Universidad de Castilla-La Mancha
Escuela Superior de Ingeniería Informática

**Trabajo Fin de Grado**
Grado en Ingeniería Informática
Tecnologías de la Información

# Programas en R ejecutables desde formularios Web

*Alonso Marcos Muñoz*

Junio, 2025

**TRABAJO FIN DE GRADO**

Grado en Ingeniería Informática

Tecnologías de la Información

# Programas en R ejecutables desde formularios Web


**Autor:** D. Alonso Marcos Muñoz

**Tutor:** D. Jesús Damián García-Consuegra Bleda

**Co-Tutor:** D. Luis Clemente Villaescusa


Junio, 2025

*Aquí va la dedicatoria que cada cual quiera escribir. El ancho se controla manualmente*

# Declaración de autoría

Yo, Alonso Marcos Muñoz, con DNI 06281718G, declaro que soy el único autor del trabajo fin de grado titulado “ … ”, que el citado trabajo no infringe las leyes en vigor sobre propiedad intelectual, y que todo el material no original contenido en dicho trabajo está apropiadamente atribuido a sus legítimos autores.

Albacete, a … de … de 20 …

Fdo.: Alonso Marcos Muñoz

# Resumen

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

# Agradecimientos

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

# Índice general

# Índice de figuras

# Índice de tablas

# Índice de algoritmos

# Índice de listados de código

# 1. Introducción

## 1.1. Introducción

En la era de la transformación digital, la ejecución de código en entornos web ha cobrado una gran relevancia, permitiendo a los usuarios realizar cálculos complejos y análisis de datos sin necesidad de instalar software en sus equipos locales. R, como lenguaje de programación especializado en estadística y ciencia de datos [Team, 2023], ha sido ampliamente adoptado en la comunidad científica, académica y empresarial debido a su flexibilidad y capacidad para la manipulación y visualización de datos. Sin embargo, la ejecución de scripts en R tradicionalmente se ha realizado en entornos de escritorio o servidores locales, lo que limita su accesibilidad y escalabilidad en aplicaciones web.

Para abordar este desafío, este Trabajo Fin de Grado (TFG) propone el desarrollo de un sistema web para la ejecución remota de programas en R, facilitando a los usuarios la interacción con scripts a través de una interfaz intuitiva y accesible. La solución se basará en la integración de RStudio Server, un entorno de ejecución remoto para R, con una API REST implementada en Plumber, que permitirá la comunicación entre el backend y el frontend, desarrollado en Angular.[Angular, 2024b]

Este enfoque permitirá a los usuarios seleccionar y ejecutar programas desde un formulario web, visualizar los resultados en diferentes formatos (numéricos y gráficos) y gestionar la ejecución de scripts sin necesidad de configurar un entorno local. Además, se han considerado aspectos clave como la seguridad en la ejecución de código y la modularidad del sistema para garantizar una arquitectura escalable y fácil de mantener.

A lo largo del desarrollo del proyecto, se emplearán metodologías de gestión de proyectos como PMI [Institute, 2021] y Kanban, asegurando una planificación estructurada y eficiente. Se llevarán a cabo pruebas de validación del sistema para evaluar su rendimiento, estabilidad y usabilidad, garantizando que cumpla con los objetivos planteados.

## 1.2. Objetivos

El objetivo principal de este Trabajo de Fin de Grado (TFG) es desarrollar una aplicación web que permita la ejecución remota de programas en R a través de un formulario web, proporcionando a los usuarios una interfaz intuitiva para la interacción con los programas y la visualización de los resultados generados. Para ello, se integrará RStudio Server como entorno de ejecución y Plumber para la creación de una API REST, que facilitará la comunicación entre el frontend en Angular y el backend en R.

Además de la ejecución de programas, la aplicación ofrecerá funcionalidades para la gestión de los programas R disponibles en el sistema, permitiendo a los usuarios añadir, modificar, eliminar y visualizar los programas existentes. Esta funcionalidad permitirá una mayor flexibilidad en la gestión de los scripts de R, facilitando su reutilización y adaptación a diferentes necesidades.

Para lograr estos objetivos generales, se han definido los siguientes objetivos específicos:

1. Desarrollar la API REST en R con Plumber, permitiendo la comunicación entre el frontend y RStudio Server para la ejecución remota de programas en R.
2. Implementar la interfaz web en Angular, proporcionando un formulario intuitivo para que los usuarios puedan seleccionar, parametrizar y ejecutar scripts en R.
3. Visualizar los resultados de los programas en R en diferentes formatos, incluyendo resultados numéricos y gráficos, para mejorar la interpretación de los datos procesados.
4. Facilitar la gestión de programas R en el sistema, permitiendo a los usuarios administrar los programas disponibles mediante una interfaz sencilla y organizada.
5. Definir un sistema de metadatos estructurado que facilite la parametrización de los programas en R, permitiendo su reutilización y ejecución con valores por defecto.

## 1.3. Estructura del proyecto

El proyecto se estructura en los siguientes capítulos:

- **Capítulo 1: Introducción** Se expone el contexto del proyecto, sus objetivos y la estructura general del documento.
- **Capítulo 2: Hipótesis de trabajo** Se presenta un resumen de las hipótesis de trabajo y los requisitos iniciales que fundamentan el desarrollo del sistema.
- **Capítulo 3: Estado del arte** Se revisan las tecnologías utilizadas en el desarrollo del sistema, incluyendo RStudio Server, Plumber para la creación de APIs en R y Angular para la interfaz web, así como un análisis de enfoques similares y su impacto en la ejecución de código en entornos web.
- **Capítulo 4: Metodología y diseño del sistema** Se detalla la metodología utilizada para la gestión del proyecto, basándose en estándares como PMI y Kanban, y se documenta el proceso de planificación y diseño de la solución.

- **Capítulo 5: Experimentos, validación y resultados** Se describen en detalle las actividades realizadas durante el desarrollo del proyecto. Se incluirán pantallazos de código, formularios y otros elementos gráficos que evidencian los experimentos, pruebas y la validación del sistema, así como la presentación final de los requisitos funcionales y no funcionales.
- **Capítulo 6: Conclusiones y trabajo futuro** Se presentan las conclusiones obtenidas del desarrollo del TFG y se proponen posibles mejoras y futuras líneas de investigación.
- **Bibliografía** Se recopilan las referencias bibliográficas utilizadas durante la investigación y desarrollo del proyecto.
- **Anexos** Se incluyen documentos complementarios, como la documentación de la API REST, el manual de usuario y ejemplos de código relevantes.

# 2. Hipótesis de trabajo

## 2.1. Introducción

En este capítulo se establecen las bases teóricas y operativas que sustentan el desarrollo del sistema. Se definen las hipótesis que orientan la solución propuesta, así como los requisitos funcionales y no funcionales que aseguran que el sistema cumpla con los objetivos planteados. La integración de tecnologías como R y Angular, junto con la ejecución de programas a través de RStudio Server, constituye el eje central para optimizar la gestión, ejecución y consulta de programas. Este enfoque permite abordar de manera integral aspectos críticos como la eficiencia, la usabilidad y la seguridad del sistema.

## 2.2. Hipótesis de trabajo

La hipótesis principal que guía este proyecto es que la integración de un sistema web basado en Angular para la gestión y ejecución de programas en R, junto con la comunicación directa con RStudio Server, mejorará significativamente la eficiencia en la administración y análisis de procesos. Con ello se asume que:

- La utilización de RStudio Server para ejecutar scripts en R mediante una API REST garantizará un procesamiento rápido y seguro de los datos.
- El desarrollo de la interfaz con Angular permitirá una experiencia de usuario intuitiva y modular, facilitando la incorporación de nuevas funcionalidades en el futuro.
- La centralización de la gestión de programas y sus metadatos permitirá al administrador mantener un control riguroso sobre los recursos disponibles, mientras que los usuarios podrán ejecutar y consultar información de manera ágil.
- La implementación de mecanismos de autenticación y autorización asegurará que solo los usuarios autorizados accedan a funcionalidades críticas, protegiendo la integridad y confidencialidad del sistema.

## 2.3. Contextualización del sistema

En el contexto actual de la transformación digital, la ejecución de código y el análisis de datos han dejado de estar limitados a entornos locales para trasladarse a plataformas web y basadas en la nube. Esta tendencia responde a la creciente necesidad de colaboración en el manejo de información, lo que ha impulsado el desarrollo de soluciones que permitan a los usuarios ejecutar procesos complejos sin depender de configuraciones locales específicas.

El sistema propuesto se enmarca dentro de este paradigma, ya que pretende facilitar la ejecución remota de programas en R. Tradicionalmente, la ejecución de scripts en R se realizaba en entornos de escritorio o en servidores locales, limitando el acceso y la capacidad de colaboración entre distintos usuarios. Con la integración de RStudio Server como IDE, incluso en su versión Community, se abre la posibilidad de gestionar y ejecutar scripts de forma remota mediante una interfaz web, permitiendo a los usuarios interactuar con el código y visualizar resultados sin necesidad de contar con instalaciones locales de R.

Además, el uso de una API REST, desarrollada con Plumber, posibilita la comunicación entre el backend y el frontend, lo que integra el procesamiento de datos en R con una interfaz de usuario dinámica desarrollada en Angular. Esta combinación de tecnologías no solo mejora la eficiencia en la ejecución de scripts, sino que también amplía el alcance del análisis de datos al permitir la integración de servicios y la centralización de recursos en un entorno web.

Por tanto, la contextualización del sistema se fundamenta en:

- **Accesibilidad y escalabilidad:** El sistema se orienta a proporcionar acceso remoto a herramientas de análisis de datos, eliminando la necesidad de configuraciones locales y facilitando el uso colaborativo.
- **Integración de tecnologías modernas:** La utilización de RStudio Server, Plumber y Angular permite combinar el poder analítico de R con la flexibilidad y usabilidad de las aplicaciones web.
- **Respuesta a las tendencias de digitalización:** La creciente demanda de soluciones basadas en la nube y la ejecución remota de procesos analíticos es el motor que impulsa el desarrollo de este sistema, en línea con las tendencias actuales en el ámbito tecnológico y científico.

Este enfoque sitúa al sistema propuesto como una respuesta innovadora a los desafíos contemporáneos del análisis de datos, permitiendo una mayor eficiencia, colaboración y adaptabilidad en un entorno digital en constante evolución.

# 3. Estado del arte

## 3.1. Introducción

En este capítulo, se ofrece una revisión detallada del estado actual de las tecnologías y metodologías relacionadas con el desarrollo de sistemas orientados a la ejecución de scripts y aplicaciones interactivas. Se abordan, entre otros, los siguientes ámbitos:

- **Lenguaje R y su ecosistema:** Utilizado para el análisis de datos y la estadística, fundamental en aplicaciones científicas e industriales.
- **Desarrollo y consumo de APIs:** Permite la integración de servicios y la automatización de procesos analíticos.
- **Frameworks para el desarrollo frontend:** Especialmente, Angular, para la creación de interfaces de usuario dinámicas y escalables.

El análisis se basa en la documentación y prácticas recomendadas por referentes en cada área. Por ejemplo:

- La robustez y versatilidad de R para el procesamiento de datos ha sido comprobada por su uso extendido en la comunidad científica y en aplicaciones industriales [Team, 2023].
- Angular se destaca por su capacidad para construir interfaces dinámicas que responden a las necesidades actuales del mercado [Angular, 2024b].

Además, la adopción de buenas prácticas en la planificación y gestión de proyectos, basadas en estándares internacionales como el PMBOK Guide [Institute, 2021], contribuye a estructurar eficazmente los procesos de diseño, desarrollo e integración de las tecnologías involucradas.

El objetivo de esta sección es proporcionar una visión general del estado del arte en las áreas clave relacionadas con el proyecto.

## 3.2. Integración de RStudio Server a través de Posit Workbench y WSL2

La creciente integración de entornos Linux en sistemas Windows ha revolucionado la manera en que se desarrollan y despliegan aplicaciones de análisis de datos. El Subsistema de Windows para Linux (WSL), especialmente en su versión 2, permite a los usuarios de Windows ejecutar distribuciones Linux de forma nativa, ofreciendo un entorno robusto y eficiente para la ejecución de aplicaciones basadas en Linux [Canonical and Microsoft, 2022]. Entre las distribuciones disponibles, Ubuntu 20.04 destaca por su estabilidad y soporte extendido, siendo ampliamente adoptada en entornos de desarrollo profesional y académico.

En este contexto, Posit Workbench se erige como una herramienta esencial para la instalación y configuración de RStudio Server. RStudio Server permite a los usuarios acceder a un entorno de desarrollo integrado para R a través de un navegador web, centralizando la ejecución de scripts y facilitando la colaboración en proyectos de análisis de datos. Aunque en este TFG se utiliza la edición Community, la robustez de RStudio Server y su capacidad para gestionar sesiones de R de forma remota lo convierten en una plataforma idónea para integrar herramientas analíticas en un entorno centralizado [RStudio, 2024a].

La combinación de WSL y Posit Workbench permite a los desarrolladores trabajar en un entorno Linux sin abandonar Windows, aprovechando las mejoras en rendimiento y compatibilidad que ofrece WSL2. Esto resulta particularmente ventajoso para proyectos que requieren un entorno de desarrollo reproducible y escalable. Además, la utilización de RStudio Server facilita la integración de funciones analíticas y la ejecución remota de scripts, elementos fundamentales para el desarrollo de aplicaciones web interactivas en R.

En resumen, la adopción de estas tecnologías en el TFG no solo optimiza el proceso de desarrollo, sino que también refleja una tendencia actual en el manejo de entornos híbridos, donde se combinan las ventajas de Linux y Windows para ofrecer soluciones flexibles y de alto rendimiento en el análisis de datos.

## 3.3. R y su ecosistema en la ejecución de scripts

El lenguaje R se ha consolidado como una herramienta esencial para el análisis de datos y la estadística, respaldada por un ecosistema en constante evolución. Entre sus principales ventajas destacan:

- **Ejecución de procesos:** Gracias a su amplia colección de paquetes y librerías, R permite ejecutar scripts de forma eficiente.
- **Plataforma abierta y flexible:** El R Project for Statistical Computing se integra fácilmente en diversos entornos, facilitando la ejecución remota de scripts y la generación de reportes reproducibles [Team, 2023].

Asimismo, el ecosistema de R incluye entornos de desarrollo integrados como RStudio, que simplifican el manejo del código. Herramientas como Shiny [RStudio, 2024b] han impulsado la creación de aplicaciones interactivas y dashboards, permitiendo la integración de R

con otros lenguajes y tecnologías a través de APIs y servicios web. La activa comunidad y la abundante documentación especializada aseguran que R se mantenga a la vanguardia en el manejo de grandes volúmenes de datos.

### 3.3.1. Uso de R en entornos web

El uso de R en entornos web ha adquirido gran relevancia en los últimos años, impulsado por el desarrollo de herramientas que permiten transformar análisis de datos complejos y modelos estadísticos en aplicaciones interactivas. En este contexto, soluciones como **Shiny** y **R Markdown** han demostrado ser tendencias importantes, ya que facilitan la creación de dashboards interactivos y la generación de informes dinámicos, respectivamente.

- **Shiny** permite crear interfaces gráficas interactivas que combinan la potencia analítica de R con la flexibilidad de la web, posibilitando que los usuarios exploren los datos de forma visual y en tiempo real.
- **R Markdown** integra código, resultados y narrativa en un solo documento, lo que favorece la reproducibilidad y la colaboración en el análisis de datos.

No obstante, es importante destacar que, en el presente TFG, se ha optado por no utilizar estas herramientas. La elección metodológica se ha orientado a enfoques distintos para la ejecución de scripts y el procesamiento de datos, en consonancia con los objetivos específicos del proyecto. La mención de Shiny y R Markdown en esta sección sirve para contextualizar las tendencias actuales dentro del ecosistema R y resaltar la diversidad de soluciones disponibles en el ámbito del análisis de datos en entornos web.

### 3.3.2. RStudio Server: Arquitectura, funcionalidades y aprendizaje de librerías

RStudio Server es una herramienta esencial que permite a los usuarios acceder de forma remota a un entorno de desarrollo integrado a través de un navegador web. Su arquitectura se basa en un modelo cliente-servidor en el que:

- Un servidor central administra las sesiones de R para múltiples usuarios de manera simultánea.
- Cada usuario opera en un entorno aislado y seguro, lo que facilita la ejecución de scripts y el análisis colaborativo de datos.

Es importante destacar que en este proyecto se utiliza la versión **Community Edition** de RStudio Server [RStudio, 2024a]. A diferencia de la versión Professional, que incluye opciones avanzadas de autenticación (como LDAP o SAML) y herramientas de administración centralizada (monitoreo y balanceo de carga), la versión Community es gratuita y de código abierto, con funcionalidades básicas. Esto limita la implementación de mecanismos de seguridad y escalabilidad avanzados.

Esta limitación es relevante, ya que el diseño del sistema debe adaptarse a las características de la versión gratuita, buscando alternativas o complementos para suplir aquellas funciones reservadas a la edición Professional.

## 3.4. Desarrollo de APIs en R

El desarrollo de APIs en R ha abierto la puerta a que las funcionalidades y modelos desarrollados sean accesibles desde aplicaciones externas mediante protocolos estándar como HTTP. Esto permite:

- Integrar procesos analíticos con otros sistemas.
- Automatizar la generación de reportes y análisis complejos.
- Promover la interoperabilidad entre aplicaciones web, móviles y otros entornos.

Al exponer funciones de R a través de APIs, se potencia el aprovechamiento de los recursos analíticos y se facilita la expansión de la funcionalidad del sistema.

### 3.4.1. Plumber: Creación de servicios REST en R

Plumber [Schloerke and Allen, 2025] es un paquete de R que ha revolucionado la creación de APIs, ya que permite transformar funciones y scripts en endpoints REST de forma ágil. Algunas características destacadas son:

- **Anotaciones en el código:** Utiliza comentarios en el código R para definir rutas, parámetros y formatos de respuesta.
- **Simplicidad:** Reduce significativamente la complejidad del desarrollo de servicios web, permitiendo exponer funcionalidades complejas en formatos comunes como JSON.
- **Escalabilidad:** Facilita la integración de R en ecosistemas más amplios, donde se requieren soluciones escalables y de fácil mantenimiento [Schloerke and Allen, 2025].

Esta herramienta permite que aplicaciones externas interactúen de forma directa con el análisis de datos realizado en R, abriendo nuevas posibilidades en la integración de sistemas.

## 3.5. Frameworks para desarrollo frontend

El desarrollo de aplicaciones web modernas se apoya en frameworks que facilitan la construcción de interfaces dinámicas, responsivas y escalables. Estos frameworks:

- Mejoran la experiencia del usuario mediante interfaces interactivas.
- Permiten la integración fluida con APIs y servicios backend.
- Garantizan el rendimiento, la modularidad y la facilidad de mantenimiento del sistema.

### 3.5.1. Introducción a Angular y su uso en aplicaciones interactivas

Angular es un framework frontend basado en TypeScript y mantenido por Google, consolidado como una opción robusta para desarrollar aplicaciones web de una sola página (SPA) con interfaces dinámicas y escalables. En este proyecto se utiliza **Angular 19**, desarrollado en Visual Studio Code y ejecutado sobre **Node.js v20.18.0**. Aunque Angular no proporciona un backend propio, su arquitectura basada en componentes permite construir una interfaz de usuario modular y altamente interactiva, que consume APIs externas para gestionar la lógica del servidor.

La estructura modular de Angular promueve la reutilización del código y facilita el mantenimiento, aspectos fundamentales en aplicaciones de gran envergadura. Entre las características clave de Angular se destacan:

- **Enlace de datos bidireccional:** Este mecanismo sincroniza automáticamente el modelo y la vista, garantizando que cualquier cambio en los datos se refleje instantáneamente en la interfaz de usuario.
- **Inyección de dependencias:** Permite compartir servicios y otros componentes de forma sencilla, promoviendo la modularización y reutilización del código.
- **Sistema de enrutamiento robusto:** Facilita la navegación entre diferentes vistas de la aplicación, permitiendo el desarrollo de SPA con transiciones fluidas y una experiencia de usuario optimizada.

Además, para lograr una maquetación y diseño consistente, se ha incorporado **Angular Material**, un conjunto de componentes de interfaz basados en las directrices de Material Design. Angular Material ofrece módulos predefinidos que facilitan la implementación de botones, íconos, tarjetas, barras de herramientas y formularios. Estos componentes no solo agilizan el desarrollo al proveer una apariencia uniforme y personalizable, sino que también mejoran la experiencia del usuario al garantizar una interfaz moderna y responsiva.

Angular facilita la integración con servicios backend mediante la creación de **Angular Services**. Estos servicios, que utilizan el módulo `HttpClient`, encapsulan la lógica para consumir APIs REST de forma reactiva a través de **RxJS**. Gracias a esta arquitectura, la aplicación puede realizar peticiones HTTP, gestionar errores y reintentos, y actualizar dinámicamente la interfaz sin interrumpir la experiencia del usuario.

En resumen, Angular se utiliza en este TFG para construir una interfaz de usuario altamente interactiva y modular, mientras que Angular Material aporta un conjunto de componentes de UI que optimizan la maquetación y el diseño. La integración fluida entre Angular y los servicios backend (desarrollados en R utilizando Plumber) garantiza una experiencia de usuario eficiente y escalable, fundamental para la ejecución remota de scripts en R [Angular, 2024b][Angular, 2024a].

## 3.6. Gestión de Usuarios y Autenticación

En el contexto del presente TFG, la gestión de usuarios y la autenticación se han diseñado para asegurar el correcto acceso a las funcionalidades críticas del sistema, sin recurrir a

soluciones complejas como LDAP. La implementación se basa en el uso de scripts en bash que, mediante comandos con privilegios (a través de `sudo`), permiten la creación y administración de usuarios en el entorno Linux. Dichos scripts, por ejemplo `crear_usuario.sh` y `crear_usuario_noadmin.sh`, se integran en la API REST desarrollada con Plumber, permitiendo ejecutar operaciones de creación, modificación y eliminación de usuarios de forma dinámica. Además, se han implementado endpoints que permiten iniciar y finalizar sesiones de usuario, garantizando que la API se levante al inicio de la sesión y se cierre al finalizarla, lo que optimiza el uso de recursos y refuerza la seguridad del sistema.

Este enfoque simplificado se adapta a la escala y necesidades del proyecto, facilitando la administración de los usuarios y asegurando que sólo aquellos autorizados puedan interactuar con el sistema. La diferenciación entre administradores y usuarios básicos se gestiona directamente en el sistema operativo, lo que permite controlar el acceso a las funciones críticas sin la necesidad de implementar un sistema de autenticación más complejo.

## 3.7. Tecnologías Complementarias y su Integración

La integración de diversas tecnologías complementarias es clave para dotar al sistema de flexibilidad, escalabilidad y facilidad de mantenimiento. En este apartado se describen dos áreas fundamentales: la dockerización y la gestión de ficheros mediante operaciones CRUD sobre JSON.

### 3.7.1. Dockerización y Gestión de Contenedores

La dockerización del sistema ha permitido aislar y estandarizar el entorno de ejecución tanto para el backend como para el frontend. En este TFG, se han definido dos contenedores principales:

- **myapi-backend:** Contenedor que aloja la API REST desarrollada con Plumber, junto con los scripts en R.
- **angular-frontend:** Contenedor dedicado a la aplicación web en Angular, gestionado a través de Node.js.

Esta separación funcional facilita el despliegue del sistema, permitiendo que los tutores o cualquier usuario con acceso puedan implementar y probar la aplicación de forma autónoma. Además, el uso de Docker garantiza que el entorno de ejecución sea reproducible y escalable, minimizando discrepancias entre los diferentes entornos de desarrollo y producción [Inc., 2023b, Inc., 2023a].

### 3.7.2. Gestión de Ficheros y Operaciones CRUD sobre JSON

Otro aspecto crucial en el desarrollo del sistema es la gestión dinámica de ficheros, la cual se aborda mediante la implementación de un servicio web en R utilizando Plumber. Este servicio se encarga de gestionar un fichero JSON que almacena información relevante, como los metadatos de los programas en R. Mediante operaciones CRUD (Crear, Leer,

Actualizar y Borrar), el servicio permite mantener y modificar de forma centralizada las entidades que componen el sistema, facilitando la integración con la API REST. Además, se ha implementado la funcionalidad para contar el número de entidades presentes en el fichero JSON, lo cual es fundamental para la administración y monitoreo de la información gestionada. Esta solución mejora la flexibilidad del sistema, permitiendo que los usuarios actualicen dinámicamente la información sin necesidad de modificar manualmente el código, lo que contribuye a una mayor eficiencia en la gestión de datos.

# 4. Metodología y Diseño del Sistema

## 4.1. Introducción

En este capítulo se describe la metodología utilizada para la planificación, definición y diseño del sistema. Se detalla el proceso seguido para recopilar y estructurar los requisitos, así como el diseño arquitectónico que integra las distintas tecnologías empleadas.

## 4.2. Metodología de desarrollo aplicada

Esta sección expone el enfoque adoptado para la gestión y desarrollo del TFG, basado en metodologías reconocidas y en herramientas de gestión de proyectos.

### 4.2.1. Planificación y gestión del TFG según el Plan de Sistemas de Información (PSI)

Se describe el proceso de planificación basado en el PSI, que ha orientado la definición de objetivos, la identificación de tareas y la distribución de responsabilidades.

### 4.2.2. Uso del estándar PMI para la gestión de proyectos

Se explica cómo se ha aplicado el estándar PMI en la organización y seguimiento del proyecto, garantizando el cumplimiento de hitos y la correcta asignación de recursos.

### 4.2.3. Planificación del trabajo mediante EDT

Se detalla la elaboración de la Estructura Desglosada del Trabajo (EDT), que ha permitido descomponer el proyecto en fases y actividades, facilitando la planificación y el control de avances.

### 4.2.4. Uso de Kanban para la gestión de tareas

Se describe la implementación de un tablero Kanban para la gestión diaria de las tareas, permitiendo un seguimiento visual y ágil del progreso del proyecto.

## 4.3. Definición de requisitos

En esta sección se documenta el proceso metodológico para la definición de los requisitos del sistema. Se explica cómo se recopilaron, analizaron y estructuraron los requisitos utilizando las plantillas oficiales del Documento de Requisitos del Sistema (DRS). Aquí se debe incluir:

- La descripción de las técnicas de recolección de información (entrevistas, revisión documental, análisis de casos similares, etc.).
- La forma en que se adaptaron las plantillas del DRS al contexto del proyecto.
- El procedimiento de validación interna (revisión con tutores, reuniones de feedback, etc.).

*Nota: La versión final y validada de los requisitos (tanto funcionales como no funcionales) se presenta en detalle en el Capítulo 5 y se complementa con la evidencia práctica en los Anexos.*

## 4.4. Diseño del sistema

Esta sección presenta la arquitectura y el diseño de la solución propuesta, describiendo la interacción entre los distintos componentes y las consideraciones de seguridad.

### 4.4.1. Arquitectura General: Frontend, Backend, RStudio Server y Docker

Se explica la estructura del sistema, que integra el frontend desarrollado en Angular, el backend basado en Node.js y la ejecución de scripts en R mediante RStudio Server.

### 4.4.2. Interacción entre componentes y comunicación entre APIs

Se detalla el flujo de información y la comunicación entre el frontend, la API REST y el entorno de ejecución en R, describiendo los protocolos y mecanismos empleados.

### 4.4.3. Seguridad en la Ejecución de Scripts

Se exponen las medidas de seguridad implementadas, con especial énfasis en la integración con RStudio Server y en la gestión de los ficheros que contienen los scripts en R.

# 5. Experimentos, validación y resultados

## 5.1. Introducción

En este capítulo se recoge el proceso de implementación, las pruebas realizadas y los resultados obtenidos durante el desarrollo del sistema. Se presenta de forma detallada la ejecución práctica de la solución, evidenciando el funcionamiento de cada uno de sus componentes.

## 5.2. Presentación detallada de los requisitos funcionales y no funcionales

En esta sección se presenta la versión final y validada de los requisitos del sistema, documentada conforme a las plantillas oficiales del DRS. A continuación se incluye el contenido completo del DRS, que abarca tanto los requisitos funcionales como los no funcionales:

### 5.2.1. Requisitos funcionales

Los requisitos funcionales describen las funcionalidades esenciales que debe ofrecer el sistema para satisfacer las necesidades de sus usuarios. Entre los principales se destacan:

1. **RF-01: Gestión de Programas.** Este requisito permite al administrador gestionar los programas disponibles en el sistema. Las acciones contempladas incluyen:
    - Acceso a la sección de gestión de programas tras la autenticación.
    - Visualización de un listado que muestra el nombre, propósito y metadatos de cada programa, junto con iconos que permiten ejecutar el programa o consultar sus detalles.
    - Opciones para añadir, modificar o eliminar programas, con secuencias alternativas que habilitan iconos adicionales para la edición y borrado en caso de que el usuario posea permisos de administrador.

- Actualización dinámica del listado tras cada operación.

2. **RF-02: Ejecución de Programas.** Este requisito permite al usuario ejecutar programas en R mediante un formulario web. El proceso incluye:
   - Presentación de un formulario con los parámetros necesarios para la ejecución, incluyendo valores predeterminados configurados por el creador del programa.
   - Permitir la modificación de dichos parámetros por parte del usuario.
   - Interacción con RStudio Server para ejecutar el programa utilizando los ficheros y metadatos actualizados.
   - Visualización de los resultados de la ejecución, junto con un resumen de los metadatos del proceso.
3. **RF-03: Consulta de Metadatos.** Este requisito posibilita que el usuario acceda a la información detallada de cada programa. La funcionalidad abarca:
   - Visualización de los metadatos, que incluyen la descripción de cada parámetro, el tipo de dato y su significado.
   - Permitir al usuario cerrar la ventana de metadatos y retornar al flujo principal del sistema.
   - Mostrar mensajes alternativos en caso de que el programa no disponga de metadatos o se muestren valores de ejemplo.

### 5.2.2. Requisitos no funcionales

Los requisitos no funcionales establecen las características de calidad y restricciones que debe cumplir el sistema, orientadas a garantizar su rendimiento, seguridad y facilidad de uso. Entre ellos se destacan:

1. **RNF-01: Integración con RStudio Server.** El sistema debe integrarse de forma eficaz con RStudio Server para la ejecución de scripts en R mediante una API REST. Esto implica garantizar una comunicación rápida y segura, así como la correcta visualización y almacenamiento de los resultados obtenidos. Adicionalmente, el sistema debe incorporar funcionalidades para la gestión de los ficheros que contienen los programas escritos en R. Esto abarca la carga, almacenamiento y recuperación de dichos ficheros de forma organizada, lo que resulta esencial para asegurar la trazabilidad de las operaciones, facilitar la depuración y mantener la integridad de los scripts utilizados.
2. **RNF-02: Lenguaje y Framework de Desarrollo (Angular).** La interfaz de usuario se desarrollará utilizando Angular, lo que posibilita la creación de componentes modulares y escalables. Esta elección tecnológica permite una interacción fluida con los servicios REST y facilita el mantenimiento y la evolución futura del sistema.
3. **RNF-03: Seguridad.** Es fundamental implementar mecanismos robustos de autenticación y autorización que aseguren el acceso controlado a las funcionalidades críticas del sistema. Esto incluye:

- La protección de la integridad y confidencialidad de los datos.
- La implementación de controles de acceso que restrinjan las operaciones de gestión de programas únicamente a usuarios autorizados.

### 5.2.3. Definición de actores

- **ACT-01: Invitado.** Representa al individuo que utiliza el sistema para ejecutar programas en R, consultar metadatos y visualizar resultados gráficos.
- **ACT-02: Administrador.** Es responsable de gestionar los programas en el sistema, incluyendo la capacidad de añadir, modificar y eliminar programas disponibles para los usuarios.
- **ACT-03: RStudio Server.** Representa el sistema RStudio Server, encargado de ejecutar los programas en R y devolver los resultados al sistema.

## 5.3. Implementación del backend

En esta sección se detalla el proceso de desarrollo e implementación del backend, incluyendo la creación de la API REST, la configuración de RStudio Server y la implementación del sistema de autenticación.

### 5.3.1. Desarrollo de la API REST con Plumber

Se describe el proceso de creación de la API REST utilizando el paquete Plumber en R, con ejemplos de código y rutas definidas.

### 5.3.2. Configuración de R, RStudio Server y dockerización del Backend

Se explica la configuración y adaptación de RStudio Server (Community Edition) para la ejecución remota de scripts en R, destacando las limitaciones y las adaptaciones realizadas.

### 5.3.3. Gestión dinámica de APIs, usuarios y sesiones

Se detalla la integración del sistema de autenticación basado en LDAP, incluyendo el flujo de autenticación y las medidas de seguridad implementadas.

## 5.4. Implementación del frontend

Se expone el desarrollo de la interfaz de usuario en Angular y su integración con la API REST.

### 5.4.1. Creación de la interfaz en Angular

Se describe el proceso de diseño y desarrollo de la interfaz, resaltando la arquitectura basada en componentes y las técnicas empleadas para asegurar la usabilidad.

### 5.4.2. Comunicación con la API REST y manejo de sesiones

Se explica cómo se establece la comunicación entre el frontend y la API REST, incluyendo la gestión de peticiones y respuestas.

### 5.4.3. Gestión de formularios y validaciones

Se detallan las funcionalidades implementadas para la gestión de formularios, validación de datos y retroalimentación al usuario.

## 5.5. Despliegue y ejecución de la aplicación

Se describe el proceso de despliegue del sistema, tanto en el frontend como en el backend, y se explica cómo se ha verificado su correcto funcionamiento.

### 5.5.1. Ejecución del frontend (Angular)

Se detalla la puesta en marcha de la aplicación Angular y las herramientas utilizadas para su despliegue.

### 5.5.2. Ejecución del backend (Node.js y scripts en R)

Se explica el proceso de integración y ejecución del backend, destacando la comunicación entre Node.js y la ejecución de los scripts en R.

## 5.6. Plan de pruebas y evaluación

Se presenta el conjunto de pruebas realizadas para evaluar el funcionamiento del sistema, desde pruebas unitarias hasta de integración y seguridad.

### 5.6.1. Pruebas unitarias de los servicios REST

Se describen las pruebas unitarias realizadas para garantizar el correcto funcionamiento de cada servicio de la API REST.

### 5.6.2. Pruebas de integración entre Angular y el backend

Se detalla el proceso de integración y las pruebas realizadas para asegurar la interoperabilidad entre el frontend y el entorno de ejecución en R.

### 5.6.3. Pruebas de seguridad en la ejecución de código R

Se explican las pruebas implementadas para validar la seguridad y robustez en la ejecución de scripts en R.

## 5.7. Resultados y análisis

Se integran y analizan los resultados obtenidos, demostrando cómo se han cumplido los objetivos del proyecto a través de evidencias prácticas (pantallazos, gráficos, fragmentos de código, etc.).

### 5.7.1. Análisis de Errores Técnicos y Soluciones

# 6. Conclusiones y trabajo futuro

6.1. Conclusiones

6.2. Limitaciones y dificultades encontradas

6.3. Trabajo desarrollado y competencias adquiridas

6.4. Posibles mejoras y trabajo futuro

# Referencia bibliográfica


[Angular, 2024a] Angular (2024a). Angular documentation. Disponible en: https://angular.io/docs.

[Angular, 2024b] Angular (2024b). The modern web framework. Disponible en: https://angular.io/.

[Canonical and Microsoft, 2022] Canonical and Microsoft (2022). Windows subsystem for linux (wsl). Disponible en: https://docs.microsoft.com/en-us/windows/wsl/.

[Inc., 2023a] Inc., D. (2023a). Docker compose documentation. Disponible en: https://docs.docker.com/compose/.

[Inc., 2023b] Inc., D. (2023b). Docker engine v27.5.1. Descargado de: https://docs.docker.com/desktop/setup/install/windows-install/. Utiliza el backend de WSL 2, por lo que los límites de recursos son gestionados por Windows.

[Institute, 2021] Institute, P. M. (2021). *PMBOK Guide – 7th Edition*. PMI Publishing.

[RStudio, 2024a] RStudio (2024a). Rstudio server community edition. Disponible en: https://rstudio.com/products/rstudio-server/.

[RStudio, 2024b] RStudio (2024b). Shiny: Web application framework for r. Disponible en: https://shiny.rstudio.com/.

[Schloerke and Allen, 2025] Schloerke, B. and Allen, J. (2025). plumber: An api generator for r. Versión del paquete R 1.3.0. Disponible en: https://www.rplumber.io.

[Team, 2023] Team, R. C. (2023). The r project for statistical computing. R Foundation for Statistical Computing. Disponible en: https://www.r-project.org.

# A. Anexo: Instalación y Configuración del Sistema

## A.1. Instalación de RStudio Server en Ubuntu

Este apartado describe detalladamente el proceso seguido para configurar un entorno Linux sobre Windows mediante WSL2 y para instalar RStudio Server utilizando Posit Workbench. Esta configuración es fundamental para disponer de un entorno centralizado, robusto y reproducible que permita la ejecución remota de scripts en R, optimizando el desarrollo y la integración de herramientas analíticas.

### A.1.1. Instalación y Configuración de Ubuntu en WSL

Para preparar el entorno de desarrollo en Windows, se utiliza el Subsistema de Windows para Linux (WSL2). El proceso consta de los siguientes pasos:

1. **Verificar los requisitos:** Se debe disponer de Windows 10 (versión 2004 o superior) o Windows 11. Se recomienda comprobar la versión ejecutando el comando `winver`.
2. **Habilitar WSL y la Plataforma de Máquinas Virtuales:** Abra Windows PowerShell con privilegios de administrador y ejecute los siguientes comandos:

   ```
   dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-for-
   dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /nor
   ```

   Tras ejecutar estos comandos, reinicie el sistema.
3. **Establecer WSL 2 como versión predeterminada:** Una vez reiniciado el sistema, ejecute:

   ```
   wsl --set-default-version 2
   ```

   Si se solicita una actualización del kernel de Linux, descargue e instálelo desde `https://aka.ms/wsl2kernel`.
4. **Instalar Ubuntu 20.04:** Descargue e instale la distribución Ubuntu 20.04 desde la Microsoft Store. Al iniciar Ubuntu, se le pedirá que configure un usuario UNIX y establezca una contraseña, la cual se utilizará posteriormente para acceder a RStudio Server.
5. **Actualizar el sistema:** Una vez configurado, ejecute en la terminal de Ubuntu:

   ```
   sudo apt-get update && sudo apt-get upgrade -y
   ```

### A.1.2. Uso de Posit Workbench para la Instalación y Configuración de RStudio Server

Con Ubuntu debidamente configurado en WSL2, se procede a instalar RStudio Server utilizando Posit Workbench. Los pasos realizados son los siguientes:

1. **Agregar el repositorio de CRAN y la clave GPG:** Ejecute:

   ```
   sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E298A3A825C0D65DFD
   sudo add-apt-repository 'deb https://cloud.r-project.org/bin/linux/ubuntu focal-
   ```

2. **Instalar R y dependencias:** Actualice la lista de paquetes e instale R y las librerías necesarias:

   ```
   sudo apt-get update
   sudo apt install -y r-base r-base-dev libcurl4-gnutls-dev libxml2-dev libssl-dev
   ```

3. **Descargar e instalar RStudio Server:** Descargue la versión estable adecuada para Ubuntu 20.04:

   ```
   wget https://download2.rstudio.org/server/focal/amd64/rstudio-server-2024.12.1-56
   sudo apt-get install -y gdebi-core
   sudo gdebi rstudio-server-2024.12.1-563-amd64.deb
   ```

   Se recomienda ignorar mensajes de advertencia como “No se pudo encontrar una implementación alternativa de telinit”, ya que son conocidos en este entorno.

4. **Iniciar RStudio Server:** Ejecute:

   ```
   sudo rstudio-server start
   ```

5. **Acceso al sistema:** RStudio Server estará disponible en `http://localhost:8787`. Para acceder, utilice las credenciales del usuario UNIX configurado en Ubuntu.

## A.2. Ejemplos de llamadas a la API

## A.3. Respuestas esperadas

# B. Anexo: Instalación y configuración del Sistema

## B.1. Instalación de RStudio Server en Ubuntu

## B.2. Configuración de Plumber para servicios REST

## B.3. Despliegue de la aplicación Angular