD-WRT. Este se encargará de coordinar a todos los paneles y todos los posibles componentes con los que será necesario intercambiar información.

Los hijos de este objeto DD-WRT son: *panel Login, panel Inicio, panel Configuración, panel Wireless activado, panel Wireless desactivado, panel seguridad wireless 1, panel seguridad wireless 2, panel MAC desactivado y panel MAC activado.*

Para implementar este game object, se ha pasado por diferentes posibilidades de configuración. Se ha decidido usar un prefab llamado DD-WRT, que nos permite realizar la implementación de forma más rápida.

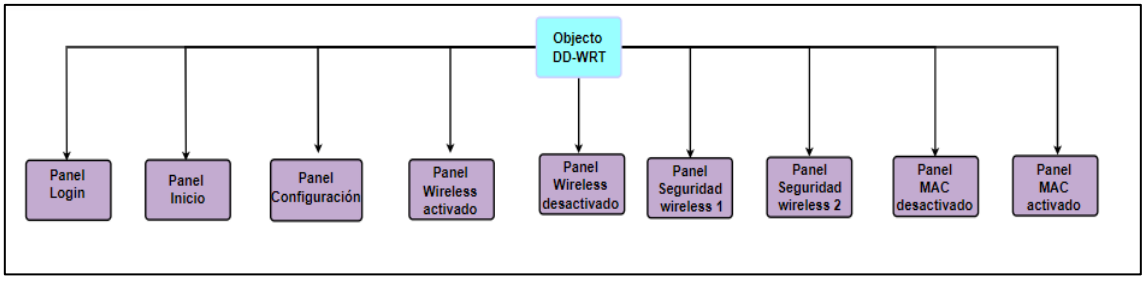

ILUSTRACIÓN 22. ESQUEMA OBJETO DD-WRT.

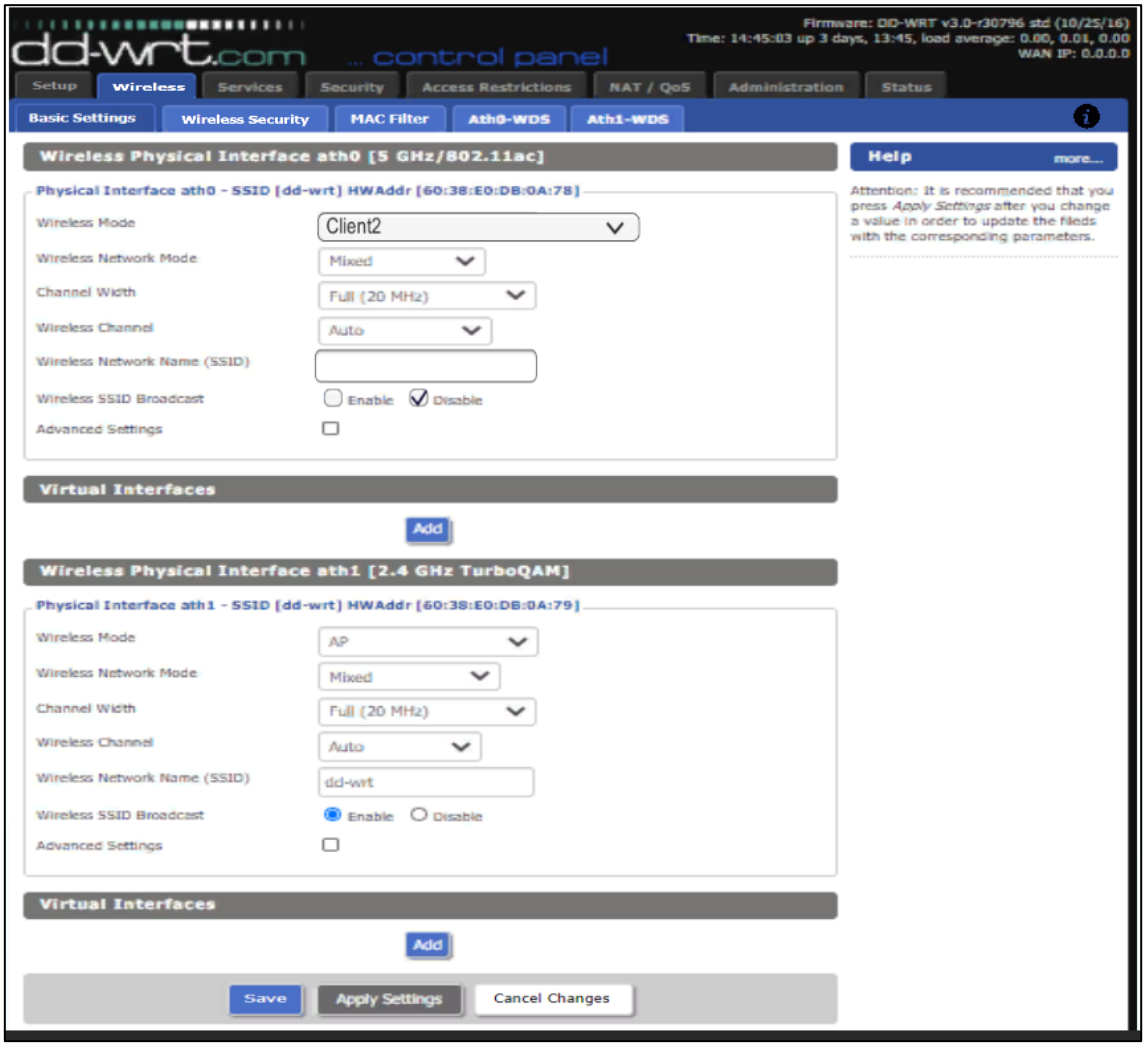

ILUSTRACIÓN 23. PANEL WIRELESS DESACTIVADO

3.2.2.4.1 Panel prefab DD-WRT

De forma análoga al prefab de Zeroshell se lleva a cabo el desarrollo de panel prefab DD-WRT. Este nos servirá para poder agrupar los elementos comunes que se comparten entre todos los paneles DD-WRT, a excepción del panel Login.

Cabe remarcar que nuestro prefab se encuentra en el ecosistema de un Canvas, de lo contrario no podríamos añadir este prefab.

Este game object estará compuesto por un game object de tipo Panel, el cual tendrá como hijo un game object de tipo Image. En esta imagen introduciremos la captura de pantalla asociada a cada uno de los paneles. Por ejemplo, en el panel Wireless

adjuntaremos la captura correspondiente en la configuración de DD-WRT donde se puede asignar el nombre al SSID de la red.

De esta imagen a su vez colgarán 3 game objects vacíos: *Wireless, Help y Configuración.* Estos tendrán cada uno asignado un hijo de tipo Button: *Botón Wireless, Botón Help y Botón Setup.* Los botones nos permiten interactuar con las funcionalidades para cada uno. En caso del botón Wireless nos redirigirá al panel Wireless con la configuración que esté guardada, mientras que el botón Configuración nos llevará al panel de configuración. El botón de ayuda será explicado con posterioridad su funcionalidad. Este botón también forma parte de los game objects del panel prefab Zeroshell.

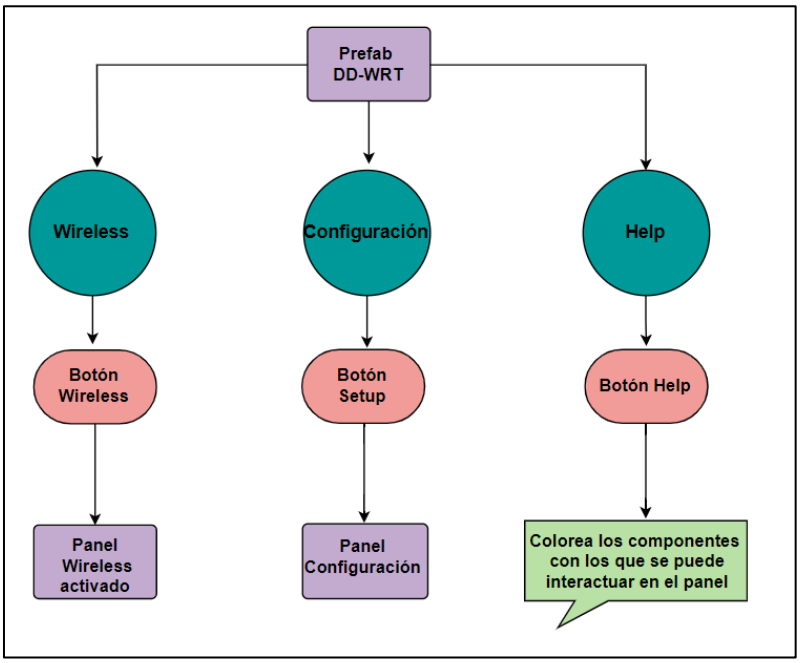

ILUSTRACIÓN 24. ESQUEMA PREFAB DD-WRT

3.2.2.4.2 Panel Login

Una vez que solicitamos en el navegador acceder a la IP donde está alojado DD-WRT, el panel Login será el que nos permita entrar al software. Este panel se encargará de comprobar que el usuario y la contraseña son los correctos.

Tendremos un game object tipo Panel, al cuál se añadirá un hijo de tipo Image. De esa imagen va a colgar toda la estructura de este panel. Este panel Login va a estar compuesto por 4 game objects: *Guardar, Ayuda, Volver y Datos Login.*

Los 3 primeros serán los objetos a los que van a pertenecer los botones para poder interactuar. En primer lugar, el botón Iniciar sesión nos permitirá comprobar que la autenticación es válida. El botón que nos permite volver al navegador será el botón Volver, que es hijo de Volver. Mientras que el botón de Ayuda se explicará más tarde. El game object datos login nos va a permitir introducir los campos para llevar a cabo la autenticación. Este objecto contará con 3 hijos: *Nombre, Contraseña y Credenciales inválidas.* Los 2 primeros son game objects de tipo Input field, que nos permitirán introducir las credenciales. En caso de que estas sean inválidas, cuando intentemos iniciar sesión nos aparecerá un texto, que corresponde con el game object Credenciales inválidas, que será de tipo Text. Por lo tanto, si no son válidas, los campos de Contraseña y Nombre se reiniciarán, para poder reintentarlo.

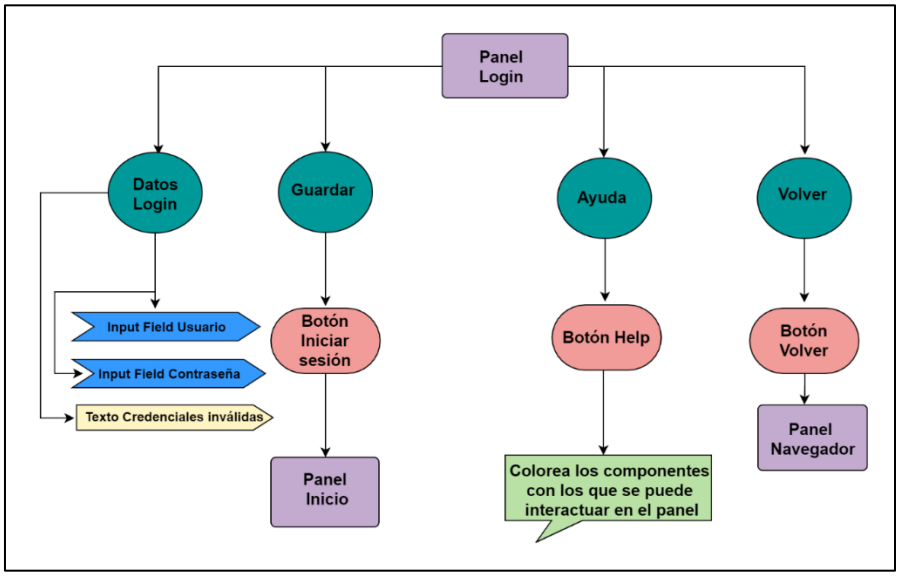

ILUSTRACIÓN 25. ESQUEMA PANEL LOGIN

3.2.2.4.3 Panel Inicio

Una vez que nos hemos autenticado, se accede al panel de control de DD-WRT, el cual está compuesto por la información sobre el sistema como el nombre del router. La forma de implementar este panel será con el uso del prefab DD-WRT. Puesto que este panel no incluye ninguna funcionalidad extra en él. Básicamente podremos acceder a los menús que son accesibles en nuestra aplicación: Configuración y Wireless.

3.2.2.4.4 Panel Configuración

Este panel está ideado con el fin de albergar la configuración del router. En nuestro caso este panel tendrá una configuración equivalente a la diseñada en el Prefab DD-WRT.

Se va a usar el prefab para generar este panel, el cuál tendremos que cambiar la imagen adjuntada al panel para que sea la del panel de configuración. Una vez dentro de la imagen, tendremos además habilitados 2 game object: Configuración básica y Volver. Ambos tienen como hijo un game object de tipo Button. El primero al ser pulsado nos carga el panel en el que estamos: panel Configuración. En caso de volver, lo que esté permitirá es salir al panel Login, para poder reiniciar los parámetros o incluso salir hacia el navegador y cambiar al software Zeroshell.

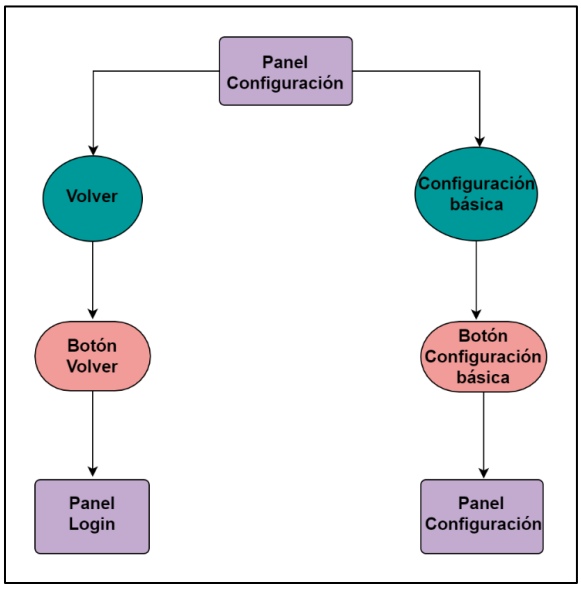

ILUSTRACIÓN 26. ESQUEMA PANEL CONFIGURACIÓN

3.2.2.4.5 Panel Wireless activado

Si lo que se pretende es configurar los parámetros de básicos de la red es necesario acceder al panel que será el mostrado por defecto al entrar en Wireless. Ya hemos dicho que vamos a usar el prefab para ayudarnos en la implementación. Así que, partiendo de él, tendremos que añadir 8 game objects: *Seguridad Wireless, Filtro MAC, Configuración Básica Wireless, Guardar, Cancelar, Faltan parámetros, Modo wireless , SSID y SSID Broadcast.*

Los 5 primeros corresponden con game objects que tendrá configurado como hijo un game object de tipo Button. Los *botones Seguridad Wireless, Filtro MAC y Configuración básica*, nos permiten interaccionar con los paneles que se pueden configurar dentro de Wireless, que permitirán configurar en cada caso los parámetros de seguridad básica, políticas de filtrado MAC y tipo de seguridad Wireless para la red.

El objecto Guardar como su nombre indica, nos permitirá almacenar los parámetros que se hayan configurado, pero sólo si se han introducido los valores propuestos en la práctica. Para lograr ese fin se añadirá al objecto un hijo *botón Guardar*. En caso de querer cancelar estos parámetros introducidos se hará uso del game object Cancelar con un hijo *botón Cancelar*, para poder reiniciar los valores del panel. Se ha explicado que no se pueden guardar los datos si no son los solicitados en la práctica, por lo tanto, se usa el game object Faltan parámetros, que tendrá un hijo de tipo Text llamado *texto Precaución.* Este se encargará de mostrar un texto de advertencia, para que se ajusten los parámetros requeridos.

El game object Broadcast SSID, estará compuesto por 2 hijos de tipo Toggle. Este tipo de objetos nos permite habilitar opciones. Los toggles son *llamados toggle SSID Broadcast ON y toggle SSID Broadcast OFF*. Con estos podremos habilitar o deshabilitar la emisión del SSID por la red. Este objecto Broadcast SSID, se encargará de configurar cuál es el panel que vamos a tener activado para la configuración de la configuración básica. Si está en activo el toggle SSID Broadcast ON nos situará en el panel que nos encontramos. En cambio, si quien se activa es toggle SSID Broadcast OFF, nos redirigirá al panel Wireless activado.

El game object SSID será de tipo Input Field, nos permitirá introducir en el campo el SSID que queramos asignarle a la red del punto de acceso.

Por último, para poder configurar el modo Wireless, usamos el game object Modo Wireless. Este objecto será padre de un game object tipo Dropdown; permite seleccionar entre varias opciones. Las opciones que se podrán seleccionar: AP, Client, Client Bridge, ADHOC, WDS Station y WDS AP.

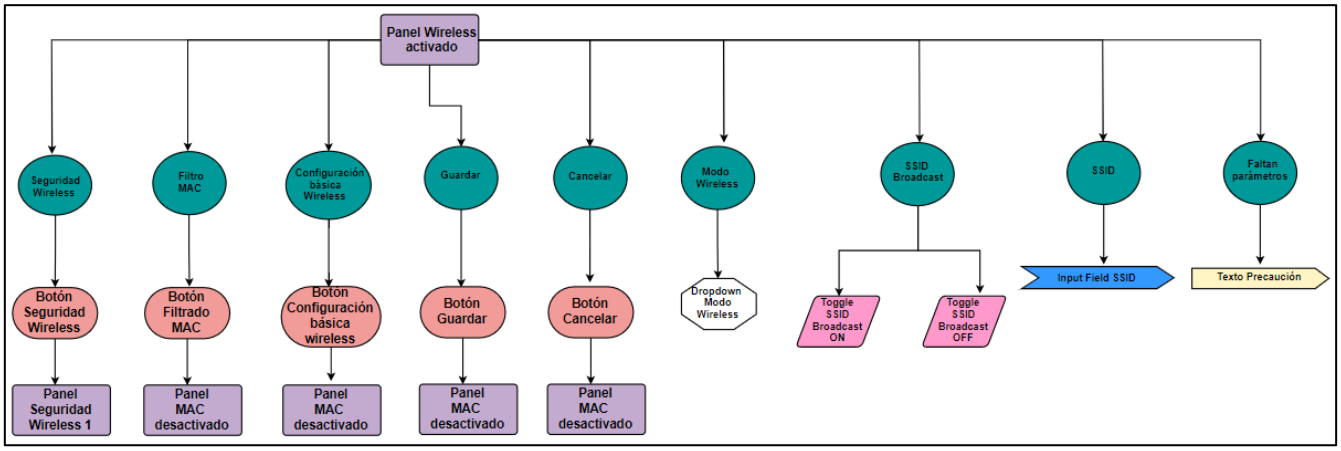

ILUSTRACIÓN 27. ESQUEMA WIRELESS ACTIVADO

3.2.2.4.6 Panel Wireless desactivado

Una vez se deshabilite el SSID Broadcast damos paso a este panel. Este panel tiene la misma finalidad que el panel anterior. En la práctica como el panel Wireless con el SSID broadcast desactivado es el que solicita configurar, nos hará permanecer en este panel por defecto cuando se accede a la configuración básica de Wireless. Este panel es idéntico al panel Wireless activado, la única diferencia es que no tendremos un texto de advertencia, porque se suponen ya configurados todos los parámetros.

3.2.2.4.7 Panel Seguridad wireless 1

Para configurar los parámetros sobre la seguridad de la red accede a este panel Seguridad Wireless.

Como en el resto, está añadido el prefab DD-WRT. Al cuál se añadirán 6 game objects: *Seguridad Wireless, Filtro MAC, Configuración básica, Guardar, Modo seguridad y Precaución.* Los 3 primeros se configuran igual que para los paneles de Seguridad activado/desactivado. En el game object Guardar tendremos configurado un hijo de tipo Button que permitirá guardar la configuración de seguridad seleccionada. El *botón Guardar*, no guardará los cambios a no ser que se seleccione el requisito de prácticas, que será WPA2 Enterprise Mixed. Para ese caso, se recibirá una notificación que es implementada en el game object Precaución. El cuál contará con un hijo de tipo Text, el cuál mostrará una advertencia.

El game object Modo seguridad, será el que nos permita seleccionar la seguridad adecuada para la red. Por lo tanto, se emplea un game object de tipo Dropdown, que será hijo de Modo seguridad. Este *dropdown Modo seguridad* podrá tener los siguientes valores: WPA personal, WPA Enterprise, WPA2 Personal, WPA2 Enterprise, WPA2 Personal mixed, RADIUS Y WEP.

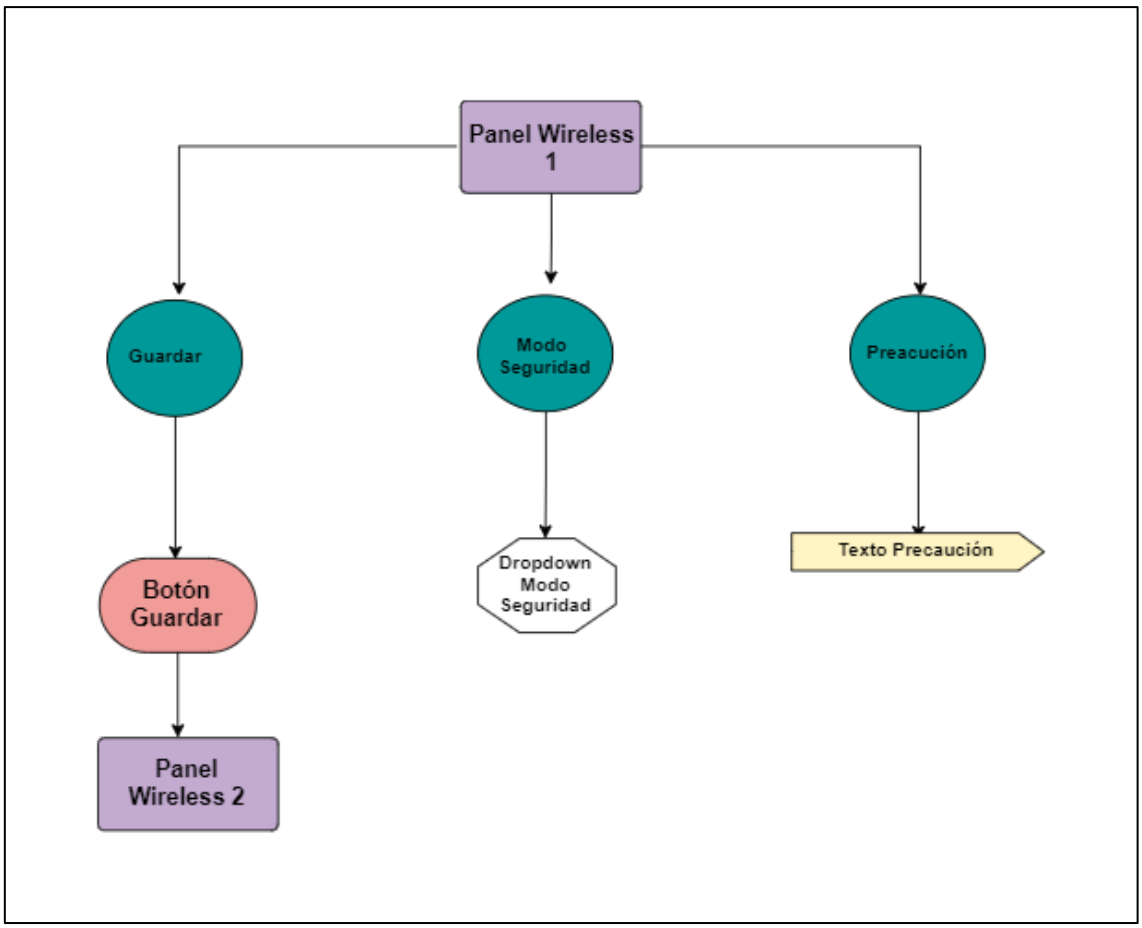

ILUSTRACIÓN 28. ESQUEMA PANEL WIRELESS 1

3.2.2.4.8 Panel Seguridad wireless 2

Una vez seleccionado el modo de seguridad que se requiere en nuestra red pasamos al panel de configurar los parámetros para esa red.

Este panel estará compuesto por 6 game objects: *Seguridad Wireless, Filtro MAC, Configuración básica, Guardar, Algoritmos WPA y Dirección servidor.* El game object Guardar, ya lo conocemos de secciones anteriores y tendrá la misma funcionalidad y

componentes. Para poder configurar el algoritmo de cifrado de WPA haremos uso del game object Algoritmos WPA, al que se adjuntará un hijo de tipo Dropdown. El game object *dropdown WPA algoritmos*, podrá tener como posibles valores: AES, TKIP + AES y TKIP.

Por otro lado, para poder asignar la dirección del servidor, se añade de forma manual mediante el game object Dirección servidor. A su vez este es padre de un game object Input Field llamado Dirección RADIUS, donde se introduce la dirección asociada al servidor.

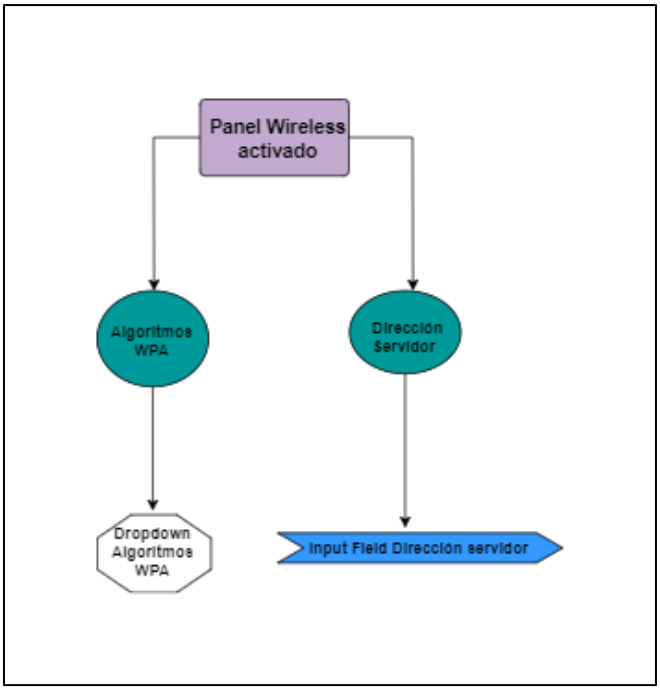

ILUSTRACIÓN 29. ESQUEMA PANEL WIRELESS 2

3.2.2.4.9 Panel MAC desactivado

Dentro de las posibles configuraciones que podemos realizar a la red Wireless, encontramos el poder realizar el filtrado por dirección MAC. Para eso nos servirá este nuevo panel.

Como ya se viene explicando, tomamos el prefab DD-WRT como referencia y a partir de ahí añadimos nuevos objetos: *Seguridad Wireless, Filtro MAC, Configuración básica, Guardar y Usar Filtro MAC.* Los 4 primeros son análogos al resto de paneles.

Nos centraremos en cómo se implementa este nuevo game object Usar Filtro MAC, será el encargado de controlar si se aplica filtrado por dirección MAC en nuestra red inalámbrica. Para realizar la implementación se añadirán 2 hijos: *Toggle Usar filtro ON y Toggle Usar filtro OFF.* Por defecto estará activo el toggle de usar filtro OFF, una vez que se pasa a activar el toggle de Usar filtro ON cambiaremos al panel MAC activado.

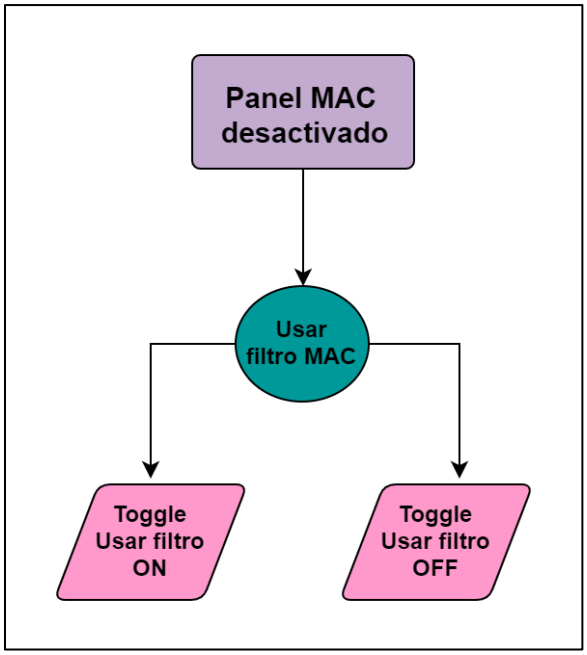

ILUSTRACIÓN 30. ESQUEMA PANEL MAC DESACTIVADO.

3.2.2.4.10 Panel MAC activado

Si lo que se pretende es activar el filtrado por MAC, entraremos en este panel para poder seleccionar el tipo de filtrado que se va a hacer sobre las direcciones.

Con el fin de implementarlo contamos con los game objects: *Seguridad Wireless, Filtro MAC, Configuración básica, Guardar, Usar Filtro MAC y Modo Filtrado.*

Ya hemos dicho cuál es el fin de este panel, así que el game object Modo Filtrado será el encargado de alojar esta funcionalidad. Tendremos dos posibilidades en el modo de filtrado, por una parte, podremos prevenir el acceso a clientes que estén en una lista de acceso o permitir solo el acceso de los que estén en la lista de acceso. Estas son implementadas mediante 2 game objects que son hijos de Modo Filtrado llamados: *toggle Permitir y toggle Prevenir*.

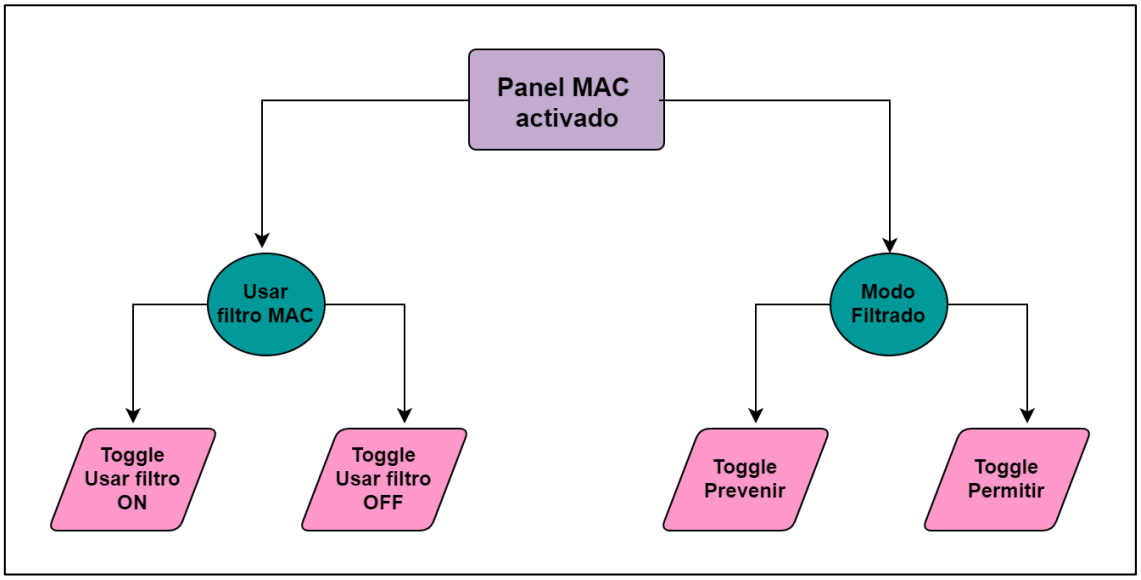

ILUSTRACIÓN 31. ESQUEMA PANEL MAC ACTIVADO

3.2.3 Ayuda

En caso de que el alumno tenga alguna duda a la hora de realizar la práctica con la aplicación se ha incorporado un botón de ayuda representado mediante el icono: ❶.

Este botón aparecerá en cada uno de los paneles configurados anteriormente. Cada vez que un botón de Ayuda sea pulsado se generará un evento OnClick, que llamará a la función *Info()* del script ControladorPrincipal.cs dónde se ejecutará el código asociado a la función.

El botón de Ayuda sombreará en verde todos los campos con los que el usuario pueda interaccionar en el panel. Los game objects interaccionables son Input fields, Buttons, Toggles y Dropdowns. Después de dos segundos se llamará a la función *ResetToWhite()* que reiniciará el color de los game objects a su color por defecto.

```
public void info()
{
    buttons = GetComponentsInChildren<Button>();
    inputs = GetComponentsInChildren<InputField>();
    toggles = GetComponentsInChildren<Toggle>();
    dropdowns = GetComponentsInChildren<Dropdown>();
    foreach (Button button in buttons)
    {
        button.GetComponent<Image>().color = Color.green;
        Invoke("ResetToWhite", 2f);
    }
    foreach (InputField inputf in inputs)
    {
        inputf.GetComponent<Image>().color = Color.green;
        Invoke("ResetToWhite", 2f);
    }

    foreach(Toggle tgl in toggles)
    {
        tgl.GetComponent<Image>().color = Color.green;
        Invoke("ResetToWhite", 2f);
    }

    foreach(Dropdown dd in dropdowns)
    {
        dd.GetComponent<Image>().color = Color.green;
        Invoke("ResetToWhite", 2f);
    }

}
```

ILUSTRACIÓN 32. FUNCIÓN INFO()

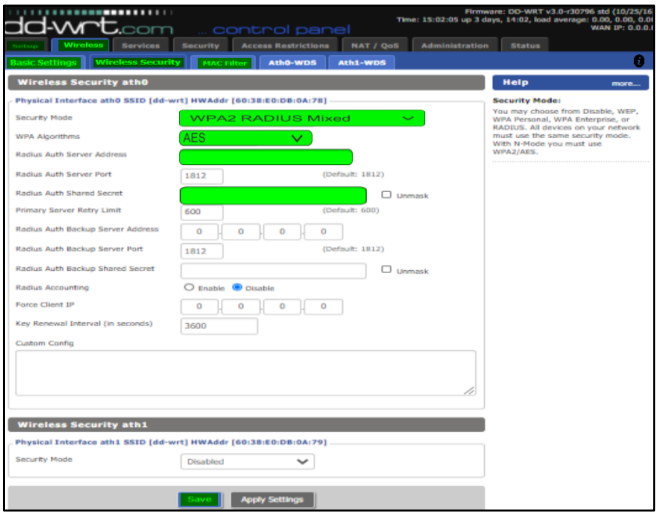

ILUSTRACIÓN 33. PANEL CON AYUDA

3.2.4 Volver

La aplicación está creada de tal forma que permita volver atrás para intercambiar entre los paneles de control de Zeroshell y DD-WRT. Es decir, que una vez entramos en cualquiera de estos paneles podremos volver atrás y acceder al navegador para poder

intercambiar entre la configuración del servidor RADIUS o del AP. El icono que representa volver es: ←.

El botón Volver estará implementado en el Navegador. Al ser pulsado generará un evento Onclick() que invocará a la función *Back()* nos llevará al inicio de la aplicación. También encontremos el botón en los paneles de inicio y de login de DD-WRT y Zeroshell, que invocarán a la función *BackNavigator().*

```
private void Back()
{
    Navegador.SetActive(false);
    Start_.SetActive(true);
}

4 referencias
private void BackNavigator()
{
    Zeroshell.SetActive(false);
    DDWRT.SetActive(false);
    Navegador.SetActive(true);
}
```

ILUSTRACIÓN 34. FUNCIÓN BACK() Y BACKNAVIGATOR()

CAPÍTULO 4. EJEMPLO DE USO

Una vez desarrollada la aplicación para los alumnos es hora de conocer su funcionamiento. Vamos a hacer un recorrido por la aplicación para saber cómo utilizarla, para poder configurar los mecanismos de seguridad.

4.1 Inicio

En el punto de partida el alumno se encontrará con un panel de Inicio para arrancar la aplicación. Una vez arrancada, nos situamos en el panel Navegador, a través de la barra de direcciones introduciremos la IP correspondiente para cada configuración.

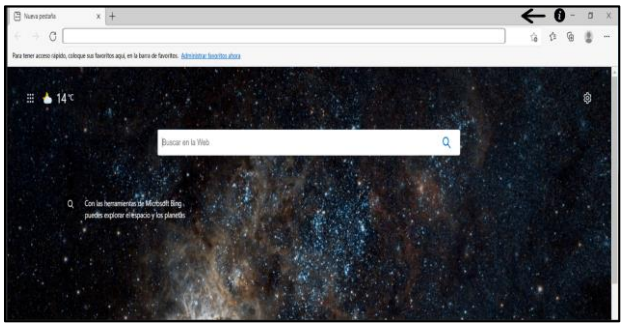

ILUSTRACIÓN 35. PANEL NAVEGADOR

4.2 Configuración del router como punto de acceso

Accederemos mediante “192.168.0.1” a configurar el panel de control DD-WRT. El primer paso, es autenticarse con usuario y contraseña: “admin”.

Para cambiar el SSID de la red entramos en Wireless → Basic Settings y actualizamos el valor del SSID. Pulsamos Save para guardar los cambios.

Con el fin de conseguir establecer la seguridad mediante WPA2 Enterprise se accede a Wireless → Wireless Security. Dentro del menú, seleccionamos el modo de seguridad a WPA2 RADIUS Mixed. La página se actualizará mostrando todos los campos que incluye el protocolo. Habrá que modificar el campo de tipo de algoritmo WPA donde se selecciona TKIP+AES. Además, en el campo Radius Auth Server Address hay que introducir la IP de la máquina correspondiente a Zeroshell. Introduciremos una

contraseña, para que puedan comunicarse el servidor y el AP en el campo Shared Secret. Por último, pulsar sobre Save para guardar la configuración.

Una vez introducimos los parámetros de seguridad en el AP, nos dirigimos a Setup y pulsamos sobre el botón Volver, que nos llevará al panel del navegador.

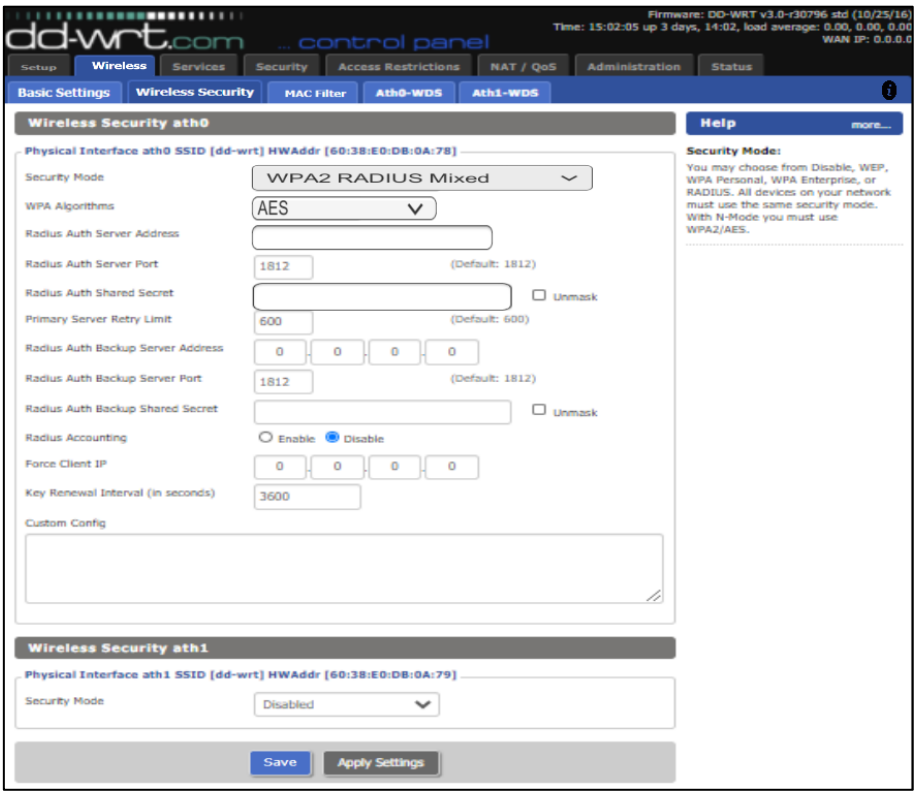

ILUSTRACIÓN 36. MENÚ CONFIGURACIÓN WPA2 ENTERPRISE

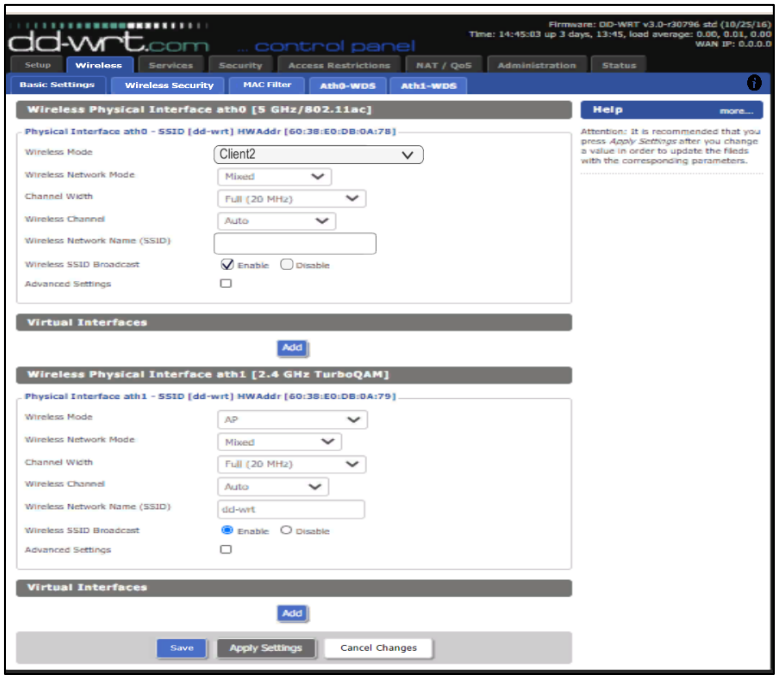

ILUSTRACIÓN 37. MENÚ CONFIGURACIÓN PUNTO DE ACCESO

4.3 Configuración servidor RADIUS con Zeroshell

De nuevo nos situamos en el Navegador donde introducimos la IP “192.168.0.2” para acceder a configurar Zeroshell. Una vez dentro, hay que autenticarse con usuario: “admin” y contraseña “zeroshell”.

Para configurar el servidor RADIUS, es necesario activarlo y comprobar su certificado. En el menú lateral ir a RADIUS y habilitar Enabled y CheckCRL, más tarde hacer click en Save.

El servidor RADIUS debe configurar los clientes que se van a conectar a él, en nuestro caso es el AP que acabamos de configurar. De este modo dentro del menú de RADIUS vamos al menú superior y hacemos click sobre Authorized Clients. En este panel introduciremos los valores de la IP y máscara , nombre AP y Shared Secret. Para añadir este nuevo usuario pulsamos sobre el botón +. Si queremos salir hacemos click el botón Close y volvemos al panel de RADIUS.

La configuración de los usuarios que podrán conectarse a la red, se almacenará en el servidor. En el menú lateral derecho seleccionamos Users y luego en ese menú hacemos click en Add. En este panel introduciremos los campos de Username, Password, Lastname, Firstname y Email.

Para abandonar la configuración de Zeroshell, en el menú lateral pulsamos sobre Setup y botón Volver.

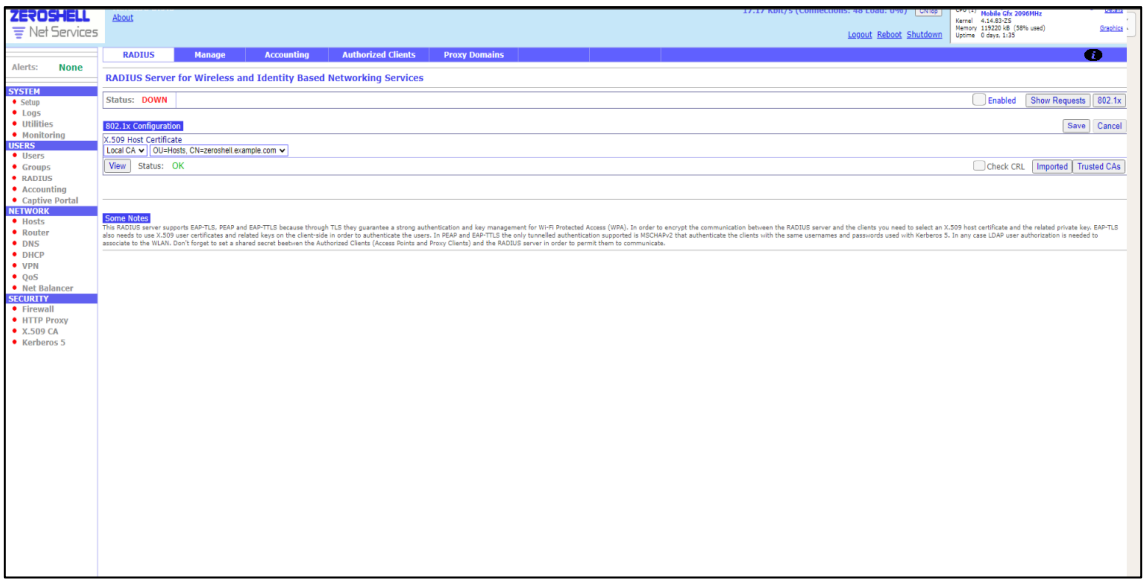

ILUSTRACIÓN 38. CONFIGURACIÓN SERVIDOR RADIUS

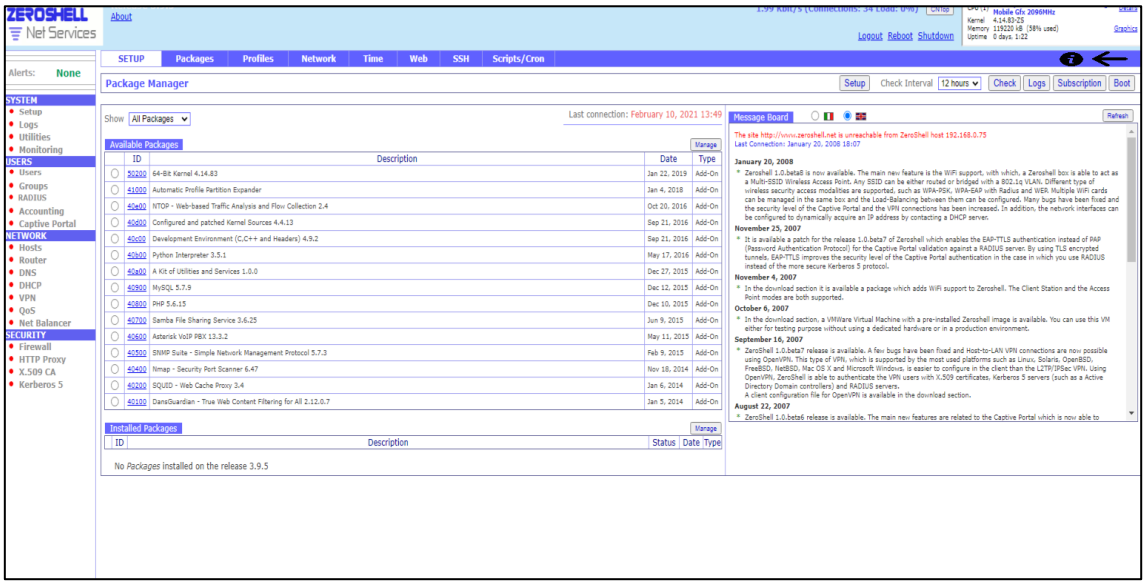

ILUSTRACIÓN 39. MENÚ CONFIGURACIÓN ZEROSHELL

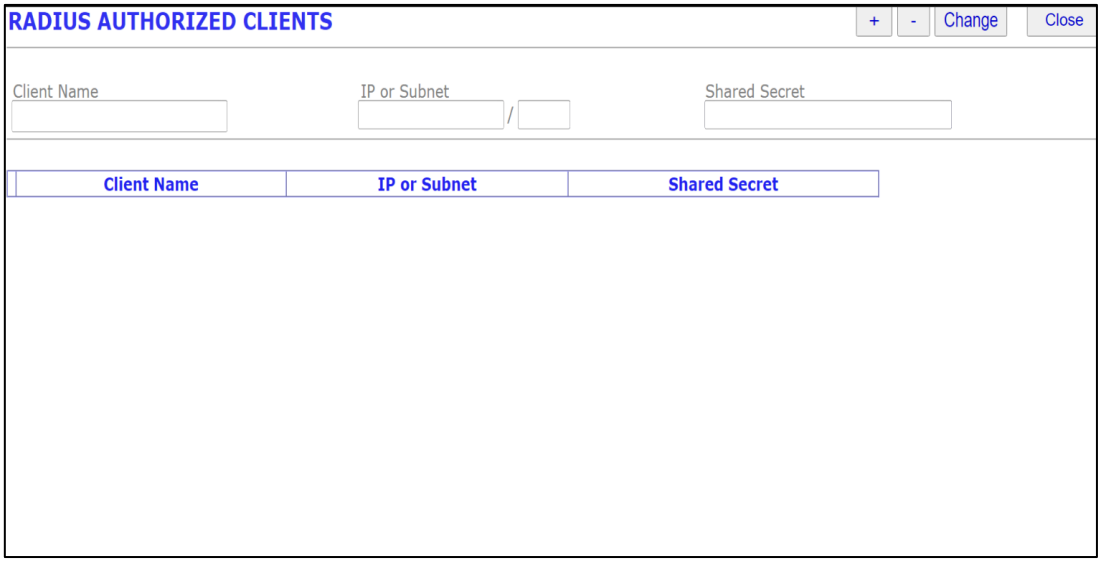

ILUSTRACIÓN 40. MENÚ CONFIGURACIÓN CLIENTES RADIUS

4.4 Otras configuraciones

Para proporcionar mayor seguridad en se puede implementar la técnica de filtrado MAC, evitando el acceso del resto de direcciones. Para ello hacemos click en Wireless → MAC Filter, donde activaremos Use Filter y marcamos en Filter Mode "Permit only clients listed to Access the wireless network".

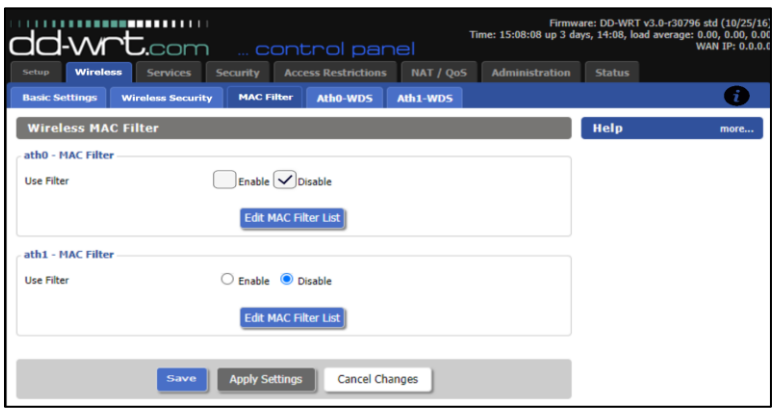

ILUSTRACIÓN 41. PANEL FILTRADO MAC

CAPÍTULO 5. CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

Con la realización del proyecto, se han conseguido adquirir amplios conocimientos en Unity y complementar los que ya tenía en seguridad de redes Wifi. Gracias a esta combinación de tecnologías, he creado una aplicación que permitirá a los alumnos realizar la práctica sobre seguridad de redes Wifi, configurando el protocolo WPA2 Enterprise.

Se ha simplificado la configuración de diferentes dispositivos, que han quedado unificados en una pieza de software, de forma que pueda ser usado en cualquier momento y en cualquier lugar, dando más flexibilidad a la práctica ya existente. Puesto que se podrá realizar de forma on-line o en el laboratorio sin necesidad de equipos adicionales.

Por otra parte, la práctica cuenta con la ventaja de que puede ser usada de forma individual. Permite a cada alumno, beneficiarse de configurar todos los mecanismos para implementar el protocolo de seguridad, o aplicar protección en la red mediante el filtrado de direcciones MAC.

5.2 Líneas futuras

De forma que pueda ampliarse el proyecto, aquí se recoge toda la información para poder seguir su implementación en Unity. Una posible mejora, es recrear mediante un entorno en 3D el laboratorio con los equipos, con el fin de poder realizar una experiencia más inmersiva. El interfaz se podría mostrar en un equipo del laboratorio, incluyendo la posibilidad de ver y manipular el entorno 3D con los equipos y cables que se conectan para realizar la configuración.

El desarrollo de entornos virtuales está creciendo de forma exponencial. Una tecnología muy interesante es la realidad virtual basada en marcadores, con la que se pueden reconocer marcadores del mundo real y proyectar un mundo virtual a través de la pantalla del móvil. Esto puede realizarse con Vuforia. En el proyecto podría implementarse esta herramienta para mejorarlo, de forma que cuando el alumno enfocase a un código QR o imagen predefinida(marcador), se mostrará la interfaz gráfica correspondiente al equipo.

ANEXOS

ControladorPrincipal.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ControladorPrincipal : MonoBehaviour
{
    public GameObject Start_;
    public GameObject Navegador;
    public GameObject DDWRT;
    public GameObject Zeroshell;

    public GameObject PanelLoginDDWRT;
    public GameObject PanelLoginZS;
    public GameObject PanelSetupDDWRT;
    public GameObject PanelSetupzS;

    public InputField IP;
    public Button StartBtn;
    public Button BtnBackStart;
    public Button BtnBackZeroshellNav;
    public Button BtnBackDDWRTNav;
    public Button BtnBackZeroshellLogin;
    public Button BtnBackDDWRTLogin;



    private GameObject[] InfoBtns;
    private Button[] buttons;
    private InputField[] inputs;
    private Toggle[] toggles;
    private Dropdown[] dropdowns;

    void Start()
    {
        Start_.SetActive(true);
        Listener();
    }

    public void Listener()
    {
        IP.onEndEdit.AddListener(checkIP);
        StartBtn.onClick.AddListener(NavPanel);
        BtnBackStart.onClick.AddListener(Back);
        BtnBackDDWRTNav.onClick.AddListener(BackNavigator);
        BtnBackZeroshellNav.onClick.AddListener(BackNavigator);
        BtnBackZeroshellLogin.onClick.AddListener(BackNavigator);
        BtnBackDDWRTLogin.onClick.AddListener(BackNavigator);
    }

    private void checkIP(string ip)
    {
        if (ip.Equals("192.168.0.1"))
        {
            Navegador.SetActive(false);
```

```
            DDWRT.SetActive(true);
            PanelLoginDDWRT.SetActive(true);
        }
        else if (ip.Equals("192.168.0.2"))
        {
            Navegador.SetActive(false);
            Zeroshell.SetActive(true);
            PanelLoginZS.SetActive(true);
        }
    }

    private void NavPanel()
    {
        Navegador.SetActive(true);
        Start_.SetActive(false);
    }

    public void info()
    {
        buttons = GetComponentsInChildren<Button>();
        inputs = GetComponentsInChildren<InputField>();
        toggles = GetComponentsInChildren<Toggle>();
        dropdowns = GetComponentsInChildren<Dropdown>();
        foreach (Button button in buttons)
        {
            button.GetComponent<Image>().color = Color.green;
            Invoke("ResetToWhite", 2f);
        }
        foreach (InputField inputf in inputs)
        {
            inputf.GetComponent<Image>().color = Color.green;
            Invoke("ResetToWhite", 2f);
        }

        foreach(Toggle tgl in toggles)
        {
            tgl.GetComponent<Image>().color = Color.green;
            Invoke("ResetToWhite", 2f);
        }

        foreach(Dropdown dd in dropdowns)
        {
            dd.GetComponent<Image>().color = Color.green;
            Invoke("ResetToWhite", 2f);
        }

    }
    private void ResetToWhite()
    {
        foreach (Button button in buttons)
        {
            button.GetComponent<Image>().color = Color.white;
        }

        foreach (InputField inputf in inputs)
        {
            inputf.GetComponent<Image>().color = Color.white;

        }

        foreach (Toggle tgl in toggles)
```

```
        {
            tgl.GetComponent<Image>().color = Color.white;
        }

        foreach (Dropdown dd in dropdowns)
        {
            dd.GetComponent<Image>().color = Color.white;
        }
    }

    private void Back()
    {
        Navegador.SetActive(false);
        Start_.SetActive(true);
    }

    private void BackNavigator()
    {
        Zeroshell.SetActive(false);
        DDWRT.SetActive(false);
        Navegador.SetActive(true);
    }


}
```

ControladorDDWRT.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class ControladorDDWRT : MonoBehaviour
{

    private GameObject[] DDWRTPanels;

    public GameObject panelLogin;
    public GameObject panelMain;
    public GameObject panelSetup;
    public GameObject panelWirelessONSSID;
    public GameObject panelWirelessOFFSSID;
    public GameObject panelWirelessSecurity1;
    public GameObject panelWirelessSecurity2;
    public GameObject panelMACEn;
    public GameObject panelMACDis;


    public Button btnSaveLoginPL;
    public InputField ifNamePL, ifPwdPL;
    public Text txtInvalidCredentialsPL;

    private string password = "admin";
    private string username = "admin";
    private string password2, name2;
    private bool isLogged = false;

    public Button btnWirelessONPM, btnSetupOffPM;

    public Button btnWirelessOFFPS, btnSetupONPS, btnSavePS, btnBasicSettingPS;
```

```
    public     Button     btnWirelessONPWON,     btnSetupOFFPWON,     btnSavePWON,
btnWirelessSecOFFPWON, btnMACOFFPWON, btnCancelPWON, btnBasicSettingsPWON;
    public Toggle tglSSIDONPWON, tglSSIDOFFPWON;
    public Text txtCautionPWONM;
    public Dropdown ddWirelessModePWON;
    public InputField ifSSID;

    private string SSID_val;
    private bool setSSID = false;
    private bool isAP = false;
    private bool isBroadcasted = true;

    public    Button    btnWirelessONPWOFF,    btnSetupOFFPWOFF,    btnSavePWOFF,
btnWirelessSecOFFPWOFF, btnBasicSettingsPWOFF, btnMACOFFPWOFF, btnCancelPWOFF ;
    public Toggle tglSSIDONPWOFF, tglSSIDOFFPWOFF;
    public Text txtCautionPWOFF;
    public Dropdown ddWirelessModePWOFF;
    public InputField ifSSIDOFF;

    private bool isWirelessOFF = false;

    public     Button     btnWirelessONPWS1,     btnSetupOFFPWS1,     btnSavePWS1,
btnWirelessSecOnPWS1, btnMACOFFPWS1, btnBasicSettingsPWS1;
    public Dropdown ddSecurityModePWS1;
    public Text txtCautionPWS1;

    private bool isWPA2Mixed = false;


    public     Button     btnWirelessONPWS2,     btnSetupOFFPWS2,     btnSavePWS2,
btnWirelessSecOnPWS2, btnMACOFFPWS2, btnBasicSettingsPWS2;
    public Dropdown ddSecurityModePWS2, ddWPAAlgorithm;
    public InputField ifAddress, ifClave;
    public Text addressText;

    private string addressRadiusS;
    private string clave_val;
    private bool isWS2 = false;
    private bool isTKIPAES = false;

    public Button btnWirelessOFFPWSMACDIS, btnSetupOFFPWSMACDIS, btnSavePWSMACDIS,
btnWirelessSecOFFPWSMACDIS, btnMACONPWSMACDIS, btnBasicSettingsPWSMACDIS;
    public Toggle tglUseFilterDisPWSMACDIS, tglUseFilterEnPWSMACDIS;

    private bool  isMACFilter = false;


    public  Button  btnWirelessOFFPWSMACEN,  btnSetupOFFPWSMACEN,  btnSavePWSMACEN,
btnWirelessSecOFFPWSMACEN, btnMACONPWSMACEN, btnBasicSettingsPWSMACEN;
    public      Toggle      tglUseFilterEnPWSMACEN,      tglUseFilterDisPWSMACEN,
tglPreventClientPWSMACEN, tglPermitClientPWSMACEN;


    private bool isMACFilter2 = false;
    private bool isPrevent = true;
    private bool isMACDis = true;



    void Start()
```

```
    {
        panelLogin.SetActive(true);
        Listener();
    }

    void Update()
    {
        DDWRTPanels = GameObject.FindGameObjectsWithTag("PanelesDDWRT");

    }



    private void Listener()
    {
        btnSaveLoginPL.onClick.AddListener(SubmitData);
        ifNamePL.onEndEdit.AddListener(SaveName);
        ifPwdPL.onEndEdit.AddListener(SavePassword);

        btnWirelessONPM.onClick.AddListener(delegate { changeTopanel(3); });
        btnSetupOffPM.onClick.AddListener(delegate { changeTopanel(2); });

        btnWirelessOFFPS.onClick.AddListener(delegate { changeTopanel(3); });
        btnSetupONPS.onClick.AddListener(delegate { changeTopanel(2); });
        btnSavePS.onClick.AddListener(delegate { changeTopanel(2); });
        btnBasicSettingPS.onClick.AddListener(delegate { changeTopanel(2); });


        btnWirelessONPWON.onClick.AddListener(delegate { changeTopanel(3); });
        btnSetupOFFPWON.onClick.AddListener(delegate { changeTopanel(2); });
        btnSavePWON.onClick.AddListener(SaveWireless);
        btnWirelessSecOFFPWON.onClick.AddListener(delegate { changeTopanel(5); });
        btnMACOFFPWON.onClick.AddListener(delegate { changeTopanel(8); });
        btnCancelPWON.onClick.AddListener(delegate { changeTopanel(3); });
        btnBasicSettingsPWON.onClick.AddListener(delegate { changeTopanel(3); });
        tglSSIDONPWON.onValueChanged.AddListener(delegate { BroadcastON(); });
        tglSSIDOFFPWON.onValueChanged.AddListener(delegate { BroadcastOFF(); });
        ddWirelessModePWON.onValueChanged.AddListener(delegate  {  WirelessMode();
});
        ifSSID.onEndEdit.AddListener(SaveSSID);


        btnWirelessONPWOFF.onClick.AddListener(delegate { changeTopanel(3); });
        btnSetupOFFPWOFF.onClick.AddListener(delegate { changeTopanel(2); });
        btnSavePWOFF.onClick.AddListener(SaveWireless);
        btnWirelessSecOFFPWOFF.onClick.AddListener(delegate  {  changeTopanel(5);
});
        btnMACOFFPWOFF.onClick.AddListener(delegate { changeTopanel(8); });
        btnCancelPWOFF.onClick.AddListener(delegate { changeTopanel(3); });
        btnBasicSettingsPWOFF.onClick.AddListener(delegate { changeTopanel(3); });
        tglSSIDONPWOFF.onValueChanged.AddListener(delegate { BroadcastON(); });
        tglSSIDOFFPWOFF.onValueChanged.AddListener(delegate { BroadcastOFF(); });
        ddWirelessModePWOFF.onValueChanged.AddListener(delegate { WirelessMode();
});
        ifSSIDOFF.onEndEdit.AddListener(SaveSSID);


        btnWirelessONPWS1.onClick.AddListener(delegate { changeTopanel(5); });
        btnSetupOFFPWS1.onClick.AddListener(delegate { changeTopanel(2); });
        btnSavePWS1.onClick.AddListener(SaveWS1);
```

```
        btnWirelessSecOnPWS1.onClick.AddListener(delegate { changeTopanel(5); });
        btnMACOFFPWS1.onClick.AddListener(delegate { changeTopanel(8); });
        btnBasicSettingsPWS1.onClick.AddListener(delegate { changeTopanel(3); });
        ddSecurityModePWS1.onValueChanged.AddListener(delegate  {  SecurityMode();
});



        btnWirelessONPWS2.onClick.AddListener(delegate { changeTopanel(5); });
        btnSetupOFFPWS2.onClick.AddListener(delegate { changeTopanel(2); });
        btnSavePWS2.onClick.AddListener(SaveWS2);
        btnWirelessSecOnPWS2.onClick.AddListener(delegate { changeTopanel(5); });
        btnMACOFFPWS2.onClick.AddListener(delegate { changeTopanel(8); });
        btnBasicSettingsPWS2.onClick.AddListener(delegate { changeTopanel(3); });
        ddSecurityModePWS2.onValueChanged.AddListener(delegate  {  SecurityMode();
});
        ddWPAAlgorithm.onValueChanged.AddListener(delegate { Algorithm(); });
        ifAddress.onEndEdit.AddListener(SaveAddress);
        ifClave.onEndEdit.AddListener(SaveSecret);


        btnWirelessOFFPWSMACDIS.onClick.AddListener(delegate  {  changeTopanel(4);
});
        btnSetupOFFPWSMACDIS.onClick.AddListener(delegate { changeTopanel(3); });
        btnSavePWSMACDIS.onClick.AddListener(delegate { SaveMACDis(); });
        btnWirelessSecOFFPWSMACDIS.onClick.AddListener(delegate                 {
changeTopanel(5); });
        btnMACONPWSMACDIS.onClick.AddListener(delegate { changeTopanel(8); });
        btnBasicSettingsPWSMACDIS.onClick.AddListener(delegate                  {
changeTopanel(3); });
        tglUseFilterEnPWSMACDIS.onValueChanged.AddListener(delegate             {
UseFilterEn(); });
        tglUseFilterDisPWSMACDIS.onValueChanged.AddListener(delegate            {
UseFilterDis(); });



        btnWirelessOFFPWSMACEN.onClick.AddListener(delegate  {  changeTopanel(5);
});
        btnSetupOFFPWSMACEN.onClick.AddListener(delegate { changeTopanel(3); });
        btnSavePWSMACEN.onClick.AddListener(delegate { SaveMACEn(); });
        btnWirelessSecOFFPWSMACEN.onClick.AddListener(delegate                  {
changeTopanel(5); });
        btnMACONPWSMACEN.onClick.AddListener(delegate { changeTopanel(8); });
        btnBasicSettingsPWSMACEN.onClick.AddListener(delegate { changeTopanel(3);
});
        tglUseFilterEnPWSMACEN.onValueChanged.AddListener(delegate              {
UseFilterEn2(); });
        tglUseFilterDisPWSMACEN.onValueChanged.AddListener(delegate             {
UseFilterDis2(); });
        tglPreventClientPWSMACEN.onValueChanged.AddListener(delegate { Prevent();
});
        tglPermitClientPWSMACEN.onValueChanged.AddListener(delegate  {  Permit();
});


    }


    private void changeTopanel(int menuID)
```

```
{
    foreach (GameObject panel in DDWRTPanels)
    {
        panel.gameObject.SetActive(false);
    }

    switch (menuID)
    {
        case 0:
            panelLogin.gameObject.SetActive(true);
            break;
        case 1:
            panelMain.gameObject.SetActive(true);
            break;
        case 2:
            panelSetup.gameObject.SetActive(true);
            break;
        case 3:
            if (isWirelessOFF == true)
            {
                panelWirelessOFFSSID.gameObject.SetActive(true);
            }
            else
            {
                panelWirelessONSSID.gameObject.SetActive(true);
            }
            break;
        case 4:
            panelWirelessOFFSSID.gameObject.SetActive(true);
            break;
        case 5:
            Debug.Log(isWS2);
            if (isWS2 == true)
            {
                panelWirelessSecurity2.gameObject.SetActive(true);
            }
            else
            {
                panelWirelessSecurity1.gameObject.SetActive(true);
            }
            break;
        case 6:
            panelWirelessSecurity2.gameObject.SetActive(true);
            break;
        case 7:
            Debug.Log("seleccionando panale" + panelMACEn);
            panelMACEn.gameObject.SetActive(true);
            break;
        case 8:
            Debug.Log("seleccionando panale is MAC DIS"+isMACDis);
            if (isMACDis == false)
            {
                panelMACEn.gameObject.SetActive(true);
            }
            else
            {
                panelMACDis.gameObject.SetActive(true);
            }

            break;
        default:
```

```
            break;
    }
}



private void SaveName(string name)
{
    if (name == username)
    {
        name2 = name;
        CheckLogin();
    }
    else
    {
        name2 = "";
    }

}

private void SavePassword(string pwd)
{
    if (pwd == password)
    {
        password2 = pwd;
        CheckLogin();
    }
    else
    {
        password2 = "";
    }
}

private void CheckLogin()
{
    if(password2 == password  && name2 == username )
    {
        isLogged = true;
    }

}

private void SubmitData()
{
    if (isLogged == true)
    {
        changeTopanel(1);
    }
    else
    {
        txtInvalidCredentialsPL.text = "Usuario y/o contraseña inválidos";
        ifNamePL.text = "";
        ifPwdPL.text = "";
        Invoke("ResetText", 5f);
    }

}

private void ResetText()
{
```

```
        txtInvalidCredentialsPL.text = "";
    }



    private void WirelessMode()
    {
        Debug.Log("Valor   wireless"   +   ddWirelessModePWON.value   +   "y"   +
ddWirelessModePWOFF.value);
        if ( ddWirelessModePWON.value == 1 || ddWirelessModePWOFF.value == 1)
        {
            isAP = true;
        }
    }

    private void BroadcastON()
    {
        isBroadcasted = true;
        tglSSIDOFFPWON.isOn = false;
    }

    private void BroadcastOFF()
    {
        isBroadcasted = false;
        tglSSIDONPWON.isOn = false;
    }

    private void SaveSSID(string ssid)
    {
        Debug.Log("Guarda ssid"+ssid);
        SSID_val = ssid;
        setSSID = true;
        ifSSID.text = SSID_val;
        ifSSIDOFF.text = SSID_val;
    }

    private void SaveWireless()
    {
        Debug.Log(SSID_val);
        Debug.Log(isAP);
        Debug.Log(setSSID);
        if (isAP == true && setSSID == true)
        {
            if (isBroadcasted == true)
            {
                ddWirelessModePWON.value = 1;
                isWirelessOFF = false;
                changeTopanel(3);
                txtCautionPWONM.text = "";
            }
            else
            {
                ddWirelessModePWOFF.value = 1;
                changeTopanel(3);
            }
        }
        else
        {
            txtCautionPWONM.text = "Debes establecer los parámetros necesarios.";

        }
```

```
}

private void CancelWireless()
{
    isBroadcasted = true;
    isAP = false;
    isWirelessOFF = false;
    txtCautionPWONM.text = "";
    txtCautionPWOFF.text = "";
    ifSSID.text = "";
}



private void SaveWS1()
{
    if (isWPA2Mixed == true)
    {
        changeTopanel(6);
        txtCautionPWS1.text = "";
    }
    else
    {
        txtCautionPWS1.text = "Debe seleccionar los parámetros requeridos";
    }
}

private void SecurityMode()
{
    if (ddSecurityModePWS1.value == 5 || ddSecurityModePWS2.value == 0)
    {
        isWPA2Mixed = true;
    }
}



private void SaveAddress(string address)
{
    addressRadiusS = address;
}

private void SaveSecret(string secret)
{
    clave_val = secret;
}

private void Algorithm()
{
    if ( ddWPAAlgorithm.value == 1)
    {
        isTKIPAES = true;
    }
}
private void SaveWS2()
{
    Debug.Log("add:" + addressRadiusS);
    Debug.Log("tkip + aes(1):"+ isTKIPAES);
    if (addressRadiusS != null && clave_val != null && isTKIPAES == true )
    {
        addressText.gameObject.SetActive(true);
```

```
        ifAddress.gameObject.SetActive(false);
        addressText.text = addressRadiusS;

        isWS2 = true;
    }
}



private void UseFilterEn()
{
    isMACFilter = true;
    tglUseFilterDisPWSMACDIS.isOn = false;
    isMACDis = false;
    changeTopanel(8);
}

private void UseFilterDis()
{
    isMACFilter = false;
    isMACDis = true;
    tglUseFilterEnPWSMACDIS.isOn = false;

}

private void SaveMACEn()
{
    if(isMACFilter == true)
    {
        changeTopanel(8);
    }

}



private void UseFilterEn2()
{
    isMACFilter2 = true;
    isMACDis = false;
    tglUseFilterDisPWSMACEN.isOn = false;
    tglUseFilterEnPWSMACEN.isOn = true;
    changeTopanel(8);

}

private void UseFilterDis2()
{
    isMACFilter2 = false;
    isMACDis = true;
    tglUseFilterEnPWSMACEN.isOn = false;
    tglUseFilterDisPWSMACDIS.isOn = true;
    changeTopanel(8);
}

private void Prevent()
{
    isPrevent = true;
    tglPermitClientPWSMACEN.isOn = false;
    tglPreventClientPWSMACEN.isOn = true;
```

```
    }

    private void Permit()
    {
        isPrevent = false;
        tglPreventClientPWSMACEN.isOn = false;
        tglPermitClientPWSMACEN.isOn = true;
    }

    private void SaveMACDis()
    {
        if (isPrevent == false && isMACFilter2 == true)
        {
            isMACDis = true;
        }
    }
}
```

ControladorZeroshell.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;


public class ZeroShellController : MonoBehaviour
{

    private GameObject[] zeroshellPanels;

    public GameObject panelLogin;
    public GameObject panelMain;
    public GameObject panelUsers;
    public GameObject panelAddUser;
    public GameObject panelUsers_Modified;
    public GameObject panelRadiusON;
    public GameObject panelRadiusOFF;
    public GameObject panelAuthClient;

    public Button btnPassword, btnLogin;
    public InputField IFpassword, IFuser;
    public Text  InvalidCredentialsTxt;
    private string username = "admin";
    private string password = "zeroshell";
    private string pwd2, name2;
    private bool isLogged = false;
    public Button btnUserM, btnRadiusM, btnSetupM;

    public Button btnUserU, btnRadiusU, btnSetupU, btnAddUserU;

    public Button btnUserAU, btnRadiusAU, btnSetupAU, btnSubmitUser, btnResetUser;
    public InputField inputName;
    public InputField inputLastname;
    public InputField inputUsername;
    public InputField inputEmail;
    public InputField inputpwd1, inputpwd2;
```

```
    public Text txtDescripcion, txtDirectorio, txtWarning, txtContraseñaInvalida;
    private string pwd1, pwd22;
    private Dictionary<string, string> fields = new Dictionary<string, string>();
       public      string      nameField,      lastnameField,      usernameField,
emailField,passwordField;
    private bool isSavedUser = false;



    public Button btnUserUM, btnRadiusUM, btnSetupUM;
    public Text txtEmail, txtDescriptionMP, txtUsername;



    public Button btnUserRON, btnRadiusRON, btnSetupRON, btnClientesAuthRON;


    public Button btnUserROFF, btnRadiusROFF, btnSetupROFF, btnSaveROFF;
    public Toggle tglCheckCRL, tglRadius;
    public Text warningTextROFF;
    private bool isCheckCRL = false;
    private bool isRADIUS = false;
    private bool isSavedPanelRadiusON = false;

    private Dictionary<string, string> fieldsAuthClient = new Dictionary<string,
string>();


    public Button btnAñadirCA, btnCerrarCA;
    public Text txtClientName, txtIPMask, txtSharedSecret;
    public InputField inputClientName, inputIP, inputMask, inputSharedSecret;

    void Start() {

        panelLogin.gameObject.SetActive(true);
        Listener();
    }

    void Update() {

        zeroshellPanels =GameObject.FindGameObjectsWithTag("PanelesZS");

    }


    public void changeTopanel(int menuID)
    {
        foreach (GameObject panel in zeroshellPanels)
        {
            panel.gameObject.SetActive(false);
        }

        switch (menuID)
        {
            case 0:
                panelLogin.gameObject.SetActive(true);
                break;
            case 1:
                panelMain.gameObject.SetActive(true);
                break;
            case 2:
```

```
            if (isSavedUser == true)
            {
               panelUsers_Modified.gameObject.SetActive(true);
            }
            else
            {
                panelUsers.gameObject.SetActive(true);
            }
            break;
        case 3:
            panelAddUser.gameObject.SetActive(true);
            break;
        case 4:
            panelUsers_Modified.gameObject.SetActive(true);
            CompleteUserInfo();
            break;
        case 5:
            panelRadiusON.gameObject.SetActive(true);
            break;
        case 6:
            if (isSavedPanelRadiusON == true)
            {
                panelRadiusON.gameObject.SetActive(true);
            }
            else
            {
                panelRadiusOFF.gameObject.SetActive(true);
            }
            break;
        case 7:
            panelAuthClient.gameObject.SetActive(true);
            break;
        default:
            break;

    }

}

public void Listener()
{

    btnLogin.onClick.AddListener(SubmitData);
    btnPassword.onClick.AddListener(SubmitData);
    IFpassword.onEndEdit.AddListener(SavePassword);
    IFuser.onEndEdit.AddListener(SaveUser);

    btnSetupM.onClick.AddListener(delegate { changeTopanel(1); });
    btnUserM.onClick.AddListener(delegate { changeTopanel(2); });
    btnRadiusM.onClick.AddListener(delegate { changeTopanel(6); });

    btnSetupU.onClick.AddListener(delegate { changeTopanel(1); });
    btnUserU.onClick.AddListener(delegate { changeTopanel(2); });
    btnRadiusU.onClick.AddListener(delegate { changeTopanel(6); });
    btnAddUserU.onClick.AddListener(delegate { changeTopanel(3); });

    btnSetupUM.onClick.AddListener(delegate { changeTopanel(1); });
    btnUserUM.onClick.AddListener(delegate { changeTopanel(2); });
    btnRadiusUM.onClick.AddListener(delegate { changeTopanel(6); });

    btnSetupAU.onClick.AddListener(delegate { changeTopanel(1); });
```

```
    btnUserAU.onClick.AddListener(delegate { changeTopanel(2); });
    btnRadiusAU.onClick.AddListener(delegate {changeTopanel(6); });
    btnResetUser.onClick.AddListener(ResetNewUser);
    btnSubmitUser.onClick.AddListener(SubmitNewUser);

    inputName.onEndEdit.AddListener(SaveName);
    inputLastname.onEndEdit.AddListener(SaveLastName);
    inputUsername.onEndEdit.AddListener(SaveUsername);
    inputEmail.onEndEdit.AddListener(SaveEmail);
    inputpwd1.onEndEdit.AddListener(SavePassword1);
    inputpwd2.onEndEdit.AddListener(SavePassword2);


    btnSetupROFF.onClick.AddListener(delegate { changeTopanel(1); });
    btnUserROFF.onClick.AddListener(delegate { changeTopanel(2); });
    btnRadiusROFF.onClick.AddListener(delegate { changeTopanel(6); });
    tglCheckCRL.onValueChanged.AddListener(delegate { CheckCRL(); });
    tglRadius.onValueChanged.AddListener(delegate { CheckRadius(); });
    btnSaveROFF.onClick.AddListener(delegate { changeTopanel(6); });

    btnSetupRON.onClick.AddListener(delegate { changeTopanel(1); });
    btnUserRON.onClick.AddListener(delegate { changeTopanel(2); });
    btnRadiusRON.onClick.AddListener(delegate { changeTopanel(6); });
    btnClientesAuthRON.onClick.AddListener(delegate { changeTopanel(7);});


    btnAñadirCA.onClick.AddListener(AddClientAuth);
    btnCerrarCA.onClick.AddListener(delegate { changeTopanel(6); });

    inputClientName.onEndEdit.AddListener(SaveClientName);
    inputIP.onEndEdit.AddListener(SaveIP);
    inputMask.onEndEdit.AddListener(SaveMask);
    inputSharedSecret.onEndEdit.AddListener(SaveShared);
}


private void SaveUser(string name)
{
   if (name == username)
    {
        name2 = name;
        CheckLogin();
    }
    else
    {
        name2 = "";
    }


private void SavePassword(string pwd)
{
    if (pwd == password)
    {
        pwd2 = pwd;
        CheckLogin();
    }
    else
    {
        pwd2 = "";
    }
}
```

```
private void CheckLogin()
{
    if (pwd2 == password && name2 == username)
    {
        isLogged = true;
        InvalidCredentialsTxt.text = "";
    }

}

private void SubmitData()
{
    if (isLogged == true)
    {
        changeTopanel(1);
    }
    else
    {
        InvalidCredentialsTxt.color = Color.red;
        InvalidCredentialsTxt.text = "Usuario y/o contraseña inválidos";
        IFuser.text = "";
        IFpassword.text = "";
        Invoke("ResetWarnings", 3.0f);
    }
}


public void SaveName(string input)
{
    fields.Add("nombre", input);
}

public void SaveLastName(string input)
{
    fields.Add("apellido", input);
    txtDescripcion.text = fields["nombre"] + fields["apellido"];
}

public void SaveUsername(string input)
{
    fields.Add("username", input);
    txtDirectorio.text = "home/"+fields["username"];

}

public void SaveEmail(string input)
{
    fields.Add("email", input);
    Debug.Log(fields["email"]);
}

public void SavePassword1(string input)
{
    pwd1 = input;
}

public void SavePassword2(string input)
{
    pwd22 = input;
    CheckPassword();
```

```
    }


    private void CheckPassword()
    {
        if (pwd1 == pwd22)
        {
            fields.Add("password", pwd1);
        }
        else if(pwd1 != pwd2)
        {

            txtContraseñaInvalida.color = Color.red;
            txtContraseñaInvalida.text = "Las contraseñas deben coincidir";
            Invoke("ResetWarnings", 3.0f);
        }

    }
    private void SubmitNewUser()
    {

        if  (fields.ContainsKey("nombre")  &&  fields.ContainsKey("apellido")  &&
fields.ContainsKey("username")       &&       fields.ContainsKey("email")       &&
fields.ContainsKey("password"))
        {

            nameField = fields["nombre"];
            lastnameField = fields["apellido"];
            usernameField = fields["username"];
            emailField = fields["email"];
            passwordField = fields["password"];
            isSavedUser = true;
            changeTopanel(4);

        }
        else
        {
            txtWarning.color = Color.red;
            txtWarning.text = "Debe establecer todos los parámetros requeridos";
            Invoke("ResetWarnings", 3.0f);

        }
    }

    private void ResetNewUser()
    {
        fields.Clear();
        txtDirectorio.text = "";
        txtDescripcion.text = "";
        inputUsername.text = "";
        inputEmail.text = "";
        inputLastname.text = "";
        inputName.text = "";
        inputpwd1.text = "";
        inputpwd2.text = "";
}



    private void CompleteUserInfo()
    {
```

```
        txtUsername.text = usernameField;
        txtEmail.text = emailField;
        txtDescriptionMP.text = nameField + lastnameField;

    }


    private void CheckCRL()
    {
        isCheckCRL = true;
        CheckingPanelOFF();
    }
    private void CheckRadius()
    {
        isRADIUS = true;
        CheckingPanelOFF();
    }
    private void CheckingPanelOFF()
    {
        if (isCheckCRL == true && isRADIUS == true )
        {
            isSavedPanelRadiusON = true;

        }
        else
        {
            warningTextROFF.color = Color.red;
            warningTextROFF.text = "Cuidado. Los cambios no se han guardado.";
        }
    }



private void SaveClientName(string clientName)
{
        fieldsAuthClient["name"] = clientName;
}
private void SaveIP(string IP)
{
        fieldsAuthClient["IP"] = IP;
}

private void SaveMask(string Mask)
{
        fieldsAuthClient["Mask"] = Mask;
}

private void SaveShared (string sharedsecret)
{

        fieldsAuthClient["secret"] = sharedsecret;
}



    private void AddClientAuth()
    {
        Debug.Log("entra a añadir");
```

```
        if                (fieldsAuthClient.ContainsKey("Mask")                &&
fieldsAuthClient.ContainsKey("IP")   &&   fieldsAuthClient.ContainsKey("name")   &&
fieldsAuthClient.ContainsKey("secret"))
        {

            inputIP.text = "";
            inputMask.text = "";
            inputSharedSecret.text = ""; ;
            inputClientName.text = "";
            txtSharedSecret.text = fieldsAuthClient["secret"];
            txtClientName.text = fieldsAuthClient["name"];
            txtIPMask.text      =      fieldsAuthClient["IP"]      +      "/"      +
fieldsAuthClient["Mask"];
            fieldsAuthClient.Clear();
        }
    }


    void ResetWarnings()
    {
        txtContraseñaInvalida.text = "";
        txtWarning.text = "";
        InvalidCredentialsTxt.text = "";
    }
}
```

Bibliografía

Cisco Meraki. *Cisco Meraki*. Obtenido de Meraki Documentation: https://documentation.meraki.com/MR/Access_Control

Download, U. *Unity*. Obtenido de Unity: https://unity3d.com/es/get-unity/download

Labster. *Labster*. Obtenido de Labster: https://www.labster.com/simulations/

Linksys. *Linksyst WRT1200 AC UserGuide.* Obtenido de Linksyst WRT1200 AC UserGuide: https://downloads.linksys.com/downloads/userguide/MAN_WRT1200AC_LNKPG-00192_RevA00_Intl.pdf

Losilla, F. (2019). *Apuntes asignatura Redes Inalámbricas para el Grado de Telemática*. Cartagena.

Losilla, F. (2020). *Apuntes asignatura Realidad virtual y aumentada del Máster en ingeniería telemática.* Cartagena.

Secure W2. Obtenido de Secure W2: https://www.securew2.com/solutions/802-1x

Unity Learn. Obtenido de Unity Learn: https://learn.unity.com/search?k=%5B%22q%3AUser+Interface%22%5D

Unity. *Unity*. Obtenido de Unity Documentation: https://docs.unity3d.com/Manual/index.html

VirtualBox. *VirtualBox*. Obtenido de VirtualBox: https://www.virtualbox.org/

Zeroshell. *Zeroshell*. Obtenido de Zeroshell: https://zeroshell.org/