E.T.S. de Ingeniería Industrial, Informática
y de Telecomunicación

ASISTENTE PARA LA CONSTRUCCIÓN Y GESTIÓN DE MICROSERVICIOS GNOSS INTEGRADOS Y OPERABLES DESDE LA PLATAFORMA SEMÁNTICA GNOSS

Grado en Ingeniería Informática

Trabajo Fin de Grado

Laura del Villar Aparicio Latorre

Alfredo Pina Calafi

Pamplona, 11 de junio

upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

RESUMEN

La plataforma GNOSS, desarrollada por la compañía RIAM INTELEARNING LAB S.L., permite la construcción y la explotación de grafos de conocimiento. Además, es extensible mediante aplicaciones web específicas de cada proyecto que permiten crear una experiencia de usuario enriquecida, con funciones que se integran a través del API de GNOSS.

Se pretende el desarrollo de una herramienta de administración que permita a los desarrolladores la gestión de estas aplicaciones web y una documentación que especifique cómo construirlas para que su integración en la plataforma GNOSS sea sencilla y eficiente. Esta herramienta permitirá crear las aplicaciones web en el servidor de desarrollo con IIS y su compilación automática a través de la aplicación Jenkins. También permitirá su despliegue en el entorno de desarrollo y se redactará una documentación con recomendaciones sobre cómo construir la aplicación para que su despliegue se pueda automatizar. Además, se crearán las aplicaciones en los entornos de preproducción y producción, así como su compilación y despliegue.

PALABRAS CLAVE

- .NET
- Desarrollo Web
- Compilación automática
- Despliegue de aplicaciones automatizado
- GNOSS Knowledge Graph Builder
- Jenkins

ÍNDICE DE CONTENIDOS

1 INTRODUCCIÓN

El presente trabajo ha sido realizado en la compañía Riam Intelearning Lab S.L. durante el periodo de prácticas curriculares del grado de Ingeniería Informática.

La compañía ha construido una plataforma llamada GNOSS que opera con el conjunto de tecnologías que concurren en el programa de Inteligencia Artificial Simbólica basada en la explotación de Grafos de Conocimiento. Esa plataforma permite la construcción y la explotación de grafos de conocimiento. Es una forma de integrar y representar la información que permite descubrir e investigar cualquier tema de una forma más profunda e intuitiva y disfrutar de una web más semánticamente consciente.

La plataforma GNOSS provee de muchas funcionalidades que permiten construir y explotar los grafos de conocimiento de diversas maneras. Algunas de esas funcionalidades son las siguientes:

1. **Construir y gestionar grafos de conocimiento y metadatos:**
    - Gestionar ontologías
    - Gestionar taxonomías y tesauros
    - Integración de datos y APIs
    - Portabilidad e interoperabilidad de datos
    - Extracción de entidades etiquetado automático.
2. **Descubrir, razonar y analizar a través de un Grafo de Conocimiento**:
    - Búsqueda semántica
    - Interfaces de búsqueda basados en facetas
    - Contextos e información enriquecida
    - Sistemas de recomendación NER
    - Semantic Analitycs & Visual Data Browsing
    - GNOSS Semantic Search Plug-ins
3. **Gestión semántica de contenidos:**
    - Páginas web semánticas
    - Publicación semántica nativa (por usuarios)
    - Generar formularios semánticos
    - Personalizar la capa de presentación
    - Gestión multisitio y multidioma
    - Semantic SEO
    - Flujos de trabajo a medida

Sin embargo, la plataforma tiene limitaciones. Cuando se necesita realizar tareas o explotaciones para las cuales la plataforma no está preparada se permite integrar aplicaciones que doten de esa funcionalidad. Por lo tanto, la plataforma GNOSS es extensible mediante aplicaciones web específicas por cada proyecto.
Por ejemplo, un proyecto realizado con GNOSS Knowledge Graph Builder es Didactalia. En los mapas interactivos de Didactalia, se quería mostrar un ranking de las personas que más habían jugado como el que se muestra en la figura 1.

Figura 1 - Mapas interactivos de Didactalia

Entonces, se vio necesario crear una aplicación web que se encargase de registrar cada partida de cada usuario, calcular el ranking de usuarios y, además, almacenar el ranking en el grafo de conocimiento.

En este proyecto, se quiere desarrollar una herramienta de administración que permita a los desarrolladores realizar la gestión de estas aplicaciones web. En esta introducción, se detallarán tanto la motivación y los objetivos de este proyecto, así como las tecnologías utilizadas para su realización.

1.1 MOTIVACIÓN

La realización de este trabajo surge debido a que, hasta ahora, para desplegar esa aplicación que se creaba especialmente para un proyecto concreto, se tenía que solicitar al administrador de sistemas que desplegase la aplicación.

Además, no solo se trata de hacer una compilación y despliegue automatizado de esta aplicación en varios entornos, sino que, se haga al mismo tiempo el despliegue de todas las configuraciones ontológicas y de UX (experiencia de usuario) que se han preparado para esa solución de Inteligencia Artificial semántica desarrollada con GNOSS Knowledge Graph Builder. Estas configuraciones son modificaciones que se han realizado en la vista, tanto de HTML como de Javascript. Entonces, tanto el HTML y las funciones creadas con

Javascript como la aplicación específica deben desplegarse al mismo tiempo, ya que sino la experiencia de usuario mientras no estén todos los componentes actualizados se verá degradada. Por ello, se ha integrado la compilación y el despliegue de las aplicaciones específicas en el sistema de integración continua que ya existía y se encargaba de desplegar del entorno de desarrollo al de preproducción y del entorno de preproducción al de producción, las configuraciones que se han ido trabajando.

Por lo tanto, gracias a este proyecto, cualquiera que necesite realizar la instalación y el despliegue de las aplicaciones específicas de un proyecto, podrá hacerlo a través de la herramienta desarrollada en este trabajo de fin de grado. Además, a la vez que se despliegan las configuraciones cada proyecto, también se desplegarán las aplicaciones específicas. Esto último prácticamente al mismo tiempo, de manera automatizada y sin interacción humana.

1.2 OBJETIVOS

El objetivo principal de este proyecto es desarrollar una herramienta para dar de alta aplicaciones web.

Para poder llevar a cabo este objetivo, hay que cumplir las siguientes tareas:

1. Creación de una interfaz de administración de las aplicaciones web
2. Creación de las aplicaciones web en el servidor con IIS
3. Compilación automática de las aplicaciones a través de Jenkins
4. Despliegue y redacción de una documentación con recomendaciones sobre cómo construir la aplicación para que su despliegue se pueda automatizar
5. Realizar pruebas con usuarios

2 TECNOLOGÍAS Y HERRAMIENTAS UTILIZADAS

2.1 LENGUAJE DE PROGRAMACIÓN

El lenguaje de programación empleado para el desarrollo del proyecto ha sido c# con la plataforma .NET y la herramienta Visual Studio.

Por otro lado, se ha utilizado también HTML y Javascript para implementar la vista de la interfaz web.

2.2 INTERNET INFORMATION SERVICES(IIS)

Es un conjunto de servicios que transforman un sistema Microsoft Windows en un servidor capaz de ofrecer servicios [1]. Se usa principalmente para alojar sitios web y otro contenido en la Web.

IIS ofrece lo que se llama Grupo de Aplicaciones (Application Pool). Es una forma de aislar unas aplicaciones web de otras, confinándolas en su propio proceso y en sus propios límites de seguridad [2]. Cada grupo de aplicaciones se ejecuta de forma independiente y, por lo tanto, los errores en cada grupo de aplicación no afectarán a las aplicaciones ejecutándose en otros grupos de aplicaciones.

IIS se emplea en este trabajo en el segundo objetivo.

2.3 JENKINS

Es un servidor automatizado open source escrito en Java. Jenkins ayuda en la automatización de parte del proceso de desarrollo del software mediante integración continua y facilita ciertos aspectos de la entrega continua. [3]

Se utiliza en el tercer objetivo del proyecto. En él, se crea una tarea para la compilación de las aplicaciones web.

2.4 NEXUS

Es un administrador de repositorios. Es un servidor que almacena y recupera archivos.

Se utiliza Nexus en la última parte de la tarea de Jenkins. Esta última, subirá al repositorio de Nexus de la empresa el paquete NuGet de la aplicación en concreto.

2.5 NUGET

NuGet es el administrador de paquetes de .NET. Las herramientas de cliente NuGet proporcionan la capacidad de producir y consumir paquetes. La galería NuGet es el repositorio de paquetes central utilizado por todos los autores y consumidores de paquetes. [4]

Se utiliza para desplegar los paquetes almacenados en Nexus.

2.6 RABBITMQ

Es un software de encolado de mensajes llamado bróker de mensajería o gestor de colas. Es un software donde se pueden definir colas, las aplicaciones se pueden conectar a dichas colas y transferir o leer mensajes en ellas.

En una cola de mensajes existen productores y consumidores. Los productores son aplicaciones clientes. Estas crean mensajes y los entregan a la cola de mensajes. Por otro lado, los consumidores se conectan a la cola y se suscriben a los mensajes que se procesarán. En lugar de publicarse directamente los mensajes en la cola, el productor envía a un exchange. Estos son agentes de enrutamiento de mensajes, definidos por virtual host dentro de RabbitMQ.

El flujo de mensajes de RabbitMQ es el siguiente. En primer lugar, el productor publica un mensaje al exchange. Después, el exchange recibe el mensaje y pasa a ser el responsable del enrutamiento del mensaje. A continuación, se establece un binding (enlace que se configura para vincular una cola a un exchange). Los mensajes permanecen en la cola hasta que sean manejados por un consumidor. Por último, el consumidor procesa el mensaje. [5]

Se utiliza como gestor de colas para enviar eventos a los servicios Windows. Por ejemplo, cuando se quiere desplegar una aplicación se envía un evento a la cola Rabbit. Ocurre lo mismo cuando se quiere ejecutar una aplicación.

2.7 ENTITY FRAMEWORK

Entity Framework es un componente de .NET Framework. Es un conjunto de tecnologías de ADO.NET que admiten el desarrollo de aplicaciones de software orientadas a datos. Entity Framework permite a los desarrolladores trabajar con datos en forma de objetos y propiedades específicos del dominio sin tener que preocuparse de las tablas y columnas de bases de datos subyacentes en las que se almacenan estos datos. Además, permite mantener y crear aplicaciones orientadas a datos con menos código que en las aplicaciones tradicionales y trabajar en un nivel más alto de abstracción cuando se trata con datos. [6]

Se emplea para trabajar con la base de datos.

2.8 SYSTEM USABILITY SCALE (SUS)

System Usability Scale, en español Sistema de Escalas de Usabilidad, proporciona una herramienta rápida y fiable para medir la usabilidad. Fue creado por John Brooke en 1986 y permite evaluar una amplia variedad de productos y servicios. Consiste en un cuestionario de diez preguntas con cinco opciones de respuesta, desde totalmente de acuerdo hasta totalmente en desacuerdo. Las diez preguntas realizadas son las siguientes. [7]

1. Creo que usaría este sistema frecuentemente
2. Encuentro este sistema innecesariamente complejo
3. Creo que el sistema fue fácil de usar
4. Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar este sistema
5. Las funciones de este sistema están bien integradas
6. Creo que el sistema es muy inconsistente
7. Imagino que la mayoría de la gente aprendería a usar este sistema en forma muy rápida
8. Encuentro que el sistema es muy difícil de usar
9. Me siento confiado al usar este sistema
10. Necesité aprender muchas cosas antes de ser capaz de usar este sistema.

El test SUS se utiliza en las pruebas de usuario.

3 ALCANCE Y PLANIFICACIÓN DEL PROYECTO

3.1 VIABILIDAD

Antes de empezar el proyecto, se realizó un estudio de viabilidad para definir el alcance del proyecto.

La empresa desarrolla las aplicaciones en .NET Framework 4.x. ASP.NET 4.x es un marco consolidado que proporciona los servicios necesarios para compilar aplicaciones web de nivel empresarial basadas en servidor en Windows [11]. En cambio, ASP.NET Core es un marco multiplataforma de código abierto que tiene como finalidad compilar modernas aplicaciones web basadas en la nube en Windows, macOS o Linux [11]. La empresa pretende en un futuro próximo realizar la migración de ASP.NET Framework a ASP.NET Core.

Docker es una tecnología de creación de contenedores que permite la creación y el uso de contenedores de Linux [12]. Debido a la posible migración a .NET Core, se investigó y valoró el uso de Docker para el despliegue de las aplicaciones específicas. Sin embargo, solo se valoró para el desarrollo de aplicaciones en .NET Core ya que las desarrolladas en .NET Framework, aunque en teoría se pueden desplegar con Docker, no es eficiente debido a que los contenedores son excesivamente grandes y carece de sentido. Finalmente, se descartó usar Docker debido a que era un trabajo a largo plazo que no se iba a poder probar correctamente con el equipo de desarrollo de GNOSS.

El proyecto se ha planificado con metodologías ágiles. Al principio, se dividió en sprints de dos semanas con los que al final de cada uno se pudiese realizar una demo. Así, el proyecto quedó dividido en seis sprints. Para realizar la planificación, se empleó la herramienta desarrollada por la empresa Atlassian para la administración de tareas, Jira.

Los sprints quedaron de la siguiente manera:

- Sprint 1: Creación de la interfaz de administración
- Sprint 2: Creación de las aplicaciones web en el servidor de desarrollo en IIS
- Sprint 3: Compilación automática con Jenkins
- Sprint 4: Despliegue de aplicaciones web en desarrollo
- Sprint 5: Creación de las aplicaciones web en preproducción y producción
- Sprint 6: Integrar con el proceso de integración continua el despliegue y compilación

3.2 REPLANIFICACIÓN

Aunque se definió el alcance descrito en el anterior apartado, se trabajó mucho más rápido de lo esperado. En el segundo sprint, como se vio que la creación de las aplicaciones web en el servidor en IIS para el servidor de desarrollo se hizo muy rápido, se decidió realizar en el mismo sprint la creación de aplicaciones web en IIS para los servidores de preproducción y producción. Además, la integración del despliegue y la compilación en el proceso de integración continua se realizó en el cuarto sprint. Por lo tanto, se tuvo que realizar una replanificación y ampliar el proyecto.

Siguiendo el principio de Yagni, el cual dice que no se debe agregar nunca una funcionalidad excepto cuando sea necesaria [13], se decidió no iniciar el desarrollo de compilar y desplegar aplicaciones .NET Core con Docker y dedicar el tiempo que le quedaba al proyecto a integrar la ejecución remota de aplicaciones.

Cuando se crea un proyecto con la plataforma GNOSS, siempre hay una sincronización de datos para crear el grafo de conocimiento del proyecto. Esas aplicaciones para la sincronización de datos, hay que ejecutarlas casi siempre en todos los entornos. Por ejemplo, en la web del Museo Del Prado, proyecto realizado con GNOSS, hay una sincronización diaria con las obras de arte que se modifican en el sistema de catalogación del Museo Del Prado. Estas aplicaciones tenían el mismo problema que las aplicaciones web ya que se necesitaba que las ejecutase el administrador de sistemas. Es por ello por lo que se vio la necesidad de realizar la creación, el despliegue y la ejecución de las aplicaciones ejecutables.

Por lo tanto, se añade un nuevo objetivo que sería la creación, compilación y despliegue de aplicaciones ejecutables. Además, se definieron dos sprints más en los que se realizaría el despliegue de aplicaciones ejecutables además de su creación, despliegue y ejecución.

Después de la replanificación, los sprints quedaron de la siguiente manera:

- Sprint 1: Creación de la interfaz de administración
- Sprint 2: Creación de las aplicaciones web en los servidores en IIS
- Sprint 3: Compilación automática con Jenkins
- Sprint 4: Despliegue de aplicaciones web
- Sprint 5: Creación, compilación y despliegue de aplicaciones ejecutables
- Sprint 6: Ejecución remota de aplicaciones ejecutables

A continuación, de la figura 2 a la 7, se muestra el Backlog (lista de trabajo ordenado por prioridades para el equipo de desarrollo) realizado en Jira.

Figura 2 - Sprint 1

Figura 3 - Sprint 2

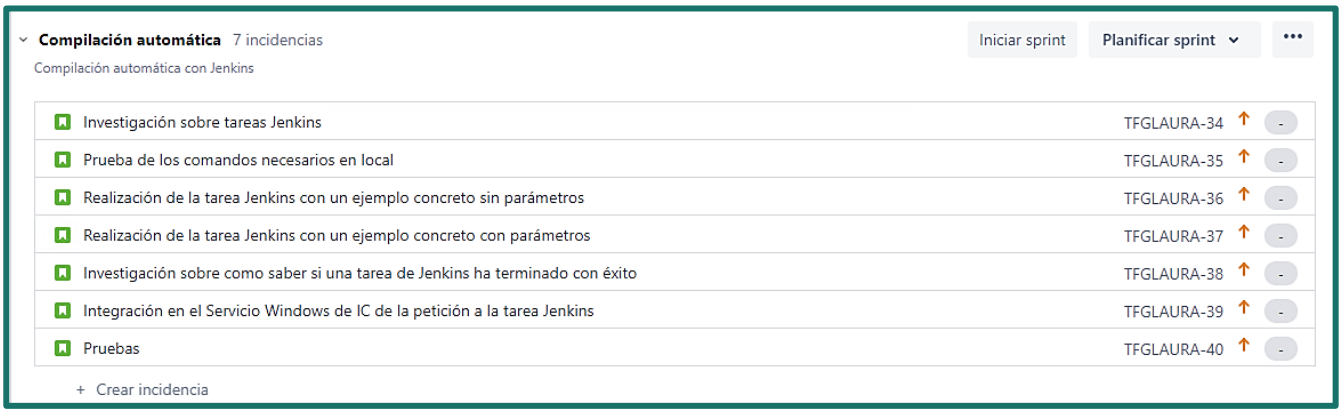


Figura 4 - Sprint 3

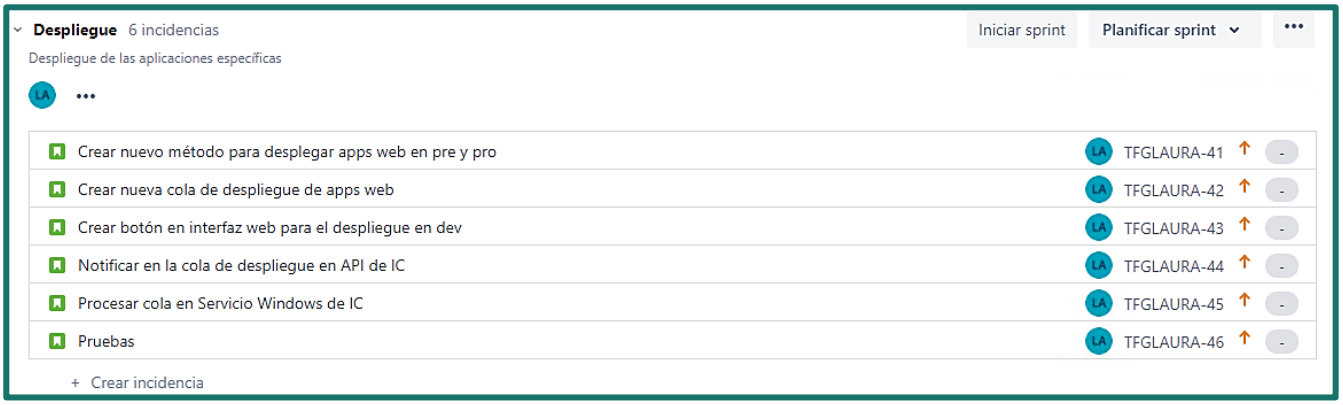


Figura 5 - Sprint 4

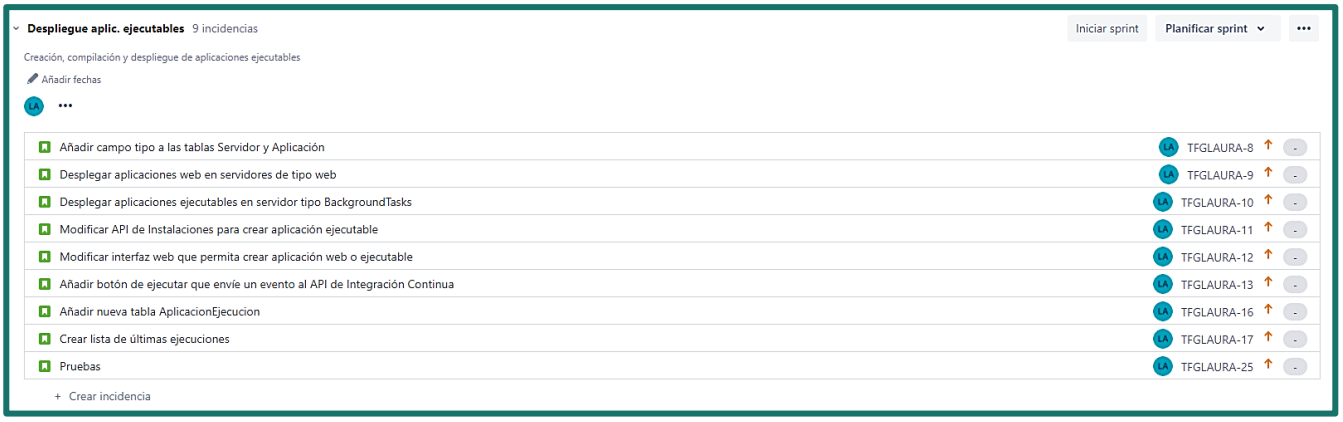


Figura 6 - Sprint 5

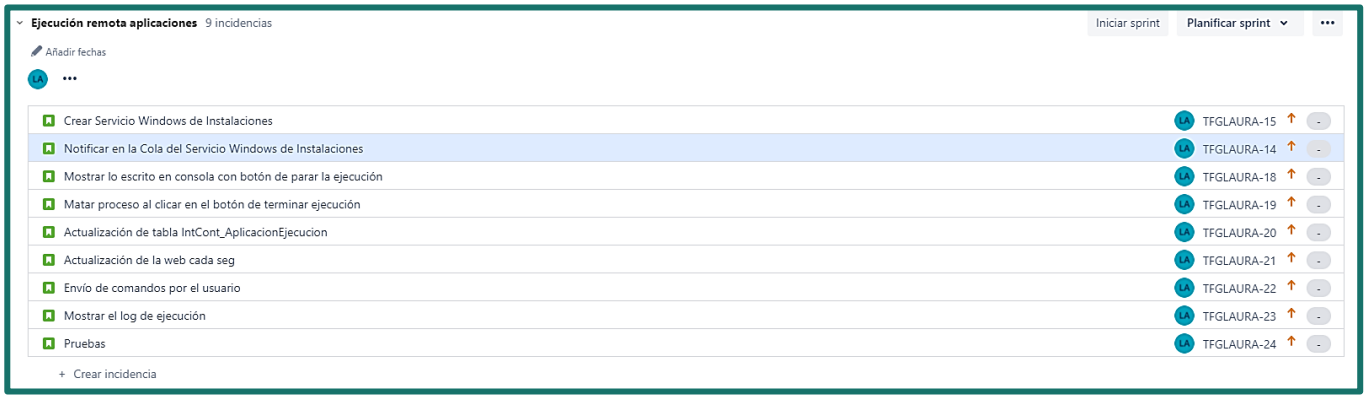


Figura 7 - Sprint 6

4 DESARROLLO E IMPLEMENTACIÓN

El proyecto se ha desarrollado en el siguiente orden según los objetivos propuestos. En primer lugar, se ha creado una interfaz de administración de las aplicaciones web en la que el usuario introducirá los datos de cada aplicación y se modificará la base de datos. Después, esas aplicaciones se crean en sus respectivos servidores con IIS y se realiza la compilación automática de las aplicaciones a través de Jenkins. A continuación, se despliega cada aplicación en el servidor de cada entorno. Por último, se realiza la creación y el despliegue de aplicaciones ejecutables.

La plataforma GNOSS está formada por varios componentes. Para la realización de este proyecto, se han empleado algunos de ellos. A continuación, se detallan.

En primer lugar, se utiliza la web. Es la aplicación web responsable de mostrar la interfaz de usuario.

Después, se ha empleado el API de Integración Continua. Es el encargado de versionar todas las actualizaciones en las configuraciones de una solución GNOSS que realizan los desarrolladores desde las GNOSS Dev Tools. Además, ahí se crearán los eventos para enviar a las colas Rabbit y se realizará la creación, modificación o borrado de las aplicaciones en la base de datos.

Por otro lado, se utiliza el Servicio Windows de Integración Continua. Es el encargado de leer los eventos de las colas Rabbit y realizar lo necesario según el tipo de cola y de evento. Por ejemplo, si lee un evento de tipo desplegar aplicación, se encargará de realizar la compilación y llamar al componente API de Instalaciones para realizar el despliegue.

Por último, el API de Instalaciones. Se emplea para realizar el despliegue de las aplicaciones. Además, se emplea también para la instalación de las aplicaciones web en IIS.

4.1 INTERFAZ DE ADMINISTRACIÓN

La interfaz de administración de las aplicaciones Web permitirá al usuario iniciar el proceso de creación de las aplicaciones de cada proyecto. Para implementarla, se ha modificado la web y el API de Integración Continua. A continuación, en la figura 8, se detallan los casos de uso de la interfaz web.

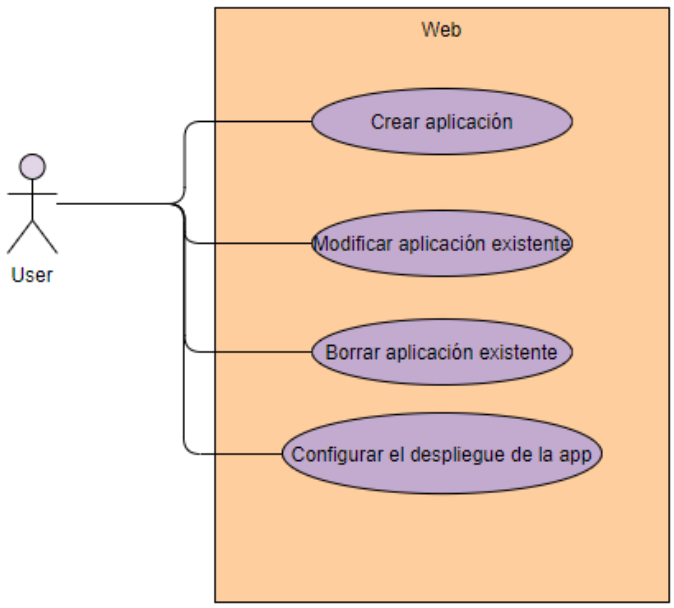


Figura 8 - Diagrama de casos de uso de la web

En la interfaz web se mostrarán las aplicaciones existentes en el proyecto. Además, habrá una opción para crear aplicaciones. Para gestionarlas, se ha creado un formulario web que permite los casos de uso anteriores. Al crear una aplicación, la web pedirá al usuario el nombre de la aplicación, otro nombre de menos de 100 caracteres que no se podrá modificar y la ruta del repositorio en la que tiene que subir su proyecto. Para que funcione correctamente, deberán estar rellenados los primeros tres campos (Nombre de la aplicación, Nombre corto de la aplicación y Ruta del repositorio). Además, existirá otro campo llamado Ruta completa del repositorio que mostrará, como su propio nombre indica, la ruta completa al repositorio Git-Lab. Así, el usuario sabrá donde tiene que subir el código de su aplicación. No se podrá escribir en este campo.

El campo Ruta del repositorio se completará al mismo tiempo que se escriba en el campo nombre corto y su valor será por defecto “Apps/NombreCorto” siendo NombreCorto el valor escrito en el campo correspondiente. También, se completará a la vez el campo Ruta completa del repositorio con el valor del campo ruta del repositorio.

Por otro lado, una vez creada la aplicación, se mostrará un botón para bien modificar los datos de la aplicación o bien para borrar la aplicación. Además, para gestionar el despliegue de la aplicación, se mostrarán otros campos para rellenar: la ruta del archivo Csproj y el nombre del archivo .Sln. Estos campos se rellenarán una vez se haya creado la aplicación. El archivo CsProj es el proyecto creado en Visual Studio y el archivo .Sln se refiere a la solución. La solución es una estructura para organizar los proyectos en Visual Studio y mantiene la información de estado de los proyectos.

A continuación, en la figura 9, se muestra el formulario que aparece al crear una nueva aplicación.

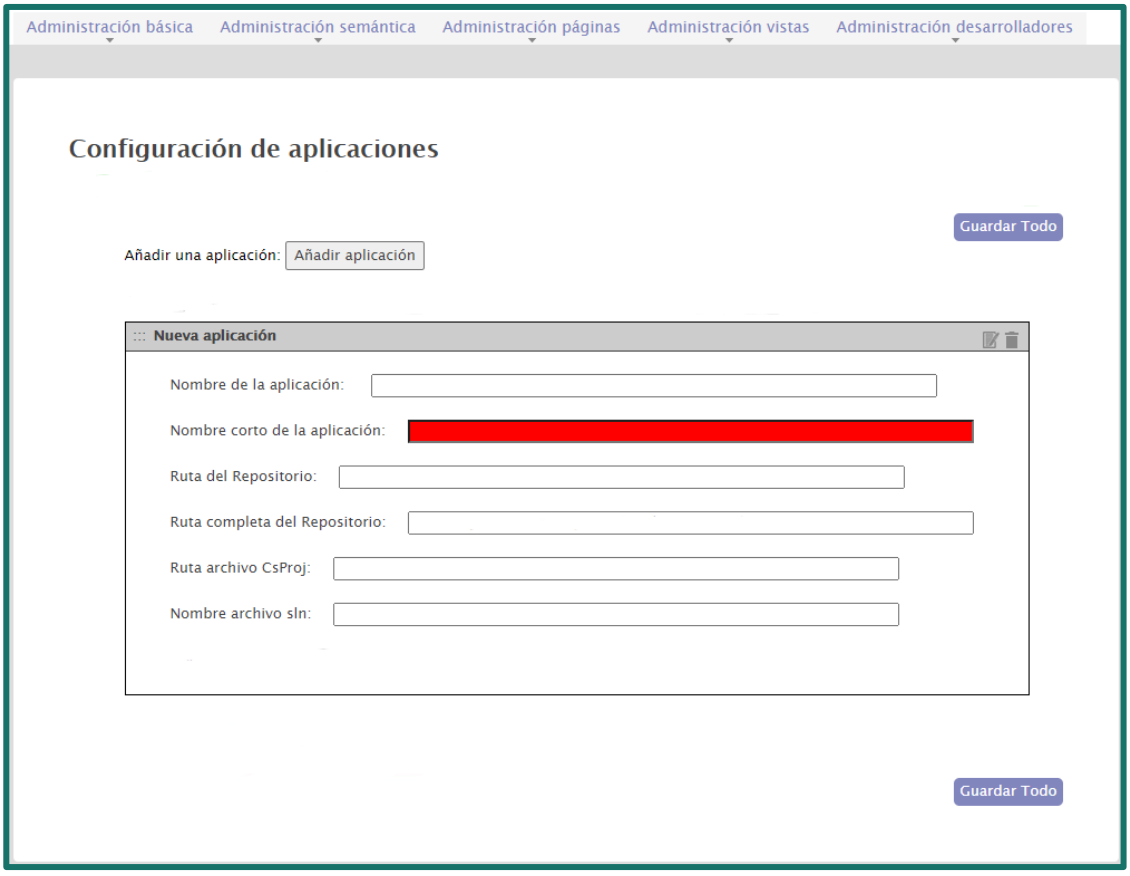


Figura 9 - Interfaz web

Para poder gestionar estas aplicaciones y guardar sus datos se han creado dos tablas en la base de datos a parte de las que ya estaban creadas: Aplicación y Servidor. Como su propio nombre indica, Aplicación se ha creado para guardar los datos de cada aplicación y Servidor para guardar los datos de cada servidor. Cada aplicación, tendrá el identificador de la personalización a la que pertenece. La personalización se refiere al proyecto al que forma parte.

Además, se trabaja con diferentes entornos de trabajo: desarrollo, preproducción y producción. El entorno de desarrollo está en continuo cambio, es dónde se programa. Cuando se completa un código perfectamente funcional y se realizan las pruebas necesarias para tener el software estable, se pasa al siguiente entorno. Este entorno es pre-producción. Su propósito principal es emular el entorno de producción con el fin de testear una solución y sus futuros evolutivos. Por último, el entorno de producción es el entorno donde finalmente se ejecuta la aplicación y donde acceden los usuarios finales.

Por lo tanto, por cada personalización existe el proyecto que se vincula al entorno de desarrollo, el proyecto que se vincula al entorno de preproducción y el que se vincula al entorno de producción. También, cada proyecto que pertenece a un entorno concreto tendrá un servidor o varios. Estas últimas entidades ya existían en la base de datos.

Además, se necesitará la tabla Repositorio para poder completar el campo de la ruta completa del repositorio. Con esta tabla, se devolverá el nombre del repositorio.

A continuación, en la figura 10, se muestra el modelo entidad-relación.

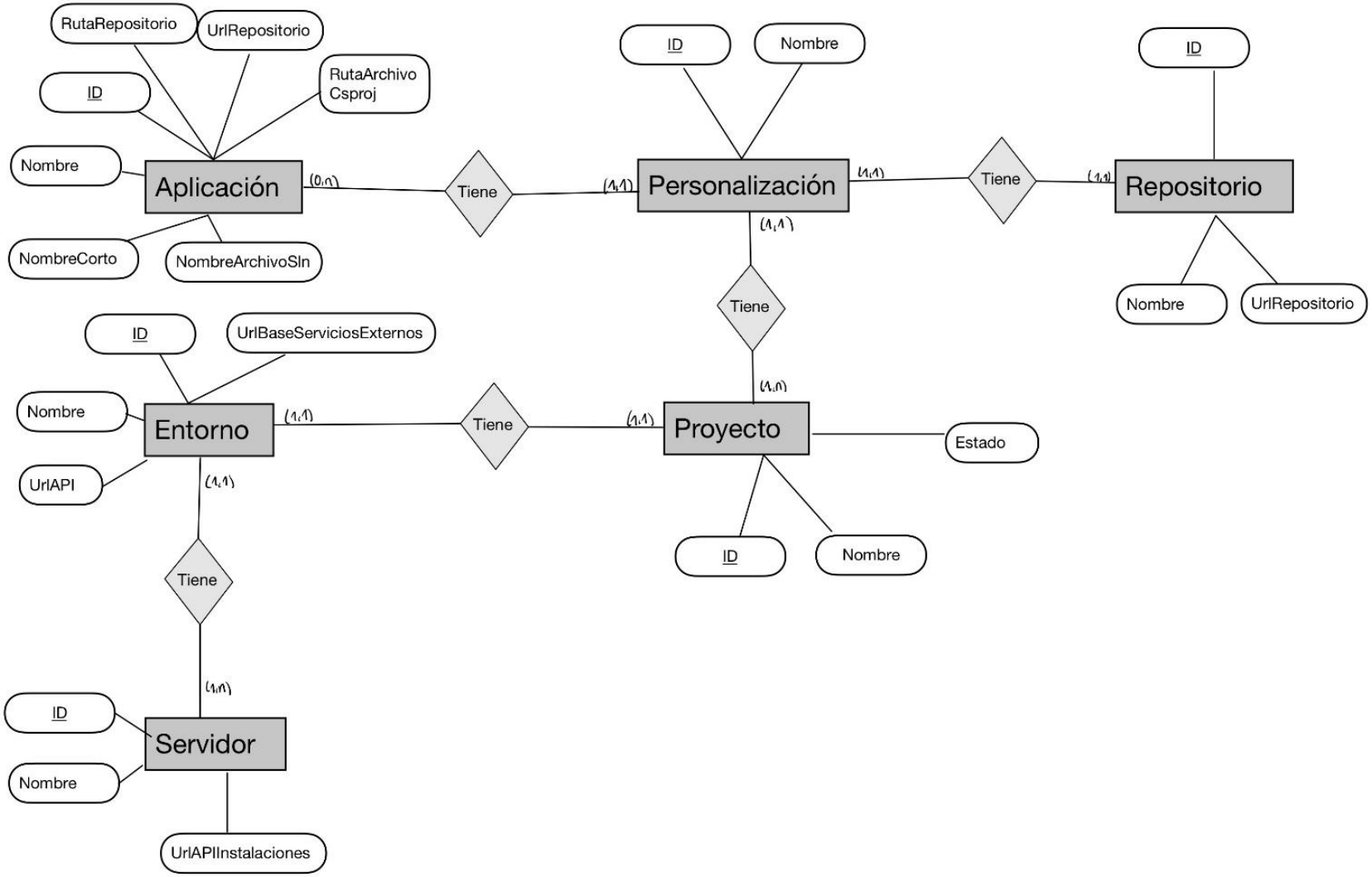


Figura 10 – Modelo Entidad-Relación

Por último, se ha creado la lógica para realizar las consultas a la base de datos. Así, se comprobará qué aplicaciones existen para ese proyecto. Si las aplicaciones que llegan por el formulario son nuevas, se insertarán en la base de datos. Si ya existen, pero se han modificado, se actualizarán en la base de datos. Si el campo booleano que muestra si se quiere eliminar la aplicación tiene el valor verdadero, se eliminará la aplicación en la base de datos.

Por otro lado, se ha realizado una gestión de errores. Si o bien hay un error al realizar las consultas o los campos no están correctamente rellenados, se mostrará un mensaje de error por pantalla. Si al contrario todo funciona correctamente, se mostrará un mensaje de que los cambios se han guardado correctamente. A continuación, se muestran dos ejemplos. La primera imagen, figura 11, es para un correcto funcionamiento mientras que la segunda imagen, figura 12, es, al contrario.

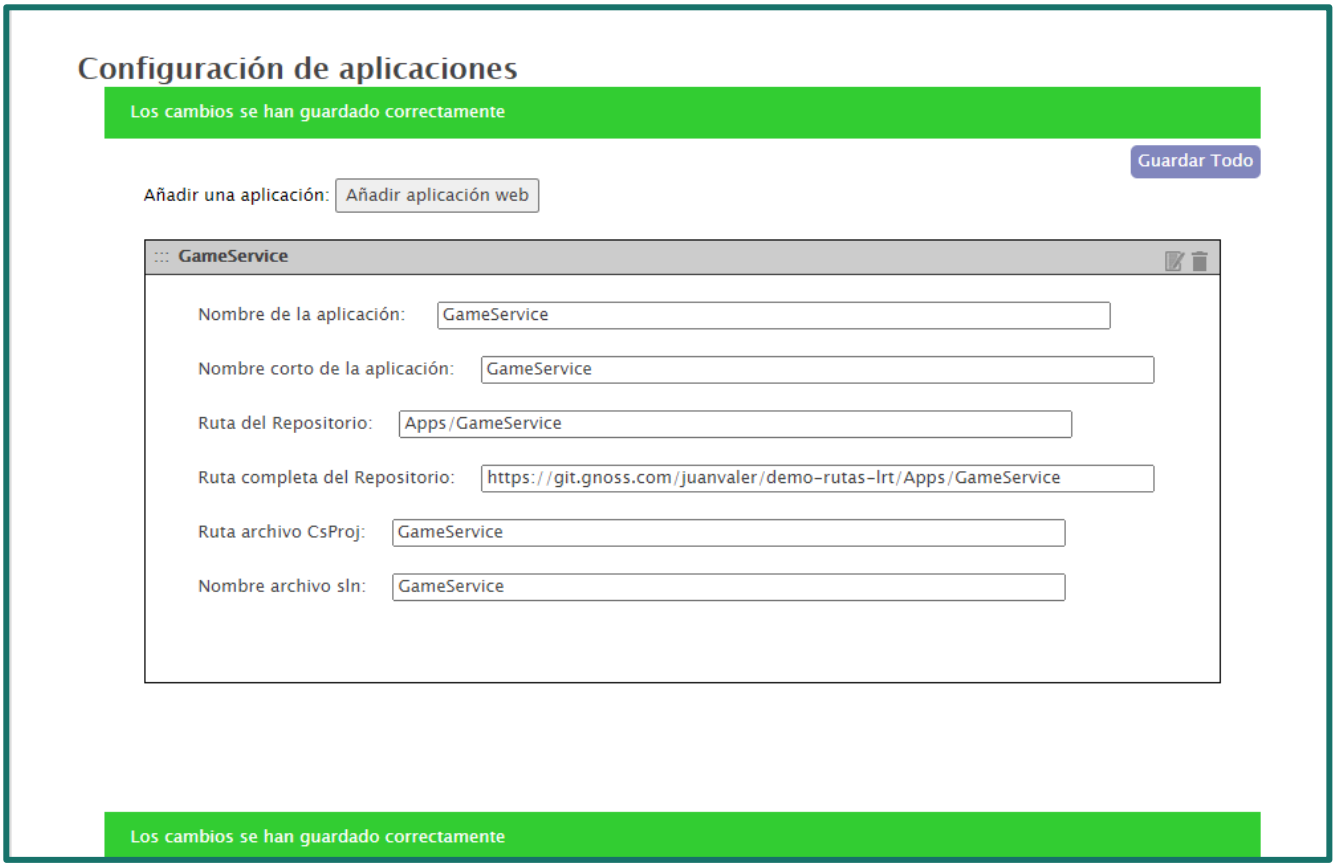


Figura 11 - Cambios guardados correctamente

Figura 12 - Error

4.2 CREACIÓN APLICACIÓN WEB EN EL SERVIDOR IIS

Es necesaria la creación de la aplicación web en todos los servidores en IIS. Por cada entorno de trabajo: desarrollo, preproducción y producción; existe un servidor. Excepto en el entorno de producción que existe más de uno. Estos últimos no están siempre todos encendidos ya que según la hora que sea habrá más demanda de ellos.

Para la creación de la aplicación web en el servidor IIS, se sigue el esquema mostrado en la figura 13.

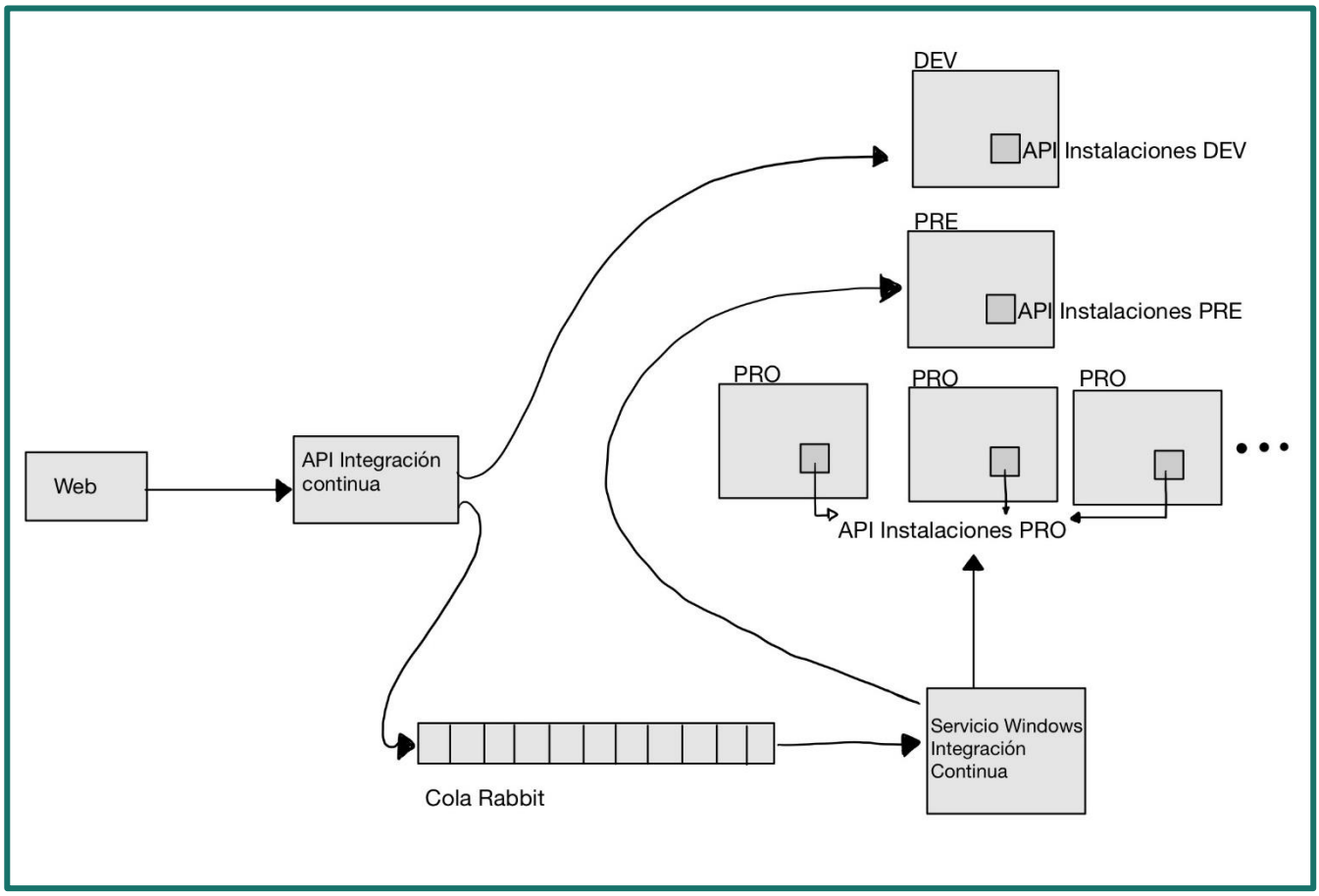


Figura 13 – Diagrama del funcionamiento de los componentes en la creación de las aplicaciones web

La web enviará los datos de las aplicaciones al API de Integración Continua. Una vez ahí, según la aplicación sea nueva o ya exista, se realizarán diferentes acciones. Si la aplicación existe, pero solo se quiere modificar sus datos, simplemente se actualizarán estos en la base de datos.

Si la aplicación recibida en el API de Integración Continua se quiere crear, se creará directamente en el servidor de desarrollo en IIS a través del API de Instalaciones. A continuación, se encolará la aplicación a una cola Rabbit. Además, existe un servicio Windows de Integración Continua que estará continuamente escuchando la cola. Cuando este servicio encuentre un evento en la cola, intentará instalarlo tanto en el servidor de preproducción como en los de producción.

Los servidores de producción al no estar siempre encendidos, puede ser que no permitan instalar la aplicación. Es por ello por lo que cuando el API de Instalaciones no pueda instalar la aplicación y devuelva un error, el servicio Windows de Integración Continua capturará este error y creará un fichero con los datos de la aplicación que no se ha podido crear.

Por otro lado, existirá también, otro hilo que se encargue de comprobar cada hora si hay aplicaciones pendientes de crear en algún servidor. Esto lo comprobará buscando si existen ficheros en un directorio concreto ya que estos se habrán creado si ha habido un error al instalar cada aplicación. Si encuentra alguno, volverá a intentar crear la aplicación en su correspondiente servidor. Si además lo consigue, se borrará el fichero correspondiente. En cambio, si el servidor sigue apagado, lo seguirá intentando cada hora hasta que lo consiga.

Por último, si la aplicación recibida en el API de Integración Continua se quiere borrar, ocurrirá lo mismo que si se quiere crear. La única diferencia es que llamará al API de Instalaciones para borrar la aplicación en vez de para crearla.

Para la instalación de las aplicaciones en IIS, se ha creado un proceso que ejecuta comandos en Appcmd. En primer lugar, se añadirá un *Application Pool*. Esto es un grupo de aplicaciones que están asociados a procesos de trabajo y contienen una o más aplicaciones, provocando aislamiento entre aplicaciones. Después, se añadirá la aplicación. Para la ejecución de estos comandos, se ha necesitado permisos especiales. A continuación, en la figura 14, se muestra el código utilizado.

```
0 referencias
public void Create(string nombreCorto, string UrlBaseServiciosExternos)
{
    string rutaBase = $@"c:\SitiosWeb";
    string path = $@"{rutaBase}\{UrlBaseServiciosExternos}\{nombreCorto}";
    if (!Directory.Exists(path))
    {
        Directory.CreateDirectory(path);
    }
    ExecuteCommand($"appcmd", $"add apppool /name:{UrlBaseServiciosExternos}_{nombreCorto}");
    ExecuteCommand("appcmd", $"add app /site.name:{UrlBaseServiciosExternos} /path:/{nombreCorto} /physicalPath:{path}");
}

3 referencias
static void ExecuteCommand(string pComando, string pParametros)
{
    ProcessStartInfo procStartInfo = new ProcessStartInfo(pComando, pParametros);
    procStartInfo.WorkingDirectory = @"C:\Windows\System32\inetsrv";
    procStartInfo.RedirectStandardOutput = true;
    procStartInfo.RedirectStandardError = true;
    procStartInfo.UseShellExecute = false;
    procStartInfo.CreateNoWindow = false;
    procStartInfo.Verb = "runas";
    Process proc = new Process();
    proc.StartInfo = procStartInfo;
    proc.Start();
    StringBuilder sb = new StringBuilder();
    while (!proc.StandardOutput.EndOfStream)
    {
        sb.AppendLine(proc.StandardOutput.ReadLine());
    }

    LoggingService.EscribirLog($"Ejecutado {pComando} {pParametros}. Exit code: {proc.ExitCode}. {Environment.NewLine}Respuesta: {sb.ToString()}");
    if (proc.ExitCode != 0)
    {
        while (!proc.StandardError.EndOfStream)
        {
            sb.AppendLine("Error: ");
            sb.AppendLine(proc.StandardError.ReadLine());
        }
        throw new Exception($"Error al ejecutar el comando appcmd con la instrucción: \n\t\"{pComando}\". Respuesta: {sb.ToString()}");
    }
}
```

Figura 14 - Instalación en IIS

Siguiendo el ejemplo del apartado anterior, cuando se quiera crear la aplicación “GameService”, se realizará una petición al API de Instalaciones como la siguiente.

```
UrlApiInstalaciones/iis/create?nombreCorto=GameService&UrlBaseServiciosExternos=servicios.didactalia.com
```

Esta petición, como se muestra en la figura 15, creará el grupo de aplicaciones “servicios.didactalia.com” y añadirá la aplicación “GameService”.

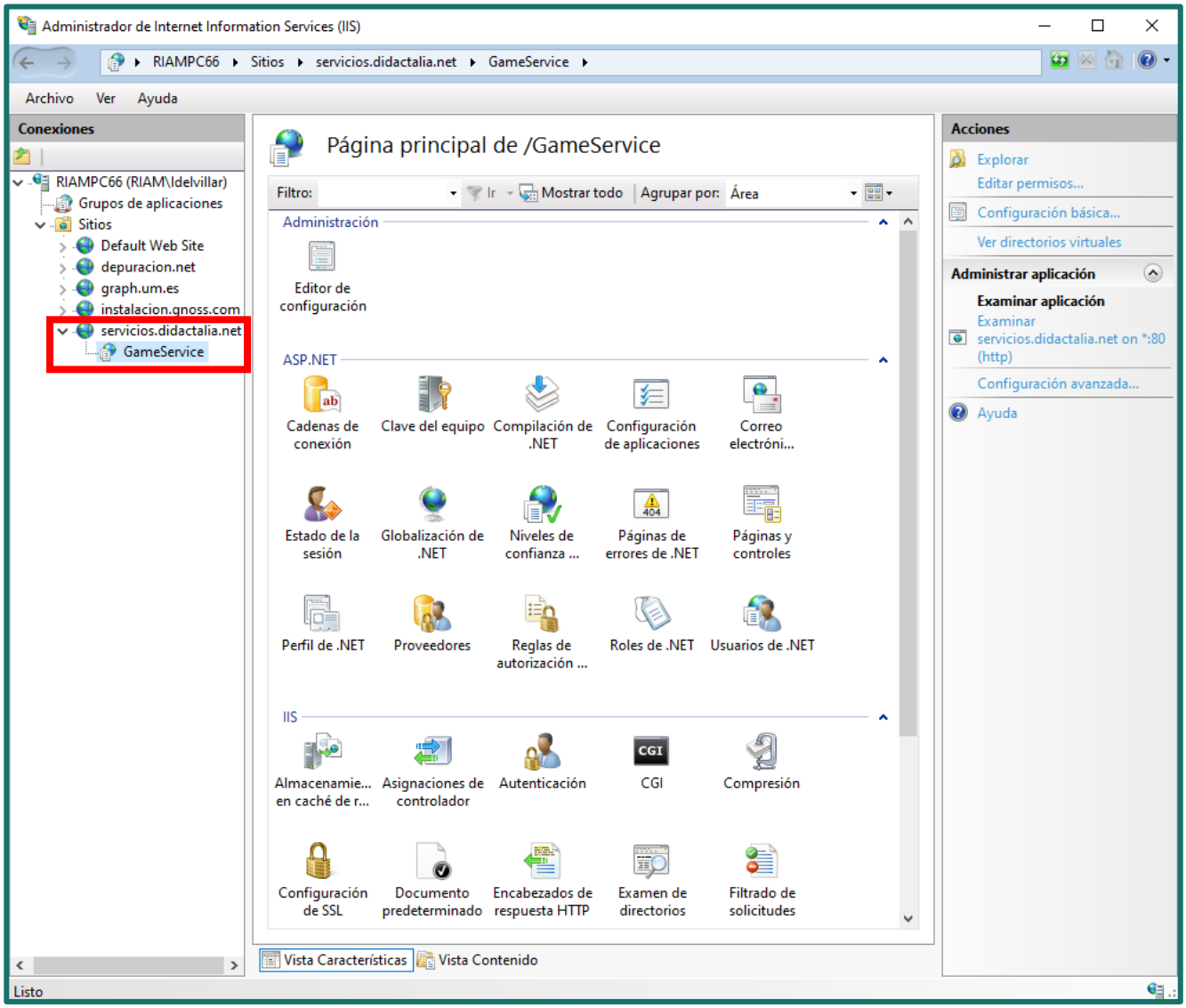


Figura 15 - Creación de la aplicación en IIS

4.3 COMPILACIÓN AUTOMÁTICA CON JENKINS

Para la compilación automática, se ha creado una tarea en Jenkins parametrizada. Así, se realizará una petición a Jenkins desde el Servicio Windows de Integración Continua mandando los parámetros necesarios.

Los parámetros enviados son los siguientes:

- NombreArchivoSln: el nombre del archivo Sln
- UrlRepositorio: ruta completa al repositorio Git donde está el código subido
- RutaRepositorio: la carpeta donde se encuentra almacenado en Git
- Rama: la rama actual en Git
- Repositorio: la ruta del repositorio Nexus
- Versión: la versión
- NombreServicio: nombre del servicio o el nombre corto de la aplicación

La tarea Jenkins se encargará de realizar la compilación de la aplicación. Para ello, realizará diferentes subtareas.

En primer lugar, ejecutará el siguiente comando en Windows:

```
c:\nuget\nuget.exe restore "%WORKSPACE%\%RutaRepositorio%\%NombreArchivoSln%.sln"
```

Realizará la restauración de los paquetes NuGet instalando todas las dependencias del proyecto que aparecen en el archivo de proyecto. %WORKSPACE% es el espacio de trabajo definido en Jenkins mientras que %NombreArchivoSln% es el parámetro para el nombre del archivo Sln.

A continuación, se construirá un proyecto o solución de Visual Studio utilizando MSBuild:

MSBuild Version: compilar con net 4.6.1 y visual studio 2017 MSBuild Build File: .\${NombreArchivoSln}\${NombreArchivoSln}.sln Command Line Arguments: /p:PublishProfile=%WORKSPACE%\%RutaRepositorio%\%NombreArchivoSln%\Properties\PublishProfiles\Nexus.pubxml
MSBuild Version: publicar con net 4.6.1 y visual studio 2017 MSBuild Build File: .\${NombreArchivoSln}\${NombreArchivoSln}.sln Command Line Arguments: /p:PublishProfile=%WORKSPACE%\%RutaRepositorio%\%NombreArchivoSln%\Properties\PublishProfiles\Nexus.pubxml /p:Targets=Publish /p:DeployOnBuild=true

Con estos comandos, se compilará la aplicación y se publicará. Nexus.pubxml es el perfil de publicación.

Por último, se ejecutarán los siguientes comandos en Windows:

```
echo ^<?xml version="1.0" encoding="utf-8"?^> > package.nuspec
echo ^<package xmlns="http://schemas.microsoft.com/packaging/2010/07/nuspec.xsd"^> >> package.nuspec
echo      ^<metadata^> >> package.nuspec
echo      ^<id^>% NombreServicio %^</id^> >> package.nuspec
echo      ^<version^>%version%^</version^> >> package.nuspec
echo ^<description^>Servicio compilado con Jenkins^</description^> >> package.nuspec
echo ^<authors^>Jenkins^</authors^> >> package.nuspec
echo ^</metadata^> >> package.nuspec
echo ^</package^> >> package.nuspec
"C:\NuGet\nuget.exe" pack
C:\NuGet\nuget.exe setapikey 19957ff2-be43-3d4a-9c0e-72e204137fb4 -source %repositorio% -NonInteractive
C:\Nuget\nuget.exe delete  % NombreServicio % %version% -Source %repositorio%  -NonInteractive
C:\NuGet\nuget.exe push *.nupkg -Source %repositorio% -NonInteractive
```

Estos comandos crearán un archivo .nuspec y un paquete de NuGet basado en este archivo. Además, subirá el paquete al repositorio de Nexus que se pasa por el parámetro llamado repositorio.

Para la comprobación de que la tarea de Jenkins ha terminado de manera exitosa se realizará lo siguiente. En primer lugar, se realizará la siguiente petición a Jenkins:

```
http://IP:PUERTO/job/CompilacionServicioEspecifico/buildWithParameters?token=TOKEN&NombreArchivoSln=NOMBREARCHIVOSLN&repositorio=http://git.gnoss.com:PUERTO/repository/serviciosweb&version=2021.03.16&RutaRepositorio=RUTAREPOSITORIO
```

Siendo IP y PUERTO la IP y el puerto del servicio Jenkins en la empresa. Además, se le pasará un token generado en Jenkins para que cualquiera no pueda realizar la petición. Esta petición devolverá un parámetro en la cabecera llamado "Location" con una URL. En la figura 16 se muestra la cabecera.

Figura 16

Después, se realizará una petición a esa URL para obtener el número de la tarea de Jenkins.

```
http://IP:PUERTO/queue/item/477/api/json?token=TOKEN
```

Esta petición responderá con un Json que tendrá entre otros datos, el número de la tarea que se está ejecutando. A continuación, se muestra.

```
{
        "_class": "hudson.model.Queue$LeftItem",
        "actions": [
                [...]
        ],
        "blocked": false,
        "buildable": false,
        "id": 477,
        "inQueueSince": 1616676256285,
        "params": "\nNombreArchivoSln=GameService\nrepositorio=http://git.gnoss.com:8081/repository/serviciosweb\nversion=2021.03.16\nRutaRepositorio=didactalia-servicios",
        "stuck": false,
        "task": {
```

```
                "_class": "hudson.model.FreeStyleProject",
                "name": "CompilacionServicioEspecifico",
                "url": "http://localhost:8080/job/CompilacionServicioEspecifico/",
                "color": "blue_anime"
        },
        "url": "queue/item/477/",
        "why": null,
        "cancelled": false,
        "executable": {
                "_class": "hudson.model.FreeStyleBuild",
                "number": 36,
                "url": "http://localhost:8080/job/CompilacionServicioEspecifico/36/"
        }
}
```


Por último, se realizará una nueva petición a Jenkins con el número de la tarea.

```
http://IP:PUERTO/job/CompilacionServicioEspecifico/36/api/json?token=TOKEN
```

Esta petición devolverá otro archivo Json que tendrá entre otros datos, el resultado de la tarea. Si la tarea ha terminado exitosa, tendrá un campo llamado “result” con el valor “SUCCESS”. Por el contrario, si falla, su valor será “FAILURE”.


```
{
        "_class": "hudson.model.FreeStyleBuild",
        "actions": [
                [...]
        ],
        "artifacts": [],
        "building": false,
        "description": null,
        "displayName": "#36",
        "duration": 135959,
        "estimatedDuration": 69370,
        "executor": null,
        "fullDisplayName": "CompilacionServicioEspecifico #36",
        "id": "36",
        "keepLog": false,
        "number": 36,
        "queueId": 477,
```

```
        "result": "SUCCESS",
        "timestamp": 1616676262610,
        "url": "http://localhost:8080/job/CompilacionServicioEspecifico/36/",
        "builtOn": "",
        "changeSet": {
                "_class": "hudson.plugins.git.GitChangeSetList",
                "items": [],
                "kind": "git"
        },
        "culprits": []
}
```


Además, como la tarea de Jenkins tarda un tiempo en terminar de ejecutarse, se realizará la petición cada segundo hasta que devuelva un resultado (Success o Failure). Esto significará que se ha terminado y que podemos lanzar el despliegue.

4.4 DESPLIEGUE

Según el entorno en el que se encuentre el proyecto, se realizará el despliegue de una manera u otra. El despliegue de las aplicaciones en los entornos de preproducción y producción se integra dentro de un proceso de Integración Continua y despliegue automatizado que ya existía en la plataforma GNOSS. Este proceso se utiliza para desplegar las configuraciones de la plataforma de un entorno a otro ya que según la comunidad en la que se esté, las configuraciones cambian. Por ejemplo, la web del Museo del Prado y Didactalia, proyectos realizados por la empresa, tendrán diferentes configuraciones.

En el momento de realizar el despliegue de las aplicaciones web, se llamará a un método que, en primer lugar, devolverá todas las aplicaciones específicas del proyecto en concreto en el que esté. Después, por cada aplicación específica intentará desplegarla en el servidor correspondiente al proyecto. Si al intentar desplegarla en el servidor, este está apagado, se creará un fichero con los datos del servidor que ha fallado y los de la aplicación que no se ha conseguido desplegar.

De manera similar a la creación de las aplicaciones en el servidor, se comprobará cada hora si hay aplicaciones pendientes de desplegar. Esto se realizará comprobando si existen ficheros en un determinado directorio y pasándolos por parámetro al método que realiza el despliegue.

El despliegue se realizará llamando al API de Instalaciones. Se le pasarán los parámetros de la URL del repositorio de Nexus, el dominio que será un campo llamado UrlBaseServiciosExternos, el nombre del servicio que es el nombre corto de la aplicación, el entorno y el tipo de aplicación.

A continuación, se muestra un ejemplo.

Siguiendo con el ejemplo GameService, si se quiere desplegar la aplicación se realizará la siguiente petición al API de Instalaciones.

```
UrlApiInstalaciones/?repositorio=http://git.gnoss.com:8081/repository/trygnoss/GameService/version&dominio=servicios.didactalia.com&servicio=GameService&serviciocompleto=GameService&version=version&entorno=develop&tipoAplicacion=0"
```

El tipo de aplicación es 0 debido a que es una aplicación web.

El API de Instalaciones desplegará la aplicación en el directorio C:/SitiosWeb/servicios.didactalia.com como se muestra en la figura 17.

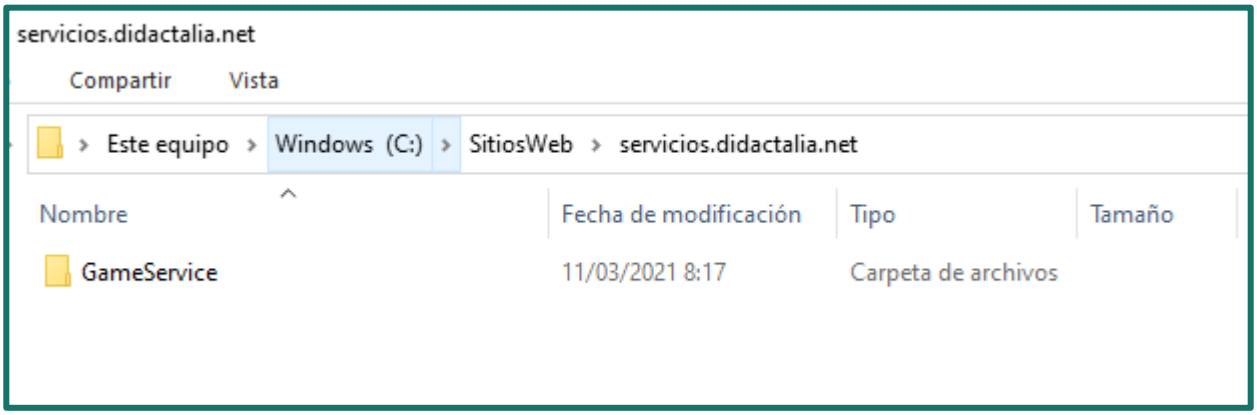


Figura 17 - Ejemplo de despliegue

Además, cada aplicación tendrá ficheros que cambien según el entorno. Por lo tanto, en el API de Instalaciones se hará una búsqueda de archivos que terminen en ".dev" y si se encuentran, según el entorno en el que se encuentre, se sustituirá el archivo original por el acabado en ".dev" si estamos en desarrollo, ".pre" si estamos en preproducción o ".pro" si estamos en producción.

Como alternativa, se podría haber usado Web Deploy en vez del Api de Instalaciones. Web Deploy es una herramienta de Microsoft que simplifica significativamente la migración, administración e implementación de servidores web y sitios web de IIS [10]. Sin embargo, se ha utilizado el Api de Instalaciones por decisión de la empresa ya que este API ya existía y ya estaba desarrollado y desplegado en todos los servidores.

Por otro lado, en la interfaz de la web se añade un botón que permitirá el despliegue de cada aplicación por separado en el servidor de desarrollo. Así, cuando se pulse dicho botón, la web enviará los datos de la aplicación concreta al API de Integración Continua. Este último, encolará el evento en una nueva cola Rabbit que se dedique únicamente al despliegue de estas aplicaciones. Después, el servicio Windows de Integración Continua comprobará si hay elementos en dicha cola y si los hay, compilará la aplicación mandando una petición a Jenkins y la desplegará únicamente en el servidor de desarrollo. Esto se realiza así debido a que el entorno de desarrollo, al ser un entorno donde se llevan a cabo muchos cambios, se tiene que realizar en repetidas ocasiones el despliegue de las aplicaciones.

A continuación, en la figura 18, se muestra una captura de la interfaz web con el botón de despliegue.

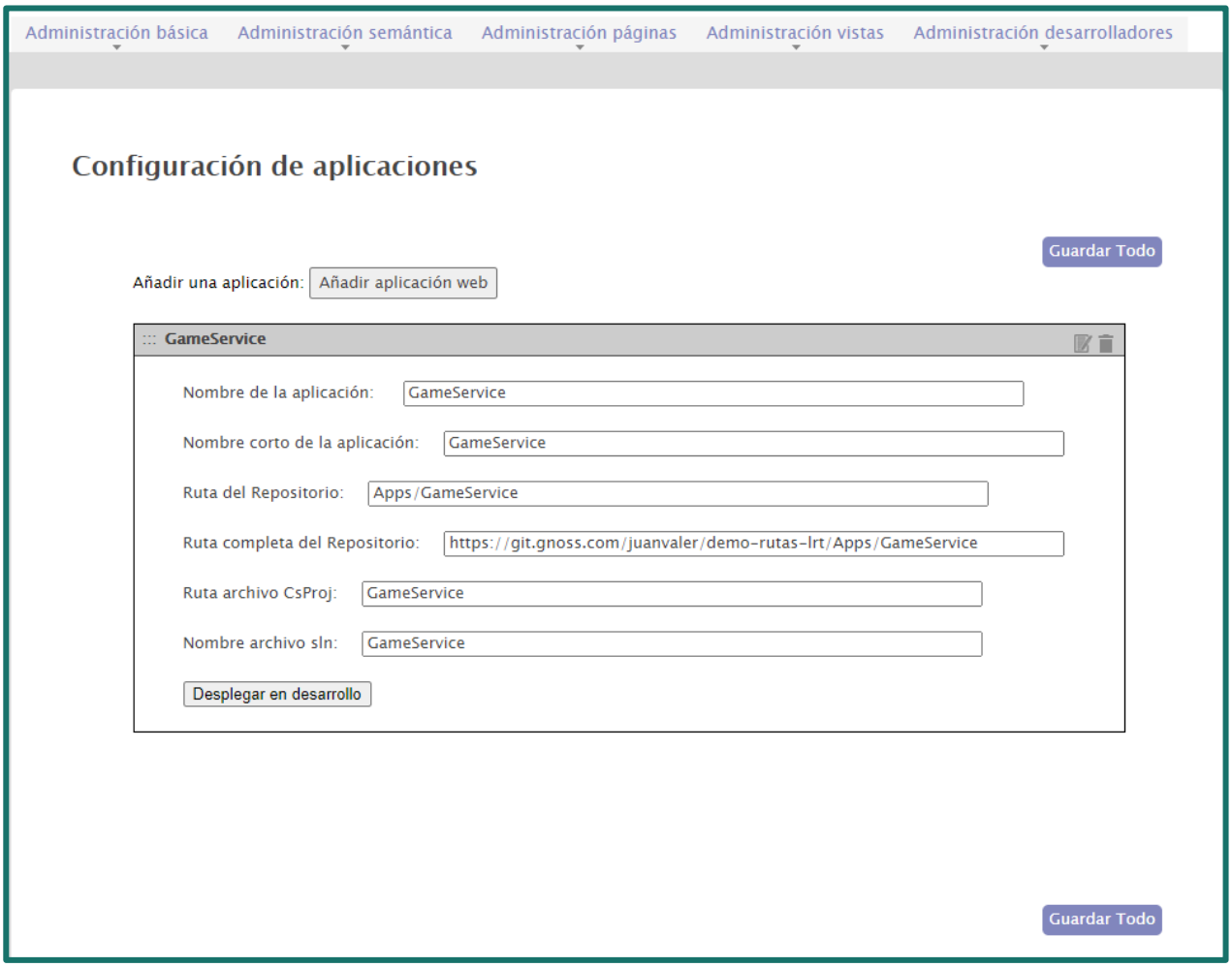


Figura 18 - Botón de despliegue en desarrollo

4.5 DESPLIEGUE DE APLICACIONES EJECUTABLES

Hasta ahora se ha creado la funcionalidad necesaria para que en Gnoss se puedan crear y desplegar aplicaciones Web. Ahora se debe incluir un nuevo tipo de aplicación, aplicación ejecutable.

En la interfaz web, el usuario debe poder elegir en el momento de la creación que tipo de aplicación se va a crear: Web o Ejecutable. La diferencia entre los dos tipos de aplicaciones es que las aplicaciones Web se exponen a través del IIS y las aplicaciones ejecutables dispondrán de un botón desde la página de administración de aplicaciones que permitirá, una vez desplegada, ejecutarla de manera remota en el servidor.

La creación de una aplicación ejecutable en el servidor es más sencilla que la creación de una aplicación Web ya que no es necesario añadir ninguna aplicación al IIS. Basta con crear el directorio en el que se va a alojar. Sin embargo, debemos solicitar al usuario un dato más, el nombre del ejecutable que posteriormente se ejecutará de manera remota.

Para ejecutar la tarea de manera remota, se debe crear un nuevo servicio Windows (servicio Windows de Instalaciones) que se encargue de arrancar la aplicación. Cuando el usuario pinche en el botón ejecutar, la web enviará al Api de Integración Continua un evento para notificarle qué aplicación se desea ejecutar. El Api de Integración Continua enviará un mensaje a una cola que leerá el servicio Windows de Instalaciones. Habrá una cola por cada servidor y entorno. Esto es, cada cola se llamará

"colaEjecutar[*NombreEntorno*][*NombreServidor*]". Siendo NombreEntorno el nombre del entorno actual y NombreServidor el nombre del servidor que hay en ese entorno. El Servicio Windows de Instalaciones cogerá el dato del entorno y servidor en el que se encuentra, en el fichero de configuración.

Por otro lado, este servicio se encarga de arrancar la aplicación y guardar la salida estándar en la base de datos. De esta manera, la web podrá mostrarle al usuario en tiempo real la salida por consola de su aplicación, y permitirle enviar comandos. Para mostrar la salida por consola al usuario, la Web mandará peticiones cada segundo al Api de Instalaciones del servidor de aplicaciones pidiendo la salida de escritura de esta aplicación. Si el usuario envía algún comando, este comando se le manda al Api de Integración Continua y este enviará un mensaje a una cola con el comando que lo leerá el servicio Windows de Aplicaciones. Además, deberá existir una opción para matar la aplicación o parar la ejecución. Esto último mandará un mensaje en la cola de aplicaciones a través del Api de Integración Continua para notificarle que debe terminar la ejecución de esa aplicación.

A continuación, en la figura 19, se muestra el diagrama de secuencia.

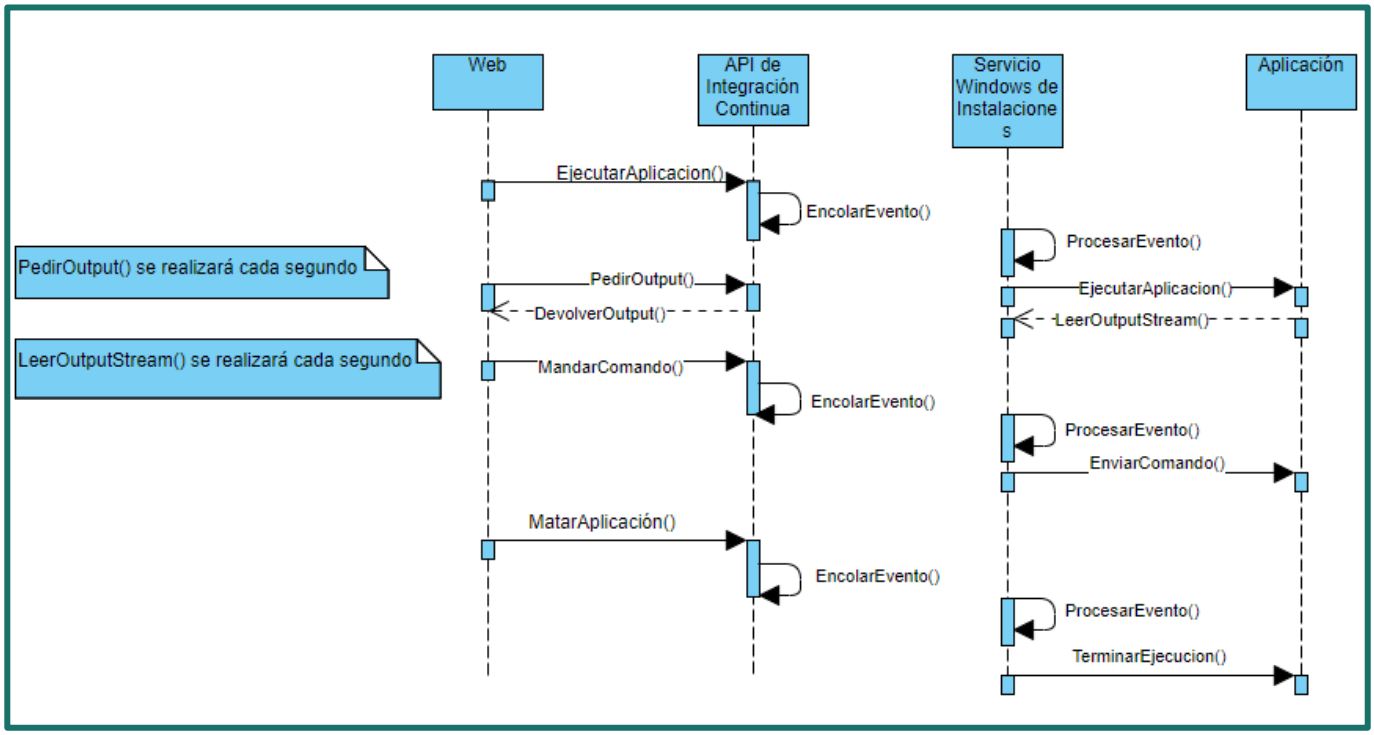


Figura 19 - Diagrama de secuencia

Para realizar todo ello, en primer lugar, se ha modificado la interfaz web para que se distinga entre cada tipo de aplicación: web o ejecutable. Además, en las aplicaciones ejecutables, se dispone de un botón para ejecutar y otro campo para guardar el nombre del archivo ejecutable.

En la figura 20 se muestra cómo queda la interfaz web.

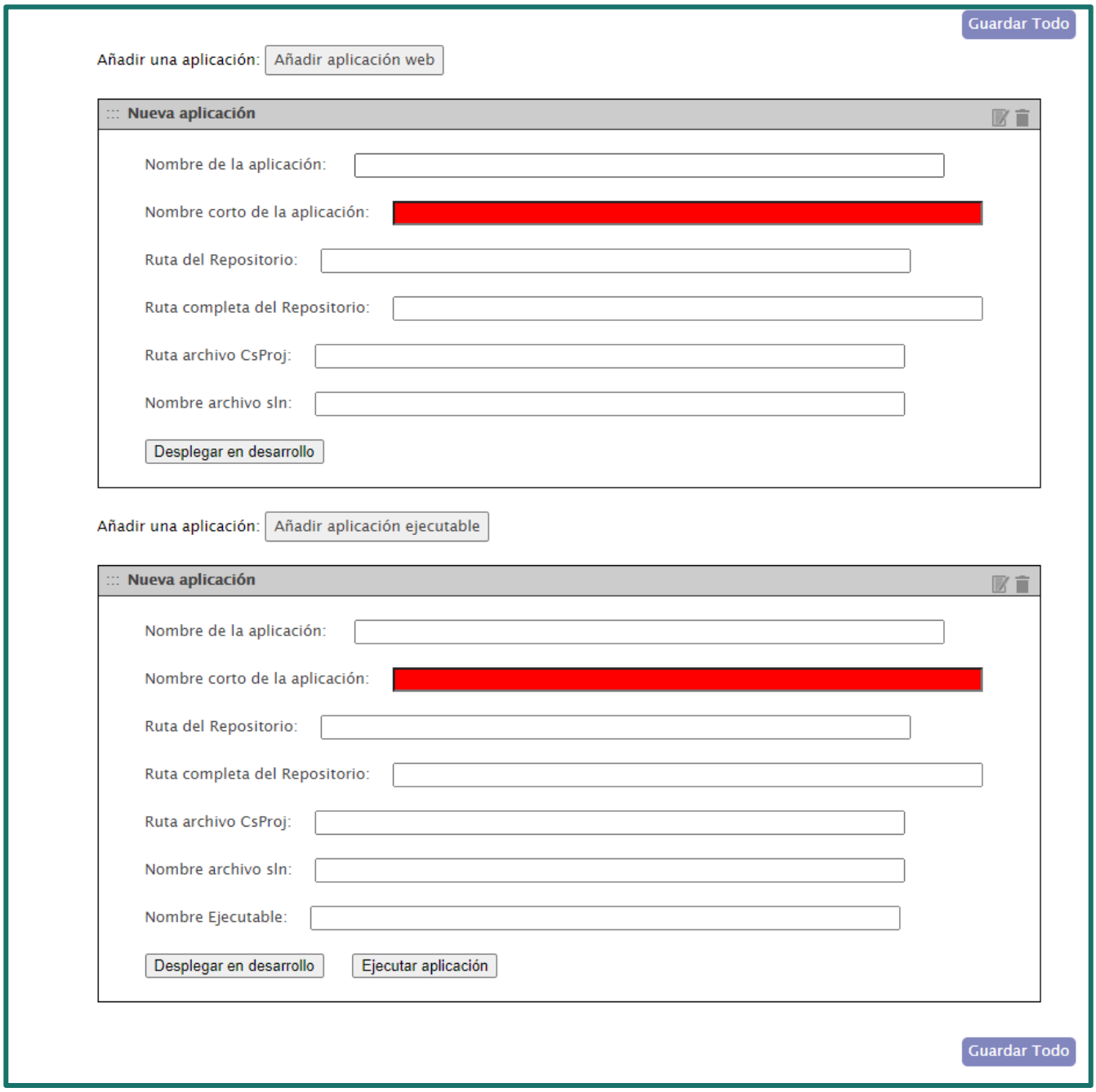


Figura 20 - Interfaz de despliegue de aplicaciones ejecutables

Cuando se añada una aplicación ejecutable, además de crearla en la base de datos, se llamará al API de Instalaciones. Este, creará un directorio en C://Tareas con el nombre de la aplicación ejecutable. Ahí es donde se desplegará posteriormente la aplicación.

Por otro lado, para almacenar los datos de las aplicaciones ejecutables se ha creado una nueva tabla llamada AplicacionEjecucion. Esta tabla tendrá un identificador, un estado que podrá ser ejecución, espera, terminado o error. Además, tendrá también dos campos llamados output y error que mostrarán lo que devuelve la consola y el error que devuelve por consola respectivamente. Por último, también mostrará la fecha de inicio de la ejecución y su fin, si esta ha terminado.

En la figura 21 se muestra el modelo entidad-relación de las entidades Aplicación y AplicacionEjecucion.

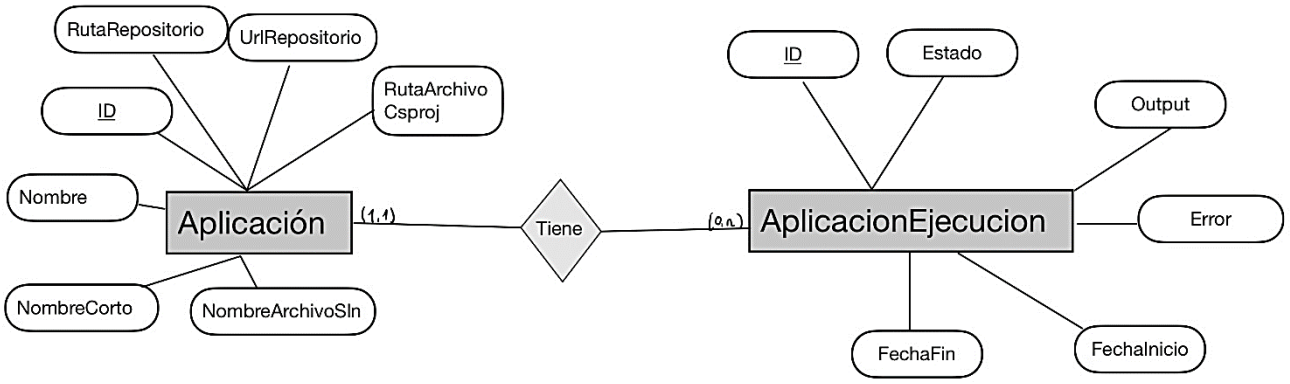


Figura 21 - Diagrama E-R AplicacionEjecucion

Después, se ha modificado el código de la compilación de las aplicaciones ya que es diferente la compilación de una aplicación web que la de una aplicación ejecutable. Por ello, se ha creado una nueva tarea en Jenkins que lo realice. La diferencia entre las dos tareas de Jenkins es que al construir un proyecto o solución de Visual Studio utilizando MSBuild no se pasará el perfil de publicación. El comando será como el siguiente.

MSBuild Version: compilar con net 4.6.1 y visual studio 2017 MSBuild Build File: .*${NombreArchivoSln}**${NombreArchivoSln}*.sln
MSBuild Version: publicar con net 4.6.1 y visual studio 2017 MSBuild Build File: .*${NombreArchivoSln}**${NombreArchivoSln}*.sln Command Line Arguments: /p:Targets=Publish /p:DeployOnBuild=true /p:outdir="c:\Publicacion"

Así, cuando se pinche en el botón de desplegar y se realice la compilación, se distinguirá si es aplicación web o aplicación ejecutable.

El servidor en el que se ejecutan estas aplicaciones será distinto a los servidores que alojan las Webs, existirá un servidor de ejecución de aplicaciones por entorno. Por lo tanto, al realizar el despliegue, se llamará al Api de Instalaciones del servidor del entorno en el que se quiera instalar. Este desplegará la aplicación en el directorio creado con anterioridad al crearse la aplicación.

A continuación, se ha creado el Servicio Windows de Instalaciones en el cual se procesa un evento de la cola y se ejecuta la aplicación del evento recibido. Distinguirá entre tres tipos de eventos según se quiera ejecutar una aplicación, terminar la ejecución de una aplicación o mandar un comando a la aplicación. Además, se dispondrá de un diccionario que almacene como clave el identificador de la aplicación y como valor el proceso que se está ejecutando. Cuando se lea un mensaje de la cola y sea del tipo ejecutar, se comprobará en el diccionario que no haya otro proceso ejecutándose de la misma aplicación. Después, creará un nuevo proceso que ejecutará la aplicación correspondiente. Si el mensaje de la cola es del tipo mandar comando, comprobará que esa aplicación esté ejecutándose en ese instante y le enviará el comando recibido. Si el mensaje es de tipo

terminar ejecución, terminará la ejecución de la aplicación. Además, la salida estándar de cada ejecución de las aplicaciones se almacenará en la base de datos.

Por último, se han realizado las siguientes modificaciones de la interfaz de la web. En primer lugar, aparecerá un listado con las últimas diez ejecuciones. En él, se podrá ver el identificador de cada ejecución, la fecha de inicio y de fin de la ejecución, el estado y el log. El log devolverá el campo output de la ejecución o el error si ha fallado. Después, si hay una ejecución en curso, se mostrará el campo output, un cuadro de texto para enviar comandos y un botón para parar la ejecución. El cuadro de texto con el campo output irá actualizándose cada segundo para mostrar en tiempo real lo que devuelve la salida estándar de la aplicación.

A continuación, en la figura 22, se muestra un ejemplo de la interfaz web con una aplicación ejecutándose.

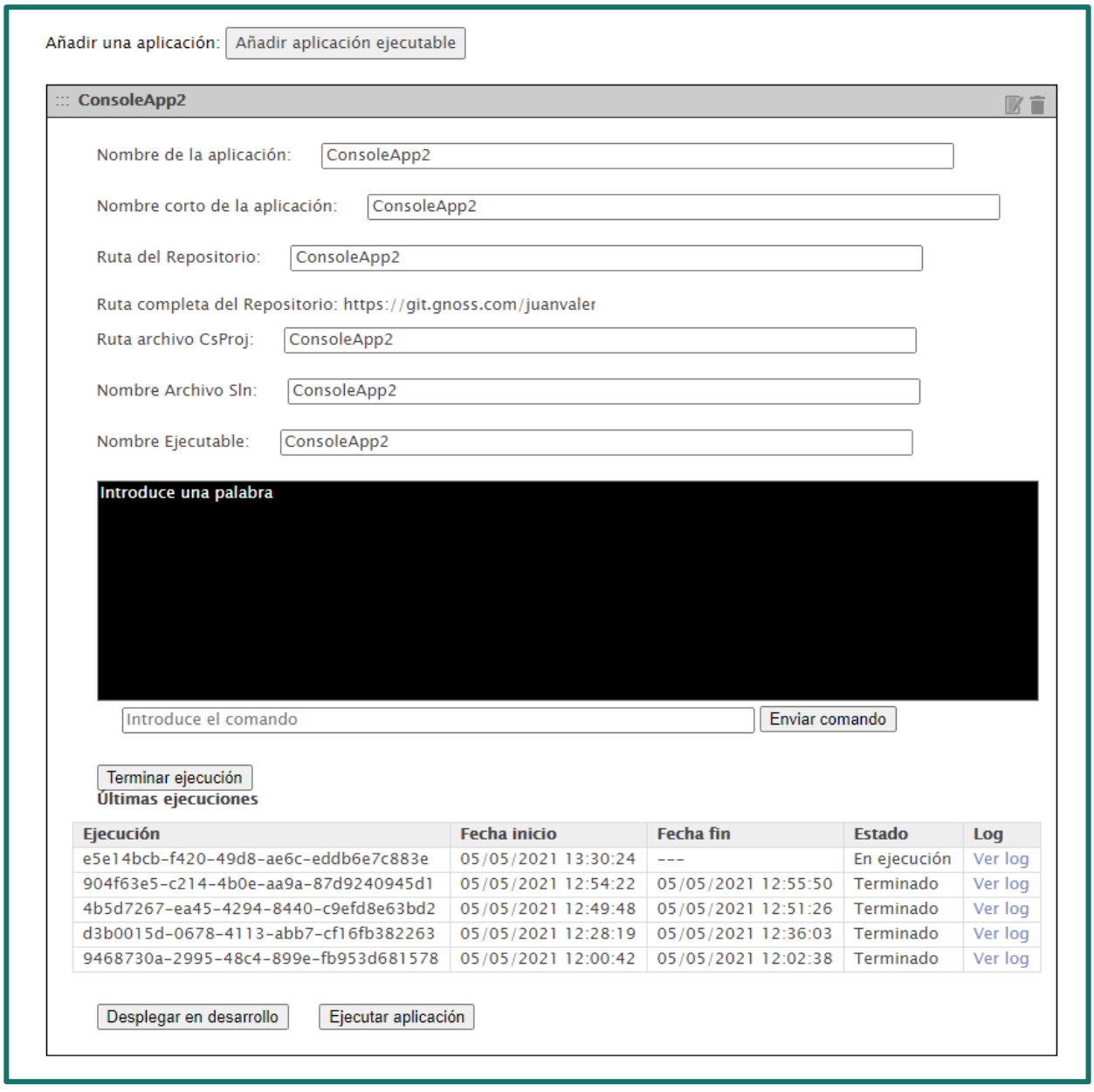


Figura 22 - Ejemplo de aplicación ejecutándose

4.6 PRUEBAS DE USUARIO

Las pruebas de usuario es un método que nos ayuda a medir el grado de eficacia, eficiencia y satisfacción de los usuarios que utilicen la herramienta. Así, se puede demostrar los problemas de usabilidad que experimentan los usuarios.

Según un estudio realizado por Nielsen se demostró que tan solo necesitamos quince personas para encontrar el 100% de los problemas de usabilidad. Aunque con dos personas ya se encontraría al menos la mitad de los problemas [9].

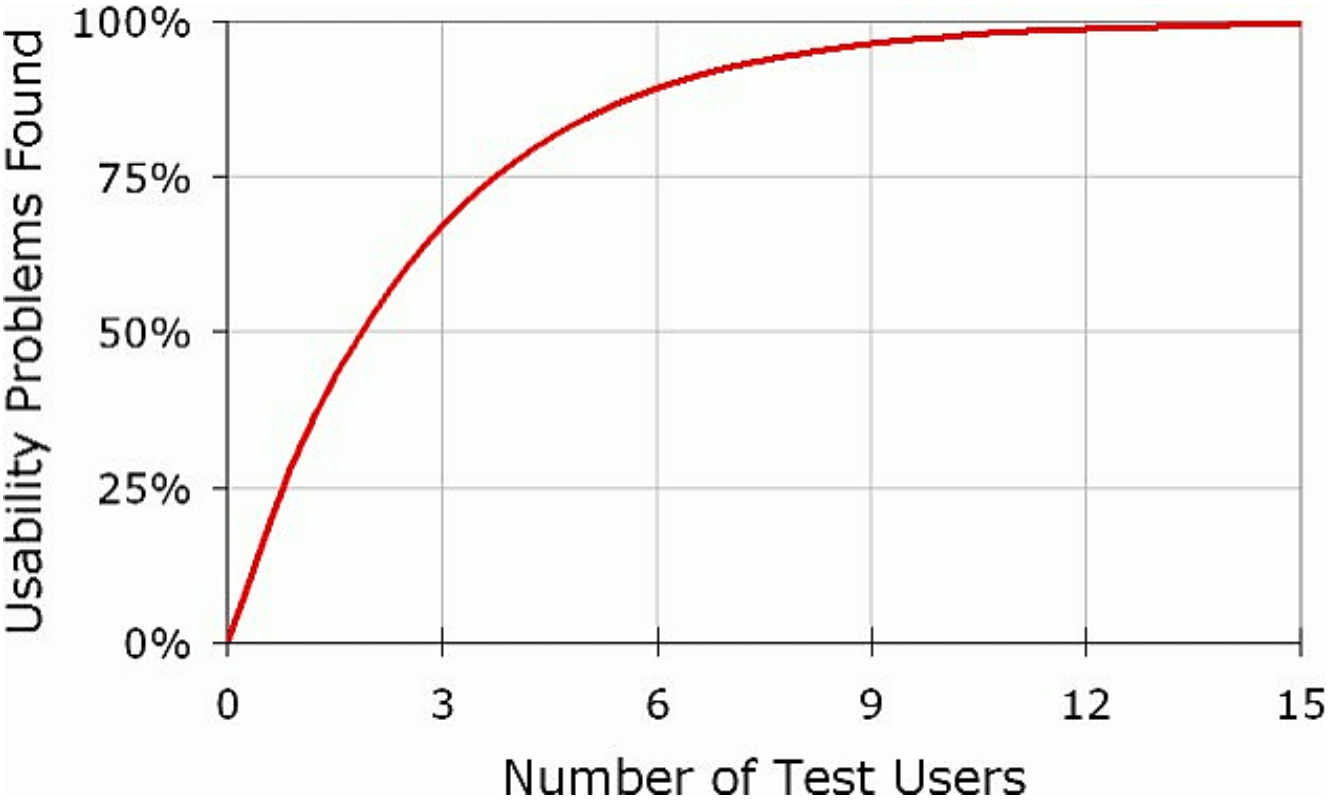


Figura 23 – Número de problemas de usabilidad encontrados respecto a pruebas de usuario realizadas [9]

Para la realización de las pruebas de usuario se emplearán dos pruebas diferentes. Se utilizará el test SUS (System Usability Scale) y un formulario con diferentes preguntas necesarias para valorar lo realizado en este proyecto. Además, se adjuntará con los test documentación para realizar la administración de aplicaciones web y ejecutables de manera correcta (Ver anexo). Con los resultados obtenidos se corregirán los fallos y se valorará mejorar el proyecto.

A continuación, se muestran las preguntas para el primer test.

1. Nombre y apellidos del usuario
2. ¿Has conseguido crear una aplicación Web o ejecutable?
3. ¿Cuándo pinchas en el botón ejecutar, se ejecuta la aplicación?
4. ¿La aplicación permite enviar comandos?
5. ¿Cuándo pinchas en el botón terminar ejecución, se para la aplicación?
6. ¿Has detectado algún fallo?
7. Si ha marcado "Sí" en la anterior pregunta explique cuál ha sido.
8. Sugerencias de mejora

El test SUS en cambio, tendrá las siguientes preguntas.

1. Creo que usaría este sistema frecuentemente
2. Encuentro este sistema innecesariamente complejo
3. Creo que el sistema fue fácil de usar
4. Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar este sistema
5. Las funciones de este sistema están bien integradas
6. Creo que el sistema es muy inconsistente
7. Imagino que la mayoría de la gente aprendería a usar este sistema en forma muy rápida
8. Encuentro que el sistema es muy difícil de usar
9. Me siento confiado al usar este sistema
10. Necesité aprender muchas cosas antes de ser capaz de usar este sistema.

Estos test se enviarán como formularios de Google en los que se registrarán las respuestas obtenidas.

Se han realizado las pruebas a 3 personas y los resultados del test SUS han sido los siguientes. Siendo 1 completamente en desacuerdo y 5 completamente de acuerdo.

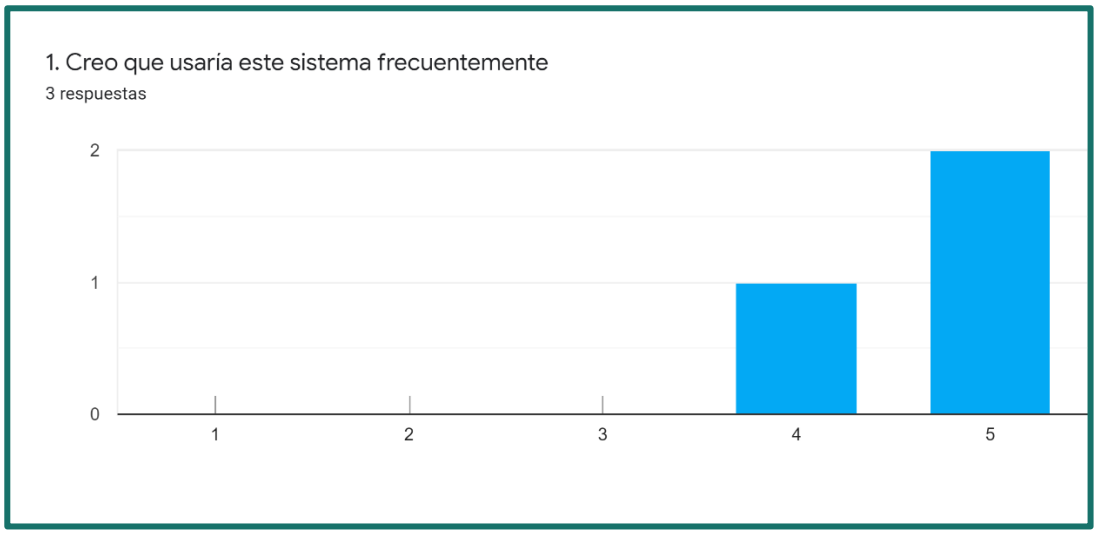


Figura 24 - Creo que usaría este sistema frecuentemente

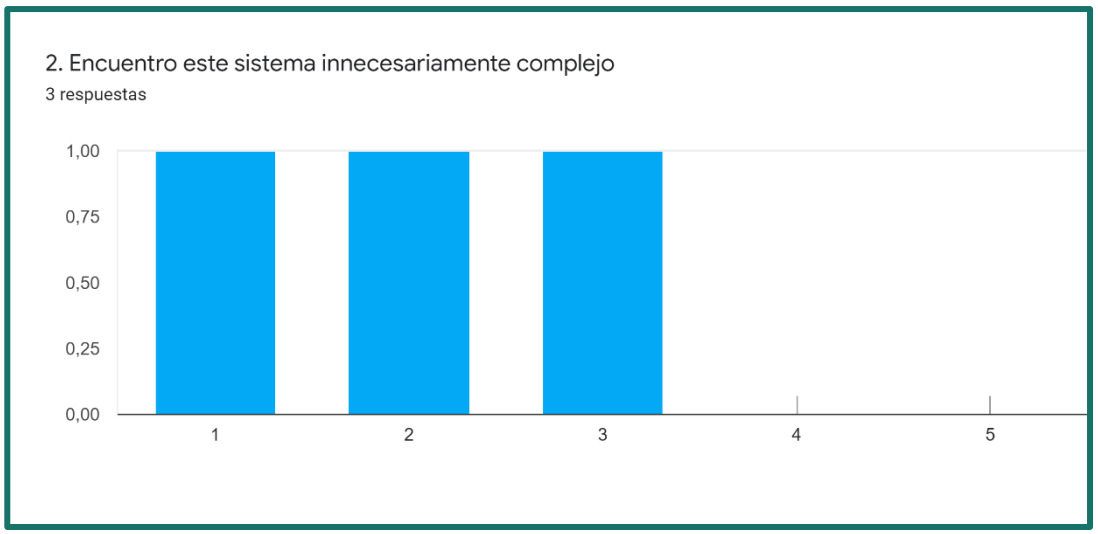


Figura 25 - Encuentro este sistema innecesariamente complejo

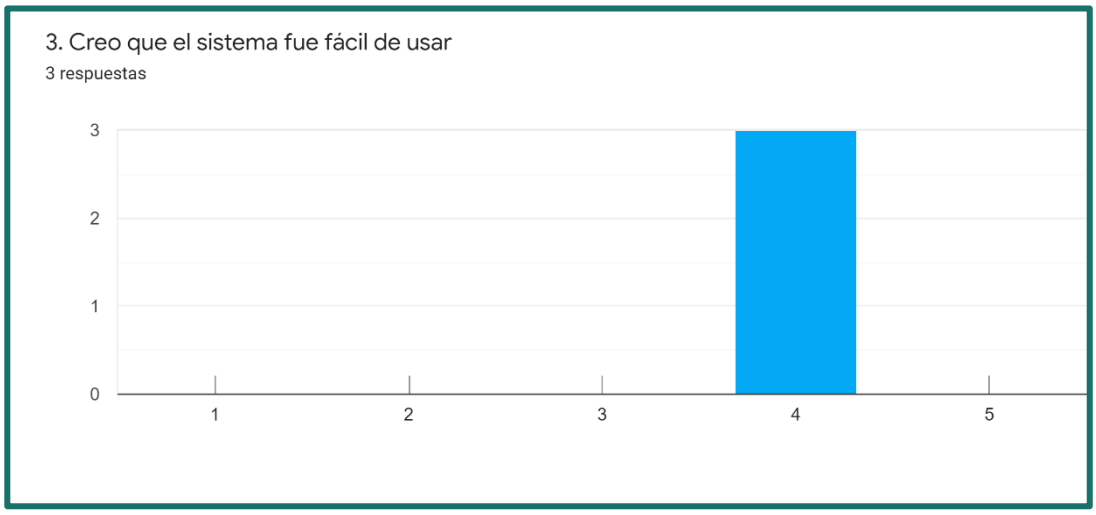


Figura 26 - Creo que el sistema fue fácil de usar

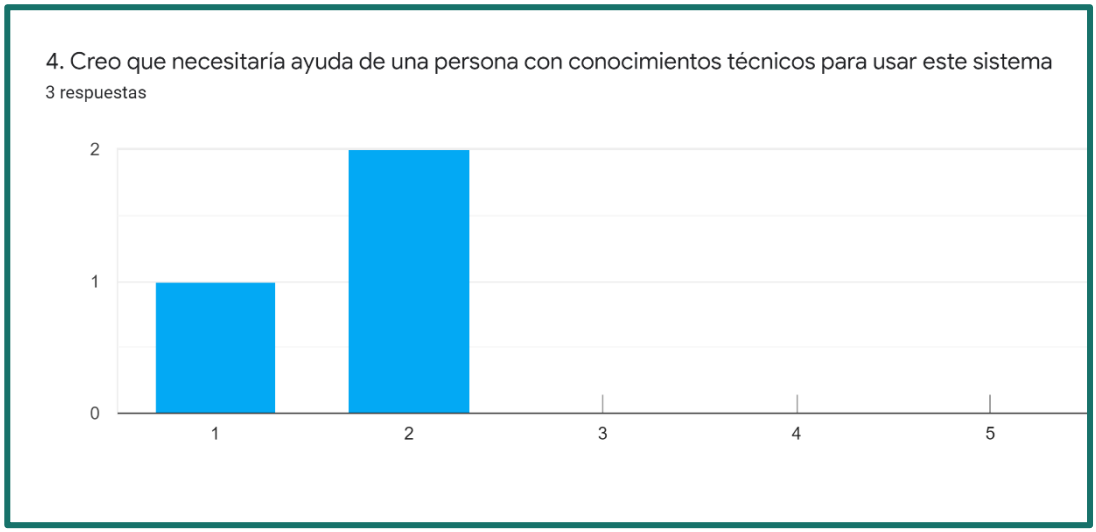


Figura 27 - Creo que necesitaría ayuda de una persona con conocimientos técnicos para usar este sistema

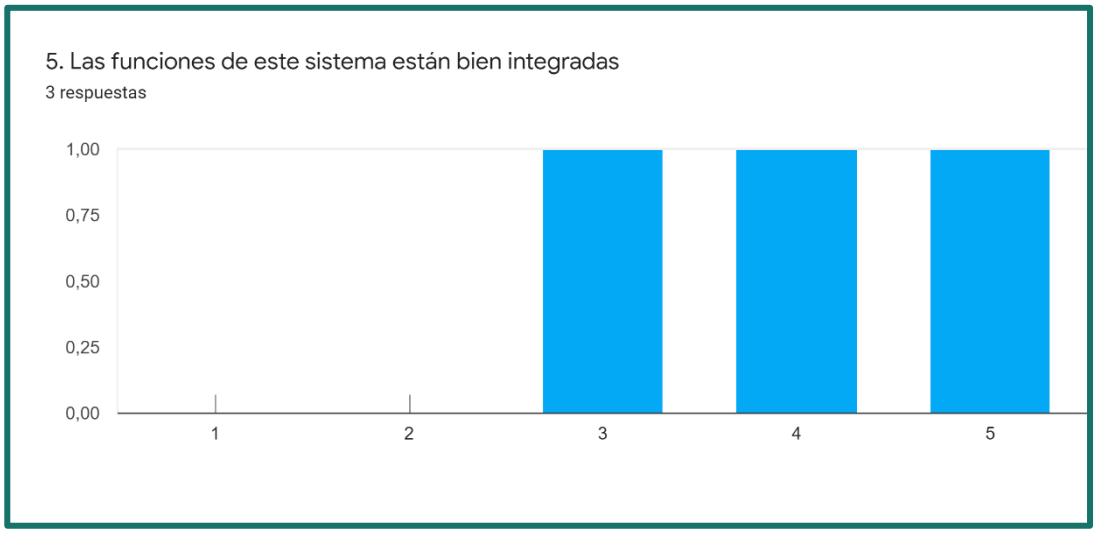


Figura 28 - Las funciones de este sistema están bien integradas

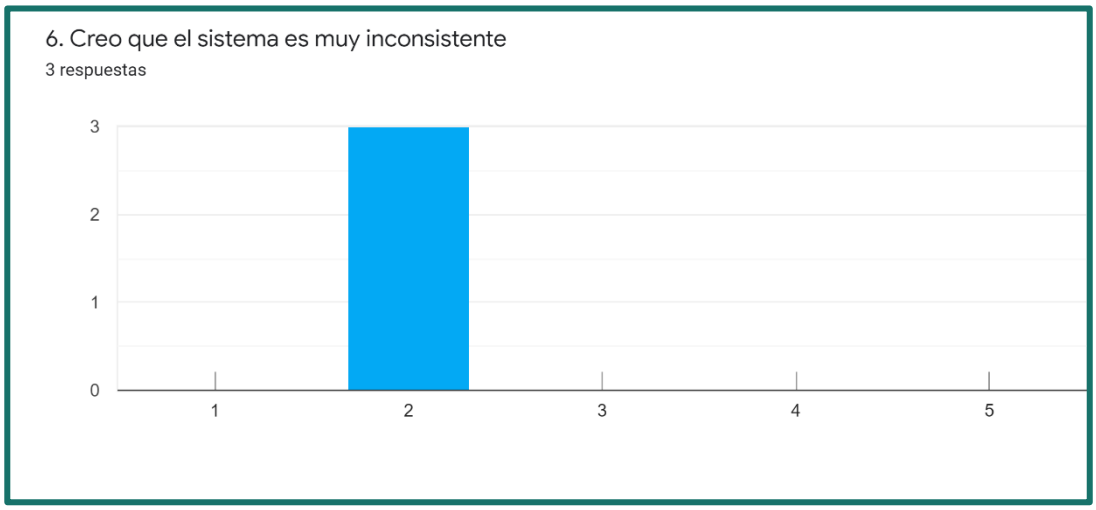


Figura 29 - Creo que el sistema es muy inconsistente

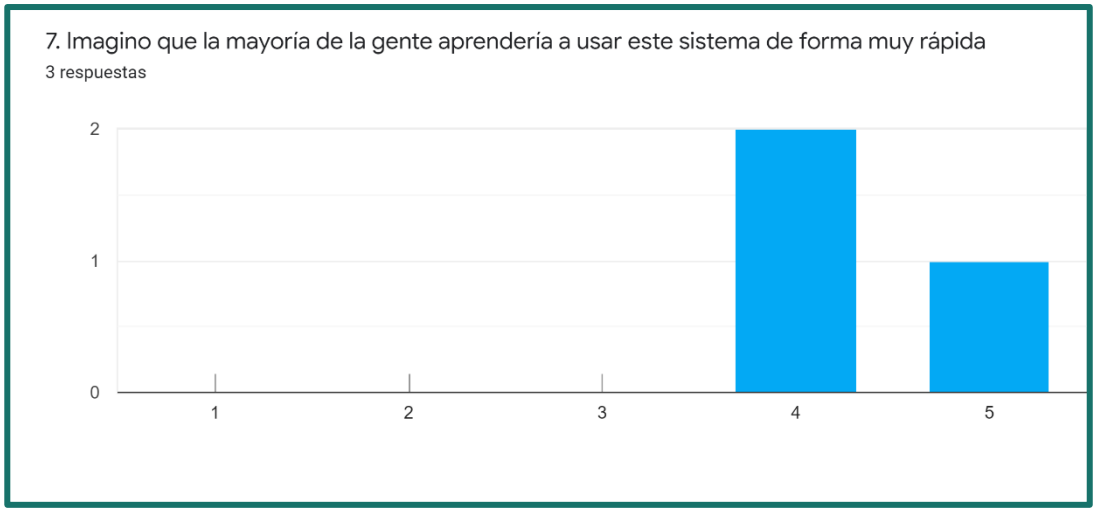


Figura 30 - Imagino que la mayoría de la gente aprendería a usar este sistema de forma muy rápida

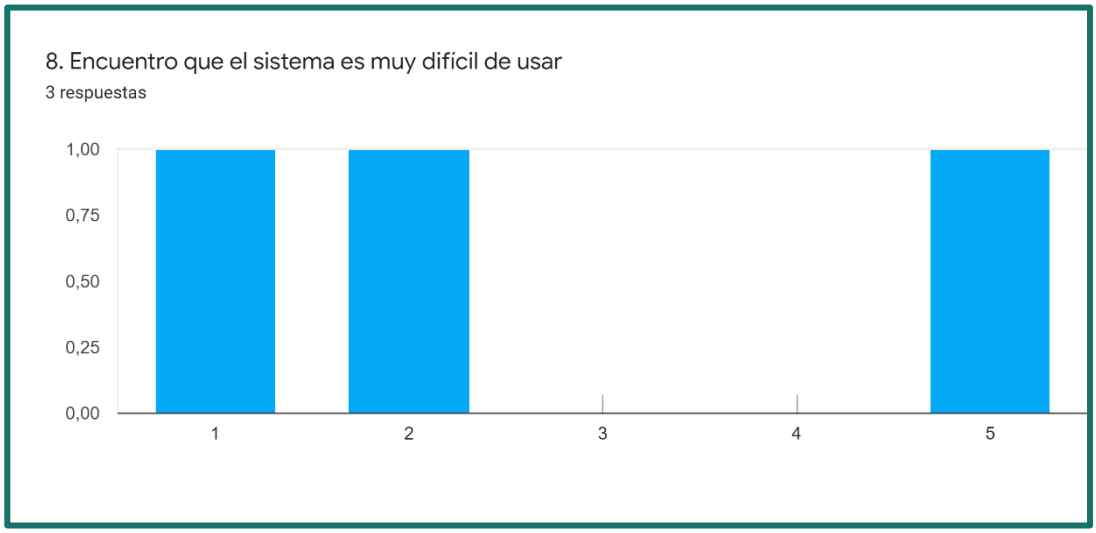


Figura 31 - Encuentro que el sistema es muy difícil de usar

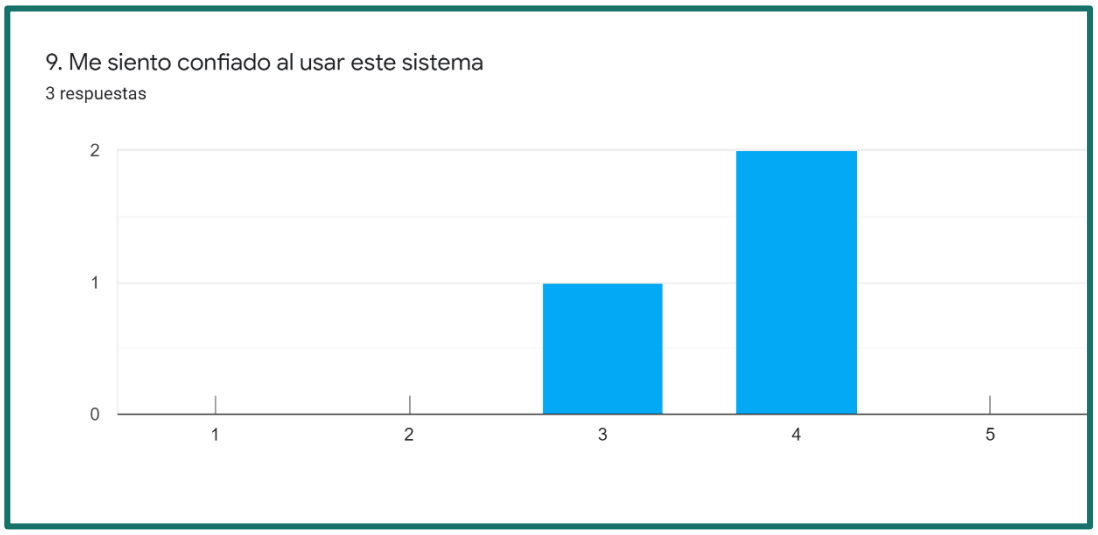


Figura 32 - Me siento confiado al usar este sistema

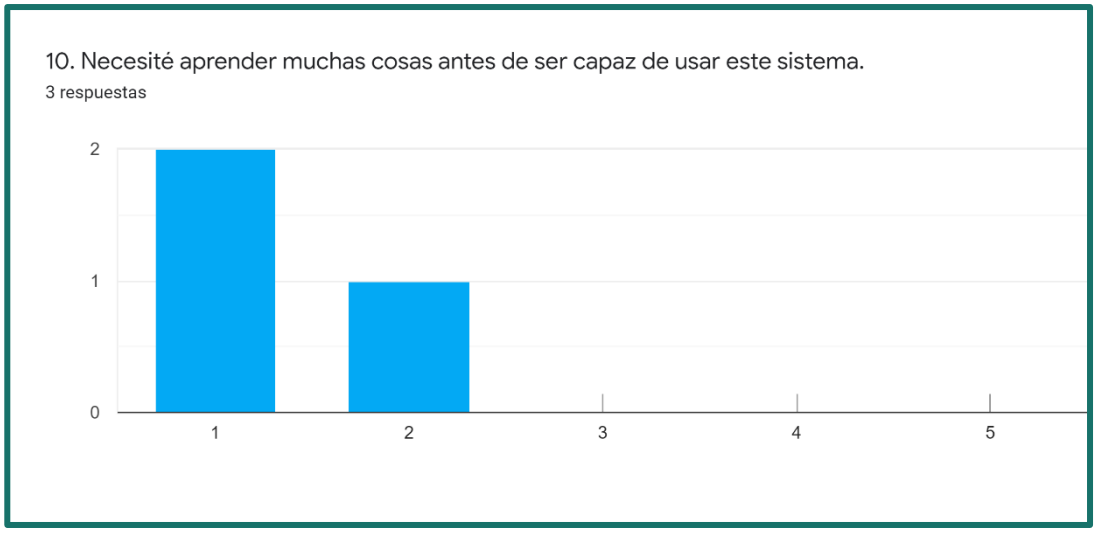


Figura 33 - Necesité aprender muchas cosas antes de ser capaz de usar este sistema

Una vez obtenidos los resultados del test SUS, para obtener el resultado final y saber su usabilidad se realizará la siguiente fórmula. Se sumarán los resultados promediados de manera que las preguntas impares se tomará el valor asignado y se le restará uno y las preguntas pares se restará a cinco el valor asignado por el usuario. Una vez obtenida la suma de los resultados, se multiplicará por 2,5.

Los resultados promediados en orden de preguntas han sido los siguientes: 5, 2, 4, 2, 4, 2, 4, 2, 4 y 1. Por lo tanto, la fórmula que resulta es la siguiente:

$$\big((5-1)+(5-2)+(4-1)+(5-2)+(4-1)+(5-2)+(4-1)+(5-2)+(4-1)+(5-1)\big)*2.5=$$

$$(4+3+3+3+3+3+3+3+3+4)*2.5=32*2.5=80$$

La puntuación SUS obtenida ha sido de 80 y dado que el máximo teórico es de 100 puntos, significa que es un sistema de gran utilidad para la empresa. [15]

Además, los resultados del segundo test han sido los siguientes.

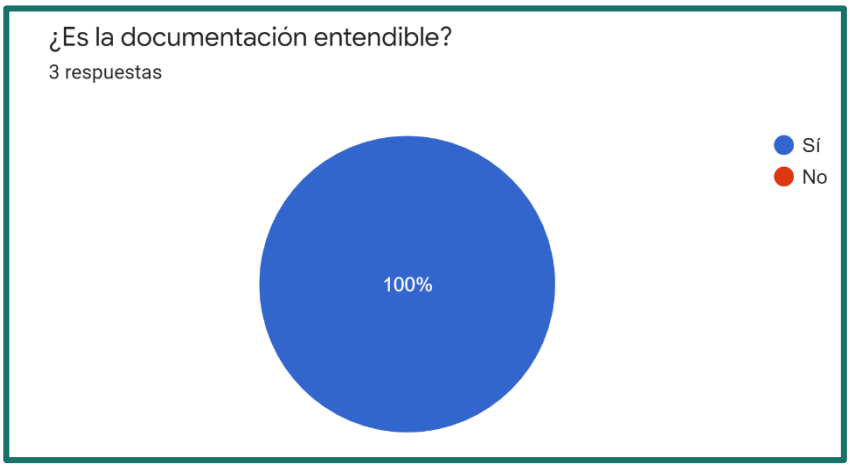


Figura 34 - ¿Es la documentación entendible?

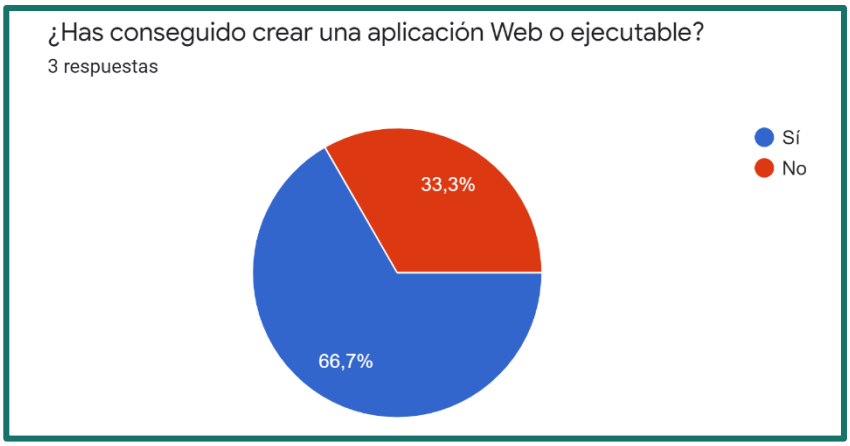


Figura 35 - ¿Has conseguido crear una aplicación Web o ejecutable?

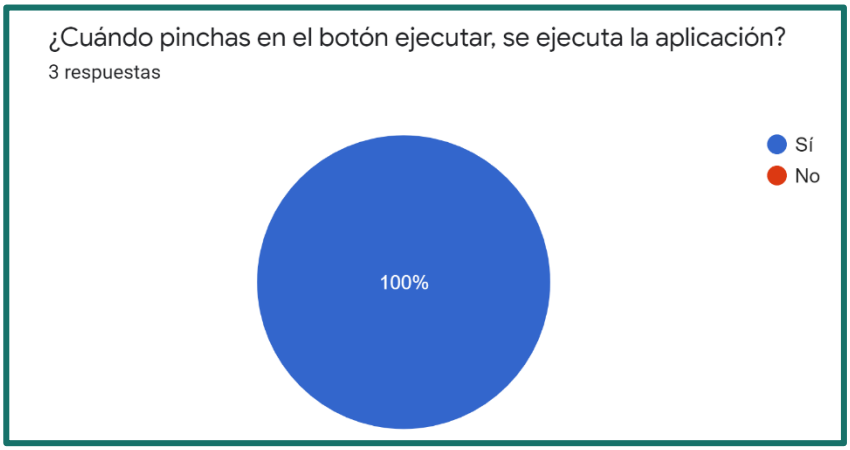


Figura 36 - ¿Cuándo pinchas en el botón ejecutar, se ejecuta la aplicación?

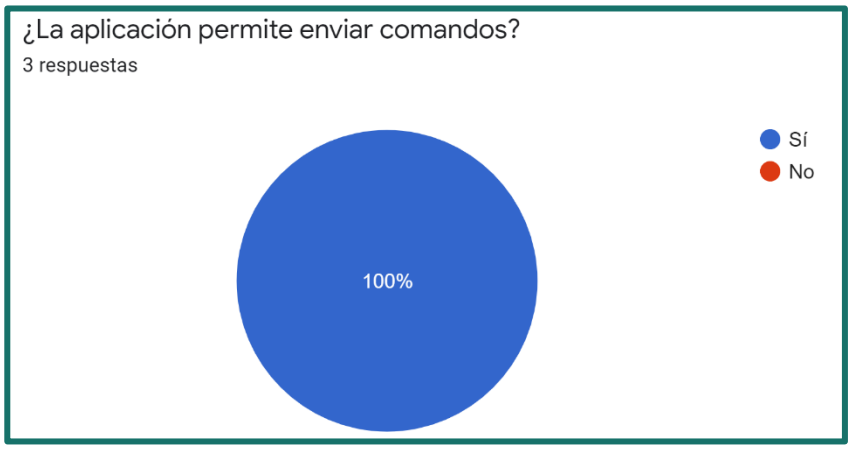


Figura 37 - ¿La aplicación permite enviar comandos?

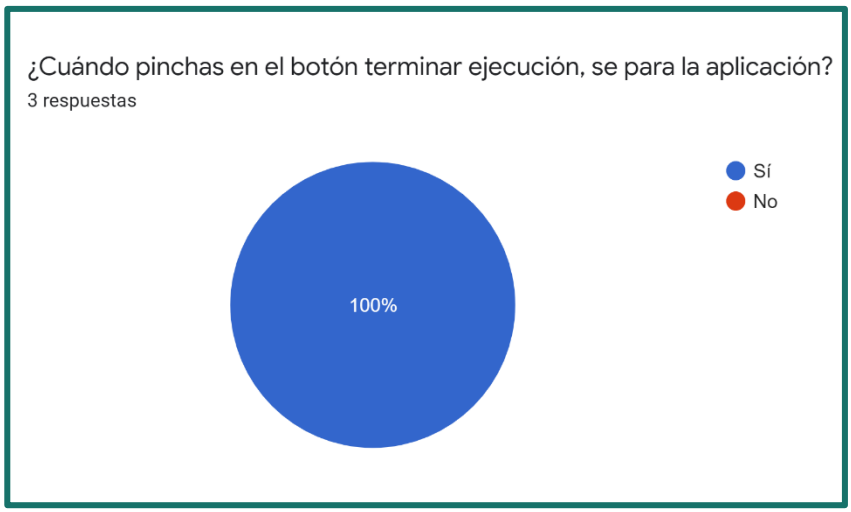


Figura 38 - ¿Cuándo pinchas en el botón terminar ejecución, se para la ejecución?

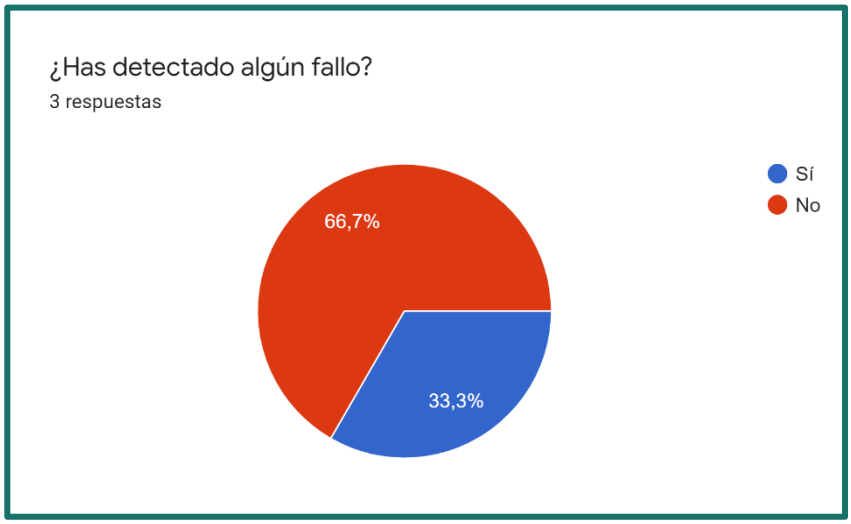


Figura 39 - ¿Has detectado algún fallo?

La persona que ha marcado que ha encontrado algún fallo, al explicar su fallo ha escrito lo siguiente:

- No se debe permitir mayúsculas, acentos u otros caracteres especiales en el nombre corto del servicio web. Esto puede dar problemas el crear el sitio en el IIS.
- Permite ejecutar la aplicación de consola sin haberla desplegado previamente, quedando esta bloqueada intentando lanzarla sin poder cancelarla.
- El manual indica que no se debe rellenar el nombre del sln pero para poder desplegarla, sí que es necesario. Si no se rellena, falla el despliegue.

En cuanto a las sugerencias de mejora, solo ha habido dos personas que han escrito alguna. Sus respuestas han sido:

- Se debe comprobar que no exista un servicio con ese nombre. ¿El servicio externo también se da de alta en la BD para poder utilizarlo con la web? Realizar tarea programada. Se puede subir un ejecutable y programar días y horas de ejecución.
- Crear los ficheros de cada entorno con las extensiones .dev, .pre y .pro hace que el editor deje de detectar el tipo de fichero que es y que se pierdan todas las ayudas del mismo. Quizás se pueda mantener la extensión del fichero y que el entorno sea un prefijo por ejemplo dev.oauth.config.

Aunque no se hayan hecho las pruebas de usuario a muchas personas, se puede observar que una persona ha encontrado fallos en el sistema y dos de ellas han mandado sugerencias de mejora. Por lo tanto, esto puede servir para mejorar el sistema y evitar que falle.

Además, en general se piensa que es un sistema necesario ya que se usaría frecuentemente y, piensan que la gente aprendería a usarlo rápidamente.

Por lo tanto, si se corrigiesen los fallos que no se habían encontrado al desarrollar el sistema y se realizase alguna de las sugerencias de mejora, es muy probable que los resultados del test SUS mejorarían notablemente.

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 CONCLUSIÓN

Finalmente, este trabajo ha permitido el desarrollo de una herramienta para la gestión de aplicaciones web y ejecutables en la plataforma GNOSS. Para ello, se han ido realizando diferentes objetivos.

En primer lugar, se ha realizado la interfaz para poder gestionar las aplicaciones web. Esta interfaz permite la creación, modificación y eliminación de estas aplicaciones. Después, se ha implementado la creación de las aplicaciones en el servidor IIS. A continuación, se ha realizado la compilación automática con Jenkins en la que se realizan peticiones a una tarea creada en Jenkins parametrizada. Después, se ha realizado el despliegue de las aplicaciones web y por último la creación y despliegue de aplicaciones ejecutables.

Además de todo ello, se han realizado pruebas de usuario para comprobar el correcto funcionamiento de la herramienta desarrollada.

Por otro lado, el proyecto realizado ha sido objetivamente complejo debido al conjunto variado de tecnologías que requiere conocer. Por lo tanto, realizarlo me ha ayudado a adquirir diversos conocimientos. Entre estos conocimientos se encuentran el lenguaje ASP.NET, el funcionamiento de Jenkins, IIS, el proceso de integración continua y las colas RabbitMQ entre otros.

Como se ha podido comprobar en las pruebas de usuario, es una herramienta de gran utilidad para la empresa ya que ha supuesto incluir dentro del flujo de Integración Continua el último componente que faltaba para dar autonomía total a los desarrolladores GNOSS en la gestión y despliegue de soluciones GNOSS.

Por último, estoy muy satisfecha con el trabajo realizado ya que he tardado menos de lo planificado inicialmente y se ha podido ampliar el proyecto.

5.2 LÍNEAS FUTURAS

La herramienta desarrollada es ampliable y por lo tanto se puede modificar para añadir nuevas funcionalidades. Estas nuevas funcionalidades no se han implementado debido al tiempo limitado del proyecto. A continuación, las explico.

En primer lugar, para la gestión de las aplicaciones ejecutables, se podría implementar la programación de la instalación de las ejecuciones. Así, poder elegir la hora y el día en el que se desea ejecutar una aplicación concreta.

Además, corregir los errores recogidos por los test de usuario además de realizar las sugerencias de mejora.

Por otro lado, como se ha explicado en el apartado del alcance del proyecto, utilizar Docker sería una línea futura cuando se realice la migración a .NET Core.

Además, se podría poner un directorio a disposición de las aplicaciones ejecutables que esté disponible a través de internet para poder publicar archivos. Así, se podría poner una casilla de verificación en la interfaz para poder disponer de un directorio virtual en IIS y poder acceder a él a través de internet.

También, ampliar el servicio FTP de GNOSS para permitir el acceso vía FTP al sistema de archivos de las aplicaciones desplegadas.

Por último, se podría ofrecer un sistema de logging unificado para las aplicaciones, tipo ELK (Elasticsearch, Logstash y Kibana) [14].

6 BIBLIOGRAFÍA

[1] "Servidor IIS: ¿Qué es? Características, Ventajas, Requisitos e Instalación", InfranetWorking, 2019. [En línea]. Disponible en: https://blog.infranetworking.com/servidor-iis/#Que_es_IIS . [Accedido: 06-abril-2021]

[2] "Seguridad: Identidades de grupos de aplicaciones en IIS 7.5 y IIS 8.0", JASoft.org, 2015. [En línea]. Disponible en: https://www.jasoft.org/Blog/post/Seguridad-Identidades-de-grupos-de-aplicaciones-en-IIS-75-y-IIS-80#:~:text=Seguridad%3A%20Identidades%20de%20grupos%20de%20aplicaciones%20en%20IIS%207.5%20y%20IIS%208.0,-por%20Jos%C3%A9%20M&text=Un%20Application%20Pool%20es%20una,sus%20propios%20l%C3%ADmites%20de%20seguridad [Accedido: 06-abril-2021]

[3] "Jenkins", Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Jenkins [Accedido: 06-abril-2021]

[4] "NuGet Gallery", Nuget. [Online]. Available: https://www.nuget.org/ [Accessed: 06-april-2021]

[5] "RabbitMQ", bigeek, 2018. [En línea]. Disponible en: https://blog.bi-geek.com/rabbitmq-para-principiantes/ [Accedido: 06-abril-2021]

[6] "Información general de Entity Framework", Microsoft, 2018. [En línea]. Disponible en: https://docs.microsoft.com/es-es/dotnet/framework/data/adonet/ef/overview [Accedido: 15-abril-2021]

[7] "El backlog del producto: la lista de tareas pendientes definitiva", Atlassian. [En línea]. Disponible en: https://www.atlassian.com/es/agile/scrum/backlogs [Accedido: 17-abril-2021]

[8] "System Usability Scale (SUS)", usability.gov. [Online]. Available: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html [Accessed: 05-mayo-2021]

[9] "Why You Only Need to Test with 5 Users", Jakob Nielsen, 2000. [Online]. Available: https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users [Accessed: 13/05/2021]

[10] "Web Deploy 3.0", Plesk. [Online]. Available: https://docs.plesk.com/en-US/12.5/deployment-guide/appendix-a-installing-thirdparty-services/web-deploy-30.69594/ [Accessed: 14/05/2021]

[11] "Elección entre ASP.NET 4.x y ASP.NET Core", Rick Anderson y olprod, 2020. [En línea]. Disponible en: https://docs.microsoft.com/es-es/aspnet/core/fundamentals/choose-aspnet-framework?view=aspnetcore-5.0 [Accedido: 18/05/2021]

[12] "¿Qué es Docker?", Red Hat. [En línea]. Disponible en: https://www.redhat.com/es/topics/containers/what-is-docker [Accedido: 18/05/2021]

[13] "Yagni", Wikipedia. [En línea]. Disponible en: https://es.wikipedia.org/wiki/YAGNI [Accedido: 21/05/2021]

[14] "¿Qué es ELK Stack?", Elastic Search. [En línea]. Disponible en: https://www.elastic.co/es/what-is/elk-stack [Accedido: 27/05/2021]

[15] "Sistema de Escalas de Usabilidad: ¿qué es y para qué sirve?", UXpañol. [En línea]. Disponible en: Sistema de Escalas de Usabilidad: ¿qué es y para qué sirve? | | UXpañol (uxpanol.com) [Accedido: 03/06/2021]

7 ANEXO 1. DOCUMENTACIÓN

INTRODUCCIÓN

La plataforma GNOSS es extensible mediante aplicaciones específicas por cada proyecto, con funciones que se integran a través del API de GNOSS. Se ha desarrollado una herramienta de administración que permite a los desarrolladores la gestión de estas aplicaciones. Esta herramienta permitirá crear las aplicaciones web o ejecutables y su compilación automática. Además, se podrá ejecutar las aplicaciones ejecutables.

ACCESO

La administración de las aplicaciones específicas está en la url "/administrar-aplicaciones-especificas" de cada comunidad. En la interfaz de administración, aparecerá una pantalla similar a la siguiente.

Figura 40

Observamos que aparecen dos listados. El primero es el de aplicaciones web y el segundo el de aplicaciones ejecutables. Estas aplicaciones son las existentes en el proyecto correspondiente. Podremos añadir nuevas aplicaciones o borrar las existentes.

APLICACIONES WEB

Cuando se quiera añadir una nueva aplicación web específica se pinchará en el botón “Añadir aplicación web”. Entonces se desplegará un formulario que solicitará los datos de la aplicación.

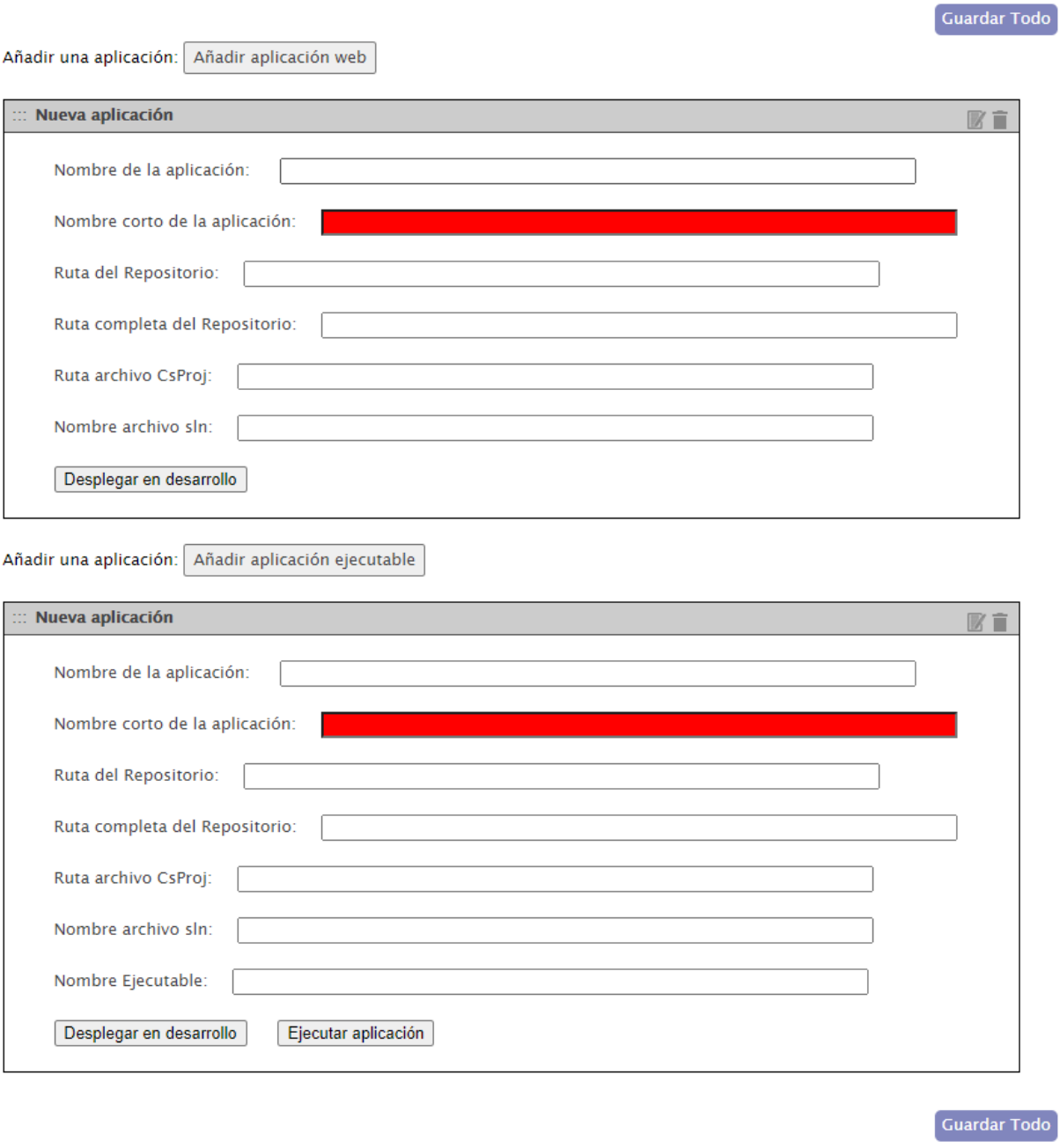


Figura 41

Aparecerá en el campo Ruta completa del Repositorio la URL de Git donde se deberá subir el código del nuevo servicio. Una vez subido el código, habrá que rellenar los campos siguientes campos:

- Nombre de la aplicación: nombre que se le quiera dar a la aplicación. Se podrá modificar cuando se desee. Se puede dejar vacío.
- Nombre corto de la aplicación: nombre asignado a la aplicación. Es único y, por lo tanto, no se puede modificar. Este campo no se puede dejar vacío.

- Ruta del repositorio: ruta en la que se encuentra el archivo .sln del servicio en Git. El campo tendrá el valor App/"nombrecorto" por defecto. Sin embargo, se puede modificar.
- Ruta completa del Repositorio: aparecerá al principio una URL donde se debe subir el código y se completará con la ruta del repositorio. No se puede modificar.
- Ruta archivo CsProj: es la ruta en la que está el archivo .csproj generado por Visual Studio. No hay que rellenarlo.
- Nombre archivo sln: nombre del archivo .sln generado por Visual Studio. No hay que rellenarlo.

Un ejemplo sería el siguiente.

Si queremos crear un servicio llamado GameService, rellenaremos los campos de la siguiente manera:

- Nombre: Game Service
- NombreCorto: GameService
- Ruta del Repositorio: GameService

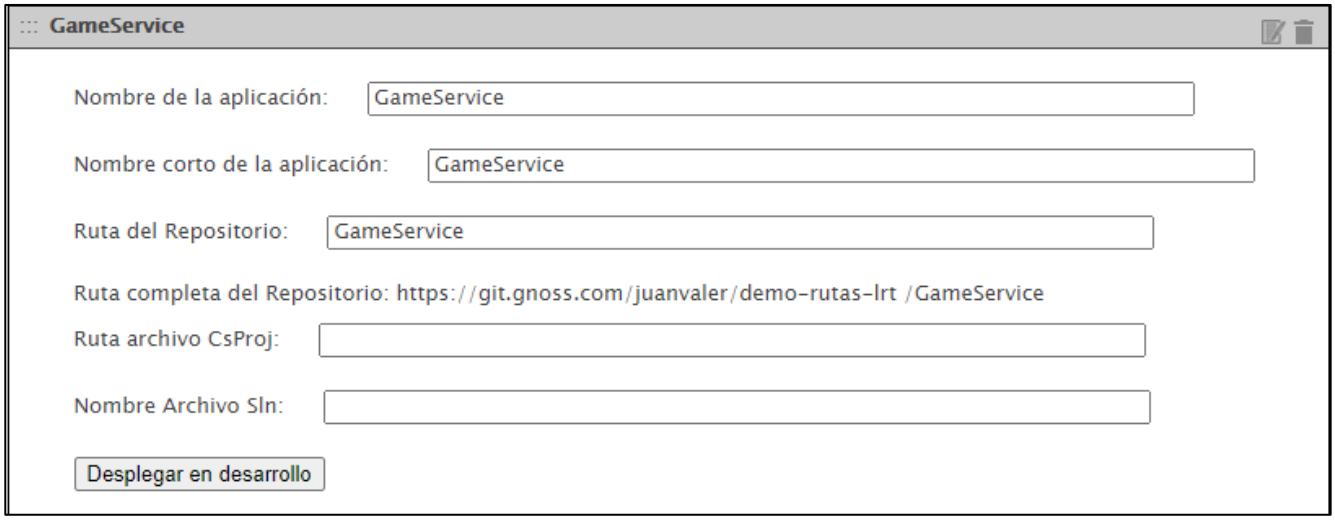


Figura 42

Una vez rellenados los campos, se pinchará en el botón guardar todo y se subirá el código a Git. El código deberá subirse en la URL proporcionada en Ruta completa del Repositorio.

Además, si existe algún fichero que cambia según el entorno se deberá subir al repositorio de Git. Según el entorno al que pertenezca, el fichero acabará en .dev (desarrollo), .pre (preproducción) y pro (producción). Se deberá subir también un perfil de publicación (Nexus.pubmxl) que diga que la URL de publicación debe ser C://Publicacion. Un ejemplo sería el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
<!--
Este archivo es utilizado por el proceso de paquete/publicación de nuestro proyecto Web. Puede personalizar el comportamiento de este proceso
editando este archivo MSBuild. Para conocer más acerca de esto, visite http://go.microsoft.com/fwlink/?LinkID=208121.
-->
<Project ToolsVersion="4.0" xmlns="http://schemas.microsoft.com/developer/msbuild/2003">
```

```
  <PropertyGroup>
    <WebPublishMethod>FileSystem</WebPublishMethod>
    <LastUsedBuildConfiguration>Release</LastUsedBuildConfiguration>
    <LastUsedPlatform>Any CPU</LastUsedPlatform>
    <SiteUrlToLaunchAfterPublish />
    <LaunchSiteAfterPublish>True</LaunchSiteAfterPublish>
    <ExcludeApp_Data>False</ExcludeApp_Data>
    <publishUrl>C:\Publicacion</publishUrl>
    <DeleteExistingFiles>True</DeleteExistingFiles>
  </PropertyGroup>
</Project>
```

Una vez realizado todo lo anterior, se habrá creado la aplicación en la base de datos y se habrá instalado en IIS.

Por otro lado, se puede desplegar la aplicación en desarrollo. Para realizarlo, se tiene que haber hecho antes todo lo anterior. Después, se podrá pinchar en el botón “Desplegar en desarrollo”. Una vez realizado esto, ya estará la aplicación instalada y desplegada.

APLICACIONES EJECUTABLES

Para la creación de una aplicación ejecutable, se debe realizar todo lo descrito en el apartado anterior. La única diferencia es que en el formulario se pedirá rellenar el campo nombre del ejecutable. En este campo se escribirá el nombre del archivo .exe.

Además, si se quiere ejecutar la aplicación, se deberá pinchar en el botón ejecutar aplicación. Entonces, aparecerá un cuadro de texto que mostrará la entrada y la salida estándar de la consola. También, se mostrará un cuadro de texto para mandar comandos y un botón para terminar la ejecución.

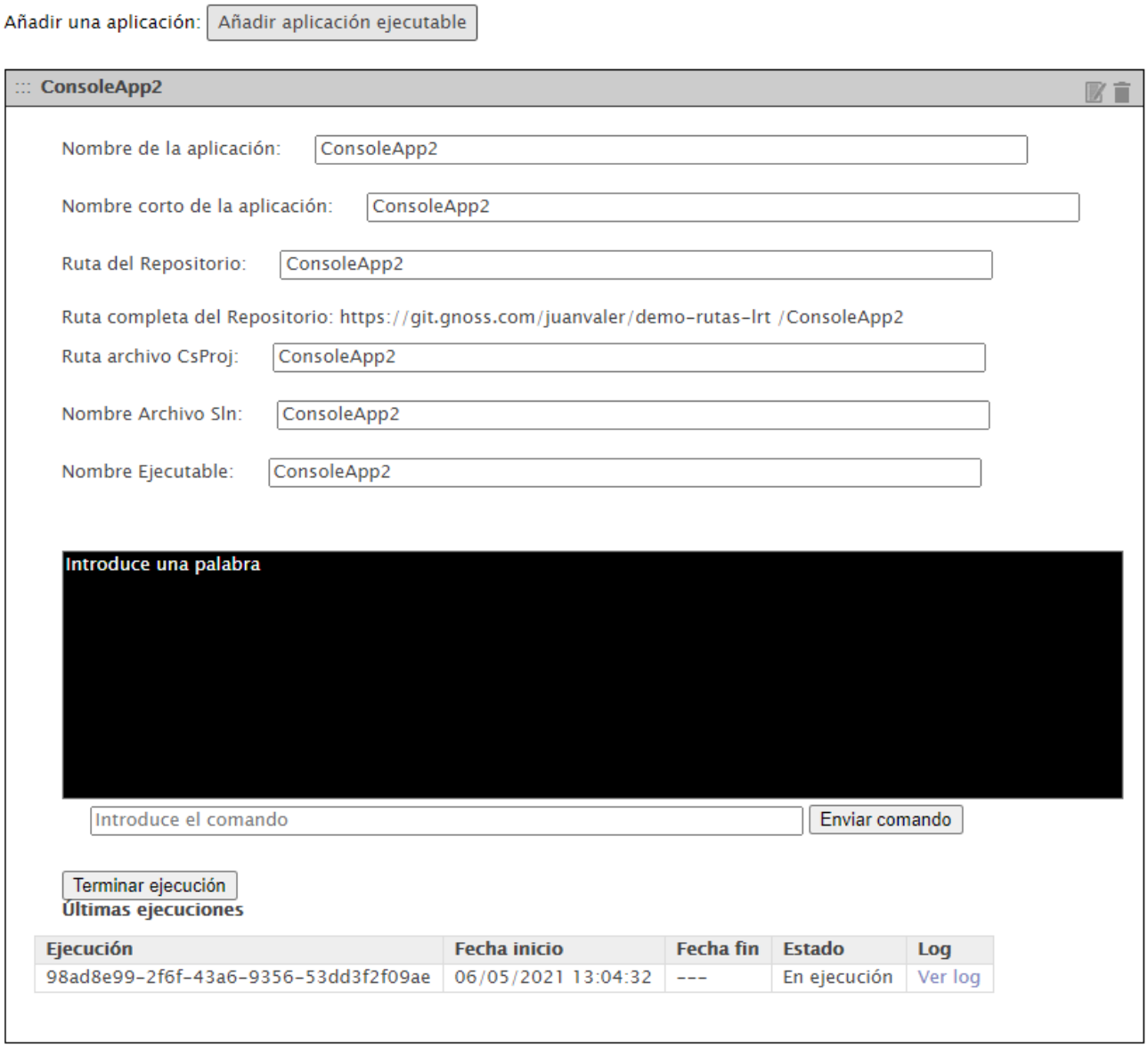
Por último, se podrá ver un listado con las últimas ejecuciones. En él, se mostrará la fecha de inicio y de fin de la ejecución, el estado en el que está y un enlace para poder ver el log.

A continuación, se muestra un ejemplo.

Figura 43

Cuando se pinche en ejecutar aplicación, se añadirá una nueva ejecución en estado “espera” y aparecerán los cuadros de texto para enviar comandos y para mostrar la salida y la entrada estándar de la consola. Cuando comience la ejecución, cambiará el estado a “en ejecución”.

Para realizar una prueba, puede crear una aplicación de consola que pueda recibir cadenas de caracteres y una vez leídos, pueda escribirlos.


Añadir una aplicación:
Añadir aplicación ejecutable
ConsoleApp2
Nombre de la aplicación:
ConsoleApp2
Nombre corto de la aplicación:
ConsoleApp2
Ruta del Repositorio:
ConsoleApp2
Ruta completa del Repositorio: https://git.gnoss.com/juanvaler/demo-rutas-lrt /ConsoleApp2
Ruta archivo CsProj:
ConsoleApp2
Nombre Archivo Sln:
ConsoleApp2
Nombre Ejecutable:
ConsoleApp2
Introduce una palabra
Introduce el comando
Enviar comando
Terminar ejecución
Últimas ejecuciones
Ejecución
Fecha inicio
Fecha fin
Estado
Log
98ad8e99-2f6f-43a6-9356-53dd3f2f09ae
06/05/2021 13:04:32

En ejecución
Ver log

Figura 44