Facultad
de
Ciencias

AUTOMATIZACIÓN DE UN PROCESO ADMINISTRATIVO ONLINE, MEDIANTE EL USO DE TECNOLOGÍA WEB Y BASES DE DATOS

(AN ADMINISTRATIVE ONLINE PROCESS AUTOMATIZATION, BY USING WEB TECHNOLOGY AND DATABASES)

Trabajo de Fin de Grado
para acceder al

GRADO EN INGENIERÍA INFORMÁTICA

Autor: Andrés Ortega Tornel

Director: Pablo Sánchez Barreiro

Co-Director: David Fernández García

Junio – 2020

ÍNDICE

RESUMEN

Una parte importante dentro de las empresas es la recolección diaria de grandes cantidades de información desde los portales web, como por ejemplo las facturas de los clientes o las reclamaciones que estos puedan poner. Realizar esta tarea manualmente supone un enorme gasto de tiempo y recursos materiales y humanos, que podrían emplearse en otro tipo de labores. Por eso mismo, se han desarrollado técnicas que permiten automatizar, por medio de la creación de robots software, este tipo de tediosos procesos administrativos. De esta forma, se disminuye considerablemente la intervención humana en tareas repetitivas que varían muy poco en cada iteración. Esta reducción contribuye a una mayor fiabilidad y eficiencia del resultado final, a la vez que libera a los encargados de estas labores para que inviertan su atención en otros asuntos que requieran mayor creatividad y toma de decisiones.

El objetivo de este Trabajo de Fin de Grado es la creación de uno de estos robots software para la automatización de la recolección diaria de información de una empresa energética en concreto. Para ello, el robot tendrá que navegar por el portal web de la empresa, mediante el uso de tecnología web, filtrando y descargando información de dicha web, para después clasificarla y almacenarla en una base de datos.

Para la realización del programa se hará uso del lenguaje de programación C# en el entorno de trabajo Microsoft Visual Studio.

Tras finalizar el proyecto y someterlo a diversas pruebas para comprobar su funcionamiento en distintas situaciones, se pudo comprobar como el tiempo invertido en llevar a cabo las descargas diarias disminuye considerablemente mediante la utilización del robot desarrollado. Además, como se puede programar su lanzamiento en horas fuera del horario laboral, el resultado es que ningún equipo ni empleado se verá interrumpido por su ejecución.

PALABRAS CLAVE

- RPA
- Robot software
- Automatización de procesos administrativos

ABSTRACT

An important part within a company is the daily recollection of large amounts of information throw web portals. An example of that information is all client's bills or the reclamations they can interpose. To manually do this task means a huge waste of time, material and human resources, which could be used in other kind of jobs. Because of that, some new automatization techniques have been developed, using software robots, in order to automatize these humdrum kinds of processes. In this way, the human intervention in repetitive tasks which are similar in each iteration, decreases notably. This reduction contributes to a bigger reliability and efficiency of the final outcome, while at the same time, it sets free the employers in charge of this labors for doing other issues that require more creativity and decision making.

The goal of this final degree project is developing one of those software robots to automatize the daily recollection of information for an energy company in particular. For this reason, the robot will have to surf the company's web portal, by using web technology, filtering and downloading information that after will be classified and stored in a database.

For the program achievement, the programming language C# in the Microsoft Visual Studio environment will be used.

After finishing the project and testing it throw various tests in order to check its correct operation in different situations, the fact that the time invested in daily downloads decreases notably using automatization robots, could be verified. Furthermore, it is possible to program the robot's launching out of working hours, so any computer or employer will be interrupt by its execution.

KEY WORDS

- RPA
- Software robot
- Backoffice automatization

1. INTRODUCCIÓN

Hoy en día, una parte importante del trabajo administrativo que se realiza en las empresas consiste en descargar diariamente miles de archivos o mensajes desde portales web, por ejemplo, para saber cuántos clientes se han dado de baja o cuáles tienen alguna deuda pendiente. Un ejemplo de esta tarea es descargar las facturas de los clientes de una empresa, de un cierto día. En este caso, el proceso de descarga consistiría en acceder y acreditarse en el portal web de la compañía, navegar hasta el apartado para el seguimiento de la información y aplicarle los siguientes filtros a la información disponible para descargar: El proceso debe ser el de facturación, la fecha de recepción tiene que ser la del día que nos interese y el tipo de información puede cambiar entre envíos y respuestas, entre otros filtros de menor importancia. Una vez se obtiene la información deseada, se descarga y se almacena. Este proceso se realiza para las facturas catalogadas como envíos y, posteriormente, se repite para las catalogadas como respuestas. Estos archivos descargados proporcionan la información necesaria para llevar a cabo un gran número de tareas administrativas.

Estas tareas pueden automatizarse mediante la creación de robots software. Por robot software se entiende un programa informático que recrea las interacciones de una persona con una página web. Para ello, el programa se guía a través de la web mediante su diseño y los elementos que esta contiene para, por ejemplo, pulsar un botón, seleccionar una opción de un desplegable o rellenar un campo de texto.

El objetivo de este trabajo es desarrollar un robot que sea capaz de realizar automáticamente una labor rutinaria como la explicada anteriormente, para un portal web específico, produciendo así un gran ahorro temporal y de recursos.

La empresa para la que he realizado este trabajo es la consultora estadounidense Accenture Limited en su sede en Santander. Esta empresa se dedica, entre otras cosas, a la consultoría o desarrollo de procesos laborales de otras empresas, generando así una mejora en su producción y eficiencia.

He de aclarar que el proyecto que concierne a este trabajo es muy diferente del realizado durante el periodo de prácticas empresariales. Decir que en las prácticas

desarrollé otros robots, y que el que se presenta en esta memoria fue realizado fuera de las horas de prácticas.

1.1 OBJETIVOS

Como se ha comentado anteriormente, la descarga diaria de información es una parte vital del trabajo diario de muchas empresas. Pero conlleva un gran inconveniente, y es que, si hay que hacerlo a mano, supone tener a unos empleados y sus correspondientes equipos totalmente ocupados entre dos y tres horas (depende del volumen de mensajes a descargar) en esta tarea, todos los días. Además de que todos los otros trabajadores que dependan de estas descargas para desempeñar sus labores, también se ven obligados a esperar. Por eso mismo, el objetivo principal de este proyecto es producir un ahorro temporal en el desempeño de esta tarea rutinaria mediante el uso de un programa informático conocido como robot software.

La consecución del objetivo principal está acompañada por otros secundarios como los siguientes:

- Ahorro de recursos humanos y materiales.
- Mejora en la calidad de los puestos de trabajo de los empleados liberados de esta repetitiva tarea.
- La reducción de errores humanos durante el proceso. Al robotizar el proceso queda asegurado que siempre se van a seguir los mismos pasos y en el mismo orden. No cabe lugar a errores ortográficos como si un nombre compuesto se escribe con guion, empieza por mayúscula, etc. O simplemente pulsar donde no se debe.
- El registro y almacenamiento de la información. Este robot almacena todos los archivos descargados en una base de datos, día tras día, facilitando así poder llevar un registro de la información y evitando saturar el espacio de almacenamiento del equipo en el que se ejecute esta tarea.

1.2 ENTORNO DE TRABAJO

Uno de los clientes de esta consultora es una empresa energética (cuyo nombre no puedo especificar por motivos de confidencialidad), con sede en Santander. La consultora es la encargada de mejorar y desarrollar los procedimientos y tareas de su cliente, con lo que dispone de un departamento dentro de las oficinas de este (departamento de Mejora), donde he prestado mis labores. La función principal del equipo de mejoras, como su nombre indica, gira entorno a desarrollar soluciones software que mejoren la eficiencia, simplifiquen y den valor añadido al trabajo diario que se realiza dentro del *BackOffice* (parte encargada de desempeñar las tareas destinadas a gestionar la propia empresa y que no mantiene contacto directo con el cliente) de la compañía energética. Para ello se realizan desarrollos software empleando distintas tecnologías entorno al concepto RPA (*Robotic Process Automation*)[1], que consiste en automatizar procesos que se pueden dividir en pasos con lógica de negocio bien definida.

El departamento de Mejora puede dividirse en dos ramas diferentes: *MAT* y *Departamentos*. MAT es un proyecto con más de un año de antigüedad en el que trabajan dos de los cinco empleados de este departamento, y que tiene como objetivo principal controlar el tiempo que tardan los miembros del resto de departamentos en realizar una tarea. Aparte de esto, permite asignar cada tarea a un solo empleado, de forma que quede muy claro cuáles están sin hacer y cuáles están siendo hechas por otras personas. Por ejemplo: Si hay que gestionar diez reclamaciones, MAT separa los diez casos y muestra qué trabajador está realizando cada caso o los que todavía están sin hacer, de forma que no se resuelva un caso dos veces por error. Como funcionalidad extra, MAT también lleva a cabo de forma interna algunas tareas repetitivas de las que hacen los trabajadores (como extracciones de datos que son siempre las mismas u otras labores que no requieren toma de decisiones), así estos quedan más libres para hacer otras cosas.

La rama Departamentos consiste en atender las peticiones de mejora o solucionar los posibles errores que puedan surgir en otros departamentos. Esto engloba desde solucionar un problema de conexión a la red de datos privada de la compañía hasta realizar la robotización de un proceso. Las peticiones llegan a través de una plataforma

propia llamada *Scarab*, donde se expone la mejora o problema a solucionar, quién la solicita y la gravedad de esta. Entonces, según lo crítica que sea y según el criterio de los componentes del departamento de Mejora se van realizando, enseñando al cliente y dando por solucionadas.

Dentro del equipo de mejoras, he realizado diversidad de tareas con una aportación real que han sido y son utilizadas por el resto de los empleados de la compañía. Algunos ejemplos de estas labores son: Realización de robots para descargas web (como el que se desarrolla en este proyecto) así como la supervisión de su ejecución diaria, creación de una aplicación para controlar las vacaciones del personal, tareas de la rama de MAT, entre otros menos destacables.

1.3 METODOLOGÍA

A continuación, se van a repasar cada una de las fases del proyecto, desde la concepción del robot hasta su despliegue, explicando cual ha sido mi aportación a cada fase.

Concepción

En este apartado no he tenido ninguna aportación, pues la idea inicial del proyecto ya existía. Yo solo fui el encargado de llevarla a cabo.

Requisitos

Los requisitos que el robot debe seguir no los extraje yo, sino que me vinieron dados en un documento en el que se explica paso por paso el procedimiento para efectuar la descarga. Esto podría haber sido suficiente de no ser porque al cambiar la página web a la que el robot debía acceder, de Internet Explorer a Google Chrome, los encargados de esta también cambiaron su diseño. Por este motivo tuve que preguntar a los empleados de mi oficina que se encargan de este tema qué se hace en las partes del proceso que difieren con el documento y actualizarlo. Estas cuestiones se explican con más detalle en el capítulo dos.

Arquitectura y diseño

Este apartado no es de realización propia, pues todos los robots de este carácter se basan en el mismo diseño, el cual ya estaba establecido. Dicho diseño se describe en el capítulo dos.

Implementación

Esta fase abarca todo el proceso de programación del robot y tiene una aportación completa por mi parte ya que me encargué totalmente de ella. Esta fase queda explicada más en detalle en el capítulo dos.

Pruebas

En esta fase he tenido una aportación completa, ya que la realización del plan de pruebas también quedó a mi cargo. En resumen, la comprobación del correcto funcionamiento del robot se basa en ir aislando partes del código, sintetizando entradas y salidas de datos si las necesita, y ver si el resultado de su ejecución es el esperado. Esta fase está más ampliamente explicada en el capítulo tres.

Despliegue

Los procedimientos de despliegue de este tipo de robots ya estaban establecidos en la empresa, por lo que mi contribución a esta fase fue la de llevar a cabo correctamente dicho procedimiento para poner el robot a funcionamiento real.

Metodología

La realización de este proyecto no ha seguido ningún tipo de metodología en especial. Aun así, he adoptado algún concepto del área de las metodologías ágiles, como el de ir probando el producto a medida que lo iba desarrollando, pero sin llegar a seguirla estrictamente. Esto se debe a que en ningún momento me fue impuesta ninguna metodología y, al ser el único programador del proyecto, disponía de total libertad organizativa.

1.4 INFRAESTRUCTURA DEL TRABAJO

En este apartado se va a explicar la infraestructura del trabajo, es decir, las herramientas y aspectos técnicos en los que se basa este proyecto.

Para la parte del código del robot me he apoyado en el entorno de desarrollo *Microsoft Visual Studio*[1], pues ya poseía conocimientos previos sobre este programa y sobre algunos de los lenguajes que soporta. Todo el código está desarrollado en C# [2], ya que es el lenguaje bandera del entorno *.NET* [3] de Microsoft y es el que se usa normalmente en la empresa. Para el desarrollo del robot me he tenido que apoyar en algunas librerías como *Selenium*[2] o *Ionic.Zip*[3].

El último recurso en referencia a la programación del programa es la estructura y diseño de la propia página web a la que este accede. Esto es primordial ya que el robot se guía mediante los atributos de los diversos elementos que conforman el código HTML [4] del portal web para llevar a cabo cada una de sus acciones, ya sea mediante un número identificador único, un nombre, el texto que incluye etc. Pero para que el robot pueda reconocer los elementos de la página web y navegar por esta, es necesario el uso de un *web driver* que proporcione las funcionalidades.

Como herramienta para el almacenamiento y extracción de datos se ha hecho uso del sistema gestor de bases de datos *SQL Server* [5].

En cuanto a la gestión de la configuración, he hecho uso de Microsoft *Visual Sourcesafe*. Dado que, durante el desarrollo el robot solo era accedido y modificado por mí y solo había una versión del proyecto, no consideré necesario el uso de ningún esquema especial de gestión de la configuración.

Sobre la gestión de la calidad no tengo nada que añadir pues no se sigue ningún proceso para controlar este aspecto del proyecto, al no existir ningún proceso de este tipo en la empresa en la que trabajaba.

He de mencionar que, durante todo el proceso de implementación del robot, se ha llevado a cabo un proceso de integración continua, de manera manual. A medida que la

[1] https://visualstudio.microsoft.com/es/vs/
[2] https://www.selenium.dev/docs/site/es/introduction/
[3] https://documentation.help/DotNetZip/

programación iba avanzando, se iba probando el código existente con el objetivo de detectar fallos y solucionarlos lo antes posible.

2. REQUISITOS, ARQUITECTURA Y DISEÑO

En este apartado voy a explicar los requisitos que el robot debe cumplir, así como todo lo relacionado con su estructura y diseño.

2.1 REQUISITOS

Para entender bien los requisitos es necesario observar el diagrama de objetivos que se representa en la Figura 1.

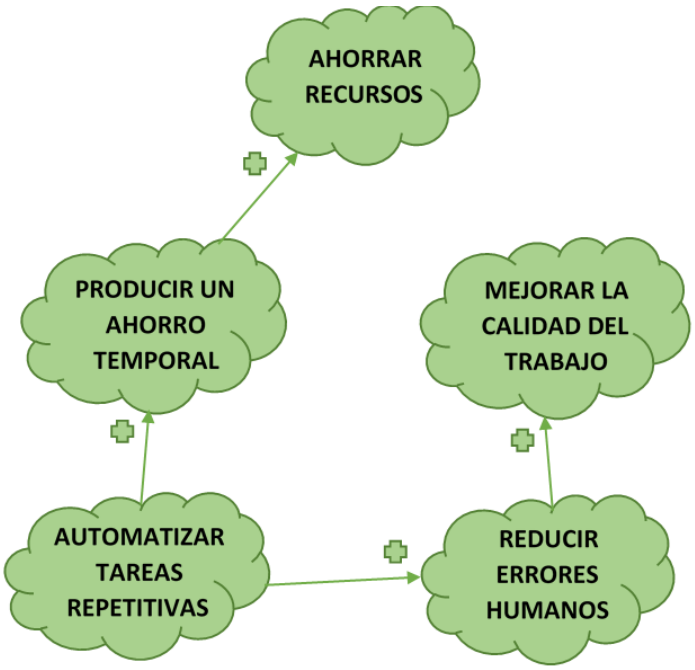

Figura 1: Objetivos que debe satisfacer el programa.

Como resultado del proceso de recogida de requisitos y, teniendo en cuenta el diagrama de objetivos, se obtienen las siguientes especificaciones:

- El robot tiene que entrar en el portal web de la empresa energética mediante el uso de las credenciales que ellos me proporcionen.
- En caso de haber problemas en el proceso de identificación, el robot no debe introducir las credenciales más de dos veces, ya que en caso de que se intente tres veces sin éxito, la cuenta queda bloqueada.
- Es necesario que descargue todos los mensajes pertenecientes al día anterior.

- Debe descargar tanto los mensajes del apartado de “Envíos” como el de “Respuestas”, para cada tipo de proceso que ellos me indicasen (por proceso se entiende como la funcionalidad de la información que se guarda en el archivo, siendo F1 las facturas, M1 modificaciones, R1 reclamaciones, etc.).
- No puede descargar los mensajes del proceso de facturación (F1) antes de las 10:00 de la mañana, ya que interferiría con otra automatización.
- La descarga de la información perteneciente a los procesos que no sean el de facturación debe estar lista antes de las 12:00 del mediodía.

Estos son los requisitos que pude obtener de las explicaciones de los empleados de otro de los departamentos y del documento que detalla todos los pasos que debe seguir el programa. Derivando de estas especificaciones, aparecen otras de carácter más técnico pensadas por mí, que ayudan a poder entender y realizar un mejor diseño del robot:

- El robot debe poder recibir parámetros que modifiquen su comportamiento, como por ejemplo las credenciales para acceder al portal, la fecha de la que se quiere obtener la información o la ruta en la que se quieren guardar las descargas.
- Algunos de estos parámetros pueden variar con el tiempo (como las credenciales de acceso), con lo que tienen que ser fácilmente accesibles y modificables.
- El resultado de las descargas debe quedar almacenado de forma que sea sencillo realizar búsquedas de mensajes o agrupaciones de estos, ya que puede ser necesario para hacer revisiones y consultas.

2.2 WEB DRIVER

En este apartado se procede a explicar brevemente qué es un driver web y para qué se usa, ya que es una pieza clave en la arquitectura del robot. *Web Driver* es una clase que simula interacciones con los elementos de una página web. Por ejemplo, puede simular una pulsación sobre un botón o la declaración de una determinada opción. El driver contiene funciones para buscar dichos elementos mediante alguno de sus atributos (nombre, identificador, etiqueta) y, dependiendo del tipo que sea, interaccionar con él

de una forma u otra (haciendo clic, desplegando, escribiendo texto, marcando una casilla).

Para que el driver web funcione correctamente, es necesario inicializarlo indicándole alguna información básica como el navegador que se va a usar, la ruta en la que depositar las descargas que realice o la URL a la que debe acceder. Para que se vea mejor cómo funciona el driver web, la Figura 2 muestra un sencillo fragmento de código del proceso de autentificación en una página web:

```
bool encontrado = false;
while (!encontrado)
{
	try
	{
		driver.driver.FindElement(By.Name("usuario")).SendKeys(user);
		driver.driver.FindElement(By.Name("clave")).SendKeys(password);
		driver.driver.FindElement(By.XPath("//td[text()='Entrar']")).Cli
		ck();
		encontrado = true;
	}catch{…}
}
```

Figura 2: Código para simular la autenticación en una página web.

La primera acción que realiza el driver es buscar un elemento cuyo nombre sea usuario, mediante la función *FindElement*. Una vez lo encuentre, como es una caja de texto, se inserta un texto. Si se ejecutase otra acción que no tenga sentido para este tipo de elemento, simplemente no pasaría nada. El último componente que se busca, en lugar de hacerlo por el nombre se hace mediante un buscador más potente, el *XPath*[4]. Con esto, se especifica que el elemento a encontrar contiene el texto *Entrar* en él.

He de aclarar que todos los nombres, etiquetas, identificadores y demás, que se usen para buscar e interactuar con los elementos, así como el diseño de la página web, se encuentran en el propio código del portal web. Así que, para que el driver funcione, es necesario conocer y entender bien y en profundidad la estructura HTML de la página.

[4] https://www.lambdatest.com/blog/complete-guide-for-using-xpath-in-selenium-with-examples/

2.3 ARQUITECTURA

En el diseño del robot se pueden diferenciar tres partes bien definidas, como muestra la Figura 3.

Figura 3: Arquitectura del robot.

Al arrancar el robot, realiza una consulta a la base de datos, que devuelve la URL de la página web de la empresa a la que se quiere acceder, el usuario y contraseña con el que se accede, la ruta en la que se descargan los mensajes y otros datos de menor importancia; A continuación, se hace otra consulta a base de datos que tiene como objetivo obtener una lista con los procesos a descargar. La parte final de esta fase es configurar el proceso de acceso a la web, especificando el motor de búsqueda a usar o la URL a la que acceder.

Una vez terminada la fase de obtención de datos, comienza el proceso de descarga de mensajes. El primer paso es identificarse en el portal con los datos adquiridos anteriormente. Una vez dentro, el robot tiene que descargar los mensajes alternando entre las pestañas Envíos y Respuestas para cada uno de los procesos, para la fecha deseada. Al final de esta fase se tiene la información descargada en una ruta de la red de la empresa, agrupada en archivos comprimidos ".zip" según si es de tipo envíos o respuestas y el proceso al que pertenece (Por ejemplo: "ENVÍOS_R1", "RESPUESTAS_F1"), así como el número final de mensajes descargados.

La tercera y última fase de esta arquitectura es el almacenamiento de los datos en la base de datos del Departamento de Mejora. Para ello, el robot recorre los mensajes uno por uno, analizando la información que contiene e insertando datos en diversas tablas. Esta parte del proceso se describe con más detalle en el capítulo 2.5.

2.4 DISEÑO DE LA BASE DE DATOS

Durante el proceso de almacenamiento de la información, esta es analizada y guardada en tres tablas diferentes de una misma base de datos, como muestra la Figura 4.

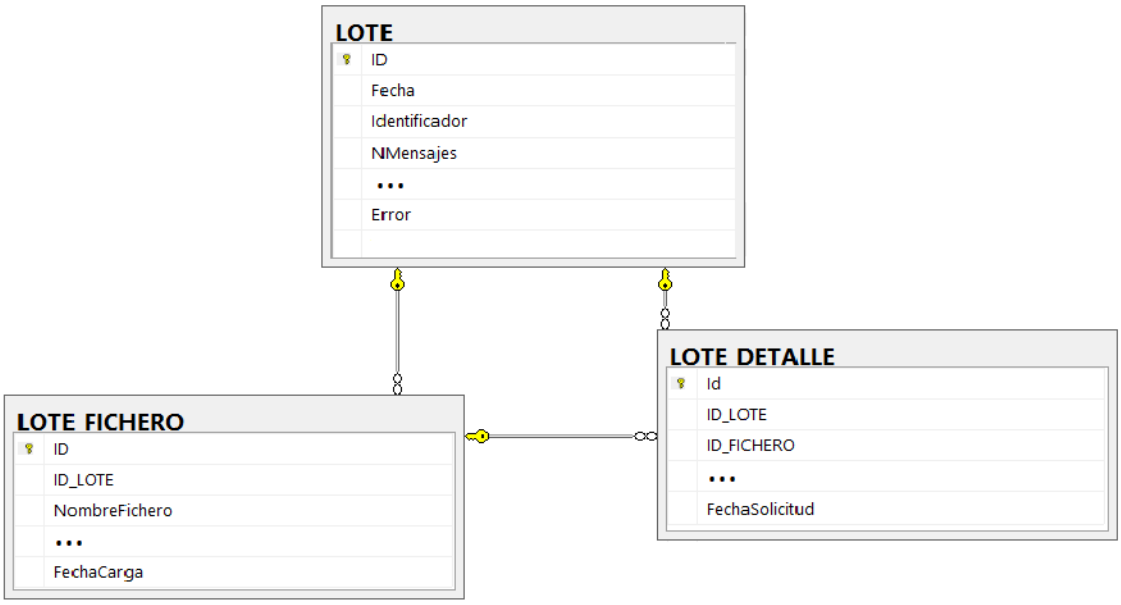

Figura 4: Diagrama de las tablas de base de datos.

El contenido de las tablas y la relación entre ellas es el siguiente:

- Lote: Cada descarga de una empresa para un día específico representa un lote (pudiendo diferenciarse de los demás por el identificador de la empresa y la fecha). En esta tabla se guardan todos los lotes de todos los días, especificando la fecha, número de mensajes descargados y si ha habido algún error, entre otras cosas.
- Lote en detalle: Aquí se puede ver información un poco más detallada (una entrada por cada archivo descargado). Están las claves foráneas que apuntan al identificador del lote al que pertenecen y el fichero al que se refieren, así como de que proceso son, la fecha y algo de la información que contiene cada mensaje.
- Lote con fichero: Esta tabla muestra la URL de cada mensaje, el nombre de este y la fecha en la que se ha cargado. También enseña el lote al que pertenece cada archivo mediante una clave foránea hacia el identificador de la tabla *Lote*.

2.5 IMPLEMENTACIÓN

En este apartado se va a explicar, paso por paso, la implementación del robot, así como los problemas que surgieron durante el proceso.

En la fase de obtención de argumentos y preparación de datos, no hay ninguna parte que haya supuesto un reto. El primer paso es coger los parámetros que se le pasan al programa a la hora de ejecutarlo y se hace una consulta, mediante un objeto de la clase SQL que permite hacer conexiones a bases de datos, con el propósito de obtener unos datos y almacenarlos.

En primer lugar, se obtienen las credenciales necesarias para acceder al portal web. En segundo lugar, se obtiene una lista con todos los procesos que se quieren descargar. Para ello hay que acceder a una tabla que contiene todos los procesos que cada empresa tiene disponibles para descargar, se rellena un *dataset*[5] con el nombre de los procesos y de ahí se pasan a una lista.

Una vez explicada la primera parte, es el turno de la fase de descarga. Lo primero de todo es, una vez accedido a la página principal, introducir las credenciales obtenidas anteriormente. La página de acceso tiene el diseño que se muestra en la Figura 5. Quedan resaltados en rojo los elementos que interesa buscar.

Figura 5: Diseño de la página de acreditación.

[5] https://docs.microsoft.com/es-es/dotnet/framework/data/adonet/dataset-datatable-dataview/

El proceso de autenticación no requiere demasiadas acciones y se realiza con un código similar al mostrado en la Figura 2 del apartado 2.2.

El proceso de descarga se ilustra en la Figura 6 y muestra la lógica que debe seguir el programa.

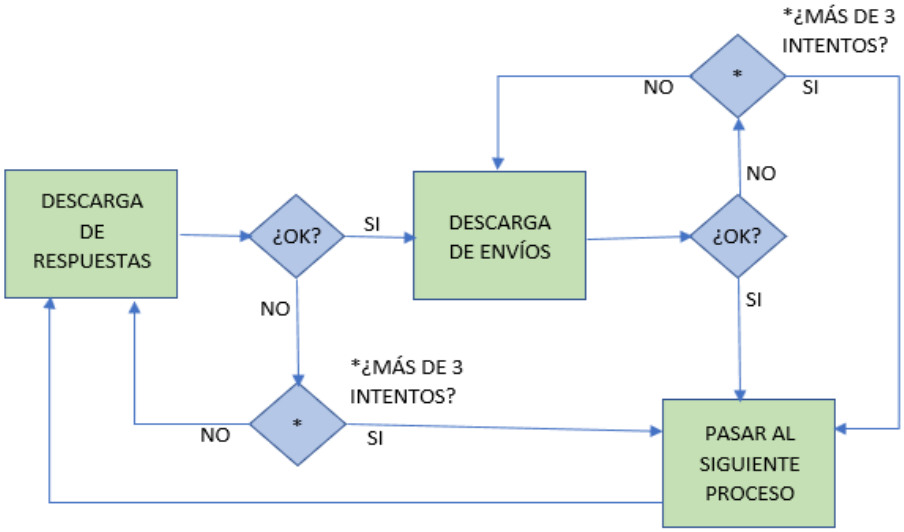

Figura 6: Diagrama del proceso de descarga.

Mediante este esquema, se consigue ejecutar el proceso de descarga para respuestas y envíos, siempre que las respuestas hayan ido bien, para cada proceso. La lógica del proceso consiste en descargar los mensajes correspondientes a las respuestas. En caso de conseguirlo, pasa a descargar los envíos. Pero si falla en las respuestas, pasa directamente al siguiente proceso. Un problema importante que surgió es que la página a veces devuelve un error durante el proceso de descarga. Este error parece ser aleatorio hasta el punto de que, si se repite exactamente el mismo proceso, lo más probable es que todo vaya bien. No obstante, no se sabe cuándo puede aparecer este error. Por este motivo hay una variable de control de intentos, que lo que hace es llevar la cuenta del número de veces que se intenta la descarga de un proceso. Mientras la descarga haya fallado y no se haya intentado más de tres veces, el robot elimina todo lo descargado durante el proceso actual, en caso de que lo haya, y vuelve a intentarlo. Al cuarto intento fallido, guarda el error y pasa al siguiente proceso. Con la parte de eliminar los mensajes descargados, se consigue que, si funciona en el segundo o tercer intento, no haya datos repetidos.

A continuación, se va a explicar la rutina mediante la cual el robot aplica los filtros requeridos para realizar las descargas. Para que se entienda mejor, se verá primero una imagen de la apariencia de la página web, con algunos elementos destacados en rojo, y después un fragmento de código con la interacción con la web mostrada en la Figura 7. La primera parte destacable corresponde con el filtrado de datos.

Figura 7: Diseño de la página de filtrado de información.

```
bool filtros = false;
while (!filtros)
{
	try
	{
		//Navegar hasta el apartado “Seguimiento mensajes”.
		driver.driver.SwitchTo().ParentFrame().SwitchTo().Frame("central
		");
		driver.cambiarAtributo(driver.driver.FindElement(By.Name("fec_re
		cep	_desde")), "value", fecha.ToString("dd/MM/yyyy"));
		//Repetir para la “fecha hasta”.
		SelectElement select = new
		SelectElement(driver.driver.FindElement(By.Name("cod_proceso")))
		;
		select.SelectByValue(proceso);
		select = new
		SelectElement(driver.driver.FindElement(By.Name("tipo")));
		select.SelectByValue("Todos");
		if (tipo == "Respuestas")
		{
			select = new
			SelectElement(driver.driver.FindElement(By.Name("cod_dest
			ino")));
			select.SelectByValue(empDestino);
		}
		else if (tipo == "Envios")
		{
			//Repetir bucle anterior para el elemento “cod_origen”.
		}
		filtros = true;
	}catch {…}
}
driver.driver.FindElement(By.XPath("//td[text()='Aceptar']")).Click();
```

Figura 8: Código del proceso de descarga.

En el fragmento de código que muestra la Figura 8, primero se introduce el rango de fechas para la descarga, que como sólo es de un día, se pone el mismo valor en ambos campos. Para ello se buscan los elementos de la web donde se introducen las fechas y se cambia el atributo *value* por la fecha en cuestión. El siguiente paso es seleccionar el proceso, el código de la empresa (este valor no ha sido explicado porque no es relevante) y el estado de la información (correcta, fallida, toda, etc.). En este caso no se trata de una caja de texto en la que insertar un valor, sino que es un desplegable con una lista cerrada de opciones, con lo que se hace uso de un objeto de la clase *SelectElement*. Este objeto permite elegir valores de un desplegable, ya sea por su valor, índice en la lista, texto, etc. Una vez seleccionado los valores pertinentes, se pulsa el

botón *Aceptar*, que se encuentra por el texto que el botón contiene en uno de sus atributos.

Una vez se completa la búsqueda, el siguiente paso es comprobar si ha producido algún resultado. Esto es importante ya que, en caso de que no haya ninguno, el robot debe terminar la rutina y pasar al siguiente proceso. También puede darse el caso de que salga un error por haber demasiados mensajes, en cuyo caso sale de la rutina y lo vuelve a intentar. Los mensajes que aparecen en estos casos son los que se muestran en las Figuras 9 y 10.

Figura 9: Mensaje para indicar la falta de resultados.

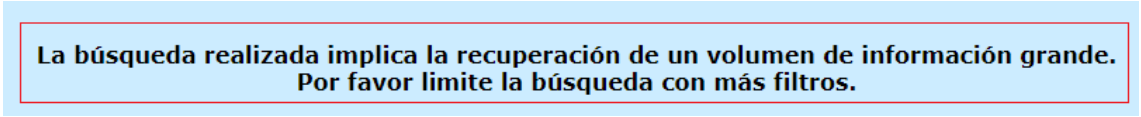

Figura 10: Mensaje para indicar el exceso de resultados.

Primero se busca el elemento que indica que no hay resultados, mediante uno de sus atributos. Si se encuentra, el robot simplemente pasa al siguiente proceso (no significa que haya habido un error, solo es que no hay nada que descargar). En caso contrario, busca si algún elemento web contiene el texto del volumen máximo de mensajes y, si lo encuentra, retorna un error y lo vuelve intentar. Si no lo encuentra es que todo ha ido bien y continúa con la descarga.

Si hay resultados, se muestra una interfaz como la de la Figura 11 (ha sido borrada toda la información visible de cada una de sus filas, por motivos de confidencialidad). Una vez aparecen los resultados de la búsqueda (en caso de que los haya), lo siguiente que se hace es ver el número de archivos disponibles y en cuántas páginas se distribuyen. Si hay más de una página de archivos, hay que ir avanzando de una en una seleccionándolos todos y descargándolos.

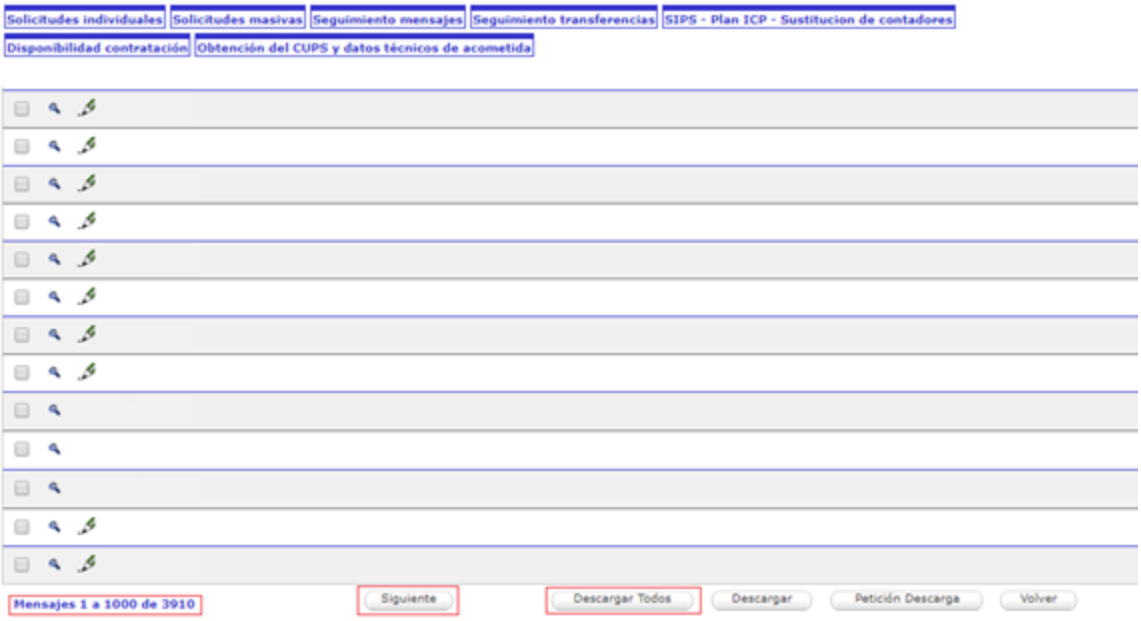

Figura 11: Interfaz con los resultados de la búsqueda.

Lo primero que se hace es usar una función propia para guardar el número total de mensajes que hay en este proceso, desde el cuadro de texto que se ve en la Figura 11, abajo a la izquierda. Con ello se puede controlar si hay más de una página de archivos disponibles y cuando se ha llegado a la página final.

En segundo lugar, se toma el número del último mensaje de la página y se pulsa el botón de *Descargar todos* (solo afecta a los de la página actual), con lo que se evita tener que ir seleccionando uno a uno todos los archivos. Después se comprueba si se ha llegado a la página final. En caso de que el número del último mensaje de la página coincida con el total de archivos, es que se ha llegado al final. Si no, se pulsa el botón *Siguiente* para avanzar.

Es importante que haya una forma de hacer esperar al robot a que se carguen los nuevos resultados tras pulsar el botón de avance. Para ello, se comprueba el número del último mensaje de la página y, mientras no se haya cargado la siguiente página y este número no haya cambiado, se usa la función *Sleep* para pausar el programa durante un segundo antes de hacer una nueva comprobación. En caso de que haya algún fallo durante la comprobación, el robot retrocede a la página anterior y vuelve a pulsar el botón *Siguiente*. Esta parte del código es importante ya que, si no se incluye, el robot volverá a pulsar el botón de *Descargar todos* antes de cargarse la información de la nueva página

y se duplicarían los últimos archivos adquiridos. Todos estos pasos se repiten mientras no se haya llegado a la última página disponible.

Un problema a tener en cuenta durante la implementación, debido al diseño de la página web, es que mientras se descarga el paquete de archivos, aparece una ventana emergente sobre la ventana principal. Esta solo contiene un mensaje que dice *Procesando* y un botón con el texto *Pulsar al finalizar la descarga*, pero es necesario cerrarla con cada descarga de mensajes.

Para solucionar este problema se ideó un sencillo código que recorre la lista de ventanas web abiertas en el buscador. Si la ventana actual no es la principal, la cual se le pasa como parámetro a la función, significa que es la emergente, con lo que se busca el botón para cerrarla y se cambia de nuevo a la principal.

Hasta ahora se han explicado las partes más relevantes de la implementación de las dos primeras fases del programa. A continuación, trataremos el código que almacena los archivos descargados en una base de datos.

Este fragmento de código se basa principalmente en hacer consultas a base de datos (ver Figura 4 en el capítulo 2.4). El primer paso es crear un nuevo lote en la tabla *Lote* y guardar su identificador. Entonces, si todos los procesos se han descargado correctamente sin error alguno, el robot ejecuta una rutina mediante la cual, se recorren uno a uno todos los archivos, desglosando la información que contiene y guardándola en las tablas *Lote en detalle* y *Lote con fichero*. Si este proceso se ejecuta correctamente, el estado del lote se actualiza a correcto. En caso contrario, el estado del lote pasa a incorrecto y se almacena un error en la carga de mensajes.

En el caso de que haya habido algún error durante la descarga de alguno de los procesos, el estado del lote pasa a incorrecto y se almacena un error en la descarga de mensajes y no se ejecuta la rutina de guardado de ficheros.

Mediante este sistema de tres tablas, toda la información queda bien ordenada y almacenada, con la ventaja de que, al guardar donde encontrar cada mensaje, su URL, en lugar de todo el mensaje en sí, se ahorra bastante espacio. Una vez descargada la información, los operarios que la necesiten la pueden coger de la ruta donde se ha

almacenado, mientras que en la base de datos solo se guarda lo esencial para poder buscar y recuperar lo que sea necesario.

Para este proyecto, como comprobación de que el robot se ha ejecutado con éxito, se ha desarrollado una consulta SQL simultánea a las tablas explicadas en el apartado 2.4, que permite ver si ha habido algún error durante la descarga de mensajes, durante su carga a la base de datos o si todo ha ido correctamente. Para ello compara el número de archivos descargados con el de almacenados y, si son diferentes, es que ha habido algún error durante su carga.

3 PRUEBAS Y DESPLIEGUE

En este capítulo se va a explicar todo lo relacionado con las pruebas de funcionamiento realizadas al robot y su despliegue, empezando por el plan de pruebas diseñado hasta la puesta en producción del programa.

3.1 ELABORACIÓN DEL PLAN DE PRUEBAS Y SU IMPLEMENTACIÓN

Durante la realización de este proyecto no se siguió un plan de pruebas propiamente definido, ya que en ningún momento la empresa lo requería. Pero sí que es cierto que el programa fue sometido a diversas pruebas manuales antes de su puesta en producción.

Para comprobar que todo iba correcto, a medida que se iba implementando el código se iba probando para corregir errores. Una vez que el robot estaba terminado y funcionaba bien en una situación normal, se fue ejecutando bajo situaciones especiales para ver qué pasaba. Por ejemplo:

- Al pasarle como parámetro una fecha no válida, el programa simplemente produce una excepción y finaliza. Lo mismo sucede si los parámetros de entrada no están en el orden establecido.
- Si se ejecutaba el robot para un día en el que el número de mensajes a descargar de un proceso es cero, el programa dejaba de funcionar (posteriormente se añadió la solución explicada en el capítulo 2.5, lo referente a la figura 9, para tratar este caso).
- Si se ejecutaba el robot para un día en el que el número de mensajes a descargar de un proceso es demasiado grande, el programa daba la ejecución por fallida (posteriormente se añadió la solución explicada en el capítulo 2.5, lo referente a la figura 10, para tratar este caso).

3.2 DESPLIEGUE

En este apartado se va a explicar el proceso que se sigue tras terminar el robot y comprobar que funciona correctamente, para ponerlo en producción. Este proceso, el cual ya estaba establecido por la empresa, consta de varios pasos:

1) Guardar el ejecutable del robot junto con las librerías que utiliza, en una ruta específica en la red de datos compartidos de la empresa, junto con el resto de los robots.
2) Crear un proyecto en Visual Studio de tipo *Integration Services (SSIS)*[6] que ejecute el programa de la ruta mencionada anteriormente. Un proyecto *Integration Services* es un flujo de tareas, donde se pueden concatenar todo tipo de labores (ejecución de programas, cargar y exportar información en base de datos, realizar consultas SQL, etc.) aplicando condicionales, filtros y otras variaciones. Este paso es necesario para poder programar la ejecución automática del robot todos los días a una determinad hora ya que, por experiencia propia de los trabajadores del Departamento de Mejora, si se trata de arrancar el programa directamente en lugar de hacerlo a través de un programa SSIS, el robot falla en ciertas ocasiones.
3) Guardar el ejecutable del SSIS en la misma ruta que el ejecutable del robot y sus librerías.
4) Crear una tarea programable de SQL Server que ejecute el programa SSIS de forma automática todos los días a la hora indicada.

Tras este proceso, el robot se ejecuta todos los días estipulados a la misma hora. Esto significa que no va a ser necesario que ningún empleado se preocupe por esta tarea. Además, como la hora de lanzamiento indicada es en horario nocturno, la ejecución del programa no va a ocupar el equipo de ningún empleado durante su jornada laboral.

[6] https://docs.microsoft.com/es-es/sql/integration-services/sql-server-integration-services?view=sql-server-ver15

4. CONCLUSIONES

La automatización de la descarga diaria de mensajes, mediante el concepto RPA (*Robotic Process Automatization*), produce un gran ahorro temporal, así como de recursos humanos y materiales. También se puede concluir que los robots que automatizan las descargas web son una herramienta totalmente fiable y eficiente, que mejora la calidad del producto final y reduce los errores humanos durante el proceso. Por último, la calidad de los puestos de trabajo también se ve notablemente aumentada debido a la automatización de esta tarea repetitiva y sin toma de decisiones.

4.1 TRABAJOS FUTUROS

Dado que las pruebas a las que se somete el robot no están establecidas y se van realizando sobre la marcha, una futura mejora que se podría añadir sería un conjunto de pruebas que, cada vez que el robot sea modificado, este tenga que superar. Estas pruebas podrían consistir en ejecutar el programa varias veces, para un día en el que se sepa bien cual es resultado final. Existiría al menos una iteración por cada situación fuera de lo común como que no haya mensajes, demasiados mensajes, credenciales incorrectas, etc. que se supone que pueda manejar el robot.

Una vez el programa haya superado las pruebas, quedaría demostrado que la modificación realizada no ha afectado negativamente a su funcionamiento.

5. BIBLIOGRAFÍA

[1] Alok Mani Tripathi. “Learning Robotic Process Automation”. Packt Publishing. 2018.

[2] Jorge Enrique Gudiño. “Programación WPF C#: Visual Studio 2019”. Publicación independiente. 2020.

[3] Fco. Javier Ceballos. “Microsoft Visual Basic.NET Curso de Programación”. “RA-MA Editorial”. 2017.

[4] Mario Rubiales Gómez. “Desarrollo Web: HTML, CSS y JavaScript”. “Grupo Anaya Publicaciones Generales”. 2017.

[5] Jérôme Gabillaud. “SQL Server 2016. Administrar Una Base De Datos Transaccional Con SQL Server Management Studio”. “Eni”. 2017.