
Grado en Ingeniería Informática

Trabajo de Final de Grado

# Desarrollo de una aplicación web para el uso de scripts en PowerShell

*Autor:*
Iván Guimerà Fontanet

*Supervisor:*
Luis Rius Gumbau
*Tutor académico:*
Ricardo Chalmeta Rosaleñ

Fecha de lectura: 13 de Julio de 2021
Curso académico 2020/2021

## Resumen

El presente documento detalla la memoria del trabajo de final de grado realizado por el alumno Iván Guimerà Fontanet de la Universitat Jaume I de Castellón durante la estancia en prácticas en la empresa Datanet Consultores S.L. Esta memoria contiene el proceso de creación de un proyecto cuyo objetivo principal consiste en el desarrollo de un sistema que permite la ejecución en remoto de scripts en PowerShell.

Para la obtención de dicho sistema se han valorado cuáles son los principales requisitos que debe cumplir el sistema, con lo cuál tras analizar y diseñar el sistema se ha definido e implementado una aplicación web escrita en .NET Core que permite la ejecución de scripts en PowerShell en cualquier servidor remoto siempre y cuando previamente se haya configurado correctamente para permitir las conexiones remotas y la ejecución de scripts.

La aplicación web permite la subida de scripts PowerShell en el servidor web a usuarios administradores para posteriormente cualquier tipo de usuario pueda ejecutar dichos scripts con los parámetros que sean necesarios en cada ocasión. Cada uno de esos scripts está asociado a un servidor remoto sobre el cuál se realiza la ejecución. Con las pruebas ejecutadas sobre 3 servidores remotos diferentes se ha podido comprobar que la aplicación funciona correctamente en todos los casos.

## Palabras clave

PowerShell, gestión de scripts, .NET Core, ejecución en remoto.

## Keywords

PowerShell, scripts management, .NET Core, remote execution.

# Índice general

# Índice de figuras

# Índice de tablas

# Capítulo 1

# Introducción

## 1.1. Contexto y motivación del proyecto

El proyecto que se presenta en este documento se ha desarrollado en Datanet Consultores S.L., cuyo logo puede verse en la Figura 1.1. Esta compañía ofrece soluciones de negocio, estrategia, desarrollo y mantenimiento de aplicaciones tecnológicas y outsourcing a diferentes sectores de nuestra sociedad tales como sanidad, industria, sector público o puertos, entre otros.

La compañía cuenta con oficinas en Valencia, Alicante, Castellón, Murcia, Madrid y Gran Canaria. Además, forma parte del grupo Alfatec Sistemas que tiene presencia en 14 países y cuenta con un total de más de 500 empleados gracias a su continua expansión con sus más de 30 años de experiencia[2]. El proyecto se ha desarrollado dentro de la Unidad de Negocio


Figura 1.1: Logotipo de la compañía Datanet Consultores.

correspondiente a BC Puertos, donde la empresa cuenta con una posición muy privilegiada a nivel nacional respecto a la solución de las necesidades específicas de las autoridades portuarias a la hora de abordar una transformación digital y modernizar sus procesos de gestión y producción.

La compañía es Gold Partner[1] de Microsoft Dynamics 365 Business Central[2], gracias a

[1]Certificación que reconoce la excelencia, capacidades y experiencia en tecnologías Microsoft.
[2]Sistema de planificación de recursos empresariales (ERP) propio de Microsoft.

sus más de 60 expertos en ERP. Con esto, puede ofrecer soluciones personalizadas para cada proyecto, con el fin de adaptarlo a las necesidades particulares de cada cliente.

Cada uno de los proyectos llevados a cabo debe ser gestionado de una manera totalmente independiente, de manera que cualquier tipo de error en algún proyecto no tiene que afectar a los demás proyectos activos. Para ello se utiliza la tecnología Docker que nos permite crear contenedores[3], cuyo propósito es conseguir esa independencia que nos permita la ejecución de varios procesos por separado para obtener un mejor rendimiento además de dar la seguridad que se obtendría si se ejecutaran en sistemas independientes. Hay que mencionar que un contenedor no es lo mismo que una máquina virtual, como se puede ver en la Figura 1.2, además de presentar ciertas ventajas tales como una mayor versatilidad, ligereza y portabilidad.

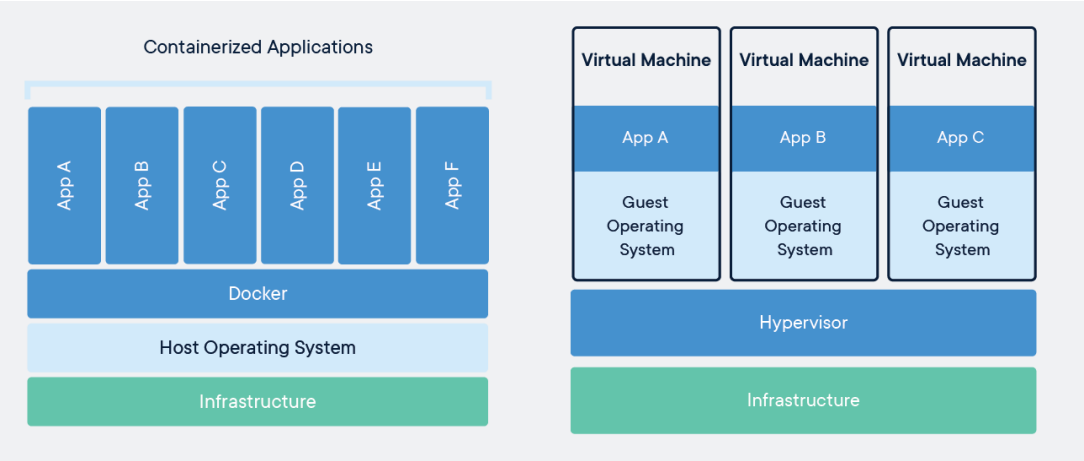


Figura 1.2: Diferencia estructural entre contenedor y máquina virtual[4].

Docker no cuenta con una interfaz en el lado del servidor, por lo cual debemos recurrir a una herramienta externa para poder contar con una interfaz y poder ver algunos datos y ejecutar algunas acciones sobre nuestros contenedores Docker. Esta herramienta que se utiliza en la empresa se llama Portainer[4] y corresponde con un software de código abierto que permite administrar y monitorear Docker (también sirve con otros programas con la misma funcionalidad de Docker, como Kubernetes, Azure ACI o Docker Swarm) de una manera sencilla e intuitiva. El problema ocurre cuando los empleados de la empresa necesitan realizar ciertas acciones que no están disponibles en Portainer y que usan con bastante frecuencia, con lo que irremediablemente tienen que recurrir a la línea de comandos.

La principal motivación del proyecto es solucionar la falta de funcionalidad que presenta Portainer respecto a las acciones que requieren realizar los empleados de la empresa. Para ello, se va a desarrollar una aplicación web que, a través de una interfaz sencilla, permita ejecutar todas las acciones de uso frecuente sin tener que recurrir a la línea de comandos.

[3]Infraestructura inmutable que permite implementar aplicaciones junto a sus dependencias.
[4]https://www.portainer.io/

## 1.2. Objetivos del proyecto

El principal objetivo de este proyecto es crear una aplicación web a través de la cual se puedan realizar acciones que actualmente solo se pueden realizar mediante la ejecución de scripts y/o comandos de PowerShell. Dicho objetivo principal se puede desglosar en los siguientes subobjetivos:

- Implementación de una aplicación web mediante un lenguaje de programación que permita la ejecución de los comandos PowerShell necesarios para la realización de las funciones requeridas.
- Permitir al administrador la creación de nuevas funcionalidades en el sitio web.
- Garantizar la seguridad adecuada del uso de las funcionalidades de la aplicación web mediante la implementación de permisos.
- Permitir al administrador la modificación de las funcionalidades existentes en el sitio web.
- Acceder de forma remota a los servidores donde se va a ejecutar cualquier funcionalidad de la aplicación web.

## 1.3. Alcance del proyecto

El alcance del proyecto se puede dividir en tres partes bien diferenciadas: alcance funcional, alcance organizativo y alcance informático:

**Alcance funcional:** Corresponde a las tareas que se le va a permitir realizar al sistema. Como se ha explicado anteriormente, se va a añadir una interfaz que permita realizar acciones que los empleados deben realizar actualmente mediante la línea de comandos, con la ejecución de varios comandos y/o scripts de una cierta complejidad.

**Alcance organizativo:** Corresponderá con el departamento de desarrollo de la empresa, que es el encargado de la creación de los contenedores Docker correspondientes a las necesidades de cada uno de los clientes y, por lo tanto, son los encargados de utilizar los scripts y/o comandos de PowerShell.

**Alcance informático:** Incluye el diseño e implementación de la aplicación web, que gestionará las interacciones con los usuarios a través de la integración de scripts PowerShell con el lenguaje de programación en el cuál se implementará la aplicación web.

## 1.4. Tecnologías y herramientas utilizadas

En esta sección se va a describir cuales han sido las tecnologías que se ha acordado utilizar para la realización del proyecto.

Como el eje principal del proyecto es la realización de una aplicación web, es fundamental determinar que herramientas web se van a utilizar:

**ASP.NET Core**

Framework[5] web de código abierto desarrollado por Microsoft que permite la compilación de aplicaciones web. Este framework se utiliza a recomendación de la empresa ya que es conocido por ellos y pueden proporcionar soporte en algunos aspectos. Además cuenta con la ventaja que es un framework cada vez más extendido y, por lo tanto, puede ser de gran utilidad de cara al futuro.

Se opta por utilizar el framework ASP.NET Core ya que, a diferencia de ASP.NET, es multiplataforma, con lo cual se puede utilizar en Windows, Linux y macOS. También resulta interesante porque se puede desplegar dentro de un contenedor Docker con lo cual se podría tener la aplicación web alojada dentro de un contenedor, cosa que facilitaría su despliegue y proporcionaría los beneficios de aislamiento propios de dicha estructura.

Además proporciona muchas otras ventajas respecto a otros frameworks, ya que es mucho más rápido que cualquier otro framework y está diseñado para adaptarse rápidamente a todas las mejoras que puedan darse en compiladores, lenguajes de programación, APIs, etc. con la ventaja que supone siempre tener el soporte de Microsoft[12].

**C#**

Lenguaje de programación compilado[6] basado en objetos, que facilita la creación de aplicaciones sólidas y seguras que se ejecutan en el ecosistema de .NET[13]. Se utiliza este lenguaje de programación ya que es el lenguaje de programación utilizado por el framework ASP.NET Core y que supone toda la implementación del backend[7] de la aplicación web creada.

**HTML**

Lenguaje de marcado que se utiliza para el desarrollo del frontend[8] de un sitio web. Se encarga de la codificación de un documento mediante etiquetas y define el estándar para el desarrollo de sitios web. En el proyecto se han utilizado ficheros con extensión ".cshtml"que son ficheros que permiten añadir código de c# dentro de un documento HTML, ya que es necesaria esa interacción para poder mostrar al usuario información obtenida en el backend y es el tipo de fichero utilizado por el framework ASP.NET Core para la interacción con el usuario.

[5]Estructura básica de trabajo utilizada para el desarrollo de software.

[6]Lenguaje de programación que necesita ser traducido a lenguaje máquina mediante un compilador antes de poder ser ejecutado.

[7]Conjunto de acciones que establecen la lógica de un sitio web.

[8]Parte del sitio web que interactua con el usuario.

**CSS**

Lenguaje de diseño que permite definir el estilo de un documento. Se utiliza de forma complementaria a un documento HTML para dotarlo de un aspecto visual mejorado.

**Javascript**

Lenguaje de programación interpretado que permite la implementación de funcionalidad en una página web. Permite el tratamiento de variables, la creación de contenido de manera dinámica, la ejecución de respuestas a eventos, etc[16].

**Bootstrap**

Framework usado para el desarrollo de sitios web. Cuenta con un gran número de plantillas con estilo (HTML+CSS) que permiten dotar al sitio web de un aspecto visual más amigable sin la necesidad de definir largos y complejos ficheros CSS.

Además de las tecnologías utilizadas en el sitio web, el proyecto también utiliza otras herramientas:

- **Jira:** Herramienta que permite realizar la gestión de un proyecto. Se ha utilizado para llevar el control del tiempo dedicado diariamente a realizar cada una de las tareas.

- **Azure DevOps:** Software que contempla la monitorización de todos los pasos de la construcción de un software y que permite una mayor conexión entre los diferentes equipos que se ocupan del desarrollo del ciclo de vida del mismo. Proporciona un gran número de herramientas relacionadas para cada una de las etapas del software. En este caso, la herramientas utilizada principalmente es GIT que es un software de control de versiones y que se ha utilizado como sistema de copias de seguridad así como para la restauración de versiones anteriores en caso de que alguna haya algún cambio de idea respecto a cualquier funcionalidad del sitio web.

- **Visual Studio:** Entorno de desarrollo compatible con múltiples lenguajes de programación y cualquier entorno que permita ejecutar .NET. Necesario para la creación de la aplicación web, aunque hay otras alternativas disponibles.

- **Docker:** Software que se dedica a la automatización del despliegue de aplicaciones dentro de contenedores. Su uso es obligatorio ya que, aunque existen otras alternativas para el despliegue de aplicaciones en contenedores, la empresa utiliza siempre este software en sus proyectos y no entra en el alcance de este proyecto barajar otras alternativas.

- **PowerShell:** Lenguaje de scripting. Existen un grupo de comandos y scripts usados frecuentemente en la empresa que usan principalmente para la gestión de los contenedores y que se pretenden usar en la aplicación web para automatizar el proceso sin necesidad de recurrir siempre a la línea de comandos.

## 1.5. Estructura de la memoria

La memoria del proyecto consta con un total de 6 capítulos.

El primer capítulo introduce y contextualiza el proyecto de una manera global. En él se describe cuál es la empresa en la que se ha desarrollado el proyecto, cuál es la finalidad del proyecto y las tareas incluidas en él y qué tecnologías se han empleado con el fin de poder ejecutarlo.

El segundo capítulo muestra qué metodología se ha seguido en la ejecución del proyecto, además de definir la distribución horaria para la ejecución de las diferentes fases del proyecto. También se incluyen los costes y recursos utilizados, además de indicar las modificaciones temporales que se han producido en algunas de las tareas realizadas.

El tercer capítulo indica cuáles son las diferente condiciones de uso de la aplicación web y qué pasos se producen con tal de proceder a su ejecución. También se indica cuáles son los requisitos necesarios para que el proyecto funcione correctamente.

El cuarto capítulo muestra como se estructura el sitio web, tanto por jerarquía de directorios como por la estructuración de la base de datos.

El quinto capítulo muestra de que manera se han desarrollado los principales puntos para la implementación del proyecto.

El sexto y último capítulo muestra los procesos de validación y verificación utilizados para que el proyecto funcione de una manera adecuada.

# Capítulo 2

# Planificación del proyecto

## 2.1. Metodología

La metodología seguida durante el desarrollo del proyecto ha sido una metodología en cascada. Debido a las características del proyecto y que ha sido desarrollado por una única persona, es la metodología más adecuada. Consiste en un conjunto de fases ejecutadas de manera secuencial, como se puede ver en la Figura 2.1, por lo que no se puede ejecutar una fase sin haber terminado la anterior.

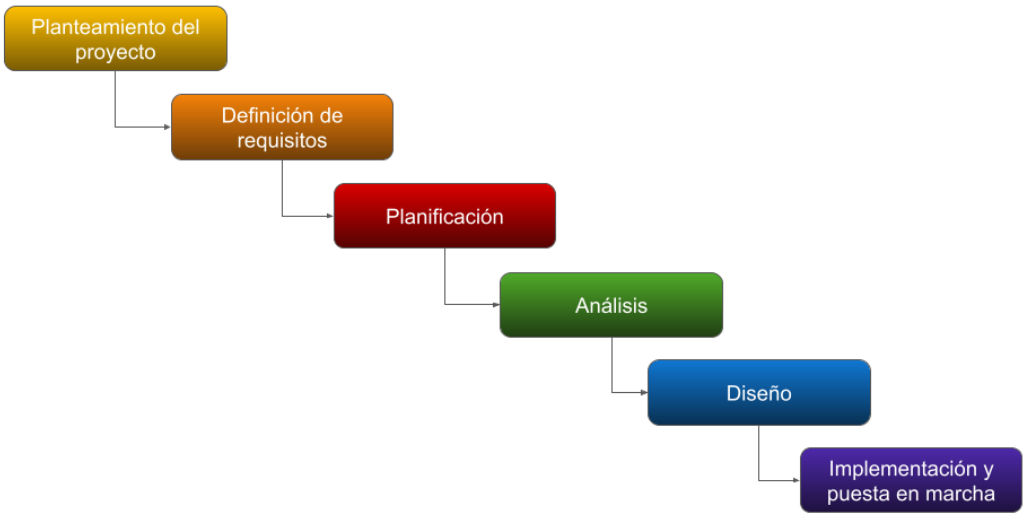


Figura 2.1: Fases a seguir en el proyecto mediante la metodología en cascada.

De hecho, la característica principal que define a la metodología en cascada consiste en que el proyecto sea un proceso secuencial y sin retrocesos. Para ello, es fundamental tener claro cuáles son los requisitos del proyecto ya que no se espera que vayan a sufrir modificaciones a lo

largo de su duración. Como los requisitos del proyecto son claros desde el principio y no se van a modificar, la metodología en cascada es ideal para afrontar este proyecto.

## 2.2. Planificación temporal del proyecto

Como se ha comentado en el apartado anterior, la metodología en cascada incluye un conjunto de fases en las cuáles se va a realizar diferentes tareas:

- **Planteamiento del proyecto:** Se realiza una descripción de cuál va a ser el proyecto a realizar y la definición de cuáles son los objetivos que se pretenden conseguir, así como la delimitación del alcance del proyecto teniendo en cuenta los objetivos indicados.

- **Definición de requisitos:** En esta fase se definen los requisitos que debe cumplir el proyecto. Los requisitos deben encontrarse dentro del alcance funcional, ser viables tecnológica, económica y temporalmente y contemplar todas las necesidades funcionales y objetivos definidos. Para esta fase es usual el uso de lenguaje unificado de modelado (UML), que define diferentes tipos de diagramas como, por ejemplo, diagramas de casos de uso.

- **Planificación:** Se debe determinar cuales son las tareas y subtareas que se van a realizar de manera secuencial para llevar a cabo el proyecto. Para ello, es importante la realización de una representación gráfica adecuada como, por ejemplo, el uso de un diagrama de Gantt o un esquema de descomposición de trabajo. En esta fase también se va a realizar la estimación de recursos software, hardware y humanos requeridos para el desarrollo del proyecto.

- **Análisis:** En esta fase se pretende transformar la definición de requisitos en especificaciones de programario que deben ser implementadas posteriormente. En esta fase se definen y desarrollan los diferentes casos de uso presentes en la aplicación web y los requisitos de datos, que son los modelos necesarios de definir junto con sus datos que van a almacenar.

- **Diseño:** Esta fase consiste en la especificación física de la documentación generada en la fase de análisis. Aquí se definen las interfaces y los procesos y arquitectura del proyecto.

- **Implementación y puesta en marcha:** Esta fase consiste en la programación y desarrollo del proyecto. Posteriormente, se realizarán las pruebas del proyecto para comprobar el correcto funcionamiento del sistema y subsanar cualquier error que pueda aparecer.

Es muy importante seguir estrictamente el orden de cada una de estas fases, además de realizar una planificación temporal de cada una de las fases y de las tareas que las componen, como se puede observar en la Tabla 2.1. Esta distribución temporal se ha realizado teniendo en cuenta el número de horas de las que se dispone para realizar el proyecto, que son 300 horas en total.

| Tarea | Horas dedicadas |
|---|---|
| **1. Planteamiento del proyecto** | **8** |
| 1.1 Descripción del proyecto a realizar | 4 |
| 1.2 Definición de objetivos y alcance | 4 |
| **2. Definición de requisitos** | **22** |
| 2.1 Identificación de requisitos | 3 |
| 2.2 Creación de diagramas de uso | 11 |
| 2.3 Validación y verificación de requisitos | 8 |
| **3. Planificación** | **12** |
| 3.1 Determinar tareas y subtareas | 4 |
| 3.2 Representación mediante diagrama de Gantt | 3 |
| 3.3 Estimación de recursos | 5 |
| **4. Análisis** | **12** |
| 4.1 Realización de diagrama de clases | 6 |
| 4.2 Realización de diagrama de actividades | 6 |
| **5. Diseño** | **15** |
| 5.1 Identificación de usuarios | 2 |
| 5.2 Diseño de interfaces | 8 |
| 5.3 Diseño de informes sobre interfaces | 5 |
| **6. Implementación y puesta en marcha** | **231** |
| 6.1 Implementación del diseño general del sitio web | 52 |
| 6.2 Implementación de scripts en la web | 67 |
| 6.3 Puesta en marcha y corrección de errores en local | 39 |
| 6.4 Implementación del sitio web en remoto | 28 |
| 6.5 Puesta en marcha y corrección de errores en remoto | 22 |
| 6.6 Depuración del código | 13 |
| 6.7 Formación a usuarios | 10 |

Tabla 2.1: Planificación temporal del proyecto

### 2.2.1. Diagrama de Gantt

Para ver de una manera más gráfica la distribución temporal de cada una de las fases y subfases, se puede observar la Figura 2.2, correspondiente con el diagrama de Gantt del proyecto En dicho diagrama se especifica las fechas de inicio de cada una de las tareas.

Figura 2.2: Diagrama Gantt del proyecto

## 2.3. Estimación de recursos y costes del proyecto

Para todo proyecto que se desarrolle, se debe tener una previsión inicial de cuáles serán sus costes, teniendo en cuenta el uso de recursos software, hardware y humanos.

### 2.3.1. Recursos software

Los programas utilizados para la realización del proyecto y para la comunicación y organización interna de la empresa son los siguientes:

- **Microsoft 365 Empresa Estándar:** Utilizado para las comunicaciones y organización dentro de la empresa. El precio por usuario es de 10,50 euros al mes. Como la estimación de la duración del proyecto es de 4 meses, debemos multiplicar ese precio por 4, por lo tanto, 42 euros. .
- **Visual Studio:** Para el proyecto a desarrollar es suficiente con utilizar la versión gratuita, por lo tanto, 0 euros.
- **Docker Desktop:** Cuenta con varios tipos de suscripciones, aunque el proyecto se puede llevar a cabo con la versión gratuita del software, por lo tanto, 0 euros.
- **Jira:** Para llevar el seguimiento de las horas dedicadas al proyecto y la organización de tareas, con la versión gratuita de este software es suficiente, con lo cual, 0 euros.
- **Azure Devops:** Ya que solo se va a utilizar la funcionalidad de Git para la realización del proyecto, se opta por usar la versión gratuita del software, por lo tanto, 0 euros

Visto todo el software utilizado en la realización del proyecto, se establece que **el coste total del software del proyecto es de 42€.**

### 2.3.2. Recursos hardware

Para la realización del proyecto se ha dispuesto de varios elementos:

- **Ordenador portátil:** Valorado en 600 euros.
- **Teclado:** Valorado en 15 euros.
- **Ratón:** Valorado en 5 euros.

Como hay que ajustar el coste de los recursos hardware al tiempo de realización del proyecto, debemos tener en cuenta el tiempo de vida aproximado del hardware utilizado. Según su uso, se estima que ese tiempo variará entre 3 y 6 años.

Suponiendo que el tiempo de vida de este hardware será de 4 años, se debe calcular en base a los 4 meses de la duración del proyecto. Por lo tanto, **el coste de los recursos hardware es de 52€.**

### 2.3.3. Recursos humanos

Se debe tener en cuenta que para la realización del proyecto es necesario contar con diferentes trabajadores, que van a realizar diferentes funciones. Como este proyecto es individual, se va a realizar las diferentes funciones adquiriendo el rol de los puestos de trabajo necesarios, cuyos costes y tiempo de dedicación se definen en la Tabla 2.2.

| Puesto | Horas dedicadas | Coste por hora | Coste total |
|---|---|---|---|
| Analista | 54 | 11,61€ | 627€ |
| Programador | 246 | 10,18€ | 2504€ |

Tabla 2.2: Coste de los diferentes roles del proyecto

- **Analista:** Se encarga de las primeras 4 fases del proyecto, donde se incluyen el planteamiento del proyecto, la definición de requisitos, la planificación y el análisis del proyecto. Para dichas tareas, la estimación inicial definida en la planificación temporal es de 54 horas. Teniendo en cuenta que el salario promedio de un analista programador junior en España es de 11,61 euros por hora[6], el coste de las funciones del analista será de 627€.
- **Programador:** Se encarga del diseño y la implementación y puesta en marcha del proyecto. Según la estimación inicial definida en la planificación inicial, la realización de dichas fases va a requerir 246 horas. Teniendo en cuenta que el salario promedio de un programador junior en España es de 10,18 euros por hora[7], el coste de las funciones del programador será de 2.504€.

El coste de cubrir los puestos de analista y programador asciende a 3131€. Además, se deben contabilizar otros costes adicionales. En primer lugar, se debe tener en cuenta los costes de contratación, que suponen un 30 % adicional al coste total de los diferentes roles que han sido necesarios para el desarrollo del proyecto. Para ello, debemos añadir ese porcentaje a los 3.131€ que suponen el pago del analista y del programador. En segundo lugar, se contabilizan otros gastos indirectos, ya sea electricidad, Internet u otros gastos y que suponen un 20 % adicional a esos 3131€. Por lo tanto, 30 % de 3.131€ = 939,30€ y 20 % de 3.131€ = 626,20€.

En la Tabla 2.3 se puede observar el resumen de todos los gastos generados respecto a los costes de recursos humanos. En conclusión, **el coste de los recursos humanos asciende a 4.696,50€**

| Concepto | Coste total |
|---|---|
| Tareas de analista | 627€ |
| Tareas de programador | 2504€ |
| Costes de contratación | 939,30€ |
| Costes indirectos | 626,20€ |

Tabla 2.3: Coste total de los recursos humanos del proyecto

### 2.3.4. Resumen de costes

Una vez se ha visto cuáles son los recursos a utilizar y qué costes generan, en la Tabla 2.4 se puede observar el coste total de cada uno de los apartados en los cuáles se ha desglosado los recursos necesarios a utilizar en el proyecto.

**El coste total de los recursos del proyecto asciende a 4.793,50€.**

| Concepto | Coste total |
|---|---|
| Recursos software | 42€ |
| Recursos hardware | 52€ |
| Recursos humanos | 4.696,50€ |

Tabla 2.4: Coste total de los recursos del proyecto

## 2.4. Seguimiento del proyecto

Una de las partes importantes del proyecto es mantener un seguimiento adecuada de todas las tareas planificadas, principalmente cuando estamos ante un proyecto que implementa una metodología tradicional donde las fases están bien definidas. No obstante, en la gran mayoría de proyectos siempre existen contratiempos que impide que la planificación inicial pueda cumplirse, bien sea por dedicar un mayor o menor tiempo a algunas tareas o por el hecho de tener que añadir a la planificación alguna tarea que no estaba prevista.

Debido a la situación sanitaria provocada por la pandemia de COVID-19 y con el fin de asegurar la salud de las personas trabajadoras de la empresa en el cual se desenvolupa el proyecto, el trabajo se ha realizado íntegramente de forma telemática. Eso ha supuesto en todo momento una pérdida de agilidad a la hora de realizar cualquiera de las fases del proyecto,

ya que todo el seguimiento del proyecto se ha realizado a través de videollamadas mediante la herramienta Microsoft Teams.

Al finalizar las primeras fases del proyecto, disminuyó el número de reuniones realizadas por semana, y se pasó a realizar una única reunión semanal para realizar el seguimiento del proyecto. Además, también se mantuvo el contacto cuando fue necesario para que el supervisor pudiese orientar al alumno cuando surgieron problemas en alguna de las tareas realizadas.

Como se puede ver en la Tabla 2.5, las principales modificaciones respecto a la planificación inicial se han producido en la fase de implementación y puesta en marcha. Es lógico que sea la parte que sufra más modificaciones ya que es muy improbable que la implementación del sistema funcione correctamente desde un primer momento, con lo cuál se hace muy difícil establecer una planificación inicial acertada.

| **Tarea** | **Horas estimadas** | **Horas dedicadas** |
|---|---|---|
| 6.1 Implementación del diseño general del sitio web | 52 | 40 |
| 6.2 Implementación de lectura de scripts en la web | 67 | 35 |
| 6.3 Puesta en marcha y corrección de errores en local | 39 | 20 |
| 6.4 Implementación del sitio web en remoto | 28 | 65 |
| 6.5 Puesta en marcha y corrección de errores en remoto | 22 | 47 |

Tabla 2.5: Variación de horas dedicadas a tareas respecto de la planificación inicial

# Capítulo 3

# Análisis del sistema

## 3.1. Actores

Un actor de cualquier sistema informático especifica un rol que adquiere una entidad que realiza una interacción directa con el sistema. En este caso todas las entidades presentes en el sistema corresponden con usuarios.

Es necesario definir cuáles son los actores del sistema con el fin de poder establecer posteriormente de manera correcta los casos de uso. El sistema cuenta con dos actores, que se pueden diferenciar en la Tabla 3.1.

| Identificador | Actor | Descripción |
|---|---|---|
| ACT01 | Administrador | Este actor representa a cada persona con permisos absolutos sobre los scripts, además de permisos sobre la gestión del actor ACT02. |
| ACT02 | Usuario | Este actor representa a cada persona con permisos de ejecución de scripts. |

Tabla 3.1: Actores del proyecto

## 3.2. Diagrama de casos de uso

Los casos de uso representan cada una de las diferentes situaciones que podrán ser dadas en el sistema. En este caso, representa la secuencia de interacciones que permitirá a los usuarios acceder y ejecutar cada una de las funcionalidades que estarán disponibles en el sitio web. Como se puede ver en la Figura 3.1, se pueden identificar 8 casos de uso diferentes, donde el actor Administrador (ACT01) tendrá una disponibilidad absoluta y el actor Usuario (ACT02) tan solo podrá llevar a cabo 2 de esos casos de uso.

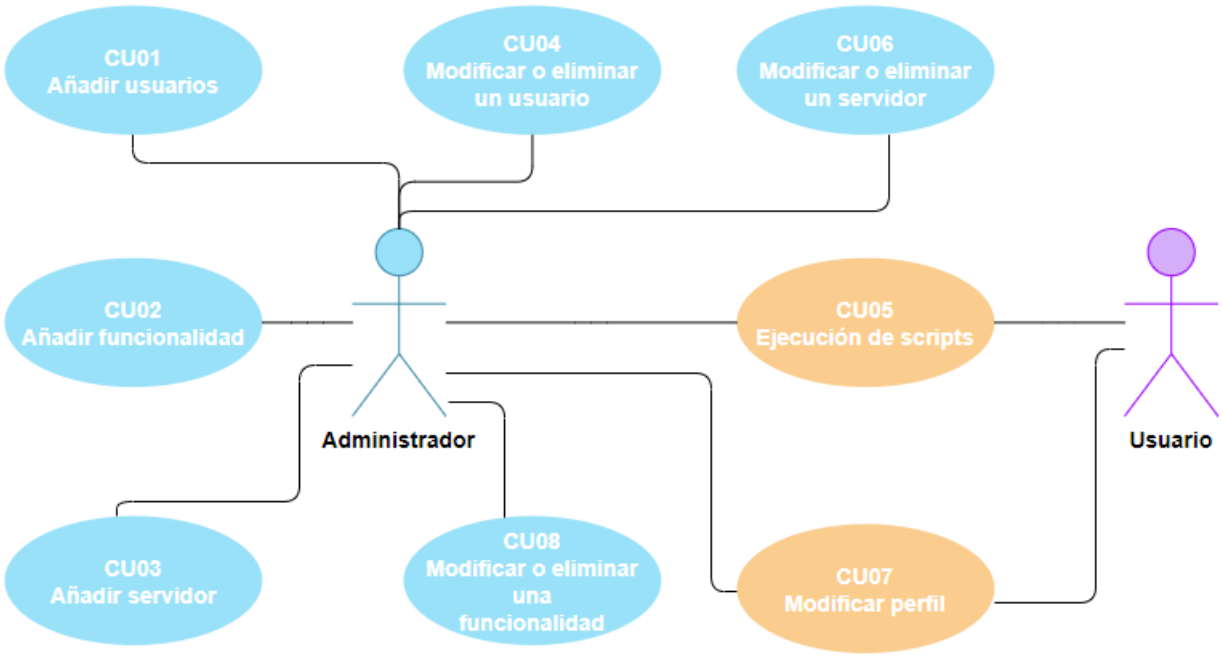


Figura 3.1: Diagrama de casos de uso

A continuación, desde la Tabla 3.2 a la Tabla 3.9 se va a realizar una descripción de la funcionalidad de cada uno de los casos de uso y cuál sería la secuencia de interacciones a realizar para que se genere la respuesta adecuada.

| Identificador: | CU01 |
|---|---|
| Nombre: | Añadir usuarios |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Actor principal: | ACT01 |
| Descripción: | Este caso de uso representa la funcionalidad que permite añadir nuevos usuarios |
| Secuencia normal: | 1.El administrador inicia sesión.<br>2. El administrador accede a la sección Más Opciones<br>3. El administrador accede a la sección Crear nuevo usuario<br>4. Se introducen los datos solicitados por el sistema para la creación del nuevo usuario<br>5. El sistema recoge los datos introducidos e inserta el nuevo usuario en la base de datos |
| Excepciones: | 1. El nombre de usuario introducido ya se encuentra en la base de datos. |

Tabla 3.2: Documentación del caso de uso CU01

| Identificador: | CU02 |
|---|---|
| Nombre: | Añadir funcionalidad |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Actor principal: | ACT01 |
| Descripción: | Este caso de uso representa la funcionalidad que permite añadir nuevos scripts en el sistema y configurar los parámetros asociados a él. |
| Secuencia normal: | 1. El administrador inicia sesión.<br>2. El administrador accede a la sección Más Opciones.<br>3. El administrador accede a la sección Añadir funcionalidad.<br>4. El sistema solicita la descripción de la funcionalidad, el servidor y el script asociado y le permite indicar el nombre de los parámetros asociados al script.<br>5. El administrador introduce los datos necesarios para el correcto funcionamiento de la nueva funcionalidad.<br>6. El sistema almacena en la base de datos el script y sus parámetros asociados. |
| Excepciones: | 1. El script asociado a la nueva funcionalidad ya se encuentra asociado al servidor introducido. |

Tabla 3.3: Documentación del caso de uso CU02

| Identificador: | CU03 |
|---|---|
| Nombre: | Añadir servidor |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Actor principal: | ACT01 |
| Descripción: | Este caso de uso representa la funcionalidad que permite añadir un nuevo servidor sobre el cuál se podrán ejecutar ciertas funcionalidades. |
| Secuencia normal: | 1. El administrador inicia sesión.<br>2. El administrador accede a la sección Más Opciones.<br>3. El administrador accede a la sección Añadir un nuevo servidor<br>4. Se introducen los datos solicitados por el sistema para añadir el nuevo servidor.<br>5. El sistema recoge los datos introducidos e inserta los datos del servidor en la base de datos. |
| Excepciones: | 1. El nombre del servidor introducido ya se encuentra en la base de datos. |

Tabla 3.4: Documentación del caso de uso CU03

| Identificador: | CU04 |
|---|---|
| Nombre: | Modificar o eliminar un usuario |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Actor principal: | ACT01 |
| Descripción: | Este caso de uso representa la funcionalidad que permite al administrador realizar modificaciones sobre los datos de los usuarios o eliminar cualquier usuario. |
| Secuencia normal: | Se desea modificar los datos de un usuario.<br>1. El administrador inicia sesión.<br>2. El administrador accede a la sección Más Opciones.<br>3. El administrador accede a la sección Modificar o eliminar usuario<br>4. El sistema muestra los usuarios guardados en la base de datos.<br>5. El administrador accede a la sección Modificar del usuario del cuál desea modificar alguno de sus datos.<br>6. El sistema modifica en la base de datos los cambios realizados en el usuario. |
| Secuencia alternativa: | Se desea eliminar un usuario.<br>1. Se repiten los pasos 1, 2, 3 y 4 de la secuencia normal<br>2. El administrador selecciona el botón Eliminar de la fila que corresponde con el usuario que desea eliminar.<br>3. El sistema elimina de la base de datos el usuario seleccionado. |

Tabla 3.5: Documentación del caso de uso CU04

| Identificador: | CU05 |
|---|---|
| Nombre: | Ejecución de scripts |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Actor principal: | ACT01, ACT02 |
| Descripción: | Este caso de uso representa la funcionalidad que permite la ejecución de los scripts de la aplicación web siempre que se posea los permisos correspondientes. |
| Secuencia normal: | 1. El usuario inicia sesión.<br>2. El usuario selecciona la funcionalidad que desea ejecutar.<br>3. El sistema solicita los datos necesarios asociados a dicha funcionalidad.<br>4. El usuario introduce los datos solicitados.<br>5. El sistema recoge los datos introducidos y ejecuta el script asociado. |
| Excepciones: | 1. Los datos de conexión al servidor no son correctos y no se puede realizar la conexión.<br>2. El servidor no está configurado correctamente y no se permite la ejecución de scripts.<br>3. Los parámetros introducidos por el usuario no son adecuados.<br>4. El administrador no ha configurado correctamente la funcionalidad. |

Tabla 3.6: Documentación del caso de uso CU05

| Identificador: | CU06 |
|---|---|
| Nombre: | Modificar o eliminar un servidor |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Actor principal: | ACT01 |
| Descripción: | Este caso de uso representa la funcionalidad que permite al administrador realizar modificaciones sobre los datos de los servidores o eliminar cualquier servidor. |
| Secuencia normal: | Se desea modificar los datos de un servidor.<br>1. El administrador inicia sesión<br>2. El administrador accede a las sección Mas Opciones.<br>3. El administrador accede a la sección Modificar o eliminar servidor.<br>4. El sistema muestra los servidores guardados en la base de datos.<br>5. El administrador accede a la sección Modificar del servidor del cuál desea modificar alguno de sus datos. |
| Secuencia alternativa: | Se desea eliminar los datos de un servidor.<br>1. Se repiten los pasos 1, 2, 3 y 4 de la secuencia normal.<br>2. El administrador selecciona el botón Eliminar de la fila que corresponde con los datos del servidor que se desea eliminar.<br>3. El sistema elimina de la base de datos los datos del servidor seleccionado. |

Tabla 3.7: Documentación del caso de uso CU06

| Identificador: | CU07 |
|---|---|
| Nombre: | Modificar perfil |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Actor principal: | ACT01,ACT02 |
| Descripción: | Este caso de uso representa la funcionalidad que permite a cualquier usuario de la aplicación web modificar sus datos. |
| Secuencia normal: | 1. El usuario inicia sesión.<br>2. El usuario hace click sobre su nombre, ubicado en la esquina superior derecha.<br>3. El sistema le muestra sus datos.<br>4. El usuario realiza las modificaciones oportunas.<br>5. El sistema modifica en la base de datos los cambios realizados en el usuario. |

Tabla 3.8: Documentación del caso de uso CU07

| Identificador: | CU08 |
|---|---|
| Nombre: | Modificar o eliminar una funcionalidad |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Actor principal: | ACT01 |
| Descripción: | Este caso de uso representa la funcionalidad que permite al administrador realizar modificaciones sobre los datos de las diferentes funcionalidades presentes en la aplicación web. |
| Secuencia normal: | Se desea modificar los datos de una funcionalidad.<br>1. El administrador inicia sesión.<br>2. El administrador accede a la sección Más Opciones.<br>3. El administrador accede a la sección Modificar o eliminar una funcionalidad.<br>4. El sistema muestra las funcionalidades guardadas en la base de datos.<br>5. El administrador accede a la sección Modificar de la funcionalidad de la cuál desea modificar alguno de sus datos.<br>6. El sistema modifica en la base de datos los cambios realizados en la funcionalidad. |
| Secuencia alternativa: | Se desea eliminar una funcionalidad.<br>1. Se repiten los pasos 1, 2, 3 y 4 de la secuencia normal<br>2. El administrador selecciona el botón Eliminar de la fila que corresponde con la funcionalidad que desea eliminar.<br>3. El sistema elimina de la base de datos la funcionalidad seleccionada. |

Tabla 3.9: Documentación del caso de uso CU08

## 3.3. Requisitos de datos

Los requisitos de datos determinan cuáles son los datos de entrada que deben ser almacenados por el sistema con el fin de poder llevar a cabo toda la funcionalidad necesaria para el correcto funcionamiento de la aplicación web.

| Identificador: | RD01 |
|---|---|
| Nombre: | Usuario |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Casos de uso asociados: | CU01, CU04, CU07 |
| Datos específicos: | Nombre de usuario, contraseña, nombre completo, email, rol. |
| Comentarios: | **Rol:** Se definen dos tipos de usuarios: usuario normal y usuario administrador. El usuario administrador podrá realizar más acciones que el usuario normal. |

Tabla 3.10: Documentación del requisito de datos RD01

Desde la Tabla 3.10 a la Tabla 3.13 se muestran los requisitos de datos necesarios para el correcto funcionamiento del proyecto.

| Identificador: | RD02 |
|---|---|
| Nombre: | Script |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Casos de uso asociados: | CU02, CU05, CU08 |
| Datos específicos: | Identificador del script, descripción de la funcionalidad, ruta del script en el servidor web, contenido, servidor asociado. |
| Comentarios: | **Identificador del script:** Identificador único de cada uno de los scripts. Se generará automáticamente.<br>**Descripción de la funcionalidad:** El administrador debe proporcionar una información clara sobre qué va a realizar la ejecución del script.<br>**Contenido:** Se almacena el contenido del script encriptado por motivos de seguridad.<br>**Servidor asociado:** Cada funcionalidad de la aplicación web se va a ejecutar sobre el único nombre del servidor que se indique. Si se quiere que una funcionalidad se ejecute sobre varios servidores, deberá crearse una nueva funcionalidad para cada servidor. |

Tabla 3.11: Documentación del requisito de datos RD02

| Identificador: | RD03 |
|---|---|
| Nombre: | Parámetro |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Casos de uso asociados: | CU02, CU05, CU08 |
| Datos específicos: | Nombre, orden, tipo, script asociado. |
| Comentarios: | **Orden:** Por defecto, al obtener los datos de una base de datos, se ordenan alfabéticamente por nombre. Con este campo se numera el orden en que aparecerán los campos a rellenar.<br>**Tipo:** Se permite añadir tanto texto como ficheros.<br>**Script asociado:** Cada uno de los parámetros se va a ejecutar sobre un script, con lo cuál se debe indicar a que script se encuentra asociado cada parámetro . |

Tabla 3.12: Documentación del requisito de datos RD03

| Identificador: | RD04 |
|---|---|
| Nombre: | Servidor |
| Versión: | 1.0 |
| Fuente: | Iván Guimerà Fontanet |
| Casos de uso asociados: | CU03, CU06 |
| Datos específicos: | Nombre, usuario, contraseña del usuario, carpeta temporal.<br><br>**Usuario:** Debe ser un usuario con privilegios de administrador para el servidor asociado, ya que en caso contrario no tendrá permisos para la ejecución de scripts. |
| Comentarios: | **Carpeta temporal:** En un principio el campo se encontrará vacío hasta que se realice la primera conexión que requiera subida de ficheros. Una vez realizada la conexión se almacenará la ruta de la carpeta temporal del servidor ya que los ficheros subidos se almacenaran en dicha carpeta y será más eficiente realizar una consulta a la base de datos que consultar por comandos PowerShell la ruta cada vez que se conecte al servidor. |

Tabla 3.13: Documentación del requisito de datos RD04

## 3.4. Requisitos de seguridad

Es fundamental la existencia de un mecanismo de seguridad que permita proteger la información sensible de la aplicación web. Se ha considerado que la aplicación contiene tres tipos de datos diferentes que es necesario proteger: la contraseña de inicio de sesión de cada usuario de la aplicación web, las credenciales de cada uno de los servidores remotos sobre los cuáles se van a ejecutar los scripts y el contenido de los propios scripts.

Como esos datos se almacenan en la base de datos, deben encriptarse antes de añadirse a la base de datos y desencriptados cuando se solicita acceder a su contenido para realizar cualquier operación en la que sean requeridos.

Por otra lado, también es necesario garantizar que ninguna persona puede acceder de ninguna manera a cualquier sitio de la aplicación de la cual no tenga permisos para acceder, con lo cuál se tiene que controlar todos las posibles formas de acceder a la aplicación web para evitar accesos no autorizados.

# Capítulo 4

# Diseño

## 4.1. Diseño de la arquitectura del sistema

La arquitectura de un sistema se define como aquel conjunto de relaciones establecidas entre cada uno de los componentes que forman parte del sistema. Como se puede ver en la Figura 4.1, en este proyecto se dispone de 6 componentes que tienen una interrelación entre ellos.

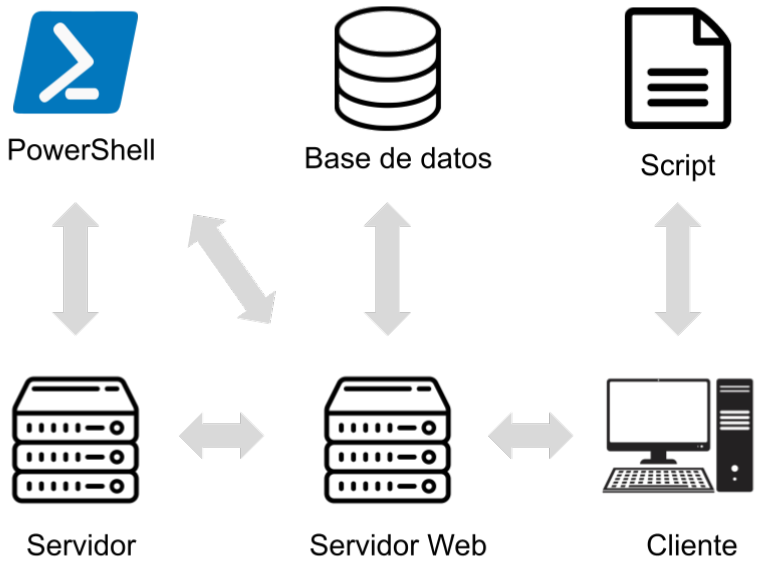


Figura 4.1: Arquitectura del sistema

Cada uno de estos elementos es muy importante en el correcto funcionamiento del sistema, ya que siempre habrá otro elemento que dependa de él y él mismo dependerá siempre de otro

elemento:

- **Script:** Representa el fichero que contiene el conjunto de comandos PowerShell que el cliente va a querer ejecutar sobre un determinado servidor. Dichos ficheros tan solo podrán ser subidos a la aplicación web mediante un usuario que tenga privilegios de administrador.
- **Cliente:** Representa cada uno de los usuarios que tienen una cuenta disponible en servidor web. La cuenta de cada cliente puede ser de tipo Usuario o de tipo Administrador. Obviamente, el usuario con cuenta de Administrador tendrá la posibilidad de tener una mayor interacción con el servidor web y será el único tipo de cuenta con posibilidad de incorporar scripts PowerShell.
- **Base de datos:** Se interrelaciona con el servidor web, ya que en la base de datos se almacenan los datos que deben permanecer en todo momento: datos de usuarios, datos de servidores, contenido de scripts, etc. El servidor web le solicita esos datos cuando el cliente los solicita y la base de datos se los envía al servidor web para que pueda enviárselos al cliente.
- **Servidor Web:** Mantiene relación directa con prácticamente todos los demás elementos del sistema. Ejecuta las órdenes del cliente, almacenando datos en la base de datos cuando sea necesario o conectando con cualquier servidor almacenado en la base de datos para realizar las operaciones contenidas en los ficheros script a través de PowerShell.
- **Servidor:** El servidor web se conecta al servidor a través de PowerShell y ejecuta un script PowerShell sobre éste. El servidor devuelve el resultado de la ejecución del script al servidor web, que a su vez será devuelto al cliente como resultado de la operación.
- **PowerShell:** Necesario para la conexión entre el servidor web y cualquier servidor cuyos datos se encuentren guardados en la base de datos del servidor web. Además de esto, los comandos que se ejecutan sobre el servidor también son de tipo PowerShell.

## 4.2. Diseño de la base de datos

Como se ha comentado previamente, la base de datos se encarga de almacenar los datos que deben permanecer siempre disponibles. Para el diseño adecuado de la base de datos se tiene en cuenta los datos obtenidos a través del análisis realizado del proyecto. En primer lugar, se realizará un diseño conceptual para definir cuáles son las diversas agrupaciones de datos que se desean almacenar; luego se evaluará el diseño lógico que indicará cuales son las características y conexiones entre las diferentes entidades; y finalmente se mostrará cuáles son los comandos encargados de generar las características deseadas con el sistema gestor de bases de datos utilizados

### 4.2.1. Diseño conceptual

El principal objetivo del diseño conceptual de una base de datos es la creación de un modelo conceptual que permita agrupar todos los requisitos de datos que se han documentado previamente[8].

Uno de los principales modelos utilizados para la realización de este diseño es el modelo entidad-relación, donde una entidad corresponde con cualquier elemento del cual se quiere almacenar información (cada uno de los requisitos de datos establecidos en el apartado 3.3) y las relaciones corresponden con cada uno de los vínculos que pueden producirse entre entidades. En la Figura 4.2 se puede observa el diseño conceptual del proyecto, donde se definen cada una de las entidades de las cuáles se quieren almacenar datos (los atributos de cada entidad) y las relaciones que hay entre ellas.

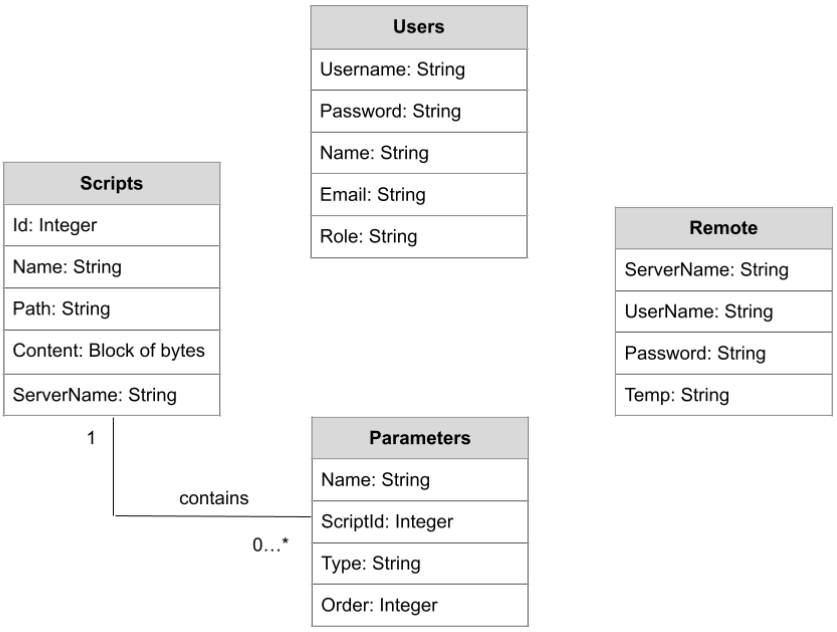


Figura 4.2: Diseño conceptual

### 4.2.2. Diseño lógico

Describe la estructura de la base de datos de tal manera que permite obtener una representación que use los recursos disponibles de la manera más eficiente posible con el fin de poder establecer una estructuración de los datos y las características y restricciones que deben ser modeladas entre los datos.

En la Tabla 4.1 se pueden apreciar las diferentes tablas que se pueden encontrar en la base de datos, donde el atributo con subrayado indica que es la clave primaria y el atributo marcado de color corresponde con una clave ajena que hace referencia a otro atributo de otra tabla.

Como se puede observar, la tabla Parameters cuenta con una clave ajena[1] (ScriptId) que hace referencia a la clave primaria Id de la tabla Scripts y se indican las reglas de integridad de

[1]Una clave ajena indica una relación entre una o varias columnas de una tabla hija y una tabla maestra donde los cambios realizados en la columna referenciada de la tabla maestra repercuten en la columna referenciada de la tabla hija.

| |
|---|
| Scripts (Id, Name, Path, Content, ServerName) |
| Parameters (ScriptId, Name, Type, Order) |
| Parameters — (ScriptId) — Scripts(Id) (NO, P, P) |
| Users (Username, Password, Name, Email, Role) |
| Remote (ServerName, UserName, Password, User) |

Tabla 4.1: Diseño lógico de la base de datos del proyecto

las claves ajenas:

**Regla de los nulos:** se establece la NO admisión de nulos en la clave ajena. No tendría sentido que una clave ajena que haga referencia a una clave primaria pueda contener nulos, ya que una clave primaria no admite nulos.

**Regla de borrado:** se establece que al eliminar un script, se propagará (P) la eliminación a todos los registros de la tabla Parameters cuyo ScripId sea el mismo que el Id del registro de la tabla Script eliminado.

**Regla de modificación:** se establece que al modificar el atributo Id de un script, se propagará (P) la modificación a todos los registros de la tabla Parameters cuyo ScripId sea el mismo que ese Id, modificando también el atributo al mismo valor modificado. En el caso de este proyecto, esta regla no se va a aplicar nunca, ya que la opción de modificar el identificador del script no se encuentra disponible (ya que no genera ninguna utilidad).

### 4.2.3. Diseño físico

El diseño físico describe la implementación de la base de datos. Debe expresarse mediante el lenguaje característico del sistema gestor de bases de datos (SGBD) que se ha utilizado en el proyecto. En este proyecto se ha utilizado SQLite como SGBD, por lo tanto, los códigos presentados a continuación están expresados con su propio lenguaje.

```
1  CREATE TABLE `Scripts` (
2      `Id`     INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT,
3      `Name`   TEXT NOT NULL,
4      `Path`   TEXT NOT NULL,
5      `Content`    BLOB NOT NULL,
6      `ServerName`     TEXT NOT NULL
7  );
```

Figura 4.3: Código de creación de la tabla Scripts

```
CREATE TABLE `Parameters` (
    `Name`  TEXT NOT NULL,
    `Order` INTEGER NOT NULL,
    `ScriptId`  INTEGER NOT NULL,
    `Type`  TEXT NOT NULL,
    CONSTRAINT `PK_Parameters` PRIMARY KEY(`Name`,`ScriptId`),
    FOREIGN KEY(`ScriptId`) REFERENCES `Scripts`(`Id`) ON DELETE CASCADE
);
```

Figura 4.4: Código de creación de la tabla Parameters

```
CREATE TABLE `Users` (
    `Username`  TEXT NOT NULL,
    `Password`  TEXT NOT NULL,
    `Name`  TEXT NOT NULL,
    `Email` TEXT NOT NULL,
    `Role`  TEXT NOT NULL,
    CONSTRAINT `PK_Users` PRIMARY KEY(`Username`)
);
```

Figura 4.5: Código de creación de la tabla Users

```
CREATE TABLE `Remote` (
    `UserName`  TEXT NOT NULL,
    `ServerName`    TEXT NOT NULL,
    `Password`  TEXT NOT NULL,
    `Temp`  TEXT,
    CONSTRAINT `PK_Remote` PRIMARY KEY(`UserName`,`ServerName`)
);
```

Figura 4.6: Código de creación de la tabla Remote

## 4.3. Sitemap

El sitemap constituye la estructuración de las páginas dentro del sitio web. En la Figura 4.7 se puede observar la estructura del sitio web, donde se pueden diferenciar las páginas que son accesibles únicamente por el administrador (en color azul) y las páginas accesibles tanto por los administradores como por los usuarios.

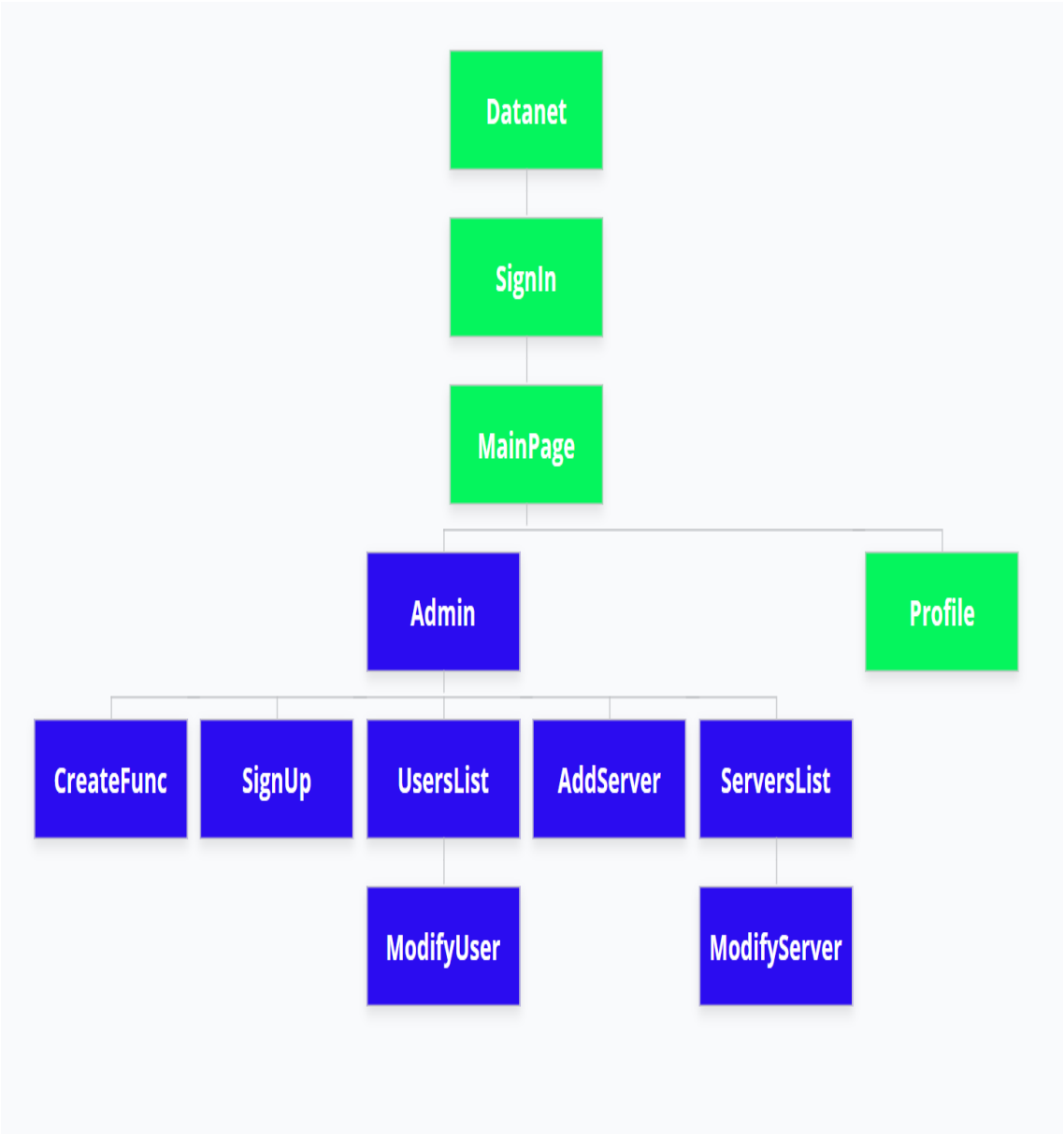


Figura 4.7: Sitemap del proyecto

## 4.4. Guía de estilos

### 4.4.1. Estructura

Como se puede ver en la Figura 4.8, la interfaz ha sido construida sobre una retícula de 3 filas:

- Cabecera: Contiene la imagen corporativa y el nombre de usuario del usuario logueado. Dicho usuario puede hacer clic sobre la imagen corporativa para ir a la página principal, puede hacer clic sobre su nombre de usuario para acceder y modificar su perfil y puede hacer clic sobre el texto Logout para salir de su sesión.
- Cuerpo: Proporciona el contenido de la aplicación web.
- Pie de página: Proporciona información sobre la fecha de creación y el copyright de la aplicación web.


Figura 4.8: Página de ejemplo de la aplicación web.

### 4.4.2. Uso de colores

Se ha optado por el uso de una gama cromática suave, donde el color predominante es un gris suave combinado con el color blanco como color de contraste. El uso del color gris permite combinarlo con cualquier color ya que se tratar de un color neutro (como el blanco y

el negro) y es muy adecuado para el uso en sitios web profesionales donde el objetivo no es atraer la atención de los usuarios sobre el contenido, ya que el objetivo es proporcionar una funcionalidad para el uso interno de la empresa. En la Figura 4.9 se puede observar cuales han sido los colores predominantes en la aplicación web, donde además del gris y blanco mencionados anteriormente, también se incluye un tono de azul para diferenciar enlaces a nuevas páginas y el negro usado como elemento de contraste en algunos elementos como por ejemplo en el uso de cabeceras en las tablas.

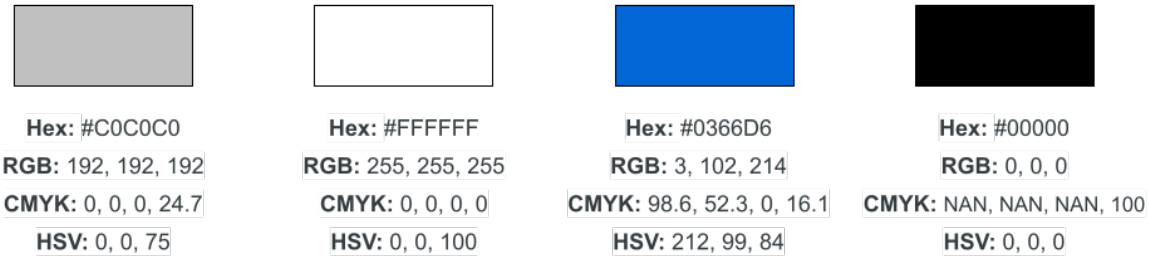


Figura 4.9: Gama de colores utilizados.

### 4.4.3. Elementos

El uso de Bootstrap ha facilitado la implementación de diferentes elementos en la interfaz web. Se han utilizado muchos de los elementos con estilos predefinidos que proporciona Bootstrap como por ejemplo el conjunto de botones, que se puede observar en la Figura 4.10, los elementos utilizados en los formularios y varios elementos decorativos.


Figura 4.10: Conjunto de botones con estilo predefinido de Bootstrap[1].

# Capítulo 5

# Implementación y pruebas

## 5.1. Patrón de diseño utilizado en el sitio web

Una vez seleccionadas las tecnologías con las que se procede a implementar el proyecto, se debe investigar las posibilidades que éstas nos ofrecen. ASP.NET Core combina con el patrón de diseño MVC para ofrecer a los usuarios un desarrollo de aplicaciones web muy interesante y potente. Un patrón de diseño MVC(Modelo-Vista-Controlador) proporciona una estructuración bien diferenciada[5]:

- **Modelo:** Representa un conjunto de clases que son las encargadas de administrar todos los datos del sistema: estructuras de bases de datos, validación, estados de sesión y control, etc. También encapsula todos los métodos que permiten interactuar con los datos.
- **Vista:** Responsable de la interfaz gráfica de usuario: formularios, botones, etc. Controla la forma en la que se muestran los datos y cómo los usuarios interactúan con ellos.
- **Controlador:** Encargado del tratamiento de eventos y la lógica de la aplicación. Acepta peticiones, solicita los datos y los prepara como respuesta, con lo cual tiene interacción tanto con el modelo como con la vista.

Como se puede ver en la Figura 5.1 hay una interacción entre estos tres componentes. Este patrón de diseño es muy importante a la hora de diseñar aplicaciones grandes, donde se cuenta con un número grande de vistas y controladores con muchas dependencias, cosa que proporciona una complejidad de implementación elevada. Una de las grandes preocupaciones en ingeniería del software que todo buen programador debe tener es el cumplimiento de los principios básicos de la programación, los llamadas principios SOLID, que son los siguientes:

**S:** Single Responsibility Principle (SRP)
**O:** Open/Closed Principle (OCP)
**L:** Liskov Substitution Principle (LSP)
**I:** Interface Segregation Principle (ISP)
**D:** Dependency Inversion Principle (DIP)

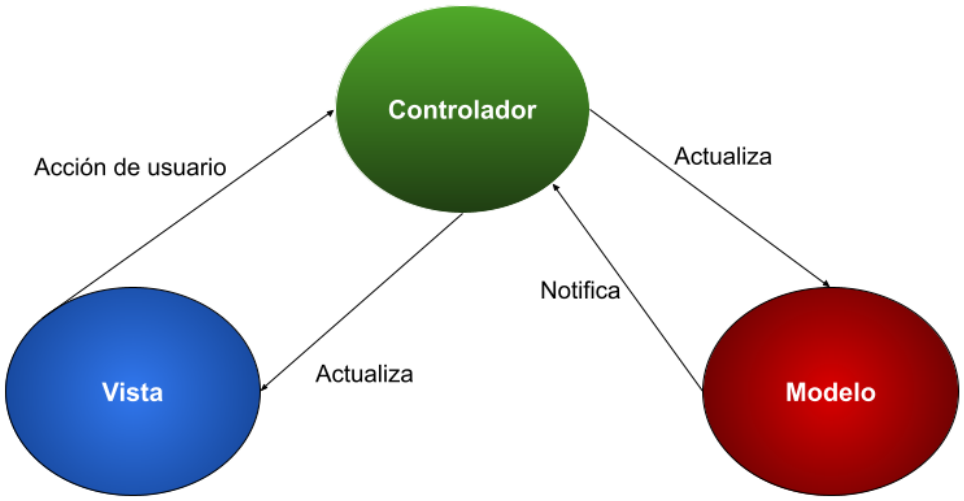


Figura 5.1: Esquema del patrón de diseño Modelo-Vista-Controlador

Cuando un controlador cuenta con un número elevado de dependencias da como resultado que muchas veces se incumpla los dos primeros principios. En primer lugar, se podría dar el caso de incumplir la condición que impone el primer principio de que una clase tan solo podrá cambiar por un único motivo; y en segundo lugar también se incrementa la posibilidad de incumplir el segundo principio que indica que una clase debe poder extenderse siempre pero sin tener que ser modificada. Para evitar incumplir los principios SOLID, se ha decidido utilizar Razor Pages, que es una sintaxis de programación que forma parte del patrón MVC ya que usa sus mismas características aunque tiene una estructuración diferente que soluciona dichos problemas con los principios SOLID ya que encapsula la lógica que se encuentra del lado del servidor para una página que contenga la lógica de una parte de la aplicación web.

## 5.2. Estructura del proyecto

### 5.2.1. Razor Pages

[17]Las Razor Pages, introducidas en la versión 2.0 de ASP.NET Core, parecen ser una buena opción a tener en cuenta, evaluando qué es lo que más se adapta a cada una de las características que debe tener cada aplicación web. Cuando se trata de aplicaciones web con una complejidad y funcionalidad menor, como es el caso de este proyecto, parece ser que las Razor Pages ofrecen grandes ventajas respecto al patrón MVC clásico:

- Enrutamiento: Coincide con las estructuración de la carpeta. Eso provoca un mejor acceso a cada uno de los elementos de la aplicación web además de una jerarquía de páginas más visual y ordenada.

- Código más limpio: Evita tener un controlador que se ocupe de gestionar un gran número de acciones (generando así una clase de difícil lectura). Nos proporciona un PageModel (que correspondería con las funciones propias del controlador y del modelo) que siempre se va a encontrar en la misma carpeta que la vista.

- Sencillez de pruebas y tratamiento de errores: Todas las acciones del PageModel ocurren cuando se ejecuta una acción GET o POST con las que se trabajan, con lo cuál, el tratamiento y búsqueda de errores estarán siempre relacionados con una de las dos acciones y será muy fácil de localizar.

Las Razor Pages cuentan con dos ficheros que ofrecen una funcionalidad característica:

- Razor Page : Corresponde con la vista del modelo MVC. Se encarga de mostrar los datos a los usuarios y de interactuar con ellos, posibilitando añadir datos al sistema. Es un fichero .cshtml, que corresponde con un fichero al cuál se le permite añadir contenido del lenguaje de programación C# dentro del lenguaje de marcado HTML.

- Page Model: Es la combinación del modelo y los métodos del controlador. Dentro de esta página se define el modelo para la vista asociada y también se dispone de handlers, que corresponden con los métodos del controlador. Es un fichero .cshtml.cs, que contiene código escrito en el lenguaje de programación C#.

### 5.2.2. Enrutamiento y archivos

Uno de los aspectos a tener en cuenta a la hora de desarrollar un sitio web es cómo enlazar las URL con los recursos alojados en el servidor. En ASP.NET Core se mapean las URL con los ficheros. Para ello, se parte de una carpeta raíz, que es donde se alojaran todo el conjunto de Razor Pages. Por defecto esta carpeta se llama Pages y se encuentra dentro del directorio raíz del proyecto, aunque se puede cambiar la ubicación de dicha carpeta.

Como se puede apreciar en la Tabla 5.1, el fichero index.cshtml es el documento por defecto, con lo cuál si intentamos acceder a una URL con el nombre de un directorio alojado en el directorio raíz Pages, automáticamente se mapeará la petición URL con el fichero index.cshtml alojado en ese directorio (en el caso de que exista).

| **URL** | **Ruta del servidor** |
|---|---|
| www.datanetproject.com | /Pages/index.cshtml |
| www.datanetproject.com/mainpage | /Pages/mainpage.cshtml |
| | /Pages/mainpage/index.cshtml |
| www.datanetproject.com/Admin/CreateFunctionality | /Pages/Admin/CreateFunctionality.cshtml |

Tabla 5.1: Ejemplos de enrutamiento del sitio web

En cuanto a los archivos del proyecto, destacan varias carpetas y ficheros[15], que han sido utilizados y modificados a lo largo de todo el proyecto:

- Pages : Como se ha indicado previamente, es la carpeta raíz en la cuál se alojan las Razor Pages. También contiene archivos auxiliares, que configura los elementos que van a ser comunes en la interfaz de todas las páginas.
- wwwroot: Contiene los archivos necesarios en tiempo de ejecución. Se alojarán ficheros como HTML, CSS, Javascript, imágenes, etc. En el proyecto tan solo se hace uso de ficheros Javascript, que serán los responsables de producir la interacción con el usuario directamente en el navegador. El código HTML viene proporcionado por las páginas razor y el código CSS se obtiene principalmente a través de la librería Bootstrap.
- Startup.cs: Contiene código que configura el comportamiento de la aplicación. En el proyecto se establece la configuración de uso del modelo MVC y de RazorPages, además de establecer la ruta donde se encuentra alojada la base de datos y cual será el sistema gestor de base de datos a través del cual se va a modelizar dicha base de datos. También se indica que se van a utilizar las sesiones para realizar un control de autenticación en la aplicación web.
- database.db: Almacena todos los datos de la aplicación web que no deben eliminarse.

## 5.3. Extensiones utilizadas en el entorno de desarrollo

Para la realización del proyecto se precisan de varias extensiones o paquetes que van a ser instalados en Visual Studio. Para ello se va a utilizar NuGet[1], que es un administrador de paquetes de .NET además de ser el repositorio central de paquetes que usan la gran mayoría de autores de contenido (entre los cuáles se encuentran Microsoft, del cuál serán la mayoría de paquetes usados) y usuarios que desean utilizar sus paquetes. Los paquetes utilizados en el proyecto son los siguientes:

- **Microsoft.PowerShell.SDK:** Para poder realizar cualquier tipo de operaciones PowerShell, hay que tener instalada la interfaz PowerShell. Con este paquete, se instala el kit de desarrollo de software correspondiente a PowerShell, con lo cuál gracias a él se van a poder ejecutar comandos y scripts.
- **System.Management.Automation:** Permite iniciar sesiones y ejecutar scripts PowerShell. Es el pilar fundamental del proyecto, ya que el objetivo central de éste es la posibilidad de ejecutar comandos PowerShell dentro de la aplicación web.
- **Microsoft.EntityFrameworkCore:** Framework que permite acceso a datos mediante modelos, que pueden ser generados manualmente o a través de una base de datos[11].
- **Microsoft.EntityFrameworkCore.Sqlite:** Paquete utilizado en EntityFrameworkCore que permite el uso de una base de datos SQLite[2]. Aunque hay otras opciones y otros

[1] https://www.nuget.org/

[2] Librería escrita en lenguaje C que implementa un sistema gestión de bases de datos sencillo, rápido y con una gran fiabilidad

sistemas de gestión de base de datos más completos, se ha acordado el uso de SQLite ya que la base de datos del proyecto no es muy extensa ni necesita muchas funcionalidades.

- **Microsoft.EntityFrameworkCore.Tools:** Paquete utilizado en EntityFrameworkCore que permite realizar diversas acciones que permiten la interacción entre la base de datos y los modelos asociados.

## 5.4. Ejecución de comandos PowerShell

Uno de los grandes objetivos del proyecto, como se ha mencionado anteriormente, es la ejecución de comandos PowerShell dentro de la aplicación web. En las extensiones vistas en el apartado 5.3, se ha mencionado dos de ellas que corresponden con la ejecución de comandos PowerShell. En primer lugar, la extensión Microsoft.PowerShell.SDK es la que va a reconocer los comandos PowerShell, su función es que podamos ejecutar comandos/scripts PowerShell y que el sistema sepa como interpretar esos comandos y realice las acciones pertinentes.

La otra extensión es System.Management.Automation, que es la extensión que permite crear sesiones de PowerShell y ejecutar comandos/scripts dentro de dicha sesión. Lógicamente, para poder crear sesiones de PowerShell, previamente debemos haber instalado el sdk de PowerShell.

Como se puede ver en la Figura 5.2, se va a crear una instancia de PowerShell gracias a la clase PowerShell[14] que nos ofrece la extensión System.Management.Automation. Cuando se ejecuta una instancia de PowerShell, se crea un runspace local con una configuración por defecto. Esa configuración por defecto trae una directiva de ejecución restringida[9] donde se permiten los comandos individualmente pero no se permiten scripts, es decir, no se permite la ejecución de todos los archivos de script: ni archivos de formato y configuración (.ps1xml), scripts de módulo (.psm1) ni los perfiles PowerShell (.ps1).

Los scripts que se van a utilizar en el proyecto son scripts de tipo perfil PowerShell, con lo cuál la directiva de ejecución por defecto no sirve. Para ello se crea el Runspace Pool, al cuál se le va a asignar una directiva de ejecución sin restricciones, donde se permite la ejecución de scripts sin necesidad de ser firmados, aunque por esta condición se debe tener en cuenta la posibilidad de la ejecución de scripts maliciosos. Como el proyecto se va a desarrollar dentro del ámbito interno de la empresa y solo los usuarios administradores tienen acceso a subir ficheros script PowerShell, no debería haber ningún inconveniente en el uso de esta directiva de ejecución. Una vez creado el Runspace Pool y la instancia de PowerShell, se va a configurar dicha instancia para que sea ejecutada bajo el RunspacePool y, por lo tanto, se puedan ejecutar scripts PowerShell dentro de la instancia.

A dicha instancia se le va a pasar el script que desea ejecutar la empresa (con un formato característico adecuado) y una estructura de datos, que contendrá los valores asociados a cada una de las variables requeridas por el script. En este caso se ha decidido utilizar un diccionario, que es una estructura de datos muy útil y eficaz. Como uno de los requisitos del proyecto es que el usuario del sitio web pueda introducir tanto texto como ficheros, se ha creado una clase Parámetro que diferenciará si para cada variable el usuario ha introducido texto o ha subido un fichero al servidor.

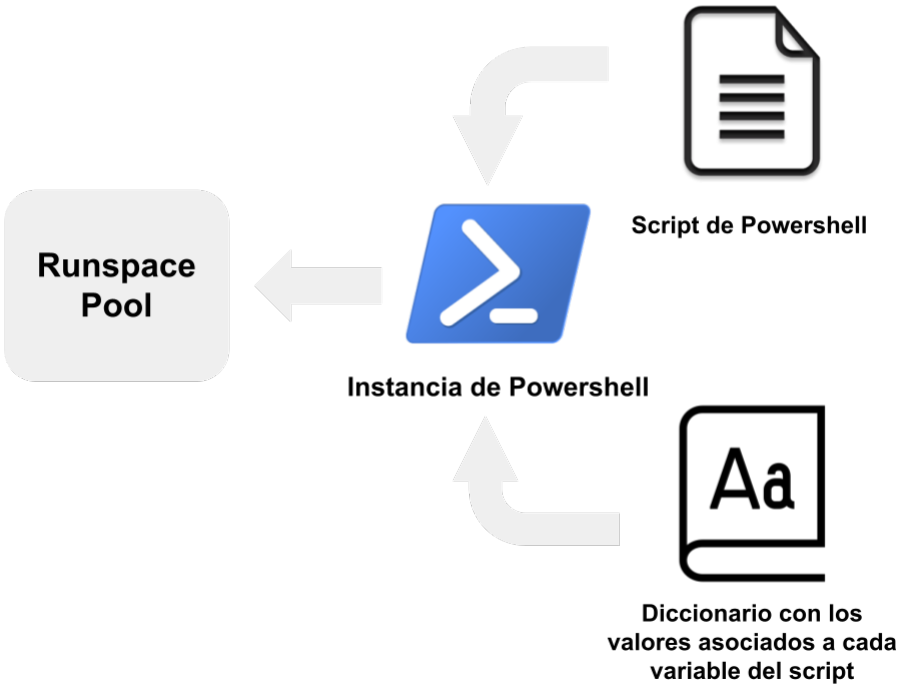


Figura 5.2: Esquema del proceso de ejecución de comandos PowerShell

La clase Parámetro se estructura de la siguiente manera:

- **Tipo:** Indica si el valor que hace referencia a esa variable es texto o es un fichero.
- **Valor:** En el caso de ser texto, se almacena el valor que el usuario ha introducido en el formulario para dicha variable. En el caso de ser un tipo fichero, se almacena el nombre local del fichero que el usuario ha seleccionado para subir al servidor web.
- **ValorWeb:** Se indica el nombre del fichero almacenado en la carpeta temporal del servidor web. Se utiliza un identificador único global (GUID) para nombrar a cada uno de los ficheros que el usuario suba al servidor web (a excepción de los scripts que son subidos por los administradores y que proporcionan la funcionalidad del sitio web). En el caso del tipo texto, este valor es null.
- **ValorServidor:** Cada fichero subido al servidor web, deberá ser posteriormente copiado a la carpeta temporal del servidor remoto mediante el uso de comandos PowerShell. Se almacena la ruta de la carpeta temporal del servidor en donde se aloja el fichero copiado desde el servidor web. En el caso del tipo texto, este valor es null.

Por lo tanto, en el caso de que el parámetro introducido sea tipo texto, tan solo interesa obtener cuál ha sido el texto introducido por el usuario. En cambio, si se trata de un fichero, se deberá incluir en el diccionario la ruta del servidor en donde se ha copiado el fichero, ya que tanto el script como los ficheros que suban los usuarios van a ser ejecutados en el servidor. En la

Figura 5.3 se puede observar que tanto $param1 como $param3 son variables de tipo texto (y por lo tanto, se guardará en el diccionario el valor contenido dentro del campo Valor de la clase Parámetro); y $param2 es un fichero, con lo cuál se guardará en el diccionario la ruta en donde se encuentra alojado dicho fichero en el servidor. Finalmente, al ejecutar el script, se mostrará por pantalla primero el texto guardado en $param1; a continuación se mostrará el contenido del fichero guardado en la ruta indicada por $param2; y finalmente se mostrará el texto guardado en $param3.

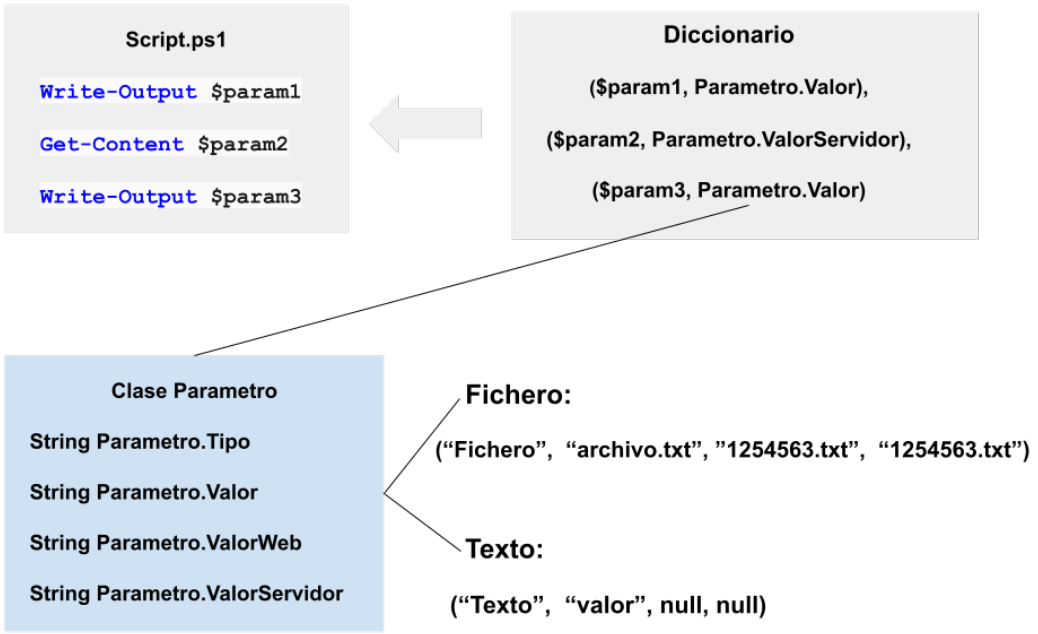


Figura 5.3: Ejemplo de objetos que recibe una instancia PowerShell

## 5.5. Conexión entre servidor web y servidor

Como se ha indicado en el apartado 5.4, como máximo se necesitará conectar con el servidor tres veces: para obtener la ruta de la carpeta temporal del servidor, para copiar los ficheros necesarios en cada que el script asociado haga referencia a ficheros, y para ejecutar el script.

De ahí se pueden obtener cuatro escenarios distintos:

- Se accede por primera vez al servidor y el script requiere el uso de ficheros adicionales: En este caso se necesita realizar las tres conexiones explicadas previamente.
- Se accede al servidor y el script requiere el uso de ficheros adicionales: En este caso ya se encuentra almacenada en la base de datos la carpeta temporal del servidor, con lo cuál

esa conexión la evitamos. Tan solo se deberá conectar para realizar el copiado de ficheros y para ejecutar el script.

- Se accede por primera vez al servidor y el script tan solo requiere texto: Se necesita realizar una conexión para obtener la ruta de la carpeta temporal del servidor y otra conexión para la ejecución del script.
- Se accede al servidor y el script tan solo requiere texto: Únicamente se necesita realizar la conexión para la ejecución del script.

Para poder realizar la conexión mediante PowerShell entre servidor web y el servidor para realizar las operaciones hay que realizar un conjunto de pasos, que se han incluido en el Anexo A, formando parte de un manual de uso donde también se incluyen los pasos a seguir para el correcto uso y funcionamiento de la aplicación web.

Una vez se tenga la posibilidad de conectar en remoto al servidor, se va a ejecutar el comando Invoke-Command de PowerShell, como se puede ver en la Figura 5.4. Antes de lanzar el comando en remoto se debe iniciar una sesión de PowerShell en el equipo remoto.

```
1  $credentials = New-Object System.Management.Automation.PSCredential -argumentList $user,
2  (ConvertTo-SecureString -String $password -AsPlainText -Force)
3  $session = New-PSSession -ComputerName $server -Credential $credentials
4  $psSession = Get-PSSession
5  Invoke-Command -Session $psSession -ScriptBlock { [Environment]::GetEnvironmentVariable("Temp","Machine") }
```

Figura 5.4: Ejecución de un comando PowerShell en remoto

Para iniciar sesión, se debe crear un objeto PSCredential que guardará el usuario y contraseña del servidor, que se van a obtener de una tabla de la base de datos que contiene todas las credenciales para cada uno de los servidores disponibles. A continuación se crea un nuevo objeto PSSesion con el nombre del servidor y sus credenciales. Cabe mencionar que el usuario asociado a dicho servidor debe tener privilegios de Administrador ya que en caso contrario se le va a denegar el acceso a la ejecución de scripts.

Una vez creada la sesión, ya se puede ejecutar el script sobre ella. En el caso del script de la Figura 5.4, el script a ejecutar devolverá la ruta de la carpeta temporal del sistema introducido en la variable $server. Para establecer la conexión en remoto siempre se va a usar una nueva sesión hacia el servidor remoto destino, pero el comando de obtención de la carpeta temporal solo se realizará cuando sea necesario (subida de fichero) y cuando la ruta de ésta no se encuentre almacenada en la base de datos.

Para la obtención de las diferentes salidas posibles, se ha suscrito al objeto PowerShell creado para el lanzamiento de scripts a los diferentes flujos de salida [10] que se generan. En este caso

se ha suscrito a tres flujos de salida: éxito, advertencia y error, aunque existen varios más. El ejemplo se puede ver en la Figura 5.5 donde se puede observar un error devuelto por PowerShell mediante el flujo de salida de error. Esto facilita muchísimo a quien ejecuta el script que pueda


Figura 5.5: Resultado de ejecución de un script con errores de Powershell

entender que ocurre en cada caso, ya que por defecto solo se muestra el flujo de éxito de ejecución y, por lo tanto, en caso de que hubiera algún error de ejecución el usuario no sabría que ocurre ya que no se mostraría ningún tipo de mensaje.

## 5.6. Seguridad

Como se ha comentado en la sección Requisitos de seguridad, es necesario la encriptación de los datos antes de ser introducidos en la base de datos. Se han diferenciado dos procesos de encriptación:

- **Encriptación de contraseñas de usuarios del sitio web:** En este caso, tan solo se necesita almacenar la contraseña del usuario en la base de datos y comprobar que cuando se intenta iniciar sesión la contraseña introducida para el usuario es exactamente la misma que la contraseña almacenada en la base de datos. Por lo tanto, en ningún momento se necesita desencriptar la contraseña del usuario para ser usada en la aplicación, tan solo se necesita realizar esa verificación. Para realizar el proceso se utiliza la función hash[3]

[3]Función hash: Algoritmo de criptografía que mapea un input de cualquier tamaño en un output cuyo contenido es totalmente diferente al input.

proporcionada por BCrypt[4] que permite encriptar la contraseña del usuario para luego almacenarla en la base de datos. Luego, cuando el usuario vaya a iniciar sesión, introduce una contraseña sobre la cuál se va a aplicar la función hash y se va a verificar que el output obtenido corresponde con el valor almacenado en la base de datos.

- **Encriptación de credenciales de servidores y contenido de los scripts:** Estos dos casos son diferentes al anterior, ya que aquí si que es necesario obtener los datos y desencriptarlos para interactuar con ellos. En este caso se ha optado por el uso de una clase[5] proporcionada por dotnet que nos proporciona todos los métodos necesarios para la implementación del estándar de cifrado avanzado (AES) que permite realizar la encriptación de texto mediante el uso de una clave y la desencriptación del texto con el uso de la misma llave.
  El proceso se puede observar en la Figura 5.6, donde en el primer uso de la aplicación se va a solicitar la creación del primer usuario (que lógicamente tendrá permisos de administrador) y que además se indique la clave de encriptación. El problema que se ha planteado aquí es donde se puede almacenar la clave de encriptación y desencriptación para que ningún usuario no autorizado pueda acceder a ella. Se ha decidido que el lugar más seguro para almacenar la clave es un fichero alojado en una carpeta oculta dentro de la raíz del disco duro, con lo cuál se conseguirá mantener oculto el fichero y además tan solo tendrá acceso a él el administrador del servidor web.

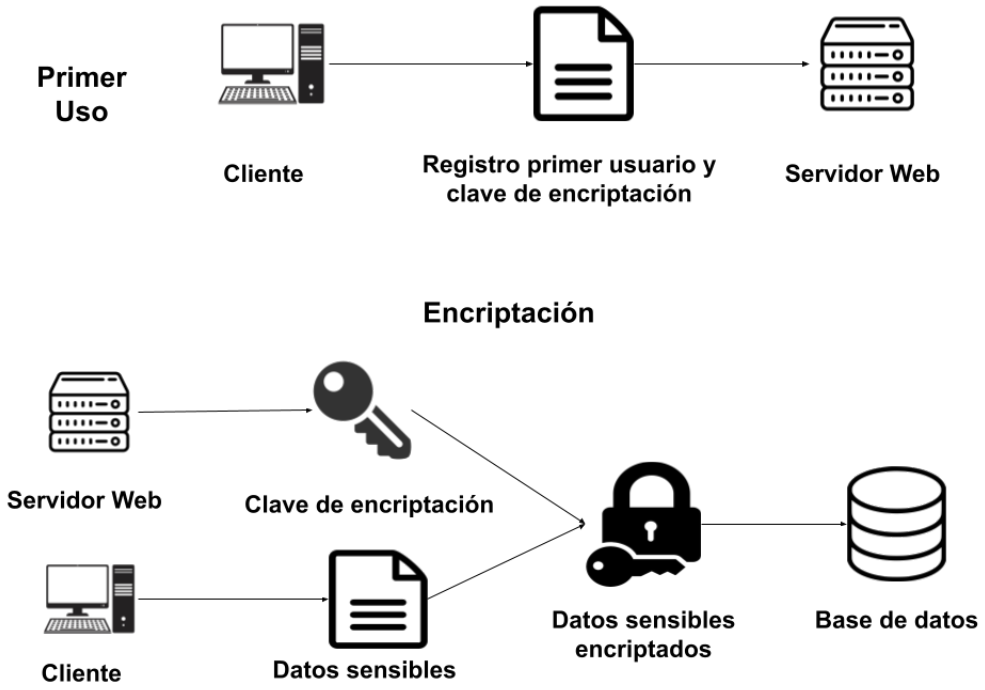


Figura 5.6: Proceso de obtención de la clave de encriptación y encriptación.

[4]https://github.com/neoKushan/BCrypt.Net-Core
[5]https://docs.microsoft.com/es-es/dotnet/api/system.security.cryptography.aes?view=net-5.0

El proceso de encriptación se produce cuando un usuario de la aplicación web introduce datos sensibles, ya sean credenciales de un servidor remoto o un script. En este caso, se procede a obtener la clave de encriptación alojada en un fichero en el servidor web y se procede a encriptar los datos sensibles introducidos para posteriormente ser guardardos en la base de datos de la aplicación de forma segura.
El proceso de desencriptación consiste en obtener de la base de datos los datos sensibles (que siempre se encontrarán encriptados) y obtener del servidor web la clave de encriptación para poder obtener el contenido que se está solicitando. Por ejemplo, para realizar la conexión a un servidor remoto, es necesario obtener de la base de datos el nombre del servidor, el nombre del usuario con el cuál se va a iniciar la sesión remotamente y su contraseña. La contraseña del usuario se va a encontrar encriptada con lo cuál va a ser necesario desencriptar esa información con el fin de proporcionar de manera correcta los datos de conexión al servidor.

## 5.7. Mejoras no implementadas

Mediante las diversas reuniones realizadas a lo largo de la estancia, se fueron acotando los objetivos y tareas del proyecto a medida que se iba avanzando en el proyecto. Debido a la complejidad del proyecto y el uso de herramientas y tecnologías no usadas previamente, se quedó pendiente de realizar pequeñas mejoras en el proyecto que no influyen en el objetivo principal del proyecto pero que habrían supuesto un extra de experiencia y una ampliación de los conocimientos adquiridos. Como la estancia en prácticas es de una duración determinada, se consiguió definir y establecer las bases de las mejoras que se querían implementar, aunque se llegó al número de horas máximo y no se pudo realizar.

El objetivo de dichas mejoras consistía en convertir el proyecto realizado en una imagen que podría ser utilizada posteriormente como base en un contenedor de Docker. Esto no es determinante en la consecución del objetivo principal de conseguir la ejecución en remoto de scripts PowerShell en un servidor web, aunque sí que resultaba interesante para profundizar un poquito más en el funcionamiento de Docker y en la exportación de proyectos para ser utilizados mediante este software.

A continuación se muestran cuales han sido las tareas que se han querido realizar aunque no se hayan podido implementar de manera completa.

### 5.7.1. Convertir código en imagen para ejecutar en contenedor

Una vez se ha desarrollado el código del sitio web, se debe compilar y convertir en una imagen para que pueda ser utilizado por Docker. Para ello Visual Studio proporciona soporte para Docker, con lo cual va a crear automáticamente el fichero Dockerfile [3], que es un documento que contiene los comandos necesarios para la creación de la imagen.

En la Figura 5.7 se puede observar el contenido del fichero Dockerfile generado automáticamente para el proyecto. En la línea 6 y 11 se puede ver el uso de la instrucción FROM. Eso indica que se van a crear dos imágenes: la primera imagen corresponde con la imagen base sobre

la cual se va a lanzar el proyecto; y la segunda imagen corresponde con la imagen utilizada para el proceso de desarrollo del proyecto. En este caso, el uso de la imagen base es ASP.NET Core 5.0 Runtime, que es el entorno de ejecución que permite lanzar aplicaciones. Para el proceso de desarrollo del proyecto, se ha utilizado la imagen .NET Core SDK, que permite el desarrollo, la compilación y el testeo de aplicaciones .NET Core.

```
6  FROM mcr.microsoft.com/dotnet/aspnet:5.0 AS base
7  WORKDIR /app
8  EXPOSE 80
9  EXPOSE 443
10
11 FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build
12 WORKDIR /src
13 COPY ["PROYECTOLOCAL/PROYECTOLOCAL.csproj", "PROYECTOLOCAL/"]
14 RUN dotnet restore "PROYECTOLOCAL/PROYECTOLOCAL.csproj"
15 COPY . .
16 WORKDIR "/src/PROYECTOLOCAL"
17 RUN dotnet build "PROYECTOLOCAL.csproj" -c Release -o /app/build
18
19 FROM build AS publish
20 RUN dotnet publish "PROYECTOLOCAL.csproj" -c Release -o /app/publish
21
22 FROM base AS final
23 WORKDIR /app
24 COPY --from=publish /app/publish .
25 ENTRYPOINT ["dotnet", "PROYECTOLOCAL.dll"]
```

Figura 5.7: Contenido del fichero Dockerfile del proyecto web

Una vez creado el documento, ya se puede compilar el proyecto, para ello se usa el siguiente comando:

**docker build -t nombreImagen .**

donde el . indica que lo que se quiere compilar es el proyecto ubicado en la ruta actual y nombreImagen es el nombre que se le asignará a la imagen generada del proyecto. Cuando se ejecute el comando, se crearan dos archivos que corresponden con las dos imágenes mencionadas anteriormente y que ya estarán listas para ser usadas por Docker.

### 5.7.2. Funcionamiento del contenedor

Una vez creada la imagen con la cuál se va a ejecutar el contenedor se deben detallar cuales serán las características de ese contenedor. Docker cuenta con un mecanismo llamado volumen, que es una carpeta física alojada en este caso en el servidor web donde se almacenan datos que también estarán presentes en el contenedor.

El uso de volúmenes es una herramienta muy importante en el uso de Docker ya que permite mantener todos los datos generados en un contenedor cuando se para, ya que en caso contrario se pierden todos los datos. En el proyecto se ha utilizado un volumen para la base de datos, que almacena prácticamente todos los datos de la aplicación web, ya sea usuarios, scripts, datos de

servidores,etc por lo tanto, no interesa que se pierda esta información cada vez que se pare el servidor. Aunque la intención del uso de un servidor es que nunca se pare su ejecución, también se debe tener en cuenta que el uso de volúmenes facilita que se tenga acceso a los datos del servidor web y posibilite una portabilidad sencilla en caso de ser necesaria. En la Figura 5.8 se muestra cuál habría sido el proceso y la estructura final del proyecto si se hubiese conseguido incluir la aplicación web en un contenedor Docker.

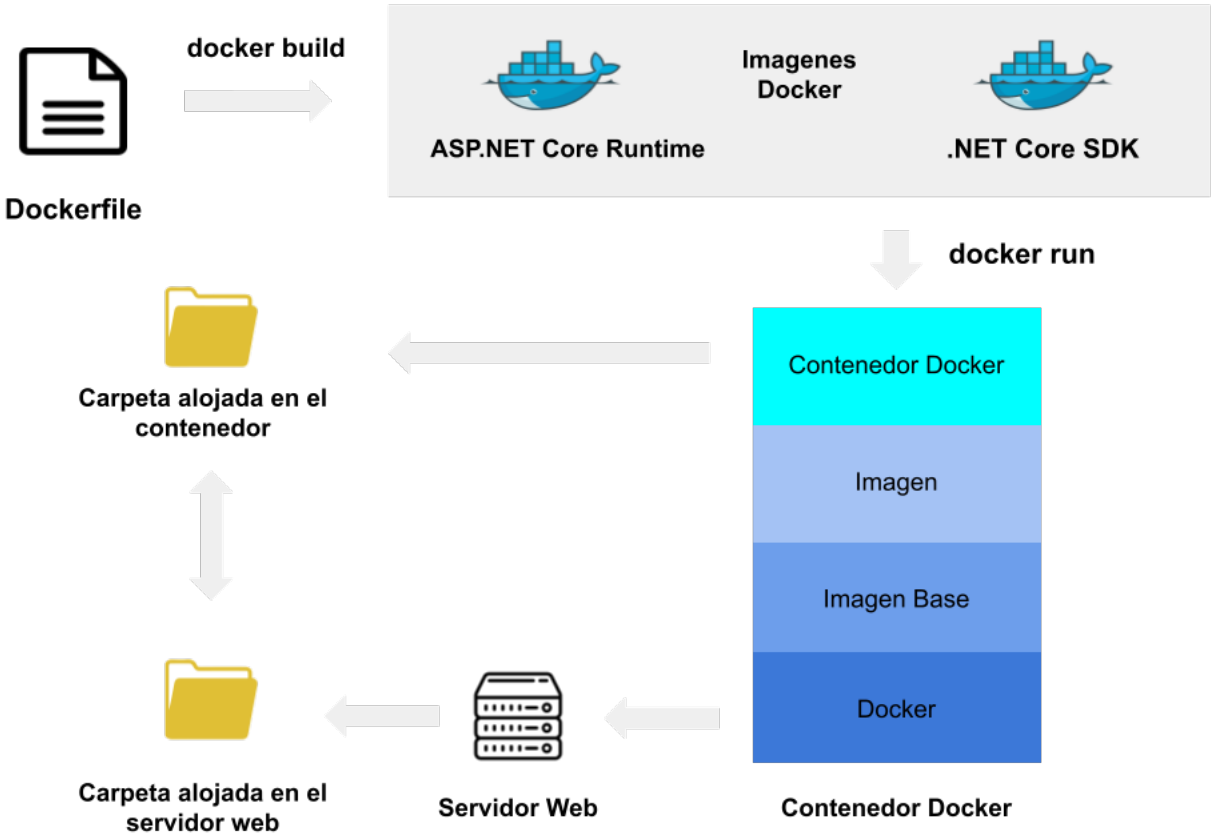


Figura 5.8: Estructura del proyecto dentro de un contenedor Docker

# Capítulo 6

# Verificación y validación

Antes de realizar la entrega del proyecto se deben realizar diversas pruebas de verificación y validación para comprobar que todo funciona correctamente.

## 6.1. Verificación

Mediante el proceso de verificación, se evalúa que el sistema funciona correctamente, es decir, que se cumplan todas los objetivos establecidos. Se pretende verificar la coherencia de la información, detectar los errores surgidos durante la programación y que todas las operaciones que se pretender realizar tengan un comportamiento adecuado.

Para realizar la verificación se pueden realizar tanto técnicas estáticas como técnicas dinámicas.

- **Técnicas estáticas:** En primer lugar se ha realizado un análisis del código donde se ha llevado una revisión exhaustiva del código en busca de defectos. Al tratarse de un proyecto cuyo código debe ser compilado antes de poder ser ejecutado, el propio compilador facilita mucho esta operación ya que cualquier fallo de sintaxis será reconocido por éste. La otra técnica estática utilizada es la verificación formal que consiste en demostrar que el programa cumple con una especificación formal dada. En este caso, las especificaciones formales más destacadas a probar podrían ser los casos de usos definidos en el apartado de análisis.
- **Técnicas dinámicas:** Consiste en experimentar cuál es el comportamiento del proyecto frente a cada una de las funcionalidades que presenta. Para obtener datos de dicha experimentación, se ha hecho uso de varias herramientas diferentes. En primer lugar, se ha utilizado la consola de Visual Studio, que permite tener un registro de errores y de información útil con el fin de entender qué es lo que ocurre en cada momento de la ejecución. En segundo lugar, se ha utilizado la herramienta de depuración de Visual Studio, que debería ser la herramienta más importante de un programador a la hora de realizar pruebas. Permite la ejecución del proyecto mientras permite analizar el código linea por

linea y entender lo que ocurre en cada momento con el uso de varias de las herramientas disponibles como los punto de interrupción. Cuando se crea un punto de interrupción se permite observar varios aspectos de la ejecución del código, como por ejemplo ver un valor almacenado en un objeto en el instante que marca el punto de interrupción, como se puede observar en la Figura 6.1 donde se observa un punto de interrupción en la línea 113 y se puede observar los valores almacenados para cada uno de los atributos de un objeto modelo. Dicho objeto pertenece a la clase Remote y cuenta con 4 atributos(Password, ServerName, Temp y UserName) de los cuáles se pueden observar sus valores en ese instante. Además cuenta con muchas más opciones que resultan muy interesantes y deberían tenerse en cuenta siempre a la hora de realizar el proceso de verificación.

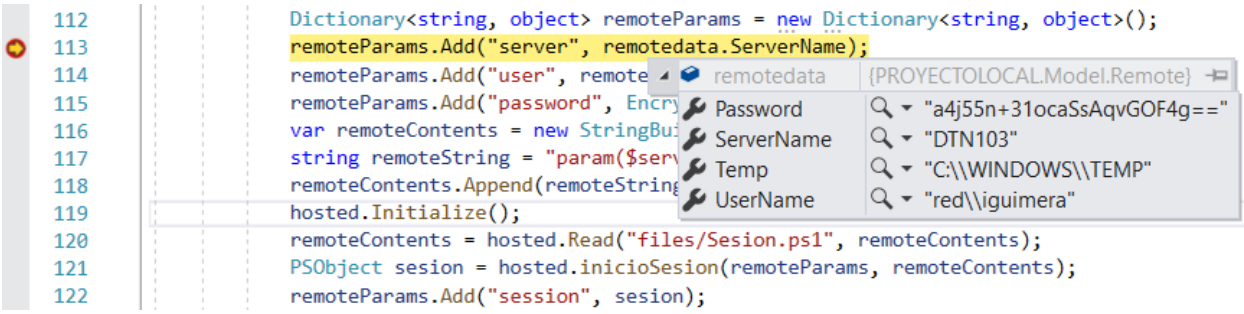


Figura 6.1: Ejemplo de depuración de código en Visual Studio

Finalmente, se ha utilizado los flujos de datos de PowerShell para obtener información útil respecto a la ejecución de comandos PowerShell, ya que es necesario diferenciar entre los errores obtenidos por el uso inadecuado del lenguaje de programación utilizado y los errores obtenidos en PowerShell. Estos fallos de PowerShell se han podido gestionar gracias a los Streams proporcionados por el SDK de PowerShell utilizado, que permitirá que cada vez que se ejecute un comando de PowerShell, no solo se obtengan los resultados cuando el comando funciona correctamente, sino que también se obtengan los mensajes de aviso o error producidos.

## 6.2. Validación

La validación del software busca evidenciar que todas aquellas funciones establecidas en el proyecto van a generar de manera consistente el resultado esperado. Para ello se usan varios métodos de validación, aunque cuando se trata de una aplicación web como es el caso de este proyecto, los métodos de validación más usados son la validación de la seguridad y la validación de interfaces.

### 6.2.1. Validación de la seguridad

Cuando se habla de validar la seguridad, se pretende garantizar que cualquier persona o sistema solo tenga acceso a los recursos para los cuales se le ha asignado permisos. Para ello se han usado las variables de sesión de HttpContext[1] que proporciona .NET con lo cual cuando un usuario inicia sesión en el sistema, se guardará una variable de sesión con su nombre de usuario (que siempre será un identificador único). Se debe valorar la seguridad del sitio web mediante dos aspectos:

- Acceso a contenido del sitio web sin autenticación: Se podría dar el caso que una persona o sistema no autenticado intentase acceder a funciones del sitio web (ya sean funciones de usuario o de administrador) mediante la inserción directa en el buscador de Internet de una URL de acceso restringido a usuarios logueados (en el caso de este proyecto todas las páginas están restringidas). Para ello se realiza en todas las páginas un control de sesión verificando que antes de acceder a la página exista una sesión Http iniciada, como se puede ver en la Figura 6.2, donde si no existe un valor asociado a la variable de sesión de nombre de usuario, se va a redirigir siempre hacia la página de inicio de sesión sin posibilidad de visualizar el contenido al cual se pretendía acceder.

```
3   @using Microsoft.AspNetCore.Http
4   @{
5       //get user session.
6       var session = @HttpContext.Session.GetString("username");
7       if (session == null)
8       {
9           Response.Redirect("SignIn");
10      }
11  }
```

Figura 6.2: Código de redirección cuando no se ha iniciado sesión

- Acceso a contenido reservado para usuarios con rol de administrador: Un usuario logueado en el sistema sin permisos de administrador podría intentar acceder a alguna de las funciones reservadas para usuarios con permisos de administrador, con lo cual hay que gestionar también esta posibilidad. Para ello, cuando se intente acceder a una página reservada para usuarios con esos privilegios, se deberá comprobar la variable de sesión (que haya un usuario logueado) y que además ese usuario tenga el rol de administrador mediante una consulta a la base de datos, como se puede observar en la Figura 6.3.

[1] https://docs.microsoft.com/en-us/dotnet/api/system.web.httpcontext.session?view=netframework-4.8

```
33      public bool IsAdmin()
34      {
35          var username = HttpContext.Session.GetString("username");
36          if (username == null) return false;
37          User user = _db.Users.SingleOrDefault(a => a.Username.Equals(username));
38          if (user.Role == "Administrador") return true;
39          return false;
40      }
```

Figura 6.3: Código para comprobar si el usuario es administrador

### 6.2.2. Validación de interfaces

Para asegurar que todos los datos introducidos por el usuario sigan el formato adecuado o que no se introduzcan valores en blanco cuando sean obligatorios de rellenar, se debe realizar una validación a nivel de interfaz. Con esto se asegura de que no haya errores con la interacción con el usuario. Está validación se realiza teniendo en cuenta dos aspectos:

- Introducción de valores en blanco: Todos los campos que el usuario debe rellenar son obligatorios (a no ser que se indique lo contrario), con lo cuál cuando un usuario envíe información, se va a comprobar que se ha rellenado toda la información y, en caso contrario, se notificará al usuario sin enviar al sistema la información rellenada. En el ejemplo mostrado en la Figura 6.4, se observa como se intenta añadir un nuevo servidor al sistema, introduciendo el usuario y contraseña, pero sin introducir el nombre del servidor. Cuando el usuario hace clic en Añadir nuevo servidor, no se podrá realizar ya que el nombre del sistema es un campo obligatorio (si se hubiese quedado en blanco únicamente el campo de carpeta temporal, si que hubiese sido válido ya que como se puede observar no es un campo obligatorio).

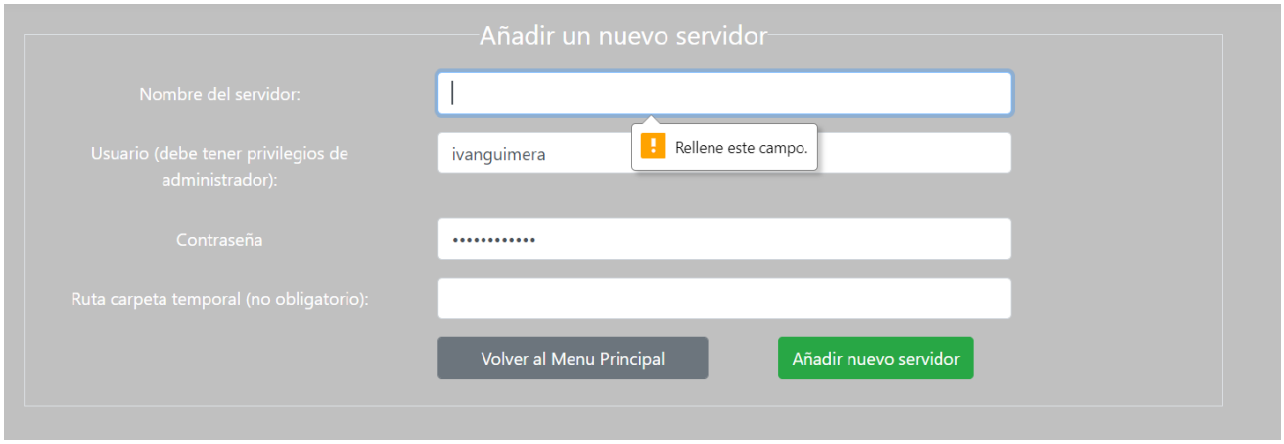


Figura 6.4: Ejemplo de envío de datos con un campo obligatorio vacío

- Valores repetidos en la base de datos: Todas las bases de datos cuentan con claves primarias, es decir, valores que deben ser únicos y que no pueden repetirse. En este caso, cuando se pretende añadir un nuevo registro en la base de datos, se va a comprobar previamente si la clave primaria del registro que se pretende introducir ya existe, en caso afirmativo se le indicará al usuario que no se ha podido añadir el registro y en caso negativo, se introducirá el nuevo registro en la base de datos. Como se puede observar en la Figura 6.5, se ha intentado introducir los datos de un nuevo usuario y se ha hecho clic en Crear Usuario. En la tabla de usuarios de la base de datos el nombre de usuario es la clave primaria con lo cuál si ya existe un registro con el nombre de usuario que se pretende introducir, se le indicará al administrador (que es quien puede crear nuevos usuarios) y no se creará el usuario con los datos que ha introducido.


Figura 6.5: Ejemplo de registro duplicado en la base de datos

# Capítulo 7

# Conclusiones

El desarrollo del proyecto se ha llevado de manera adecuada. Se han cumplido todas las expectativas a pesar de enfrentarme a un framework totalmente desconocido como es .NET Core y un lenguaje de programación poco utilizado como es C#, aunque para ello se haya tenido que invertir un número importante de horas en formación. No obstante, me ha parecido de gran utilidad enfrentarme a este framework, ya que es un proyecto desarrollado por Microsoft y eso casi siempre garantiza buenas proyecciones de futuro con lo cuál en todo momento me pareció apasionante empezar cuanto antes en el desarrollo de este proyecto.

En cuanto al ámbito profesional, me hubiese gustado poder realizar la estancia en prácticas de manera presencial aunque debido a la situación sanitaria causada por la pandemia del COVID-19 no ha sido posible. Aún así, la ayuda y formación obtenida por parte de los trabajadores de la empresa me han ayudado muchísimo a la hora de conseguir avanzar en la empresa; especialmente la ayuda de mi supervisor Luis Ruis que ha hecho que mi estancia en la empresa sea muy agradable, preocupándose en todo momento de cualquier problema que me haya podido surgir; y la ayuda de Jose Antonio López que se ha preocupado de solventar cualquier duda técnica que he podido tener y me ha proporcionado una gran formación durante todo el proyecto.

Finalmente, en el ámbito personal, la estancia en prácticas me ha servido principalmente para enfrentarme a un entorno laboral diferente marcado por el teletrabajo, que considero que es un ambiente bastante complejo ya que es fácil distraerse y requiere un alto nivel de organización y concentración que normalmente encuentras cuando acudes presencialmente a la empresa. No obstante, esta situación creo que proporciona un extra de madurez a nivel laboral y el haber conseguido desarrollar satisfactoriamente el proyecto me hace valorar mi estancia en prácticas de una manera muy positiva.

# Bibliografía


[1] Bootstrap. Buttons. https://getbootstrap.com/docs/4.0/components/buttons/. [Consulta: 04 de Junio de 2021].

[2] Datanet Consultores. Información de la empresa. https://www.datanetconsultores.es. [Consulta: 19 de Marzo de 2021].

[3] Docker. Dockerfile reference. https://docs.docker.com/engine/reference/builder/. [Consulta: 23 de Abril de 2021].

[4] Docker. What is a container? https://www.docker.com/resources/what-container. [Consulta: 19 de Marzo de 2021].

[5] Adam Altar Dragos-Paul Pop*. Designing an mvc model for rapid web application development. *Romanian-American University*, 2013.

[6] Indeed. Salario de un analista programador junior en españa. https://es.indeed.com/career/analista-programador-junior/salaries. [Consulta: 20 de Marzo de 2021].

[7] Indeed. Salario de un programador junior en españa. https://es.indeed.com/career/programador-junior/salaries. [Consulta: 20 de Marzo de 2021].

[8] MariaDB. Database design phase 2: Conceptual design. https://mariadb.com/kb/en/database-design-phase-2-conceptual-design/. [Consulta: 09 de Mayo de 2021].

[9] Microsoft. Acerca de las directivas de ejecución. https://docs.microsoft.com/es-es/powershell/module/microsoft.powershell.core/about/about_execution_policies?view=powershell-7.1. [Consulta: 18 de Abril de 2021].

[10] Microsoft. Acerca de los flujos de salida. https://docs.microsoft.com/es-es/powershell/module/microsoft.powershell.core/about/about_output_streams?view=powershell-7.1. [Consulta: 01 de Mayo de 2021].

[11] Microsoft. Entity framework core. https://docs.microsoft.com/es-es/ef/core/. [Consulta: 14 de Abril de 2021].

[12] Microsoft. Introducción a asp.net core. https://docs.microsoft.com/es-es/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-5.0. [Consulta: 20 de Marzo de 2021].

[13] Microsoft. Paseo por el lenguaje c#. https://docs.microsoft.com/es-es/dotnet/csharp/tour-of-csharp/. [Consulta: 20 de Marzo de 2021].

[14] Microsoft. Powershell class. https://docs.microsoft.com/en-us/dotnet/api/system.management.automation.powershell?view=powershellsdk-7.0.0. [Consulta: 18 de Abril de 2021].

[15] Microsoft. Tutorial: Introducción a razor pages en asp.net core. https://docs.microsoft.com/es-es/aspnet/core/tutorials/razor-pages/razor-pages-start?view=aspnetcore-5.0&tabs=visual-studio. [Consulta: 17 de Abril de 2021].

[16] Mozilla. ¿qué es javascript? https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript. [Consulta: 20 de Marzo de 2021].

[17] Steve "ardalis" Smith. *Architecting Modern Web Applications with ASP.NET Core and Microsoft Azure*. EDITION v5.0. Microsoft, 2020.

# Anexo A

# Manual de uso

En las siguientes páginas se adjunta el documento de formación para el uso correcto de la aplicación web. Este documento describe los pasos que se deben usar para configurar tanto el servidor web que contiene la aplicación como el servidor remoto donde se van a ejecutar los scripts; además de como realizar los pasos para el correcto funcionamiento de la aplicación web.


# Documentación de uso y configuración de sistemas del proyecto

*Autor:*
Iván Guimerà Fontanet

Mayo 2021

# Resumen

Este documento corresponde a una guía de uso del proyecto desarrollado por el alumno Iván Guimerà Fontanet de la Universitat Jaume I durante su estancia en prácticas en la empresa Datanet Consultores S.L. El contenido que se expone a continuación consta de dos partes:

- Configuración de sistemas: Para poder ejecutar scripts en un servidor remoto, hay que configurar previamente tanto cliente como servidor con el fin de permitir la conexión entre ellos y permitir la ejecución de scripts PowerShell remotos.
- Uso de scripts: Para el correcto funcionamiento de ejecución de scripts se debe seguir una estructura adecuada. También es necesario que un usuario con permisos de administrador añada una funcionalidad al sitio web de manera adecuada para poder insertar y ejecutar cualquier script en el sitio web.

## A.1. Configuración de sistemas

### A.1.1. Conexión entre servidor web y servidor

Para poder realizar la conexión mediante PowerShell entre servidor web y el servidor para realizar las operaciones hay que realizar un conjunto de pasos:

1. Se debe configurar el Firewall en ambos sistemas. Para ello se debe crear una regla de entrada que permite escuchar y recibir tráfico de red entrante entre los dos sistemas. Como se puede ver en la Figura A.1, se puede establecer el ámbito de conexión, indicando una única IP, un rango de IP's o una subred. En este caso, se ha establecido que todas las IP's de la subred 192.168.0.0 puedan realizar conexiones entre ellas.

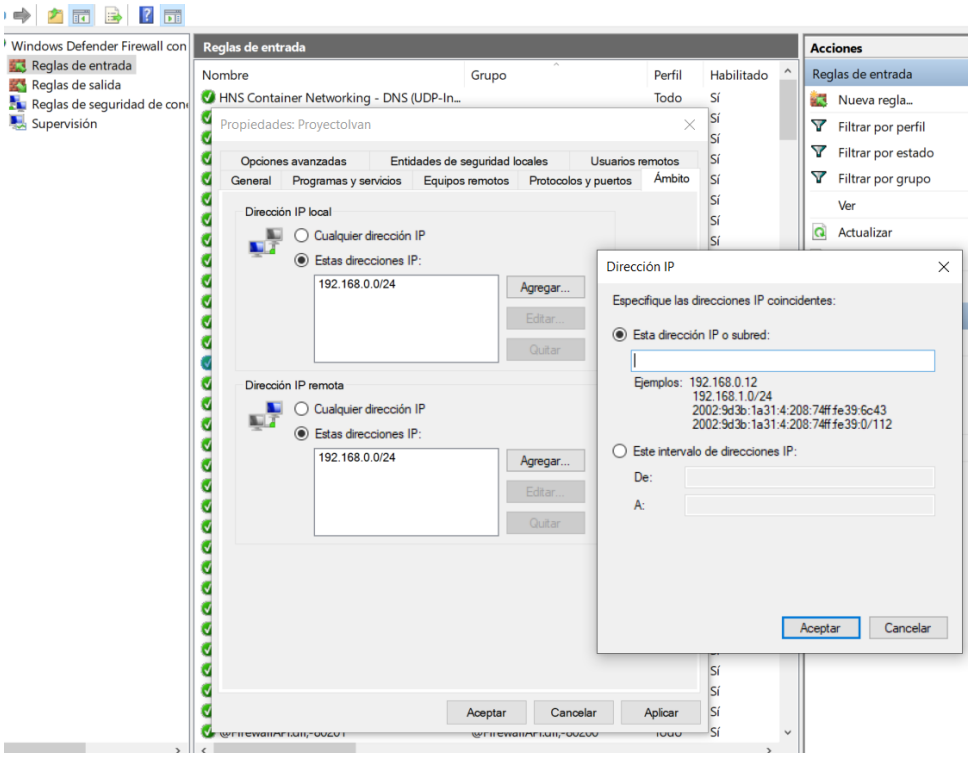


Figura A.1: Regla de entrada de Windows Defender Firewall

Obviamente, esta opción no es la óptima en términos de seguridad, ya que sería más lógico permitir la conexión entre las IP's estrictamente implicadas o crear una nueva subred entre los sistemas implicados. Una vez creada la regla de entrada en ambos equipos, ya se puede establecer la conexión entre ambos (se puede comprobar realizando un ping entre ambos).

2. Configurar el listener de WinRM en el servidor. Para ello se debe ejecutar el siguiente comando, que se encargará de iniciar el servicio WinRM y empezará a escuchar peticiones y configurará la excepción del Firewall pertinente:

**Enable-PSRemoting -SkipNetworkProfileCheck**

El parámetro SkipNetworkProfileCheck habilita el uso de PowerShell remoto en versiones cliente del SO Windows ya que cuando un sistema se encuentra en una red pública normalmente se encuentra deshabilitado. Este parámetro elimina dicha restricción y crea una regla en el firewall que permite el acceso remoto en las redes públicas.

3. Tanto en el servidor web como en el servidor se debe añadir al otro sistema en la lista de hosts confiables de WsMan (Web Services for Management). Como se puede ver en la Figura 1.2, se modifica el fichero contenido en la ruta añadiendo el nombre del host remoto en el listado de hosts confiables del sistema local. Dicho fichero está vacío por defecto, y contiene el conjunto de ordenadores remotos que pueden realizar operaciones sobre el sistema local en remoto. Para obtener el nombre de host de cada sistema simplemente hay que ejecutar el comando **hostname** de PowerShell o CMD en el sistema.

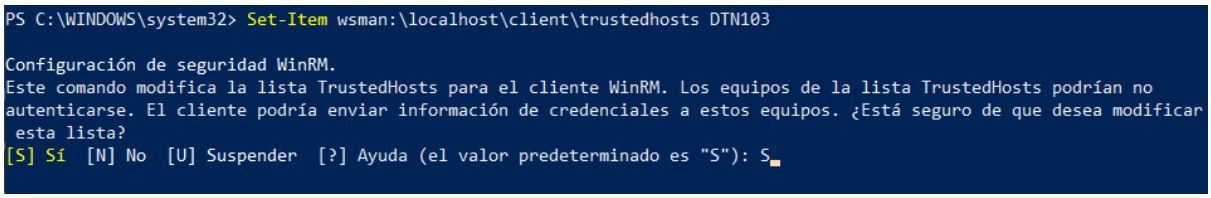


Figura A.2: Añadir un host al listado de hosts confiables

4. Finalmente hay que reiniciar el servicio Windows Remote Management (WinRM). Dicho servicio corresponde con la implementación del protocolo WSMan. Como hemos actualizado el listado de hosts confiables de WsMan de ambos sistemas, se debe reiniciar el servicio en ambos sistemas. Para la ejecución del reinicio del servicio simplemente se debe ejecutar el comando PowerShell:

**Restart-Service WinRM**

### A.1.2. Añadir servidor en el sitio web

Una vez se han realizado las operaciones para permitir la ejecución remota de scripts en el servidor, hay que añadir dicho servidor en el sitio web. Esta operación solo puede ser realizada por un usuario con permisos de Administrador. Para ello se inicia sesión en el sitio web y se hace clic en el botón *Más Opciones*. Luego se selecciona la opción *Añadir un nuevo servidor* y se mostrará la pantalla correspondiente con la Figura A.3.

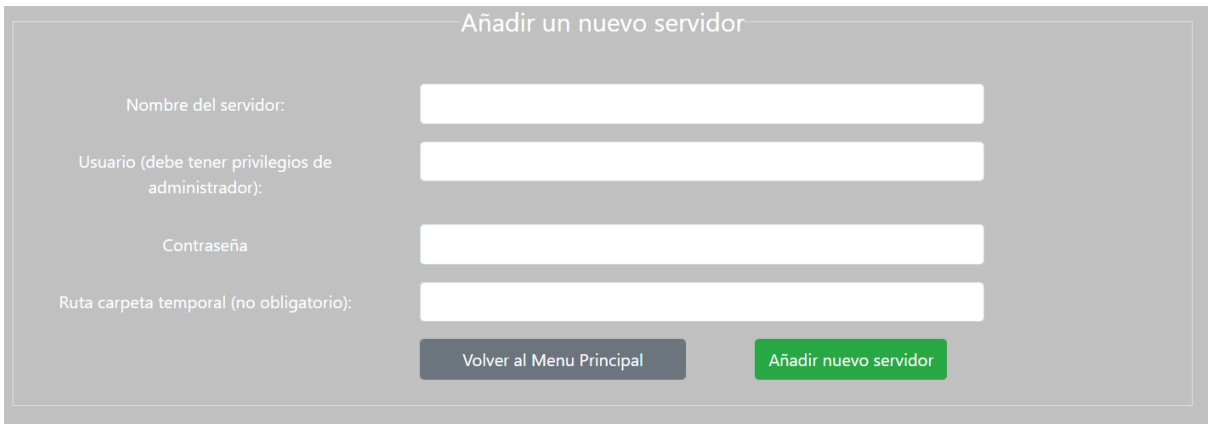


Figura A.3: Añadir un nuevo servidor al sitio web

Se observa que se solicitan cuatro campos:

- Nombre del servidor: Para obtener el nombre del equipo en el cual nos encontrado tan solo deberemos teclear el comando **hostname** en CMD o en la consola de PowerShell.
- Usuario: Debe ser un nombre de usuario del servidor remoto que tenga privilegios de administrador. Eso es debido a que la ejecución de scripts en un sistema tan solo puede ser realizada por usuarios que tengan esos privilegios; en caso contrario, no se podrá ejecutar el script. Para obtener el nombre de usuario que estoy utilizando actualmente, tan solo deberemos teclear el comando **whoami** en CMD o en la consola de PowerShell.
- Contraseña: Corresponde con la contraseña del usuario con privilegios de administrador que acabamos de indicar en el apartado anterior.
- Ruta carpeta temporal: Donde se van a alojar los ficheros que sean necesarios para el uso de los scripts (cuando se precise usar un algún fichero). No es obligatorio indicarlo por dos motivos: se pueden ejecutar scripts en remoto que no requieran conocer la ubicación de dicha carpeta (no se requiera subida de ficheros); y porque cuando se ejecuta una funcionalidad en el sitio web ya se comprueba (en el caso que se suban ficheros) en la base de datos si se ha almacenado la carpeta temporal del servidor y en caso contrario se lanzaría automáticamente el comando correspondiente con la obtención de dicha ruta antes de realizar la ejecución del script. Para obtener la ruta de la carpeta temporal de un servidor tan solo hay que abrir un terminal de PowerShell y ejecutar el siguiente comando:

**[Environment]::GetEnvironmentVariable("Temp","Machine")**

## A.2. Uso de scripts

### A.2.1. Estructura del fichero PowerShell

Para el correcto funcionamiento del script PowerShell en el sitio web se debe seguir una estructura adecuada. Simplemente se debe indicar la variable $parametroX (donde la X corresponde con el número del parámetro correspondiente, la variable $parametro1 corresponderá con el primer valor que debe introducir el usuario, $parametro2 corresponderá con el segundo valor que debe introducir el usuario, y así sucesivamente) para cada uno de los parámetros que el usuario deberá introducir en el sitio web. Se puede observar mejor con el script de ejemplo que se muestra en la Figura A.4.

```
1  Write-Output "Contenido de la licencia"
2  Get-Content $parametro1
3  Write-Output ""
4  Write-Output "Contenido del parametro 2: $parametro2"
5  Write-Output ""
6  Write-Output "Contenido del fichero"
7  Get-Content $parametro3
```

Figura A.4: Script de ejemplo utilizado en la demostración de uso del sitio web

Este script de ejemplo simplemente lo que hace es mostrar el contenido de un fichero que el usuario debe subir a través del sitio web ($parametro1), a continuación muestra un texto que el usuario debe introducir en el sitio web ($parametro2) y finalmente muestra el contenido de otro fichero que el usuario debe subir a través del sitio web ($parametro3).

### A.2.2. Configuración de la funcionalidad en el sitio web

Para insertar el script en el sitio web y que éste se encuentre disponible para su ejecución por parte de cualquier usuario registrado del sitio web, es necesario que un usuario con permisos de administrador añada la funcionalidad en el sitio web. Para ello el administrador inicia sesión y selecciona el botón *Más Opciones*. A continuación se mostrará la página con las diferentes acciones que puede realizar el usuario administrador, como se puede ver en la Figura A.5.

Luego debe seleccionar la opción *Añadir una nueva funcionalidad*, y en primer lugar deberá indicar una descripción de cual es el objetivo de la nueva funcionalidad, cuál es el servidor sobre el cuál se va a ejecutar el script y subir el script, que siempre será un script PowerShell con extensión .ps1.


Figura A.5: Funciones del administrador

Finalmente se deben añadir los parámetros que el usuario debe introducir para la correcta ejecución del script. Cada uno de esos parámetros corresponden con una variable del script y puede corresponder a texto o a un fichero (se sube el fichero en el servidor remoto y al script se le indicaría la ruta donde se encuentra alojado dicho fichero). También es necesario especificar cuál es el orden en que van a ser ejecutados dichos parámetros, es decir, por orden de aparición en el script (a menor número, mayor prioridad) ya que la base de datos almacena los parámetros por orden alfabético por defecto.


Figura A.6: Añadir una nueva funcionalidad en el sitio web

En la Figura A.6 se puede observar como se deben añadir los parámetros para el script visto en la Figura A.4. Se han añadido tres parámetros: dos que corresponden a ficheros (llamados Licencia y Fichero) y uno que corresponde a texto plano (llamado Parámetro con texto). Muy importante rellenar los campos de Posición de cada parámetro para determinar su orden, ya que si no se indicase ninguna prioridad, la base de datos asociaría los parámetros al script alfabéticamente: primero el parámetro Fichero (en lugar de Licencia), luego el parámetro Licencia (en lugar del parámetro de tipo texto), y finalmente el parámetro Parámetro con texto (en lugar del parámetro Fichero); con lo cuál se obtendría un error de ejecución. Como se vio en el apartado *Estructura del script*, el primer parámetro que queremos ejecutar corresponde con un fichero de tipo licencia (la ruta a esa licencia), en segundo lugar texto plano y finalmente un fichero cualquiera; con lo cuál indicamos ese orden en el apartado Posición.

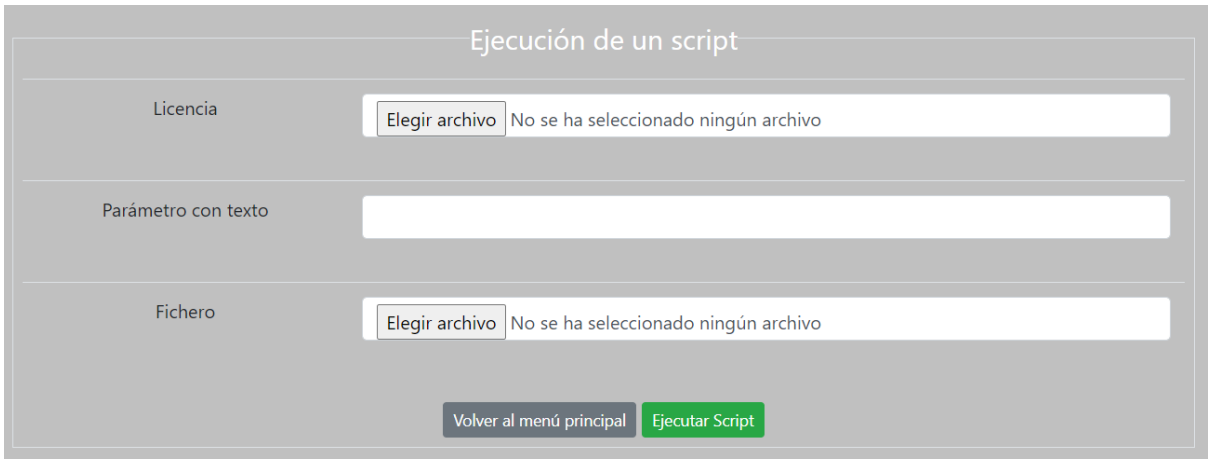


Figura A.7: Introducción de parámetros para la ejecución de un script

Una vez se ha hecho clic sobre el botón *Crear Nueva Funcionalidad*, ya se encontrará en el listado de funcionalidades para que tanto usuarios como administradores puedan ejecutar dicho script con los parámetros que indiquen, como se puede ver en la Figura A.7.

El usuario deberá indicar en primer lugar el fichero correspondiente a la Licencia alojado en su ordenador, a continuación introducirá el texto plano y finalmente seleccionará otro fichero alojado en su ordenador correspondiente con el parámetro Fichero. Al hacer clic sobre *Ejecutar Script*, el sistema subirá ambos ficheros al servidor remoto indicado previamente.Al script le indicará que el valor asociado para la variable $parametro1 es la ruta donde se encuentra alojado el fichero correspondiente con Licencia, que el valor asociado para la variable $parametro2 es el texto que el usuario haya introducido, y que el valor asociado para la variable $parametro3 es la ruta donde se encuentra alojado el fichero correspondiente con Fichero.

Finalmente, tras hacer clic sobre el botón *Ejecutar Script* y esperar unos segundos, se podrá observar en pantalla el resultado de la ejecución del script, como se puede observar en la Figura A.8.

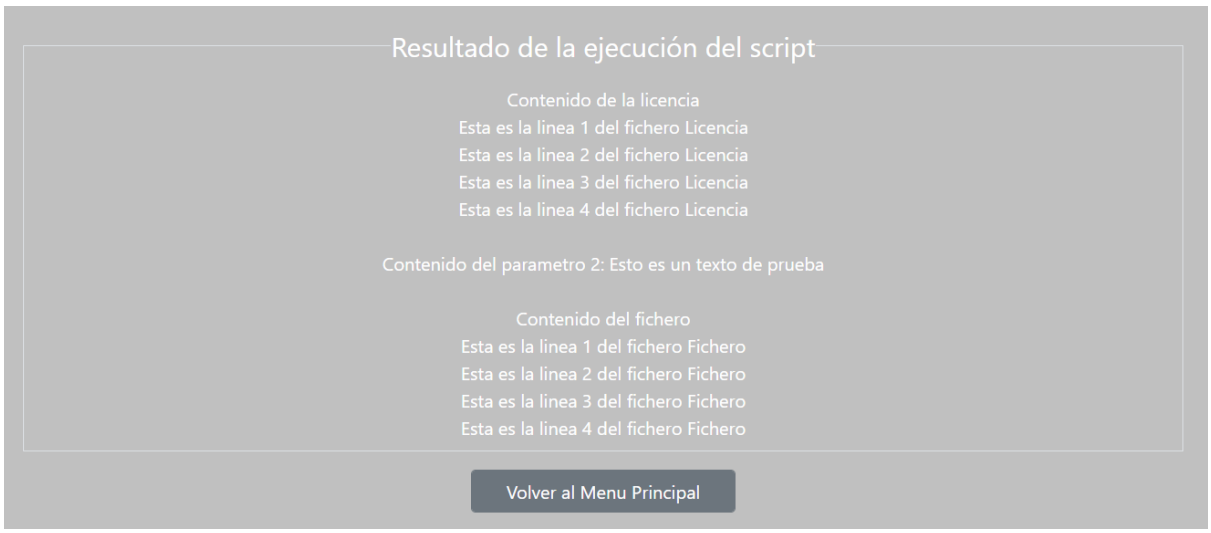


Figura A.8: Resultado de la ejecución de un script en un servidor remoto

Como se puede observar, el script de ejemplo ejecutado tan solo muestra por pantalla el contenido de ambos ficheros y el texto introducido por el usuario. En caso de que hubiese algún tipo de error tanto producido por la conexión al servidor remoto como por la ejecución del script, también se mostraría por pantalla.