UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

GRADO EN INGENIERÍA DEL SOFTWARE

Desarrollo de una aplicación web para la gestión de datos obtenidos mediante técnicas de automatización.

Web application for management data got by automation techniques development.

Realizado por

Manuel Esquivel Marín

Tutorizado por

José Luis Pastrana Brincones

Departamento

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2023

UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADUADA/O EN INGENIERÍA DEL SOFTWARE

Desarrollo de una aplicación web para la gestión de datos obtenidos mediante técnicas de automatización.

Web application for management data got by automation techniques development.

Realizado por

Manuel Esquivel Marín

Tutorizado por

José Luis Pastrana Brincones

Departamento

Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO DE 2023

Fecha defensa: septiembre de 2023

Resumen

Este Trabajo de Fin de Grado (TFG) tiene como objetivo diseñar e implementar una aplicación web usable que simplifique la ejecución y visualización de tareas automatizables, con el fin de optimizar la productividad de los usuarios que la utilicen, así como el uso eficiente del tiempo. Actuando como un enlace entre desarrolladores experimentados y usuarios sin experiencia técnica, la plataforma busca democratizar el acceso a las tecnologías de automatización.

La aplicación se centra en dos categorías de tareas: 'tareas específicas', que están preconfiguradas en el sistema y se basan en aplicaciones ya conocidas; y 'tareas genéricas', que ofrecen a los usuarios un margen de personalización más amplio. A través de un panel de usuario intuitivo, se pueden seleccionar tanto la tecnología subyacente (como aplicaciones web o de escritorio) como la aplicación específica (por dirección web o su descripción). Posteriormente, los usuarios pueden configurar y ejecutar estas tareas según sus necesidades particulares.

Adicionalmente, el sistema cuenta con un panel de historial que registra detalles de cada tarea ejecutada, como la fecha de ejecución, el éxito o fracaso, y los datos obtenidos. Un sistema de roles diferencia entre 'usuarios básicos' y 'desarrolladores', permitiendo una escalabilidad y evolución continua del conjunto de tareas y tecnologías soportadas.

Palabras clave: automatización de tareas, interfaz de usuario, desarrollo web, roles de usuario, escalabilidad

Abstract

This Final Project aims to design and implement an usable web application to streamline task automation, thereby enhancing user productivity and time efficiency. Serving as a bridge between skilled developers and users lacking technical expertise, the platform endeavors to democratize access to automation technologies.

The application targets two types of tasks: 'specific tasks,' pre-configured within the system and tailored for well-known applications, and 'generic tasks,' offering users a wider scope for customization. An intuitive user panel facilitates the selection of both the underlying technology, such as web or desktop applications, and the specific application by its web address or description. Users can then configure and execute these tasks according to their unique requirements.

Additionally, the system incorporates a history panel, logging crucial details of each executed task, including the execution date, success or failure, and data retrieved. A role-based system distinguishes between 'basic users' and 'developers,' allowing for the platform's continual scalability and the evolution of its supported task and technology set.

Keywords: task automation, user interface, web development, role-based system, scalability

Tabla de contenido

Índice de tablas

Índice de figuras

1

Introducción

1.1 Motivación

Desde los comienzos en la informática se ha observado el potencial de la automatización, conseguir que el cálculo de operaciones costosas se viera reducido gracias a la programación y el uso eficiente de los algoritmos ha supuesto un avance del que nos beneficiamos a diario en la actualidad. En un mundo donde el volumen de datos crece cada día y los usuarios buscan realizar tareas de mayor valor que continuas repeticiones de la misma actividad, la necesidad de automatizar procesos que resuelvan problemas similares de forma simplificada se ha convertido en una preocupación tanto individual como empresarial.

Sin embargo, a pesar de que existen importantes avances en el área de la automatización y herramientas que permiten configurar y ejecutar procesos de forma autónoma [1], existen barreras que impiden que usuarios con poco conocimiento técnico se beneficien de esta área. Además, el costo y complejidad de implementación de estas soluciones suponen un obstáculo inicial que plantea un escenario en el que buscar una alternativa más accesible que permita a cualquier usuario, independientemente de su nivel de experiencia, beneficiarse de la automatización de tareas.

Al democratizar el acceso a las tecnologías de automatización, además de buscar la productividad del usuario, se pretende encontrar una solución escalable que recoja en un sólo lugar las tareas, y los resultados de ejecutar estas, de forma que pueda adaptarse a las diferentes necesidades de automatización en el tiempo.

1.2 Objetivos

El principal objetivo de este Trabajo de Fin de Grado (TFG) es desarrollar una aplicación web orientada a la gestión de datos con procedencia del resultado de ejecutar las distintas tareas de automatización sobre una aplicación externa conocida, o una tecnología específica. Como consecuencia, la aplicación mejora la productividad del usuario que la utiliza teniendo en un mismo sitio de forma agrupada las tareas que pueden ejecutarse y las que ya se han ejecutado en algún momento.

El sistema inicial cuenta con dos tareas automatizables específicas:

- Firmado de documentos PDF usando la librería conocida como 'iText Core' [2], a través de la tecnología .NET [3]. Para la configuración de la tarea, el usuario debe seleccionar los ficheros que van a firmarse.
- Obtención de noticias recientes de la aplicación web de la Universidad de Málaga (UMA) [4], a través de la tecnología Selenium [5]. Para la configuración de la tarea, el usuario debe especificar un número máximo de las noticias que el sistema debería recuperar.

Además, incorpora una tarea automatizable genérica con una configuración que sería usada por usuarios con conocimientos más avanzados:

- Obtención de datos de forma estructurada de una dirección web especificada por el usuario, a través de llamadas HTTP [6]. Para la configuración de la tarea, el usuario debe especificar también una expresión en el lenguaje XPath [7] sobre la que el sistema pueda comenzar a trabajar para recuperar la información deseada.

Para lograr el objetivo principal, se han definido algunos objetivos específicos.

1.2.1 Diseño usable e intuitivo

La aplicación web debe diseñarse de forma que el uso de esta sea intuitivo [8], y así lograr que la ejecución de tareas y visualización de estas no supongan un obstáculo inicial para el usuario.

Para conseguirlo, se definen dos interfaces de usuario: una ventana de selección de la tarea automatizable y, otra ventana, para la visualización histórica de las últimas tareas ejecutadas con los resultados obtenidos por estas. Aquí la aplicación debe facilitar el uso de filtros.

1.2.2 Interoperabilidad de tecnologías

El sistema cuenta con el uso de diferentes tecnologías para la realización de las diferentes tareas y la construcción de la propia aplicación.

1.2.3 Escalabilidad y mantenibilidad

El sistema debe construirse con el objetivo de facilitar en el futuro la adición de nuevas funcionalidades, como tareas automatizables y tecnologías.

1.2.4 Gestión de roles

Para que la aplicación tenga un continuo crecimiento, los usuarios de esta deben diferenciarse en, al menos, dos tipos de roles diferenciados: 'usuarios básicos', que puedan tener alguna facilidad para proponer nuevas tareas automatizables; 'desarrolladores', que además de ser usuarios básicos, tienen visibilidad de las propuestas realizadas por otros usuarios.

1.2.5 Seguimiento y análisis

Los usuarios de la aplicación deben contar con la visibilidad de un histórico personal de tareas que han lanzado, no solo para conocer el resultado de estas, sino también para consultar las operaciones que se han realizado y eliminarlas en caso de que se desee.

1.3 Estructura de la memoria

La presente memoria se estructura de forma que se facilite la comprensión de la metodología empleada para el desarrollo del TFG. A continuación, se detalla cada capítulo con objeto de ofrecer una visión completa del proyecto:

1. **Capítulo 1: Introducción**

   Se introduce la contextualización del problema que se plantea sobre la automatización de tareas, estableciendo los motivos y objetivos que guían la realización del proyecto. Además, se especifica una descripción detallada de las funcionalidades a alto nivel del desarrollo de software que se va a realizar y, por último, se proporciona un resumen de la estructura que sigue este documento.

2. **Capítulo 2: Estado del arte**

   Este capítulo recoge un análisis de las herramientas y tecnologías existentes en el ámbito de la automatización de procesos. Se identifican las limitaciones actuales de estas y cómo este proyecto busca abordarlas. Finalmente, se especifican las diferentes tecnologías con las que el sistema va a trabajar.

3. **Capítulo 3: Metodología ágil**

   Se realiza una descripción de la metodología ágil empleada para el desarrollo del TFG, y cómo se ha adaptado para la realización de este a partir de iteraciones. Además, se presenta una planificación temporal alineada con esta metodología.

4. **Capítulo 4: Análisis de requisitos**

   Se describen los detalles iniciales específicos de las funcionalidades que debe tener el sistema en las dos primeras iteraciones, es decir, los requisitos funcionales y no funcionales. Por otro lado, se muestra el mínimo número de casos de uso que definen las principales interacciones del usuario con la aplicación web.

5. **Capítulo 5: Primera iteración**

   Este capítulo aborda el diseño e implementación inicial del sistema, incluyendo la arquitectura, los componentes principales, y los elementos de diseño que se alinean con los objetivos de la aplicación. Por último, se presentan los métodos y resultados de las pruebas realizadas en esta iteración para discutir con un usuario los hitos que se han logrado en esta primera fase del proyecto.

6. **Capítulo 6: Segunda iteración**

   Se continúa con la segunda iteración, fase en la que se abordan las mejoras discutidas de la primera iteración. Y, de forma similar a esta, se describen las pruebas e hitos conseguidos en esta fase.

7. **Capítulo 7: Tercera iteración**

   Esta tercera fase del proyecto tiene como objetivo principal la observación de cómo respondería el sistema ante un cambio identificado en los requisitos de este. Como se venía haciendo, también se detallan las actualizaciones y modificaciones en el diseño e implementación de la aplicación, así como las pruebas e hitos logrados.

8. **Capítulo 8: Cuarta iteración**

   En la última fase se revisan los últimos detalles del proyecto, pero también se centra en la incorporación de un nuevo requisito.

9. **Capítulo 9: Plan de pruebas**

   El principal propósito de este capítulo es presentar el plan de pruebas de forma detallada para garantizar que los requisitos funcionales y no funcionales del sistema se han cumplido en tiempo y forma. Entre las pruebas realizadas se encuentran: unitarias, de integración y de carga.

10. **Capítulo 10: Conclusiones y líneas futuras**

    Se resume el trabajo realizado con una perspectiva global para valorar si se han cumplido los objetivos previstos para este, así como las conclusiones generales. Por otro lado, se analiza si el proyecto puede ser un punto de partida para trabajos futuros que continúen en la misma línea.

2

Estado del arte

Durante este capítulo, se explora el paisaje actual de las herramientas de automatización existentes que buscan conseguir objetivos similares al proyecto que se presenta en esta memoria.

En primer lugar, cabe destacar que las soluciones analizadas aquí comparten una visión parecida a la hora de abordar algún desafío de automatización que se intenta resolver. En concreto, la mayoría de las plataformas ofrecen un sistema de trabajo basado en flujos, como los conocidos ampliamente 'diagramas de flujo' [9], y tratan de facilitar a los usuarios la construcción de estos sin tener que aprender directamente un lenguaje de programación.

Por lo tanto, se va a realizar un análisis que permita entender mejor el contexto en el cual se inscribe este trabajo y, de esta manera, ofrecer una visión clara del área donde puede aportar una innovación significativa.

Al final del capítulo, se presentan las herramientas y tecnologías empleadas en el desarrollo del proyecto.

2.1 Herramientas de automatización

2.1.1 Blue Prism

Figura 1. Logo de Blue Prism

Blue Prism [10] es un software de automatización de procesos. Está diseñado específicamente para ejecutar flujos secuenciales en forma de diagramas para realizar tareas específicas sobre una amplia gama de aplicaciones. Aunque se trata de una solución potente para automatizar diversas tareas, presenta una limitación para el usuario final porque requiere de un aprendizaje previo de la herramienta para usarlo de forma eficiente. Además, al tratarse de un software comercial, las licencias pueden resultar costosas y complejas de implementar para usuarios individuales o pequeñas empresas. Blue Prism, aunque muestra de forma intuitiva qué ha trabajado mediante un panel de cola de trabajo de la tarea automatizada, no ofrece ninguna alternativa cuando se necesita observar los datos de la cola ni los resultados finales de la ejecución realizada.

2.1.2 UiPath

Figura 2. Logo de UiPath

UiPath [11], al igual que Blue Prism, es otra herramienta de automatización de procesos. Se trata de una alternativa más fácil de usar que la anterior y que ofrece una versión de la comunidad para que pueda ser utilizada y probada por usuarios individuales. Sin embargo, existe el mismo desafío, UiPath no es una herramienta que ofrezca al usuario final una visibilidad intuitiva de los resultados obtenidos tras haberse ejecutado las diferentes tareas.

2.1.3 Microsoft Power Automate

Figura 3. Logo de Power Automate

Microsoft Power Automate [12] es quizás la herramienta que más se alinee con los objetivos de este proyecto, ya que ofrece una plataforma usable e intuitiva para ejecutar flujos de trabajo de tareas repetitivas y que organiza estas para el usuario final. Es compatible con una amplia gama de productos y servicios de Microsoft, y el usuario puede organizar el

resultado de las ejecuciones fácilmente con hojas de cálculo. Sin embargo, uno de los propósitos de esta herramienta es intentar que un usuario sin experiencia técnica sea capaz de automatizar flujos sin ayuda de desarrolladores, pero uno de los objetivos que se buscan en este proyecto es que usuarios y desarrolladores se comuniquen para crear una plataforma de automatización colaborativa y robusta.

2.1.4 Zapier

Figura 4. Logo de Zapier

Zapier [13] es un producto que ofrece a los usuarios de la herramienta conectar aplicaciones y servicios diferentes para automatizar flujos de trabajo entre ellas, es decir, su objetivo es tratar de hacer intuitivo y versátil el intercambio de información entre diferentes aplicaciones. Zapier se alinea también con los objetivos de Microsoft Power Automate, por lo que nuestro proyecto compartiría algunos objetivos. Sin embargo, mientras que esta herramienta ofrece una serie de aplicaciones conocidas, el proyecto trata de hacer escalable la incorporación de cualquier tecnología o aplicación para ofrecer un rango más amplio de posibilidades. Aunque Zapier ofrece un registro de tareas ejecutadas, la profundidad de este no se alinea con los objetivos de este proyecto.

2.1.5 Selenium

Figura 5. Logo de Selenium

Selenium, que ha sido mencionado en los objetivos de la memoria, es una librería de código abierto que soporta automatización de navegadores web. Aunque el propósito original de Selenium han sido siempre las pruebas funcionales sobre una aplicación web desarrollada, se ha incluido en este apartado porque es una tecnología que ha crecido lo suficiente como para proporcionar las herramientas necesarias para interactuar con un navegador y realizar tareas repetitivas sobre él. Por otro lado, al tratarse de una librería, requiere de conocimientos técnicos para usarse y uno de los objetivos de este trabajo es trabajar con ella para que el usuario final pueda acercarse al uso de esta tecnología. Por último, Selenium está limitado a la tecnología web

que quiere automatizar y no ofrece un sistema intuitivo para gestionar los resultados obtenidos de ejecutar sus tareas.

2.2 Tecnologías a utilizar

2.2.1 Backend: ASP.NET Core

Figura 6. Logo de ASP.NET

ASP.NET Core [14] es una versión moderna de ASP.NET, un marco de desarrollo de aplicaciones web desarrollado por Microsoft. Se utiliza principalmente en el backend de un sistema para implementar y mantener aplicaciones y servicios web. Y, aunque ofrece una amplia gama de lenguajes de programación para utilizarse, en el desarrollo del proyecto se emplea el lenguaje C# [15]. Se trata de una potente tecnología que ofrece facilidades para conectarse con bases de datos y construir aplicaciones escalables, que es uno de los objetivos de este proyecto. Es capaz de gestionar de forma eficiente el CRUD [16] del modelo del proyecto. La versión de .NET utilizada en el sistema inicial es la 6.0.

2.2.2 Frontend: React

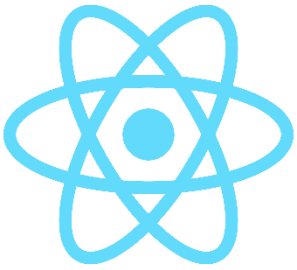

Figura 7. Logo de React

React [17] es una biblioteca de JavaScript [18] mantenida por Meta y que se ejecuta sobre el marco de trabajo de 'Next.js' [19], por lo que será necesario tener instalado el entorno 'Node.js' [20] en el backend del sistema. Se encuentra bien reconocida por su gran eficiencia en la actualización y renderización de componentes, lo que permite una creación sencilla de interfaces de usuario. Se alinea bien con el proyecto porque sigue una filosofía de modularizado de componentes que permite la reutilización de estos para facilitar la escalabilidad de la interfaz de la aplicación. Cabe destacar que utiliza una sintaxis similar a HTML [21] denominada JavaScript XML (JSX) [22], la cual proporciona una mejor legibilidad del código. La versión de React utilizada en el sistema es la 18.2.0.

2.2.3 Base de datos: SQL Server

Figura 8. Logo de SQL Server

SQL Server [23] se utiliza en el proyecto como sistema de gestión de bases de datos (RDBMS) [24] para almacenar y administrar la información relacionada con el sistema. Mantenido por Microsoft, SQL Server es una tecnología segura que ofrece el rendimiento necesario para soportar un gran volumen de datos, además de garantizar la integridad de estos. La versión de SQL Server utilizada en el sistema es la de 2022. Adicionalmente, para que la administración de los datos sea más intuitiva se hace uso de una herramienta gráfica denominada 'SQL Server Management Studio' [25] en su versión 19.1.

2.2.4 Gestor de paquetes: NuGet

Figura 9. Logo de NuGet

NuGet [26] es el gestor de paquetes de datos en .NET. Facilita la instalación de librerías que, a continuación, se describen en esta memoria. Es una herramienta que proporciona seguridad, ya que se encarga de gestionar todas las dependencias del proyecto, así como las compatibilidades de estas. NuGet se utiliza desde el propio entorno de desarrollo (IDE) [27].

2.2.5 Librería: RestSharp

Figura 10. Logo de RestSharp

RestSharp [28] es una librería que se utiliza en este trabajo para realizar llamadas HTTP en el Backend con el objetivo de proporcionar al proyecto una tecnología que sirva como apoyo a la automatización de alguna tarea que requiera consumir de un servicio web. La versión de RestSharp utilizada en el sistema es la 110.2.0.

2.2.6 Librería: iText Core

Figura 11. Logo de iText Core

iText Core es un marco de trabajo, mantenido por Apryse, que se caracteriza por ser de código abierto y comercial. Su principal objetivo es la creación y manipulación de ficheros PDF, pero también ofrece una amplia cobertura de funcionalidades para trabajar con estos. Para este proyecto se hará uso de su capacidad para realizar firmas digitales. La versión de iText Core utilizada en el sistema es la 8.0.1.

2.2.7 Librería: Selenium

Aunque esta tecnología se ha descrito y mencionado ya en la memoria presente, se destaca el uso del lenguaje de consulta XPath, que se usa con la librería, para recuperar información de un documento HTML durante alguna tarea.

2.2.8 Librería: HTMLAgilityPack

Figura 12. Logo de HTMLAgilityPack

HTMLAgilityPack [29] es una librería de .NET que facilita el análisis de documentos HTML en el proyecto, con soporte del lenguaje de consulta XPath. Es una herramienta útil en el proyecto para recuperar información de una página web de la que se desee extraer datos. La versión de HTMLAgilityPack utilizada en el sistema es la 1.11.52.

2.2.9 Librería: Json.NET

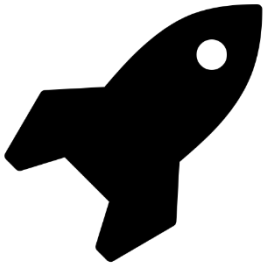

Figura 13. Logo de Json.NET

Json.NET [30] es una librería de .NET desarrollada por Newtonsoft que permite la serialización y deserialización de objetos en formato JSON [31]. Dado que el resultado obtenido por la ejecución de una tarea debe poder representarse en un formato estructurado, se ha decidido utilizar esta biblioteca para optimizar la representación de esta

información. La versión utilizada de la librería en el sistema es la 13.0.3.

2.2.10 Herramienta: Azure DevOps

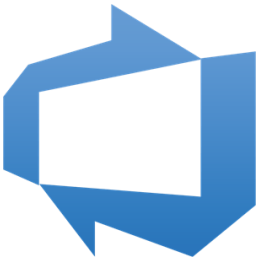

Figura 14. Logo de Azure DevOps

Azure DevOps [32] es una plataforma que proporciona múltiples servicios orientados al desarrollo de software. Sin embargo, se ha optado por utilizar como apoyo únicamente el servicio 'Azure Boards' con el objetivo de tener un tablero Kanban que facilite la gestión de las diferentes tareas y objetivos en el proyecto según se especifica en las metodologías ágiles.

2.2.11 Herramienta: StarUML

Figura 15. Logo de StarUML

StarUML [33] es una herramienta de modelado para el Lenguaje Unificado de Modelado (UML) [34]. Es útil para el proyecto porque permite realizar diagramas de clases [35] durante la fase de diseño de este, y así comprender mejor el funcionamiento del sistema que se está implementando. No obstante, es una herramienta extensible que proporciona una gran variedad de diagramas diferentes. La versión utilizada del programa es la 5.1.0.

2.2.12 Herramienta: Microsoft Visual Studio

Figura 16. Logo de Visual Studio

Visual Studio [36] es un entorno de desarrollo (IDE) empleado durante la realización del proyecto para la construcción del sistema. Es una herramienta que permite el desarrollo de una amplia variedad de aplicaciones, proporcionando facilidades al desarrollador que la utiliza. Entre las opciones que ofrece se encuentran las plantillas de proyecto, autocompletado de código, depuración y herramientas para pruebas. La versión utilizada del entorno de desarrollo es la de 2022 para la edición de la comunidad.

3

Metodología ágil

3.1 Descripción

Para la elaboración del TFG, se ha decidido seguir una metodología de desarrollo iterativa e incremental, que permita al proyecto adaptarse ante nuevos cambios en los requisitos de este. Este enfoque ágil [37] es ampliamente utilizado en el desarrollo de software porque acerca e implica más al usuario del producto en la construcción de la aplicación web durante el desarrollo de esta.

La idea principal de esta visión es realizar entregas de funcionalidades del producto de forma que, según una temporalización basada en periodos cortos de tiempo (sprints) [38], un equipo de desarrollo de software pueda gestionarse para alcanzar ciertos objetivos en cada intervalo temporal. Además, en cada periodo los usuarios pueden implicarse en cualquier fase para aportar retroalimentación del trabajo que se está realizando.

La implementación de la descripción que se ha realizado en este subcapítulo es lo que se conoce como marco de trabajo Scrum y, normalmente, la forma en la que se suele trabajar en cada periodo de tiempo suele consistir en una fase inicial de observación del estado del producto y requerimientos actuales, para continuar con la elaboración o cambios en el diseño. Seguidamente, se procede a la implementación inicial, o las modificaciones de esta, para

terminar con las pruebas y retrospección del periodo que permita validar con el equipo, y los usuarios, los hitos conseguidos durante la fase establecida.

3.2 Adaptación para el TFG

En el contexto académico del TFG, este ha sido adaptado para cubrir los requisitos temporales del mismo. Dada la naturaleza de este contexto, un sprint que pudiera extenderse en un periodo de tiempo más amplio se ha reducido para que cumpla con una duración de entre 1 y 2 semanas. Llevando esto a cabo, parte del tiempo puede ser empleado a la construcción del sistema, mientras que el tiempo restante es utilizado para discutir los logros del periodo y los cambios necesarios en el desarrollo de la aplicación.

Por otro lado, puesto que se trata de un trabajo individual que no se encuentra en un ámbito empresarial con un equipo, ciertos roles que se aplican en este marco de trabajo han sido asumidos por el estudiante y otros por el tutor. A continuación, se resumen algunos de estos roles:

- Scrum Master (estudiante). La función de la persona que tiene este rol es de llevar un seguimiento de la planificación del sprint, así como revisar este.
- Product Owner (estudiante). Este rol implica la responsabilidad de definir los requisitos del producto, así como su priorización y revisión de que se están cumpliendo los objetivos.
- Stakeholder (tutor). Se trata del rol que define a la persona que tiene algún interés en el producto. En el caso del proyecto, es un usuario que prueba el producto y ofrece una retroalimentación del desarrollo realizado.

3.3 Planificación

Como consecuencia de la adaptación temporal del TFG, se han establecido cuatro iteraciones de dos semanas cada una. Antes de comenzar a describirlas, se explica cómo ha sido estructurado el documento para seguir la planificación, con el propósito de facilitar la lectura de este.

En primer lugar, se han separado todos los requisitos en el siguiente capítulo para que estén recogidos en un solo sitio. El objetivo de la descripción de cada iteración es llevar un registro de qué se ha conseguido y qué ha cambiado para tener una visión del proyecto más completa desde el principio, ya que se conocen en primera instancia todos los objetivos de este.

En segundo lugar, las pruebas del producto en cada iteración han sido orientadas a ser pruebas de aceptación (UAT) [39], por la naturaleza del proyecto y lo que se quiere conseguir con la metodología que ha sido descrita. De esta forma, se reserva un capítulo final de plan de pruebas más orientado a la revisión final del producto.

A continuación, se definen las cuatro iteraciones que han sido propuestas en el marco de trabajo:

1. **Primera iteración**

   Al tratarse de la fase inicial del proyecto, tiene como objetivo principal el diseño e implementación de un sistema funcional con los requisitos más críticos. Para ello, primero se elabora un primer diseño de la aplicación que pueda reflejarse en la implementación, y después se inicia la configuración del entorno de desarrollo, así como la instalación de librerías y herramientas y el comienzo de la codificación.

   Entre los objetivos principales se encuentran el inicio de sesión en la aplicación web, la gestión de roles y la ejecución de tareas de al menos una tecnología.

2. **Segunda iteración**

   En esta fase se quiere mejorar el diseño inicial e incluir en el sistema la visualización del histórico de tareas, así como el resultado de la ejecución de estas.

3. **Tercera iteración**

   La tercera iteración busca comprobar la adaptabilidad del sistema, desarrollado en las iteraciones previas, ante la incorporación de una nueva tecnología automatizable. Por lo tanto, la aplicación web puede ofrecer una nueva tarea en la interfaz de usuario.

4. **Cuarta iteración**

   En esta última iteración, se analiza con el usuario la incorporación en la aplicación de algún sistema que le permita comunicarse con algún desarrollador de esta (sea por email o a través de un sistema de propuestas).

4

Análisis de requisitos

En este capítulo se analizan y detallan las características principales que debe tener el sistema que va a desarrollarse, por lo que se trata de una sección crítica de la presente memoria del TFG. Aquí se establecen las bases para la construcción del proyecto, identificando cada una de las necesidades que deben satisfacerse para alcanzar los objetivos planteados inicialmente.

En primer lugar, se identifican los requisitos funcionales [40] del sistema descrito que son las capacidades que este debe incluir. En el caso del proyecto actual, la posibilidad de ejecutar una tarea automatizada y visualizarla representa a alto nivel la capacidad principal que la aplicación ha de tener.

En segundo lugar, se describen los requisitos no funcionales que abordan aspectos que debe cumplir el sistema para aportarle mayor calidad al proyecto, pero no características cuyo detalle sea visible con la interacción de la aplicación y su objetivo principal. Entre estos se encuentran la seguridad, escalabilidad y tiempos de respuesta del sistema.

Por último, se especifican los casos de uso [41] que debe respetar el sistema, es decir, se realiza una descripción funcional más detallada de las interacciones del usuario final con el sistema para garantizar que durante las pruebas de aceptación con el mismo se consigan los objetivos planteados en el proyecto.

Se opta por representar cada uno de estos subcapítulos en formato de tabla para facilitar la retroalimentación con el usuario y su futura referencia en el índice recogido del documento.

4.1 Requisitos funcionales

Tabla 1. Requisitos funcionales de la primera iteración

ID del requisito	Descripción	Actores	Prioridad
RF1	Iniciar sesión en la aplicación web mediante credenciales de usuario previamente almacenadas por el sistema	Usuario, Sistema	Alta
RF2	Salir de la aplicación web	Usuario	Media
RF3	Diferenciar entre rol de usuario y desarrollador en el acceso de la aplicación	Sistema	Alta
RF4	Visualizar las tareas disponibles que ofrece el sistema para ser ejecutadas	Usuario	Alta
RF5	Ejecutar una tarea específica de firma automática de documentos PDF	Usuario	Alta
RF5.1	Seleccionar un conjunto de ficheros PDF para enviarlos al sistema	Usuario	Alta
RF6	Ejecutar una tarea genérica de obtención de datos web por HTTP	Usuario	Media
RF6.1	Configurar la tarea anterior mediante la escritura de una dirección web y una expresión XPath	Usuario	Media

Tabla 2. Requisitos funcionales de la segunda iteración

ID del requisito	Descripción	Actores	Prioridad
RF7	Visualizar histórico de tareas ejecutadas	Usuario	Alta

RF7.1	Mostrar información de la fecha de ejecución y su estado	Sistema	Alta
RF8	Filtrar por estado de la ejecución de la tarea	Usuario	Media
RF9	Visualizar los resultados obtenidos de la ejecución	Usuario	Alta
RF9.1	Descargar los documentos PDF firmados por el sistema	Usuario	Alta
RF9.2	Mostrar la información estructurada obtenida de la tarea genérica basada en HTTP	Sistema	Media

Tabla 3. Requisitos funcionales de la tercera iteración

ID del requisito	Descripción	Actores	Prioridad
RF10	Añadir al sistema la tarea específica de obtención de noticias recientes de la página web de la UMA por Selenium	Desarrollador	Alta
RF10.1	Ejecutar esta nueva tarea específica desde la aplicación web	Usuario	Alta
RF10.2	Configurar la tarea anterior mediante la escritura del número de noticias a adquirir	Usuario	Baja

Tabla 4. Requisitos funcionales de la cuarta iteración

ID del requisito	Descripción	Actores	Prioridad
RF11	Realizar una propuesta de automatización mediante correo electrónico	Usuario	Alta
RF11.1	Asignar la propuesta al desarrollador más desatendido	Sistema	Baja

RF12	Eliminar tareas ejecutadas del histórico	Usuario	Media

4.2 Requisitos no funcionales

Tabla 5. Requisitos no funcionales del sistema

ID del requisito	Descripción	Categoría	Prioridad
NFR1	El sistema debe ser intuitivo y fácil de usar	Usabilidad	Alta
NFR2	Debe proporcionarse un manual de usuario para entender mejor el uso de la aplicación	Usabilidad	Alta
NFR3	Las tareas de automatización deben cumplir con tiempos máximos de realización	Rendimiento	Media
NFR4	El sistema debe ser capaz de firmar un mínimo número de documentos PDF sin proporcionar inestabilidad a este	Rendimiento	Alta
NFR5	Las credenciales del usuario son almacenadas en la base de datos de forma encriptada	Seguridad	Alta
NFR6	La clave privada del certificado digital del usuario se encuentra almacenado en una ruta del servidor, así como su contraseña de forma encriptada en la base de datos del sistema	Seguridad	Alta
NFR7	La información obtenida por el resultado de ejecutar una tarea pertenece al usuario que la ejecuta	Seguridad	Alta
NFR8	El sistema debe ser diseñado de forma que incorporar una nueva tarea de	Escalabilidad	Alta

	automatización no cambie la arquitectura de este en gran medida		
NFR9	Posibilidad de incluir nuevas tecnologías en el futuro para tareas de automatización	Interoperabilidad	Media

4.3 Casos de uso

Tabla 6. Caso de uso 1 de la primera iteración

Campo	Descripción
ID del caso de uso	CU-001
ID del requisito funcional	RF1
Nombre	Iniciar sesión en la aplicación web
Actor	Usuario
Precondiciones	El usuario debe tener una cuenta registrada en el sistema.
Postcondiciones	El usuario queda autenticado y tiene acceso a las funcionalidades del sistema según su rol.
Flujo normal	1. El usuario accede a la página de inicio de sesión. 2. El usuario introduce su nombre de usuario y contraseña. 3. El usuario hace clic en el botón 'Iniciar sesión'. 4. El sistema valida las credenciales introducidas. 5. El usuario es redirigido a la página principal del sistema.
Flujo alternativo	4. a) Las credenciales introducidas son incorrectas y el sistema muestra un mensaje de error que indica 'Inténtelo de nuevo'.
Excepciones	• No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.
Prioridad	Alta
Frecuencia	Muy frecuente

Tabla 7. Caso de uso 2 de la primera iteración

Campo	Descripción
ID del caso de uso	CU-002
ID del requisito funcional	RF2
Nombre	Salir de la aplicación web
Actor	Usuario autenticado
Precondiciones	El usuario debe estar autenticado y tener una sesión activa en el sistema.
Postcondiciones	El usuario queda desautenticado y se cierra su sesión en el sistema.
Flujo normal	1. El usuario autenticado se encuentra en cualquier página de la aplicación. 2. El usuario hace clic en el botón 'Salir'. 3. El sistema cierra la sesión del usuario. 4. El usuario es redirigido a la página de inicio de sesión.
Flujo alternativo	Ninguno.
Excepciones	• No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.
Prioridad	Media
Frecuencia	Frecuente

Tabla 8. Caso de uso 3 de la primera iteración

Campo	Descripción
ID del caso de uso	CU-003
ID del requisito funcional	RF3
Nombre	Diferenciar entre rol de usuario básico y desarrollador en el acceso de la aplicación

Actor	Usuario, Desarrollador
Precondiciones	El usuario o desarrollador debe estar autenticado y tener una sesión activa en el sistema.
Postcondiciones	El sistema muestra un mensaje de bienvenida acorde al rol del actor que ha iniciado sesión.
Flujo normal	1. El usuario o desarrollador se autentica en el sistema. 2. El sistema verifica el rol asociado del actor autenticado. 3. El sistema muestra un mensaje de bienvenida acorde al rol identificado.
Flujo alternativo	1. a) Las credenciales introducidas son incorrectas y el sistema muestra un mensaje de error que indica 'Inténtelo de nuevo'.
Excepciones	• No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.
Prioridad	Alta
Frecuencia	Muy frecuente

Tabla 9. Caso de uso 4 de la primera iteración

Campo	Descripción
ID del caso de uso	CU-004
ID del requisito funcional	RF4
Nombre	Visualizar las tareas disponibles que ofrece el sistema
Actor	Usuario, Desarrollador
Precondiciones	El usuario o desarrollador debe estar autenticado y tener una sesión activa en el sistema.
Postcondiciones	Se muestra al usuario una lista de todas las tareas que están disponibles para ser ejecutadas.
Flujo normal	1. El usuario o desarrollador navega a la sección de tareas del sistema.

	2. El sistema recopila todas las tareas disponibles desde la base de datos. 3. El sistema muestra la lista de tareas disponibles para el usuario o desarrollador.
Flujo alternativo	3. a) Si no hay tareas disponibles, se muestra un mensaje informando de la situación.
Excepciones	• No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.
Prioridad	Alta
Frecuencia	Muy frecuente

Tabla 10. Caso de uso 5 de la primera iteración

Campo	**Descripción**
ID del caso de uso	CU-005
ID del requisito funcional	RF5
Nombre	Ejecutar una tarea específica de firma automática de documentos PDF
Actor	Usuario, Desarrollador
Precondiciones	El usuario debe estar autenticado y tener una sesión activa en el sistema. Además, debe haber visualizado las tareas disponibles y seleccionado la tarea específica de firma de documentos PDF.
Postcondiciones	Los documentos PDF seleccionados por el usuario son firmados automáticamente y se almacenan en el historial del usuario.
Flujo normal	1. El usuario selecciona la tarea específica de firma automática de documentos PDF desde la lista de tareas disponibles. 2. El sistema muestra una interfaz para que el usuario pueda seleccionar los documentos PDF a firmar. 3. El usuario selecciona los documentos PDF y confirma la acción.

	4. El sistema ejecuta la tarea de firma automática de los documentos seleccionados. 5. El sistema guarda un registro de la tarea ejecutada en el historial del usuario.
Flujo alternativo	3. a) Si el usuario cancela la acción, se retorna a la lista de tareas disponibles.
Excepciones	• Error en la firma de los documentos PDF. • No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.
Prioridad	Alta
Frecuencia	Frecuente

Tabla 11. Caso de uso 6 de la primera iteración

Campo	Descripción
ID del caso de uso	CU-006
ID del requisito funcional	RF6
Nombre	Ejecutar una tarea genérica de obtención de datos web por HTTP
Actor	Usuario, Desarrollador
Precondiciones	El usuario debe estar autenticado y tener una sesión activa en el sistema. Además, este se encuentra en la sección para ejecutar tareas.
Postcondiciones	La tarea genérica es ejecutada por el sistema y los resultados se almacenan en el historial del usuario.
Flujo normal	1. El usuario selecciona la tarea genérica de obtención de datos web por HTTP desde la lista de tareas disponibles. 2. El sistema muestra una interfaz para que el usuario pueda introducir la URL destino y la expresión XPath deseada. 3. El usuario completa los campos y pulsa 'Ejecutar'.

	4. El sistema hace una validación de la URL especificada y también del XPath indicado. La respuesta HTTP deber ser basada en XML. 5. El sistema ejecuta la tarea y guarda los resultados en el historial de usuario.
Flujo alternativo	3. a) Si el usuario cancela la acción, se retorna a la lista de tareas disponibles. 4. a) Si la URL o la expresión XPath son inválidas, se muestra un mensaje de error y se da la opción de ser corregidas.
Excepciones	• Error en la ejecución de la tarea. • No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.
Prioridad	Media
Frecuencia	Media

Tabla 12. Caso de uso 7 de la segunda iteración

Campo	Descripción
ID del caso de uso	CU-007
ID del requisito funcional	RF7
Nombre	Visualizar histórico de tareas ejecutadas
Actor	Usuario, Desarrollador
Precondiciones	El usuario debe estar autenticado y tener una sesión activa en el sistema.
Postcondiciones	El usuario ha visualizado su histórico de tareas ejecutadas.
Flujo normal	1. El usuario navega a la sección de histórico del sistema. 2. El sistema muestra una lista de tareas ejecutadas por el usuario con información del estado y fecha de ejecución de estas. 3. El usuario puede filtrar o navegar por las tareas para obtener más detalles.

	4. El usuario selecciona una tarea para ver más detalles. 5. El sistema muestra los detalles de la tarea seleccionada.
Flujo alternativo	2. a) Si no hay tareas ejecutadas en el histórico, el sistema muestra un mensaje indicando que el usuario aún no ha ejecutado ninguna.
Excepciones	• No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.
Prioridad	Alta
Frecuencia	Frecuente

Tabla 13. Caso de uso 8 de la segunda iteración

Campo	Descripción
ID del caso de uso	CU-008
ID del requisito funcional	RF8
Nombre	Filtrar por estado de ejecución de la tarea
Actor	Usuario, Desarrollador
Precondiciones	El usuario debe estar autenticado y tener una sesión activa en el sistema. Además, este se encuentra en la sección del histórico de tareas ejecutadas.
Postcondiciones	El histórico de tareas ejecutadas muestra sólo las tareas que corresponden al estado seleccionado por el usuario.
Flujo normal	1. El usuario selecciona un estado de la lista de filtros (completada, en proceso o fallida). 2. El sistema filtra las tareas del histórico y muestra sólo las que corresponden al estado seleccionado por el usuario.
Flujo alternativo	Ninguno.
Excepciones	• No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.

Prioridad	Media
Frecuencia	Media

Tabla 14. Caso de uso 9 de la segunda iteración

Campo	Descripción
ID del caso de uso	CU-009
ID del requisito funcional	RF9
Nombre	Visualizar los resultados obtenidos de la ejecución
Actor	Usuario, Desarrollador
Precondiciones	El usuario debe estar autenticado y tener una sesión activa en el sistema. Además, este debe haber ejecutado al menos una tarea en el sistema y encontrarse en la sección del histórico de tareas ejecutadas.
Postcondiciones	El usuario puede ver los resultados de la tarea ejecutada.
Flujo normal	1. El usuario selecciona una tarea para ver más detalles. 2. El sistema muestra los detalles de la tarea seleccionada según el tipo de resultado obtenido (documentos PDF o información estructurada).
Flujo alternativo	2. a) Si no se han generado resultados, el sistema muestra un mensaje informativo de la situación.
Excepciones	• No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.
Prioridad	Alta
Frecuencia	Frecuente

Tabla 15. Caso de uso 10 de la tercera iteración

Campo	Descripción
ID del caso de uso	CU-010

ID del requisito funcional	RF10
Nombre	Ejecutar una tarea específica de obtención de noticias recientes de la página web de la UMA
Actor	Usuario, Desarrollador
Precondiciones	El usuario debe estar autenticado y tener una sesión activa en el sistema. Además, la tarea debe existir en el sistema y el usuario encontrarse en la sección para ejecutar tareas.
Postcondiciones	La tarea específica es ejecutada por el sistema y los resultados se almacenan en el historial del usuario.
Flujo normal	1. El usuario selecciona la tarea específica de obtención de noticias recientes de la página web de la UMA. 2. El sistema muestra una interfaz para que el usuario pueda indicar un límite de noticias a adquirir por el sistema. 3. El usuario pulsa 'Ejecutar'. 4. El sistema ejecuta la tarea y guarda los resultados en el historial del usuario.
Flujo alternativo	3. a) Si el usuario cancela la acción, se retorna a la lista de tareas disponibles.
Excepciones	• Error en la ejecución de la tarea. • No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.
Prioridad	Alta
Frecuencia	Frecuente

Tabla 16. Caso de uso 11 de la cuarta iteración

Campo	Descripción
ID del caso de uso	CU-011

ID del requisito funcional	RF11
Nombre	Realizar una propuesta de automatización mediante correo electrónico
Actor	Usuario
Precondiciones	El usuario debe estar autenticado y tener una sesión activa en el sistema. Además, este debe encontrarse en la página principal de la aplicación. El rol del usuario debe ser básico.
Postcondiciones	Se ha enviado una propuesta de automatización al desarrollador más desatendido por correo electrónico.
Flujo normal	1. El usuario escribe su correo electrónico, un asunto y un cuerpo en los campos requeridos. 2. El usuario hace clic en 'Enviar propuesta'. 3. El sistema valida la información y envía la propuesta al desarrollador más desatendido.
Flujo alternativo	3. a) Si la información es inválida, se muestra un mensaje de error y se da la opción de corregirse.
Excepciones	• Error en el envío del correo electrónico. • No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.
Prioridad	Alta
Frecuencia	Media

Tabla 17. Caso de uso 12 de la cuarta iteración

Campo	Descripción
ID del caso de uso	CU-012
ID del requisito funcional	RF12
Nombre	Eliminar tareas ejecutadas del histórico
Actor	Usuario, Desarrollador

Precondiciones	El usuario debe estar autenticado y tener una sesión activa en el sistema. Además, este debe encontrarse en la sección del histórico de tareas ejecutadas.
Postcondiciones	La tarea seleccionada ha sido eliminada del histórico.
Flujo normal	1. El usuario selecciona la tarea que desea eliminar. 2. El usuario confirma la acción de eliminar la tarea. 3. El sistema elimina la tarea del histórico.
Flujo alternativo	2. a) Si el usuario cancela la acción, se retorna al histórico de tareas.
Excepciones	• No hay conexión con el servidor del sistema. • Error en la conexión del sistema con la base de datos.
Prioridad	Media
Frecuencia	Poco frecuente

5

Primera iteración

5.1 Diseño inicial

Este subcapítulo establece las bases arquitectónicas del sistema que se va a construir en el desarrollo del proyecto. En primer lugar, se presenta una breve introducción del modelo arquitectónico que sigue el sistema dentro del ámbito del desarrollo de software, para continuar con la presentación del modelo de datos de la aplicación; y finalizar, con una descripción de la interfaz de usuario.

5.1.1 Modelo cliente-servidor

El sistema general sigue un modelo cliente-servidor [42], ya que este permite separar la tecnología de la aplicación web en dos partes diferenciadas. Con este modelo se consigue gestionar las interacciones y el flujo de datos entre los usuarios de la aplicación y el sistema de tareas automatizables. De este modo, se facilita la escalabilidad del proyecto, así como su mantenibilidad en el futuro.

La comunicación entre cliente y servidor se realiza a través del protocolo de comunicación HTTP, ya mencionado con anterioridad, utilizando una API RESTful [43] para la interacción, e intercambio de información, entre las dos partes que componen el sistema.

A continuación, se especifica el rol que cumple cada parte en el sistema:

- **Cliente**: Dada la naturaleza escalable y modular del proyecto, con el propósito de hacer una aplicación web cuyos componentes de esta sean reutilizables, se ha tomado la decisión de utilizar la tecnología React. El lado del cliente está caracterizado por responsabilizarse de la interfaz de usuario de la aplicación, es decir, se encarga de diseñar las vistas que interactúan directamente con el usuario para mandarle información, posteriormente, al servidor.
- **Servidor**: En este lado, se ha optado por utilizar la tecnología ASP.NET porque ofrece un marco de trabajo para el desarrollo de aplicaciones web robustas y escalables. Se responsabiliza de procesar las solicitudes que llegan del lado del cliente, tales como lanzar tareas automatizables o interactuar con la base de datos del sistema, según los requisitos que han sido descritos.

Por último, se presenta un ejemplo de interacción de estas partes para este modelo, a partir del caso de uso que ha sido definido como 'CU-001. Iniciar sesión en la aplicación web', y con el propósito de entender cómo funciona el sistema expuesto:

1. La responsabilidad que toma el lado del cliente es la de enviar la información de las credenciales del usuario al lado del servidor, de forma encriptada por motivos de seguridad.
2. El lado del servidor, por otro lado, se encarga de validar las credenciales de usuario que ha recibido para dar acceso al cliente en la aplicación web.
3. Una vez el acceso ha sido concedido por el servidor, el cliente renderiza la página principal de la aplicación.

5.1.2 Diseño de base de datos

En esta sección se resuelve el modelo de información que debe respetar el sistema en la iteración que se encuentra, según los requisitos que han sido especificados.

Para representar el diseño de base de datos se utiliza un diagrama de entidad-relación (ER) [44], que en esta fase del proyecto se plantea así:

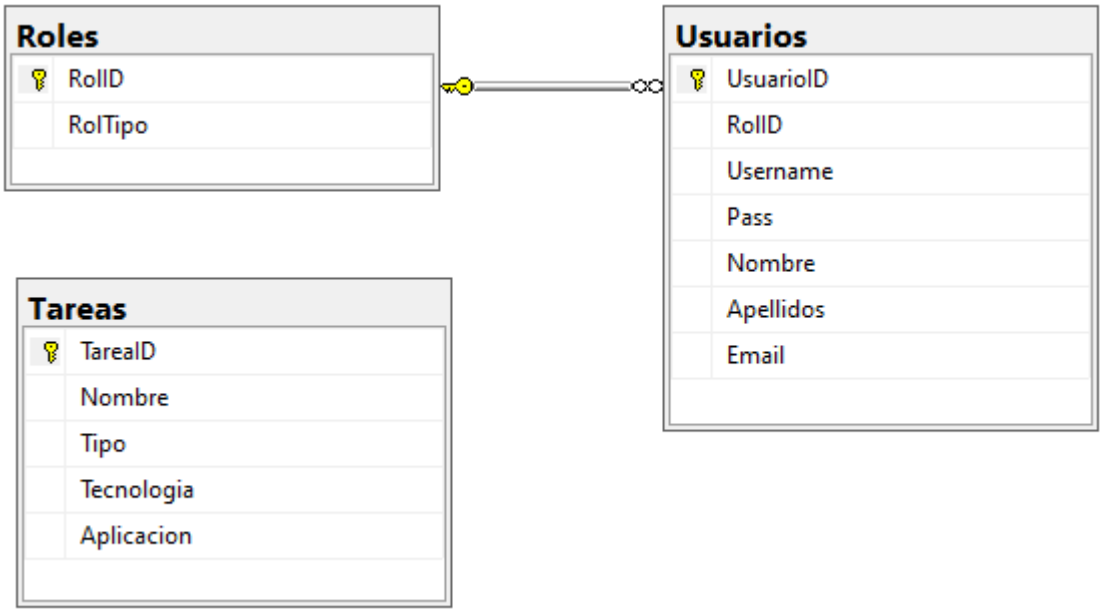

Figura 17. Diseño de base de datos de la primera iteración

Las entidades principales del sistema en esta fase son Tareas, Usuarios y Rol. En este modelo se observan ciertas restricciones:

- Los usuarios sólo pueden tener un rol. Los roles de la aplicación pueden asociarse a cualquier usuario. No hay jerarquía de roles, ni flexibilidad en permisos de usuario.
- Las tareas no se relacionan con ninguna entidad porque son independientes en este momento del diseño del sistema.
- No hay persistencia de los datos de las ejecuciones realizadas de las tareas hasta la segunda iteración.

Puesto que no se han definido requisitos de registro en la aplicación, y la gestión de los datos de las tareas se realiza por parte de los desarrolladores, se han creado datos iniciales para trabajar con las tablas. Durante la sección de la implementación inicial, se especifican los detalles técnicos.

Por último, resaltar que durante la implementación de la tarea automatizada de firmado de documentos PDF surge la necesidad de añadir dos columnas adicionales a la tabla de usuarios, para incluir la ruta del certificado digital y la contraseña de este.

5.1.3 Diagrama de clases

En este subcapítulo se muestra una aproximación más detallada de la aplicación, ya que un diagrama de clases es una representación gráfica a alto nivel del modelo de diseño de un sistema. Gracias a este diagrama, se puede verificar que el modelo cumple con los requisitos del proyecto, creando objetos que representen instancias de las clases reflejadas y estudiando si el sistema es robusto y escalable.

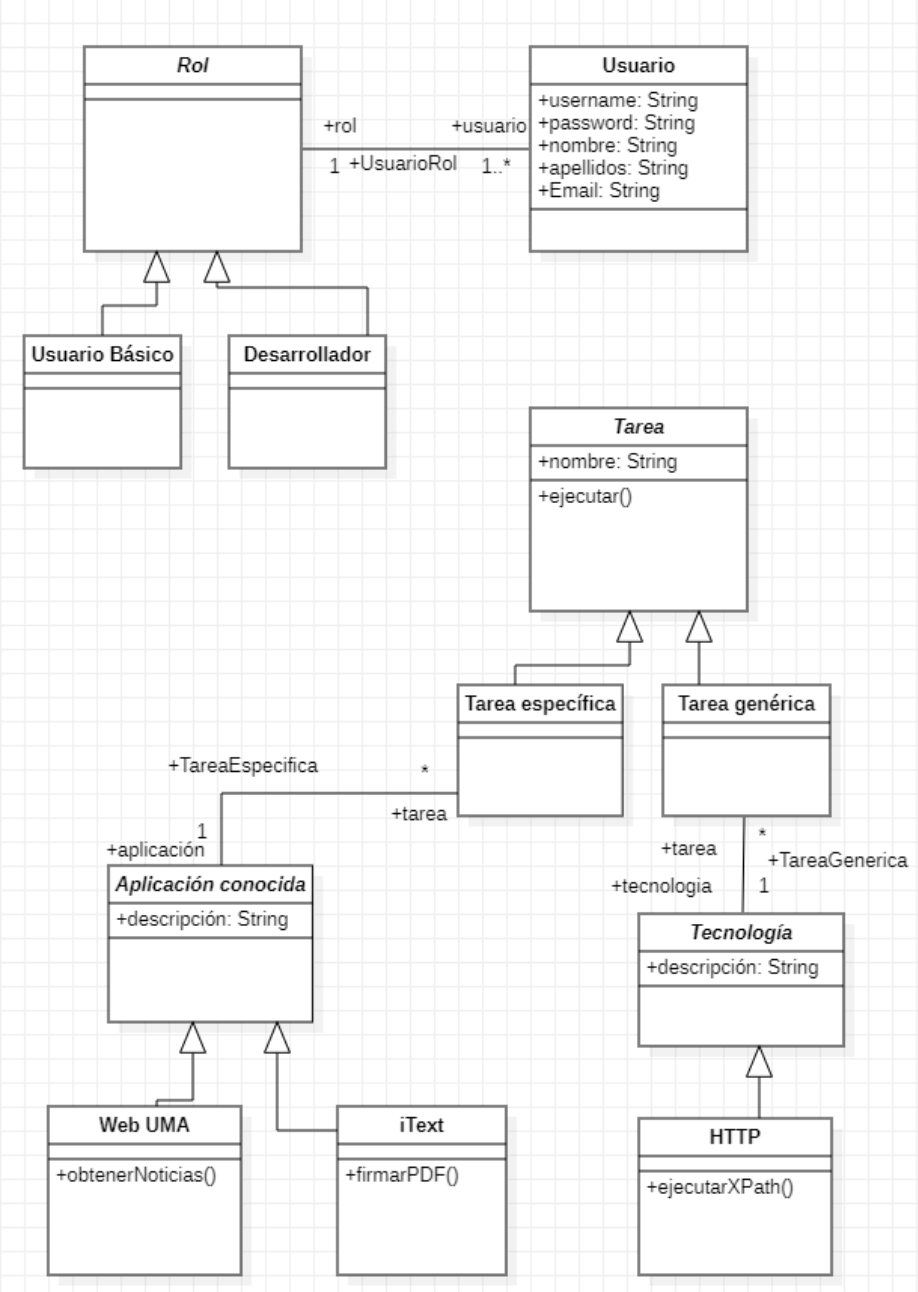

Figura 18. Diagrama de clases en la primera iteración

El anterior diagrama especifica las principales clases de la aplicación. Se observa cómo quiere comportarse el sistema ante la ejecución de tareas específicas y genéricas. El usuario no tiene visibilidad de las tareas que ejecuta, pero puede realizar la operación ejecutar si le dan acceso a la instancia en el sistema.

5.2 Implementación inicial

Esta sección se presenta como una demostración práctica del diseño que se ha realizado del sistema y los requisitos definidos durante la discusión de estos en las diferentes iteraciones del proyecto.

Dado este enfoque iterativo, se realiza en cada fase un análisis de las funcionalidades implementadas durante la codificación del sistema y, en caso de encontrarlos, los desafíos que se han observado.

5.2.1 Patrón de arquitectura: Modelo Vista Controlador

El patrón Modelo Vista Controlador (MVC) [45] es un patrón de arquitectura que permite descomponer el desarrollo de un sistema en tres partes diferenciadas: modelo, vista y controlador. La decisión que se ha tomado para seleccionar esta solución se alinea con los objetivos del proyecto de construir una aplicación modular y escalable, además de que es soportado de forma inherente en la tecnología ASP.NET.

A continuación, se realiza una descripción de la separación de responsabilidades existentes en el proyecto:

- **Modelo**. Componente de la aplicación web que se encarga de la lógica de negocio y el acceso a los datos, es decir, maneja todas las interacciones con la base de datos y define las reglas y funciones principales que el sistema tiene. En este proyecto, esta responsabilidad pertenece al servidor, por ejemplo, mediante la clase de tareas, así como sus clases asociadas. Para la construcción del modelo e interacción con la base de datos, se utiliza Entity Framework [46] en .NET.
- **Vista**. Componente de la aplicación web que se encarga de representar la interfaz de usuario del sistema, además de la presentación y visualización de los datos que

devuelve el servidor. En este proyecto, la responsabilidad le pertenece a cada uno de los componentes que son diseñados con la tecnología de React.

- **Controlador**. Componente de la aplicación web que actúa como intermediario entre el modelo y la vista, es decir, recibe las interacciones del usuario con la interfaz de usuario y las procesa para un fin. Se trata de la parte de la aplicación que puede realizar acciones directas sobre el modelo, como actualizarlo o modificarlo. En este proyecto, la responsabilidad le pertenece al servidor, el cual recibe las comunicaciones entrantes HTTP, interactúa con el modelo y devuelve una respuesta al cliente en un formato que es representado en la vista.

5.2.2 Configuración del entorno de desarrollo

Durante la redacción de esta memoria, se han mencionado algunas decisiones técnicas para la selección de las tecnologías a utilizar, como ASP.NET en el lado del servidor, o React en el lado del cliente. Adicionalmente, se especifican algunas de las configuraciones que el proyecto incluye para las tecnologías y herramientas seleccionadas:

- **Instalación del kit de desarrollo (SDK) de .NET**. Es gestionado automáticamente por la herramienta de Visual Studio, que también se ha instalado.
- **Instalación de las dependencias del proyecto**. Es gestionado también por el gestor de paquetes NuGet desde Visual Studio.
- **Instalación del motor SQL Server para la base de datos**.
- **Instalación de 'Node.js' para React.**

Por otro lado, dado que el sistema sigue el patrón MVC, la estructura de carpetas del proyecto se organiza de forma que se aprecie cada una de las tres partes que se definen en el patrón.

5.2.3 Backend de la aplicación

Se desarrollan los componentes principales del sistema, generando el modelo desde la base de datos, en el proyecto de la aplicación. El controlador principal es la clase Tareas.

Por otro lado, las tareas de automatización especificadas para el sistema en los requisitos del producto de esta fase son implementadas sobre las clases de tecnología y aplicaciones conocidas.

5.2.4 Frontal web

Se crea la vista inicial del sistema, que se renderiza con la interfaz para el inicio de sesión del usuario. Además, se crea la vista para la página principal de la aplicación web que contiene los componentes React para interactuar con las tareas disponibles que pueden ejecutarse.

5.2.5 Base de datos

En esta sección se muestra la creación, en lenguaje SQL, de la creación de la base de datos:

```
CREATE DATABASE AutomateSolution;
GO

USE AutomateSolution;
GO

CREATE TABLE Roles (
    RolID INT PRIMARY KEY IDENTITY(1,1),
    RolTipo NVARCHAR(50)      -- basico, desarrollador
);
GO

CREATE TABLE Usuarios (
    UsuarioID INT PRIMARY KEY IDENTITY(1,1),
    RolID INT FOREIGN KEY REFERENCES Roles(RolID),
    Username NVARCHAR(50),
    Pass NVARCHAR(50),
    Nombre NVARCHAR(100),
    Apellidos NVARCHAR(100),
    Email NVARCHAR(100)
);
GO

CREATE TABLE Tareas (
    TareaID INT PRIMARY KEY IDENTITY(1,1),
    Nombre NVARCHAR(300),
    Tipo NVARCHAR(100),       -- especifica o generica
    Tecnologia NVARCHAR(100), -- selenium, jab, ...
    Aplicacion NVARCHAR(100)  -- uma, autofirma, ...
);
GO
```

Figura 19. Código de creación de la base de datos en la primera iteración

Y la incorporación de algunos datos en esta:

```
INSERT INTO [Roles] ([RolTipo])
VALUES ('Basico')
INSERT INTO [Roles] ([RolTipo])
VALUES ('Desarrollador')

INSERT INTO [Usuarios] ([RolID], [Username], [Pass], [Nombre], [Apellidos], [Email])
VALUES (2, 'manuel', HASHBYTES('SHA2_256', 'manuel'), 'Manuel', 'Esquivel', 'memqs2s@uma.es')
INSERT INTO [Usuarios] ([RolID], [Username], [Pass], [Nombre], [Apellidos], [Email])
VALUES (1, 'jose luis', HASHBYTES('SHA2_256', 'jose'), 'Jose', 'Luis', 'jlpastrana@uma.es')

INSERT INTO [Tareas] ([Nombre], [Tipo], [Tecnologia], [Aplicacion])
VALUES ('Firma de PDFs', 'Especifica', '.NET', 'iText')
INSERT INTO [Tareas] ([Nombre], [Tipo], [Tecnologia], [Aplicacion])
VALUES ('Obtencion datos por HTTP', 'Generica', '.NET', 'RestSharp')
```

Figura 20. Código de actualización de la base de datos en la primera iteración

La generación del modelo con Entity Framework y SQL Server se realiza a través del siguiente comando:

Tabla 18. Generación del modelo con Entity Framework

Scaffold-DbContext "Server=(local); DataBase=AutomateSolution; Integrated Security=true; TrustServerCertificate=true" Microsoft.EntityFrameworkCore.SqlServer -OutPutDir Models

5.3 Pruebas de aceptación

Este tipo de pruebas consisten en validar con el usuario, trabajando de forma activa con este sobre la aplicación web, que los requisitos funcionales especificados en la iteración cumplan con los objetivos del proyecto. Como se venía haciendo, se ha representado en formato tabla las pruebas realizadas, indicándose en la columna 'Resultado' la validación final por parte del usuario con 'OK', en caso de haber sido una prueba exitosa; y 'NOK', en el caso contrario.

5.3.1 Casos de prueba

Tabla 19. Casos de prueba de la primera iteración

ID de la prueba	Caso de uso asociado	Descripción	Resultado
UAT-1.1	CU-001	El usuario introduce sus credenciales en la aplicación y observa que accede a la página principal. Además, se comete una equivocación durante la introducción de la contraseña y el usuario comprueba que aparece un mensaje informativo.	OK
UAT-1.2	CU-003	El usuario comprueba que el sistema le reconoce en la página principal de la aplicación con su rol mediante un mensaje informativo.	OK
UAT-1.3	CU-004	El usuario accede a la lista de tareas disponibles del sistema y le aparecen resultados.	OK
UAT-1.4	CU-005	El usuario selecciona una tarea de firma de documentos PDF, después realiza clic en la subida de los ficheros y, por último, ejecuta la tarea. Se espera que el sistema haya realizado la firma y el usuario lo comprueba con un desarrollador.	OK
UAT-1.5	CU-006	El usuario selecciona una tarea de obtención de datos por HTTP, después escribe la URL y XPath asociados a esta para terminar ejecutando la tarea. Se espera que el sistema haya realizado la obtención y el usuario lo comprueba con un desarrollador. Además, se comete una equivocación en el formato de la URL introducida y el usuario comprueba que aparece un mensaje informativo.	OK

5.4 Retrospectiva

En este capítulo, se analiza si los objetivos de la iteración se han conseguido, así como los puntos de mejora para la siguiente iteración.

5.4.1 Hitos conseguidos

La primera iteración planificaba el desarrollo funcional del proyecto, con sus objetivos más críticos: inicio de sesión en la web, ver las tareas disponibles para ejecutar y diferenciar entre rol de usuario y de desarrollador.

Se han logrado cada uno de los objetivos generales de la iteración, probando con el usuario los casos de uso de esta con éxito.

5.4.2 Áreas de mejora

El usuario necesita retroalimentación de la tarea que ha ejecutado. Por lo tanto, se incorpora al desarrollo de la segunda iteración la necesidad de un historial de ejecuciones y resultados.

6

Segunda iteración

6.1 Diseño mejorado

6.1.1 Actualización de la base de datos

Se actualiza el diseño de la base de datos:

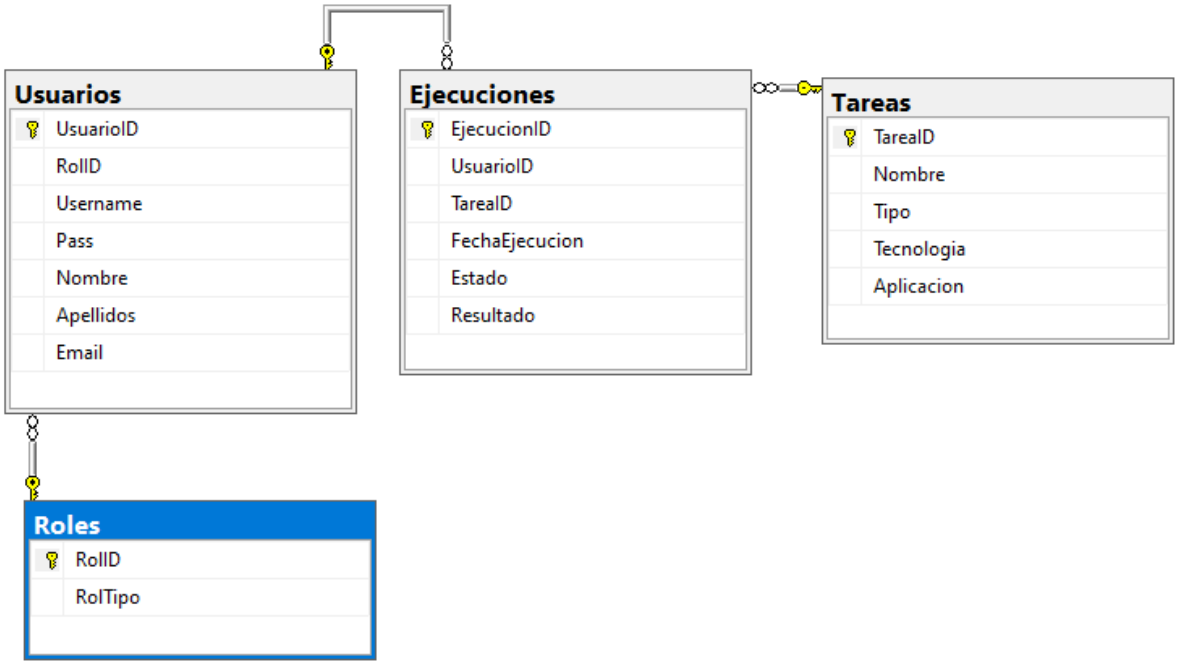

Figura 21. Actualización diseño de base de datos en la segunda iteración

En este diseño actualizado se incorpora la entidad Ejecuciones, que es muy importante en el proyecto porque después el usuario, directa o indirectamente, trabaja con los registros de esta. En esta tabla se almacena qué usuario ha ejecutado alguna tarea, a qué hora, cuál ha sido el estado de la ejecución y, la información más importante, qué resultados ha obtenido de esta ejecución.

6.1.2 Modificación del diagrama de clases

Se modifica el diagrama de clases para soportar el histórico de tareas:

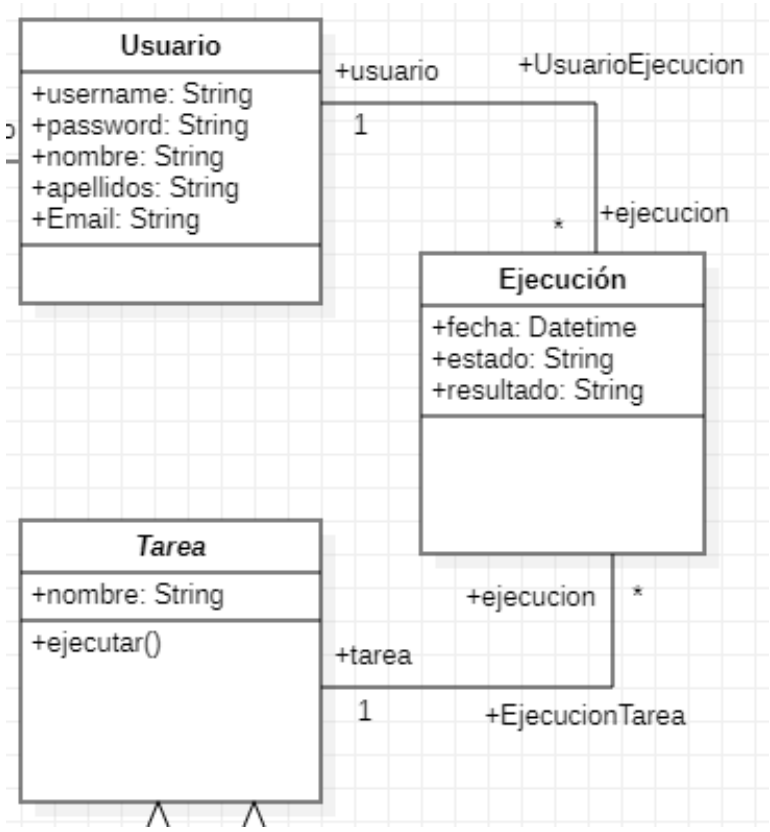

Figura 22. Diagrama de clases modificado en la segunda iteración

6.2 Implementación

6.2.1 Backend de la aplicación

Una nueva clase aparece en el modelo de la aplicación para representar las ejecuciones del usuario y, en consecuencia, se crea un controlador. En esta fase, se toman decisiones de

diseño para realizar una correcta persistencia de los datos al ejecutar las tareas, así como buenas prácticas en el uso de las clases de la aplicación.

6.2.2 Frontal web

Se crean dos nuevas vistas, una para listar el historial de tareas ejecutadas del usuario; y otra, para mostrar los resultados obtenidos por tarea ejecutada desde este historial.

6.2.3 Base de datos

Se muestra la codificación de la creación de la tabla 'Ejecuciones':

```
CREATE TABLE Ejecuciones (
    EjecucionID INT PRIMARY KEY IDENTITY(1,1),
    UsuarioID INT FOREIGN KEY REFERENCES Usuarios(UsuarioID),
    TareaID INT FOREIGN KEY REFERENCES Tareas(TareaID),
    FechaEjecucion DATETIME,
    Estado NVARCHAR(20),      -- pendiente, completada, fallida
    Resultado NVARCHAR(MAX)   -- json
);
GO
```

Figura 23. Código de creación de la tabla ejecuciones en la segunda iteración

6.3 Pruebas de aceptación

6.3.1 Casos de prueba

Tabla 20. Casos de prueba de la segunda iteración

ID de la prueba	Caso de uso asociado	Descripción	Resultado
UAT-2.1	CU-007	El usuario accede al historial de tareas ejecutadas de la aplicación y observa que no tiene ningún resultado. Tras esto, lanza una tarea y comprueba que le aparece un único resultado.	OK
UAT-2.2	CU-008	El usuario filtra por cada uno de los estados diferentes que puede tener una tarea	OK

		ejecutada en el historial de tareas. Después, comprueba que la aplicación muestra las tareas según el filtro que ha especificado.	
UAT-2.3	CU-009	El usuario selecciona alguna tarea ejecutada y comprueba que aparecen los resultados obtenidos de la ejecución.	OK

6.4 Retrospectiva

6.4.1 Hitos conseguidos

La segunda iteración se planificaba para mejorar el diseño inicial y, gracias a que el usuario ha realizado pruebas en la aplicación durante la primera iteración, ha observado la necesidad de que el sistema sea transparente con las ejecuciones que ha realizado y los resultados que estas han conseguido.

Por lo tanto, el historial de ejecuciones y resultados se ha logrado durante esta fase del proyecto y se ha probado con el usuario cada caso de uso con éxito.

6.4.2 Áreas de mejora

El usuario propone una nueva tarea de automatización de obtención de resultados de noticias en la página web de la UMA. Por lo tanto, se decide documentar la decisión para ejecutar el desarrollo en la tercera iteración.

7

Tercera iteración

7.1 Diseño adaptado

7.1.1 Cambios de los requisitos funcionales

El usuario necesita añadir una nueva tarea específica en la aplicación, pero esta debe ser implementada por un desarrollador. Actualmente, añadir una nueva tarea al sistema implica incluir una nueva funcionalidad en la implementación del producto. Sin embargo, es necesario que previamente se haga un registro en la información del sistema para incluir esta nueva tarea de 'Obtención de noticias en la web de la UMA'. Se detalla esta tarea técnica durante la sección de implementación.

7.1.2 Modificaciones en el diseño

El diseño de la base de datos no se ha visto modificado, pero el diagrama de clases presentaría la clase 'Web UMA' heredando de 'Tarea específica' como se ve en la primera iteración.

7.2 Implementación

7.2.1 Backend de la aplicación

Esta nueva tarea específica se codifica en la aplicación y se realizan los cambios necesarios para que interactúe con el resto de los componentes del sistema.

7.2.2 Frontal web

No se realizan cambios significativos en las vistas del sistema, pero se mejoran algunos de los componentes de React.

7.2.3 Base de datos

Se añade a la base de datos la nueva tarea 'Obtención noticias web UMA':

```
INSERT INTO [Tareas] ([Nombre], [Tipo], [Tecnologia], [Aplicacion])
VALUES ('Obtencion noticias web UMA', 'Especifica', 'Selenium', 'UMA')
```

Figura 24. Código de adición de nueva tarea específica en la base de datos

7.3 Pruebas de aceptación

7.3.1 Casos de prueba

Tabla 21. Casos de prueba de la tercera iteración

ID de la prueba	Caso de uso asociado	Descripción	Resultado
UAT-3.1	CU-010	El usuario selecciona una tarea de obtención de noticias recientes de la web de la UMA, después escribe el límite de noticias a recuperar y, por último, ejecuta la tarea. Se espera que el sistema haya obtenido las noticias y el usuario lo comprueba en el historial de tareas ejecutadas.	OK

7.4 Retrospectiva

7.4.1 Hitos conseguidos

La tercera iteración se planificaba para comprobar la adaptabilidad del sistema ante un nuevo requisito. En este caso, el requisito planteado por el usuario ha sido obtener las noticias más recientes de la página web de la UMA y ha sido desarrollado con éxito, gracias al diseño inicial del sistema.

Por lo tanto, la construcción de una nueva tarea específica en el sistema se ha llevado a cabo y se ha probado con el usuario el caso de uso principal con éxito.

7.4.2 Áreas de mejora

El usuario indica que el sistema debería tener algún sistema mínimo de comunicación que facilitase la proposición de nuevas automatizaciones al sistema, ya que ahora estamos en fase de desarrollo del producto y ha sido propuesto durante esta. Sin embargo, en un entorno productivo de la aplicación es necesario este requisito para que la aplicación sea escalable en el tiempo. Por lo tanto, esta nueva funcionalidad se añade a la cuarta iteración.

Por otro lado, surge la necesidad de eliminar tareas ejecutadas del histórico en función de la necesidad del usuario.

8

Cuarta iteración

8.1 Diseño actualizado

8.1.1 Identificación y priorización de nuevos requisitos

El usuario plantea la necesidad de incluir una funcionalidad en el sistema para comunicación, además de tener la posibilidad de eliminar tareas del histórico. Se prioriza lo segundo y no se aprecian cambios significativos en el diseño.

8.1.2 Actualización del diseño

No hay cambios en la base de datos para la nueva funcionalidad, pero se discute crear una entidad 'Propuestas' en esta. Eliminar una tarea del histórico es una operación del usuario en el diagrama de clases que elimina la relación con la ejecución y su tarea.

8.2 Implementación

8.2.1 Backend de la aplicación

La nueva funcionalidad para comunicarse con desarrolladores opera principalmente en el frontend de la aplicación.

8.2.2 Frontal web

Se crea un nuevo componente en la vista principal con un formulario para el envío del correo electrónico. Además, se crea otro componente similar para la eliminación de una tarea del histórico de ejecuciones.

8.2.3 Base de datos

No se realizan operaciones en el sistema gestor de base de datos.

8.3 Pruebas de aceptación

8.3.1 Casos de prueba

Tabla 22. Casos de prueba de la cuarta iteración

ID de la prueba	Caso de uso asociado	Descripción	Resultado
UAT-4.1	CU-011	En la página principal de la aplicación, el usuario escribe los campos necesarios para enviar un correo electrónico. El sistema envía la propuesta y el usuario observa confirmación en la página. Además, se comete una equivocación durante la introducción de los datos y el usuario comprueba que aparece un mensaje informativo.	OK
UAT-4.2	CU-012	En la página del historial de tareas, el usuario selecciona una tarea para ser eliminada y confirma la acción. El usuario comprueba que la tarea desaparece de la vista de la aplicación.	OK

8.4 Retrospectiva

8.4.1 Hitos conseguidos

La cuarta iteración se planificaba para mejorar el sistema con la mínima incorporación de una herramienta comunicativa con los desarrolladores del producto para facilitar la adición de nuevas tareas automatizables en el tiempo.

Dado que se ha logrado llevar a cabo con éxito este nuevo requerimiento, la cuarta iteración cumple con los objetivos del proyecto y se prueba con el usuario los últimos casos de uso.

8.4.2 Áreas de mejora

A partir de este punto, queda cerrado el trabajo del desarrollo de software, pero durante el capítulo de conclusiones y líneas futuras se realiza una reflexión sobre las áreas en las que el proyecto puede mejorar.

9

Plan de pruebas

En este capítulo se ha querido dar un enfoque ágil también a las pruebas mediante la elaboración de un plan predefinido durante el desarrollo del proyecto. Esta visión pretende lograr que las pruebas sobre las funcionalidades descritas del sistema, en el análisis de requisitos realizado con los usuarios en cada iteración, queden bien documentadas para asegurar un mayor éxito cuando se realicen sobre la implementación desarrollada. Por lo tanto, se presta especial atención al flujo normal que debe seguir el sistema.

Esta metodología que se ha llevado se acerca al proceso de desarrollo software 'Test-Driven Development' (TDD) [47], ya que este consiste en realizar pruebas sobre los requisitos de un proyecto antes, incluso, de la implementación de estos. Este estudio tiene la interesante particularidad de que, en un principio, se asume que todas las pruebas no tendrán un resultado exitoso hasta que la funcionalidad esté terminada y cumpla con los requisitos establecidos que validen el funcionamiento final del sistema.

Del mismo modo en que se ha realizado el capítulo de análisis de requisitos y las pruebas de aceptación, se opta por presentar en formato tabla cada subcapítulo con objeto de facilitar la retroalimentación con el usuario y su futura referencia en el índice recogido del documento.

9.1 Pruebas unitarias

Las pruebas unitarias [48] en un desarrollo de software son esenciales para garantizar que los componentes principales en el diseño de un sistema (clases, métodos, funcionalidades) cumplan con los requisitos especificados para el mismo, así como las posibles excepciones que puedan darse.

Entre los objetivos principales de estas pruebas, se encuentran verificar que las unidades de código funcionan como se espera para identificar errores en el diseño del sistema en una etapa temprana del desarrollo de este, y facilitar la refactorización futura del diseño.

A continuación, se presenta el mínimo lote de pruebas que el sistema debe satisfacer para los requisitos descritos en el proyecto.

Tabla 23. Pruebas unitarias del sistema

ID de la prueba	Requisito asociado	Nombre	Objetivo	Entrada	Salida esperada
PU-001	RF1	Test_ExitoLogin	Probar inicio de sesión exitoso.	Credenciales de usuario	Usuario autenticado
PU-002	RF4	Test_ListarTareasTecnologia	Probar obtención de tareas del sistema	Tecnología de automatización	Tareas de automatización listadas
PU-003	RF5	Test_ListarHistorialEjecuciones	Probar obtención de tareas ejecutadas	Usuario cualquiera	Tareas ejecutadas listadas del usuario
PU-004	RF7	Test_EjecutarTareaFirmaPDF	Probar el firmado de	Ruta local de documentos PDF válidas	Documentos PDF firmados

			documentos PDF		
PU-005	RF10	Test_EjecutarTareaNoticiasUMA	Probar obtención de noticias de la UMA	Límite de obtención de noticias	Noticias web obtenidas

Entre las herramientas ofrecidas por ASP.NET para pruebas en el backend se encuentra la herramienta Xunit [49]. Mientras que en el caso del frontend, React ofrece el marco de trabajo Jest [50].

9.2 Pruebas de integración

Las pruebas de integración [51] en un desarrollo de software son esenciales para garantizar la cohesión entre los componentes principales de un sistema, es decir, que se encuentren correctamente integrados entre ellos. Por lo tanto, permite verificar que la interacción entre los módulos del desarrollo de software funcione como se espera.

A continuación, se presenta el mínimo lote de pruebas que el sistema debe satisfacer para los requisitos descritos en el proyecto.

Tabla 24. Pruebas de integración del sistema

ID de la prueba	Requisito asociado	Nombre	Objetivo	Entrada	Salida esperada
PI-001	RF1, RF4	Test_LoginListarTareas	Probar la visualización de tareas tras iniciar sesión	Credenciales de usuario	Lista de tareas disponibles mostrada
PI-002	RF4, RF5	Test_ListarTareasEjecutarTareaFirmaPDF	Probar la ejecución de la tarea tras	Ruta local de documentos PDF válidas	Documentos PDF firmados

			mostrar la lista de tareas disponibles		
PI-003	RF10.1, RF7	Test_EjecutarTareaNoticiasUMAListarHistorialEjecuciones	Probar la visualización de la tarea ejecutada tras la ejecución de esta	Límite de obtención de noticias	La tarea ejecutada aparece en el histórico
PI-004	RF10.1, RF9	Test_EjecutarTareaNoticiasUMAListarResultados	Probar la obtención de noticias de la UMA tras la ejecución de la tarea	Límite de obtención de noticias	Las noticias obtenidas aparecen en los resultados de la tarea en el histórico

Las pruebas de integración también pueden ser realizadas con las herramientas Xunit y Jest, respectivamente.

9.3 Pruebas de carga

Las pruebas de carga [52] en un desarrollo de software son esenciales para garantizar que el sistema es estable ante la transacción simultánea de una elevada carga de datos. Por lo tanto, permite examinar el comportamiento del sistema, y extraer estadísticas de los resultados de ejecutar las pruebas.

A continuación, se presenta el mínimo lote de pruebas que el sistema debe satisfacer para los requisitos descritos en el proyecto.

Tabla 25. Pruebas de carga del sistema

ID de la prueba	Requisito asociado	Nombre	Objetivo	Entrada	Salida esperada
PC-001	NFR3	Test_CargaTiempsEjecionUnaTarea	Simular la ejecución de una tarea específica con tiempo máximo de 1 minuto	Entrada de la tarea específica	Tiempo de respuesta menor a 1 minuto con resultado exitoso
PC-002	NFR4	Test_CargaSistemaFirmadoPDFs	Simular la ejecución de una tarea de firmado de 10 documentos PDF	Ruta local de documentos PDF válidas	Los tiempos de respuesta del sistema son menores a 5 segundos durante la ejecución de la tarea
PC-003	RF5, RF6, RF10	Test_CargaSistemaEjecucionUnaTarea	Simular la ejecución de 2 tareas en paralelo	Entrada de la tarea	Los tiempos de respuesta del sistema son menores a 5 segundos durante la ejecución de las tareas

Las pruebas de carga se han realizado mediante acciones manuales en la aplicación web para observar el comportamiento de esta. Sin embargo, existen herramientas que pueden ser usadas con este objetivo:

- En el caso de ASP.NET, se puede utilizar Apache JMeter [53] para realizar pruebas de estrés o rendimiento. Es muy popular entre los desarrolladores, ya que permite probar el comportamiento de un sistema sometido a interacciones de un elevado número de usuarios concurrentes.
- En el caso de React, Jest ofrece la posibilidad de ejecutar pruebas de rendimiento sobre las pruebas unitarias especificadas en la aplicación.

10

Conclusiones y líneas futuras

10.1 Conclusiones

El proyecto que se presenta en esta memoria ha permitido establecer un buen punto de partida para la construcción de una plataforma que tiene el objetivo de facilitar a los usuarios finales un lugar donde hacer uso de la automatización para mejorar su productividad. Dada la ambición de este, se han analizado algunas de las primeras opciones que presentaría la herramienta para que pueda verse el potencial de esta, ya que uno de los propósitos finales del proyecto es que sea una plataforma escalable basada en scripting [54]. Esta última afirmación presenta una filosofía de trabajo en la que se busca conseguir una base basada en componentes de tareas individuales automatizables con el objetivo también de reducir la construcción de nuevas aplicaciones desde el principio con lo que ello implica: diseño de una nueva base de datos, modelado de un nuevo sistema, etcétera, ya que en muchas ocasiones una aplicación que no ha sido diseñada para ser escalable, y que presenta un objetivo demasiado específico, puede llegar a verse simplificada en una única tarea que resuelva el problema que esta plantea.

En general, el desarrollo de este proyecto ha cumplido con los objetivos iniciales. Durante el mismo, han aparecido diferentes retos y dificultades que han conseguido resolverse a tiempo y han servido como toma de contacto para entender el potencial que presentaría el continuo trabajo en la construcción de esta ambiciosa plataforma.

Por último, el aprendizaje de este proyecto ha sido una experiencia enriquecedora. En primer lugar, se ha tratado de llevar a cabo una metodología que se acerque a un entorno real de desarrollo de software y esto implica el estudio de las diferentes partes que componen el desarrollo cíclico de las iteraciones del proyecto. Por otro lado, el análisis y posterior uso de herramientas y tecnologías desconocidas ha permitido probar nuevas formas de implementar soluciones dentro del ámbito de la informática.

10.2 Líneas futuras

Este subcapítulo tiene especial interés para la plataforma presentada, ya que esta no quiere ser una herramienta cerrada porque busca ser escalable y reutilizable en relación con la incorporación de nuevas tecnologías al mismo y tareas específicas o genéricas que necesiten ser automatizadas.

Cabe destacar un reto que ha sido analizado durante el desarrollo del proyecto para ser incorporado como tarea específica en el mismo: el análisis de la tecnología Java Access Bridge (JAB) [55] para el firmado de documentos PDF en la aplicación Java 'Autofirma' [56]. Aunque esta última se describe como una tecnología que permite que aplicaciones Java de escritorio sean accesibles para usuarios con discapacidades, funciona como un puente entre estas aplicaciones Java y otras tecnologías de asistencia en sistemas operativos como Windows.

Además de la incorporación de nuevas tareas o tecnologías al proyecto, pueden realizarse ciertas mejoras en otras áreas que son también importantes:

1. **Mejoras del histórico de tareas**. Implementación de una funcionalidad para exportar los datos registrados en diferentes formatos como una hoja de cálculo o un documento PDF.

2. **Garantizar el rendimiento y concurrencia de las tareas en el sistema**. Actualmente, el sistema no controla la carga de trabajo de las tareas ejecutadas sobre la máquina que se ejecutan. Es una tarea importante que se realice una paralelización controlada de las tareas en ejecución, por ejemplo, limitar la apertura de navegadores web con Selenium a la vez. Además, se puede plantear la incorporación de un sistema de ejecuciones en diferentes máquinas de un mismo ecosistema de automatización.
3. **Mejoras en la comunicación entre usuarios**. Alineado con el objetivo del proyecto de ser una herramienta escalable, puede incluirse un sistema completo de comunicación en el sistema basado en mensajería o foro interno.
4. **Incorporación de un sistema de propuestas más complejo**. Con el objetivo de hacer una plataforma más colaborativa entre usuarios y desarrolladores, se puede incluir al sistema actual un sistema de propuestas basado en la aprobación y rechazo de estas.
5. **Optimizar la interfaz y experiencia del usuario**. Este proyecto ha sido desarrollado priorizando la funcionalidad crítica del sistema, pero la experiencia de usuario de este puede verse mejorada en una línea futura.
6. **Incorporación de algoritmos de Inteligencia Artificial (IA)**. Una línea también podría buscar el objetivo de realizar tareas automatizables con una base de inteligencia artificial. Por ejemplo, que un algoritmo buscara patrones en el comportamiento del usuario para observar futuras tareas susceptibles de ser automatizadas.
7. **Mejoras en la seguridad del sistema**. Dada la búsqueda funcional de los requisitos del proyecto la seguridad es básica, pero puede verse mejorada realizando una mejor gestión y análisis de los datos que se procesan cuando un usuario quiere ejecutar una tarea en el sistema.
8. **Despliegue en la nube de la aplicación**. La aplicación se encuentra desarrollada en un entorno local para visualizar de forma más eficiente la interacción del sistema con las tecnologías automatizadas. Sin embargo, una línea puede abarcar el despliegue de la herramienta en un entorno de nube para facilitar el acceso remoto.

Bibliografía

[1] W. van der Aalst, M. Bichler and A. Heinzl, "Robotic Process Automation," *Bus. Inf. Syst. Eng.,* vol. 60, p. 269–272, 2018.

[2] Apryse, «iText 8.0.1 API: Main Page,» [En línea]. Available: https://api.itextpdf.com/iText/dotnet/8.0.1/. [Último acceso: Agosto 2023].

[3] Microsoft, «¿Qué es .NET? Una plataforma para desarrolladores de código abierto.,» [En línea]. Available: https://dotnet.microsoft.com/es-es/learn/dotnet/what-is-dotnet. [Último acceso: Abril 2023].

[4] Universidad de Málaga, «Sala de Prensa - Noticias sala de prensa - Universidad de Málaga,» [En línea]. Available: https://www.uma.es/sala-de-prensa. [Último acceso: Agosto 2023].

[5] Software Freedom Conservancy, «The Selenium Browser Automation Project | Selenium,» [En línea]. Available: https://www.selenium.dev/documentation/. [Último acceso: Julio 2023].

[6] Refsnes Data, «HTTP Methods GET vs POST,» [En línea]. Available: https://www.w3schools.com/tags/ref_httpmethods.asp. [Último acceso: Abril 2023].

[7] Refsnes Data, «XPath Syntax,» [En línea]. Available: https://www.w3schools.com/xml/xpath_syntax.asp. [Último acceso: Abril 2023].

[8] J. Nielsen, Designing Web Usability, Indianapolis, Ind.: New Riders, 2000.

[9] American Society for Quality, «What is a Flowchart? Process Flow Diagrams & Maps | ASQ,» [En línea]. Available: https://asq.org/quality-resources/flowchart. [Último acceso: Abril 2023].

[10] Blue Prism Limited, «SS&C Blue Prism | Intelligent Automation Platform | Leaders in Automation,» [En línea]. Available: https://www.blueprism.com/. [Último acceso: Abril 2023].

[11] UiPath, «AI-Powered Business Automation Platform™ - Leader in RPA & Automation | UiPath,» [En línea]. Available: https://www.uipath.com/. [Último acceso: Abril 2023].

[12] Microsoft, «Power Automate | Microsoft Power Platform,» [En línea]. Available: https://powerautomate.microsoft.com/en-us/. [Último acceso: Abril 2023].

[13] Zapier Inc., «Zapier | Automation that moves you forward,» [En línea]. Available: https://zapier.com/. [Último acceso: Abril 2023].

[14] Microsoft, «Overview of ASP.NET Core | Microsoft Learn,» [En línea]. Available: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-7.0. [Último acceso: Agosto 2023].

[15] H. Schildt, C# 4.0: the complete reference, US: McGraw-Hill, 2010.

[16] Codecademy, «What is CRUD? | Codecademy,» [En línea]. Available: https://www.codecademy.com/article/what-is-crud. [Último acceso: Junio 2023].

[17] Meta, «Built-in React Hooks – React,» [En línea]. Available: https://react.dev/reference/react. [Último acceso: Agosto 2023].

[18] Refsnes Data, «JavaScript Introduction,» [En línea]. Available: https://www.w3schools.com/js/js_intro.asp. [Último acceso: Julio 2023].

[19] Vercel, Inc., «Docs | Next.js,» [En línea]. Available: https://nextjs.org/docs. [Último acceso: Agosto 2023].

[20] The OpenJS Foundation, «Node.js,» [En línea]. Available: https://nodejs.org/en. [Último acceso: Julio 2023].

[21] Refsnes Data, «Introduction to HTML,» [En línea]. Available: https://www.w3schools.com/html/html_intro.asp. [Último acceso: Junio 2023].

[22] Refsnes Data, «React JSX,» [En línea]. Available: https://www.w3schools.com/react/react_jsx.asp. [Último acceso: Junio 2023].

[23] Microsoft, «SQL Server technical documentation - SQL Server | Microsoft Learn,» [En línea]. Available: https://learn.microsoft.com/en-us/sql/sql-server/?view=sql-server-ver16. [Último acceso: Julio 2023].

[24] H. Garcia-Molina, J. D. Ullman y J. Widom, Database Systems: The Complete Book, Upper Saddle River, N.J.: Pearson Prentice Hall, 2009.

[25] Microsoft, «Download SQL Server Management Studio (SSMS) - SQL Server Management Studio (SSMS) | Microsoft Learn,» [En línea]. Available: https://learn.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16. [Último acceso: Abril 2023].

[26] Microsoft, «NuGet Gallery | Home,» [En línea]. Available: https://www.nuget.org/. [Último acceso: Agosto 2023].

[27] Amazon Web Services, Inc., «What is an IDE? - Integrated Development Environment Explained - AWS,» [En línea]. Available: https://aws.amazon.com/what-is/ide/. [Último acceso: Junio 2023].

[28] .NET Foundation, «Quick start | RestSharp,» [En línea]. Available: https://restsharp.dev/intro.html#introduction. [Último acceso: Julio 2023].

[29] zzzprojects, «documentation in Html Agility Pack (HAP),» [En línea]. Available: https://html-agility-pack.net/documentation. [Último acceso: Julio 2023].

[30] Newtonsoft, «Introduction,» [En línea]. Available: https://www.newtonsoft.com/json/help/html/Introduction.htm. [Último acceso: Julio 2023].

[31] Refsnes Data, «JSON Syntax,» [En línea]. Available: https://www.w3schools.com/js/js_json_syntax.asp. [Último acceso: Julio 2023].

[32] Microsoft, «Azure DevOps Services | Microsoft Azure,» [En línea]. Available: https://azure.microsoft.com/en-us/products/devops. [Último acceso: Abril 2023].

[33] MKLabs Co.,Ltd., «Introduction - StarUML documentation,» [En línea]. Available: https://docs.staruml.io/. [Último acceso: Agosto 2023].

[34] Object Management Group®, Inc., «Welcome To UML Web Site!,» [En línea]. Available: https://www.uml.org/. [Último acceso: Julio 2023].

[35] uml-diagrams.org, «UML Class and Object Diagrams Overview - common types of UML structure diagrams.,» [En línea]. Available: https://www.uml-diagrams.org/class-diagrams-overview.html. [Último acceso: Julio 2023].

[36] Microsoft, «Download Visual Studio Tools - Install Free for Windows, Mac, Linux,» [En línea]. Available: https://visualstudio.microsoft.com/downloads/. [Último acceso: Abril 2023].

[37] A. Agrawal, «Agile Methodology: Incremental and Iterative way of development | by Ashutosh Agrawal | Medium,» 4 Diciembre 2019. [En línea]. Available: https://medium.com/@ashutoshagrawal1010/agile-methodology-incremental-and-iterative-way-of-development-a6614116ae68. [Último acceso: Mayo 2023].

[38] M. Rehkopf, «Scrum Sprints: Everything You Need to Know | Atlassian,» [En línea]. Available: https://www.atlassian.com/agile/scrum/sprints. [Último acceso: Mayo 2023].

[39] P. Pandit y S. Tahiliani, «AgileUAT: A framework for user acceptance testing based on user stories and acceptance criteria,» *International Journal of Computer Applications,* vol. 120, nº 10, pp. 1-6, 2015.

[40] O. López, M. A. Laguna y J. M. Marqués, «Reutilización del Software a partir de Requisitos Funcionales en el Modelo de Mecano: Comparación de Escenarios,» de *Actas de IDEAS 2001*, San José, Costa Rica, 2001.

[41] U.S. General Services Administration, «Use Cases | Usability.gov,» [En línea]. Available: https://www.usability.gov/how-to-and-tools/methods/use-cases.html. [Último acceso: Junio 2023].

[42] Sun Microsystems, «Distributed Application Architecture,» 6 Abril 2011. [En línea]. Available: https://web.archive.org/web/20110406121920/http://java.sun.com/developer/Books/jdbc/ch07.pdf. [Último acceso: Junio 2023].

[43] Amazon Web Services, Inc., «¿Qué es una API de RESTful? - Explicación de API de RESTful - AWS,» [En línea]. Available: https://aws.amazon.com/es/what-is/restful-api/. [Último acceso: Junio 2023].

[44] R. Peterson, «Entity Relationship (ER) Diagram Model with DBMS Example,» 20 Julio 2023. [En línea]. Available: https://www.guru99.com/er-diagram-tutorial-dbms.html. [Último acceso: Agosto 2023].

[45] A. Leff y J. Rayfield, «Web-application development using the model/view/controller design pattern,» de *International Conference on Enterprise Distributed Object Computing*, Seattle, US, 2001.

[46] Microsoft, «Entity Framework documentation hub | Microsoft Learn,» [En línea]. Available: https://learn.microsoft.com/en-us/ef/. [Último acceso: Julio 2023].

[47] K. Beck, Test driven development: By example, Boston: Addison-Wesley Professional, 2022.

[48] M. Olan, «Unit testing: test early, test often,» *Journal of Computing Sciences in Colleges,* vol. 19, nº 2, pp. 319-328, 2003.

[49] .NET Foundation, «Home | xUnit.net,» [En línea]. Available: https://xunit.net/. [Último acceso: Agosto 2023].

[50] Meta Platforms, Inc., «Jest - Delightful JavaScript Testing,» [En línea]. Available: https://jestjs.io/. [Último acceso: Agosto 2023].

[51] P. C. Jorgensen y C. Erickson, «Object-oriented integration testing,» *Communications of the ACM,* vol. 37, nº 9, pp. 30-38, 1994.

[52] T. Hamilton, «Load Testing Tutorial: What is? How to? (Examples),» 12 Agosto 2023. [En línea]. Available: https://www.guru99.com/load-testing-tutorial.html. [Último acceso: Agosto 2023].

[53] J. Sampaio, «Load/stress testing .NET apps with Apache JMeter,» 26 Marzo 2021. [En línea]. Available: https://www.red-gate.com/simple-talk/devops/testing/load-stress-testing-net-apps-with-apache-jmeter/. [Último acceso: Agosto 2023].

[54] J. K. Ousterhout, «Scripting: Higher level programming for the 21st century,» *Computer,* vol. 31, nº 3, pp. 23-30, 1998.

[55] Oracle, «Java SE Desktop Accessibility - Java Access Bridge For Windows OS,» [En línea]. Available: https://www.oracle.com/java/technologies/javase/javase-tech-access-bridge.html. [Último acceso: Julio 2023].

[56] Gobierno de España. Ministerio de Industria, Comercio y Turismo, «Sede electrónica del Ministerio de Industria, Comercio y Turismo - AutoFirma,» [En línea]. Available: https://sede.serviciosmin.gob.es/es-es/firmaelectronica/paginas/autofirma.aspx. [Último acceso: Abril 2023].

Apéndice A

Manual de instalación

A.1 Requerimientos

Las tecnologías y dependencias del proyecto, así como sus versiones, han sido especificadas en el capítulo del estado del arte de la memoria y, adicionalmente, en la configuración del entorno de desarrollo de la primera iteración del proyecto. El entregable asociado al TFG es un archivo compreso en formato ZIP que puede ser importado en Visual Studio para su despliegue y uso. Además, Visual Studio es una herramienta que gestiona las dependencias automáticamente con el gestor de paquetes NuGet, lo que facilita la importación. Entre las acciones manuales que realizar para el correcto funcionamiento del proyecto son:

- Instalar el entorno de JavaScript 'Node.js', ya que es utilizado por React.
- El controlador de Selenium se encuentra ubicado en el proyecto.
- Instalar SQL Server y activar el servicio local. Después, ejecutar el script de la base de datos del proyecto que contiene el modelado de información del sistema presentado y que se encuentra en la carpeta backend de la aplicación.

A.2 Endpoints API disponibles

- api/tarea/GetTarea
- /api/ejecucione/CreaEjecucion
- /api/usuario/Autenticar
- /api/role/GetTipo

Apéndice B

Manual de usuario

B.1 Inicio de sesión

Cuando el servidor ha levantado todo el sistema de la aplicación web, y se ha redirigido a la URL de acceso *http://localhost:44462/*, aparece el inicio de sesión con el que se puede interactuar escribiendo las credenciales de usuario.

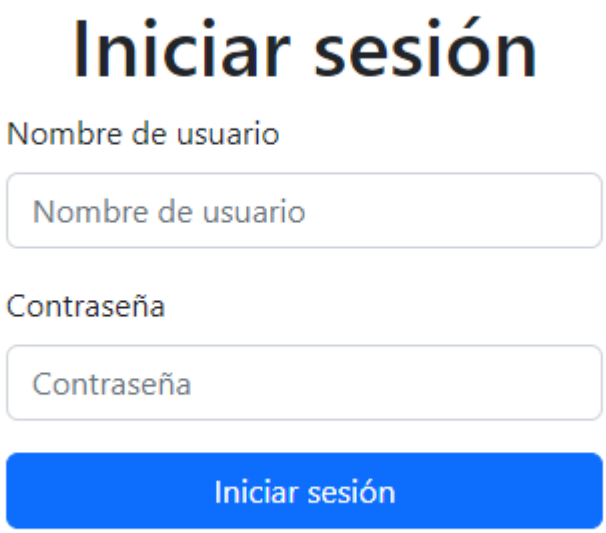

Figura 25. Ventana de inicio de sesión de la aplicación web

B.2 Ejecutar una tarea

Posteriormente, debe aparecer la página principal de la aplicación con el listado de tareas disponibles. Puede ejecutarse una tarea si se realiza clic en el botón 'Ejecutar'.

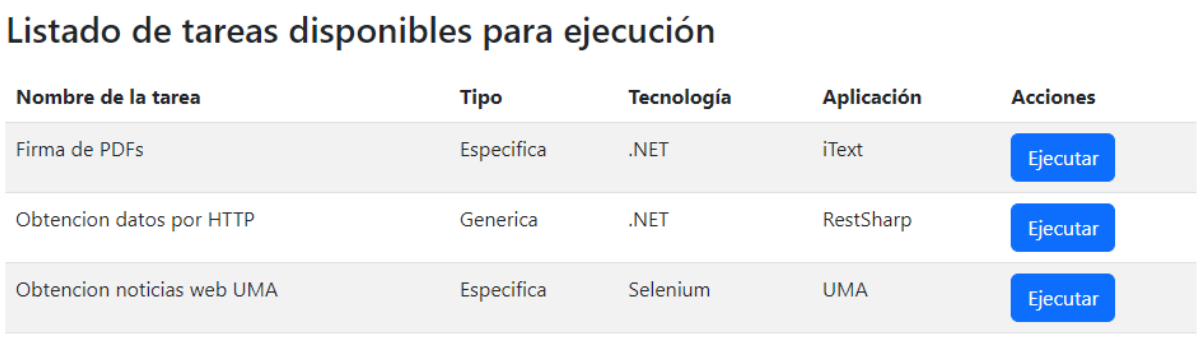

Figura 26. Ventana principal de la aplicación web mostrando las tareas disponibles

Al lado del botón 'Ejecutar', si aplica, aparece el parámetro de entrada que puede añadirse a la tarea. En el caso de firma de PDFs, se refleja así:

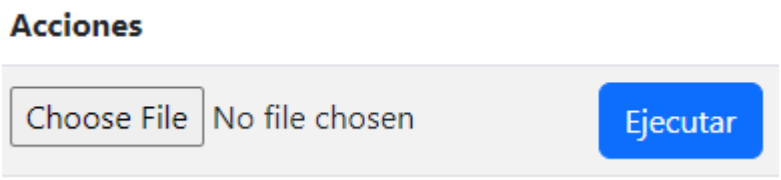

Figura 27. Interfaz de selección de documentos PDF

B.3 Historial de tareas ejecutadas

En la página principal de la aplicación también pueden verse qué tareas han sido ejecutadas y realizar clic en el botón 'Ver resultados' para la visualización de los datos trabajados en la ejecución de la tarea.

Se muestra una vista global de la página con ambos paneles incluidos:

Listado de tareas disponibles para ejecución

Nombre de la tarea	Tipo	Tecnología	Aplicación	Acciones	
Firma de PDFs	Especifica	.NET	iText	Choose File No file chosen	Ejecutar
Obtencion datos por HTTP	Generica	.NET	RestSharp	Choose File No file chosen	Ejecutar
Obtencion noticias web UMA	Especifica	Selenium	UMA	Choose File No file chosen	Ejecutar

Historial de tareas ejecutadas

Fecha de ejecución	Usuario	Tarea	Estado	Acciones
2023-09-06T19:05:30.597	manuel	Obtencion datos por HTTP	Pendiente	Ver resultados
2023-09-06T20:12:09.377	manuel	Firma de PDFs	Pendiente	Ver resultados
2023-09-06T20:12:57.59	manuel	Firma de PDFs	Pendiente	Ver resultados
2023-09-06T20:15:54.897	manuel	Firma de PDFs	Pendiente	Ver resultados
2023-09-06T20:17:40.833	manuel	Firma de PDFs	Pendiente	Ver resultados
2023-09-06T20:17:47.14	manuel	Firma de PDFs	Pendiente	Ver resultados
2023-09-06T20:17:58.687	manuel	Firma de PDFs	Pendiente	Ver resultados
2023-09-06T20:18:00.203	manuel	Obtencion datos por HTTP	Pendiente	Ver resultados
2023-09-06T20:21:10.117	manuel	Firma de PDFs	Pendiente	Ver resultados

Figura 28. Visualización general de la interfaz con el histórico de resultados

UNIVERSIDAD DE MÁLAGA | uma.es

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga