Informe Proyecto de Pasantía:

Automatización de Procesos dentro de las Operaciones en We Techs para Resolución de Problemas

Estudiante: Martín Castro

Carrera: Ingeniería Civil Informática

Empresa: We Techs

Supervisor: Jorge Rojas

Profesor Guía: Víctor Nivia

Índice

1.1)Resumen Ejecutivo

El presente informe expone el Proyecto de Pasantía realizado en la empresa chilena We Techs, fundada el año 2012, se dedica a la gestión de fluidos industriales en cuatro principales verticales: APR (Agua Potable Rural), SSR (Servicios Sanitarios Rurales), Energía y Minería. Sus clientes son otras empresas a las cuales se les entrega un software que acompañado de dispositivos IoT y diferentes sensores, permiten a sus clientes medir y observar datos en tiempo real, además de tener la posibilidad de revisar datos históricos. Se identifica una oportunidad para mejorar flujos de trabajo relacionados a múltiples tareas repetitivas realizadas manualmente en el área de Operaciones de la empresa. El objetivo del proyecto es aumentar la eficiencia operativa y liberar HH en el área de Operaciones para que puedan utilizarse en tareas que entregan mayor valor. Esto se logrará a través de la implementación de la automatización de tareas, la cual se hará mediante programación Python y utilización de herramientas de Microsoft Power Platform. El proyecto contempla desarrollar e implementar soluciones para 3 problemas: 1) La generación, revisión, formato y envío de reportes; 2) Mantención de Uptime e identificación de desconexiones; 3) Recolección de información de salidas a terreno. Luego de la implementación se observa una liberación aproximada de 48 horas mensuales, sumado a otros beneficios. Los trabajadores beneficiados y la empresa quedan satisfechos con los resultados del proyecto, con interés de expandirlo en el futuro.

1.2)Abstract

This report presents the Internship Project conducted at the Chilean company We Techs, founded in 2012. The company specializes in management of industrial fluids across four core business lines: Rural Drinking Water (APR), Rural Sanitation Services (SSR), Energy, and Mining. Its clients are other companies to whom We Techs provides software accompanied by IoT devices and various sensors. This technology allows clients to measure and observe their data in real-time, as well as review historical data. An opportunity was identified to enhance workflows related to multiple repetitive tasks manually performed in the company's Operations department. The project's goal is to increase operational efficiency and free up manpower in the Operations department for tasks that deliver greater value. This will be achieved through the implementation of task automation using Python programming and Microsoft Power Platform tools. The project aims to develop and implement solutions for three issues: 1) Report generation, format, review, and delivery; 2) Maintenance of Uptime and identification of disconnections; 3) Recollection of information from visits to clients. Following the implementation approximately 48 hours are saved monthly, along with other benefits. Both the benefited employees and the company express satisfaction with the project's results, showing interest in expanding it in the future.

2) Introducción

La pasantía se realizó en We Techs, empresa chilena fundada en el año 2012. Se dedica a la gestión de fluidos industriales y se concentra en las siguientes líneas de negocio: APR (Agua Potable Rural), SSR (Servicios Sanitarios Rurales), Energía y Minería. Está principalmente enfocada en el agua, pero también aborda otros sectores como la gasolina y el gas. Sus clientes más importantes son del sector minero, pero también ofrecen sus servicios a empresas que distribuyen estos recursos y más. El servicio que ofrecen está compuesto por un software propio y dispositivos IoT que permiten a sus clientes medir y observar en tiempo real datos relacionados a su operación, como niveles de tanques de agua, caudales, presión de gases, entre otros. La pasantía se realizó en el área TI de la empresa, la cual se compone de 4 células de trabajo: Producto, encargada del desarrollo de nuevos productos y mantención de los softwares existentes de la empresa; IoT, trabaja en el desarrollo de la parte sensorial del servicio, directamente con microcontroladores, sensores y también con la manera en la que estos transmiten la data obtenida; Data y Analítica Avanzada, la célula más nueva del área, se enfoca en Machine Learning, especialmente herramientas de predicción/forecasting que se buscan implementar de cara al cliente; DevSecOps, es la encargada de manejar la infraestructura cloud de la empresa que se encuentra en AWS.

En conversaciones con el área de Operaciones se identificaron una serie de tareas de carácter repetitivo, se pueden ver compiladas en la Tabla 1, acompañadas del impacto causado en la empresa el hecho de que la tarea se haga de manera manual.

A partir de lo observado en la empresa, se identifica una oportunidad de encontrar una mejor forma de hacer las cosas, buscando una solución que permita aumentar el valor de las horas hombres dedicadas a estos procesos, permitiendo que los empleados destinen ese tiempo a algo más significativo, como por ejemplo un análisis más profundo de los datos entregados a los clientes, resolución de problemas y más.

Una vez identificada la lista de tareas manuales que estaban afectando de diferentes maneras al área de Operaciones, se concluyó que el tiempo no era suficiente para solucionarlas todas, por lo que se generó una instancia de diálogo y análisis con las personas del área, con el objetivo de priorizar la lista y seleccionar las tareas que formarían parte del proyecto. Como resultado de esa reunión se seleccionaron las siguientes tareas (algunas combinaciones de tareas relacionadas):

- Generación y Envío de Reportes.
- Investigación de Desconexión de Dispositivos.
- Recolección de Información de Salidas a Terreno.

Tarea	Área	Frecuencia	Impacto
Investigación de Desconexión de Dispositivos	Servicio	Semanal x2	El trabajador se entera de desconexiones cuando los clientes se dan cuenta que no están recibiendo datos. Se demora más de una hora en identificar manualmente el/los dispositivos/s desconectados.
Envío de SMS para reset de Dispositivos	Servicio	Semanal x4	Se hace muy seguido, muchas veces es lo que se necesita para solucionar una desconexión, implicando que mientras no se haga, el dispositivo continuará desconectado.
Manejo de Mapas de Memoria en software propietario	Automatización y Hardware	Semanal x1-2	No está estandarizado, complica el trabajo de las personas involucradas en su manipulación.
Agregar nuevas variables medidas al software	Automatización y Hardware	Variable	Cuando se agregan nuevas variables, estas se deben registrar en 2 lugares diferentes de la plataforma, olvidarlo o hacerlo mal lleva a errores.
Visualización de detalles de equipos instalados en un punto de monitoreo	Operaciones, Reportabilidad	Variable	Es información que se necesita seguido, pero está separada en varias fuentes de datos diferentes, tomando tiempo cuando a veces se requiere obtener la información con apuro.
Creación de Fórmulas	Operaciones	Variable	Complicado, mucho tiempo invertido.
Recolección de Información de salidas a terreno	Operaciones	Diario	Actualmente se hace con lápiz y papel. Documentos físicos pueden perderse o guardarse de manera incorrecta, dificultando el seguimiento.
Configuración de Tableros	Automatización y Hardware	Variable	La configuración base es siempre igual para todos los tableros instalados, sin embargo, se hace de manera manual para cada nuevo tablero.
Reporte A: Generación de Portadas y Subportadas	Reportabilidad	Mensual	La tarea es exactamente la misma cada vez e impide al trabajador de realizar sus labores regulares.
Reportes B y C: Formato, revisión y envío.	Reportabilidad	Semanal x2	Toma mucho tiempo y es muy sensible a error humano, posiblemente entregando datos erróneos al cliente.
Generación de Reporte D.	Reportabilidad	Semanal	

Tabla 1: Principales tareas manuales que se realizan en el área de Operaciones.

3) Objetivos

Objetivo General:

Aumentar la eficiencia operativa y reducir el uso de HH en tareas repetitivas y/o mecánicas de la empresa We Techs mediante un desarrollo e implementación de automatización de procesos en un periodo de 4 meses.

Objetivos Específicos:

1. Seleccionar las herramientas de automatización necesarias para implementar los procesos identificados. La investigación y elección de las mejores herramientas para abordar cada proceso es esencial tanto para el correcto desarrollo del proyecto, como para su vida luego de finalizar el proyecto.
2. Diseñar y desarrollar la automatización de los flujos de trabajo de los procesos seleccionados. La implementación de las soluciones automatizadas debe ser efectiva y eficiente, es decir, las soluciones deben tener un impacto positivo en el trabajador, haciendo la tarea en cuestión más fácil e intuitiva de realizar, en lugar de implementar una solución posiblemente confusa que incluso puede terminar impactando negativamente la ejecución de la tarea, por ejemplo, aumentando el tiempo que toma en realizarse. Además, se busca cambiar lo menos posible los hábitos de las personas involucradas, fuera del proceso en cuestión.
3. Las soluciones desarrolladas se deben implementar y efectivamente utilizarse en los procesos de la empresa.

4) Estado del Arte

4.1) Introducción

La automatización de procesos es la imitación mediante tecnología de un agente humano con el objetivo de mejorar la eficiencia operativa, reducir costos, minimizar errores y liberar recursos humanos. Se pueden automatizar procesos basados en reglas que involucran tareas rutinarias, datos estructurados y resultados deterministas (Aguirre & Rodríguez, 2017). En otras palabras, si se quiere implementar automatización en un proceso o tarea, estos deben cumplir con una serie de requerimientos. Deben ser rutinarios, no tiene sentido automatizar cuando el trabajo en cuestión se realizará una sola vez. Los datos involucrados en el proceso deben ser estructurados y siempre presentarse de la misma manera, en el caso contrario la solución puede complicarse mucho si la automatización debe estar preparada para tratar con datos que pueden verse de múltiples maneras. Por último, deben tener resultados deterministas, esto quiere decir

que, así como los datos deben estar estructurados y bien definidos, también lo debe estar el resultado final del proceso o tarea.

La automatización de procesos puede traer un gran número de beneficios a una empresa que la implementa. La empresa colombiana Consensus, dedicada a la solución de problemas tecnológicos e informáticos de empresas tanto en Colombia como Estados Unidos, declara una serie de beneficios que otorga la automatización de procesos (Consensus, 2020), mostrándonos la importancia de esta y una idea de los impactos que puede tener en una empresa. A continuación, se muestran algunos de los más relevantes:

- Prioriza tareas que mantienen la satisfacción del cliente, es decir, permite a los trabajadores estar más disponibles para clientes en situaciones de recepción y ser más eficiente en los procesos administrativos y así garantizar una mayor satisfacción a las expectativas del cliente afectando su experiencia positivamente.
- Asume tareas humanas repetitivas y de esta manera permite a los trabajadores centrarse en tareas de alto valor.
- Hay mayor motivación y aumento de la productividad.
- Menores costos contables y de gestión que garantizan un alto retorno de la inversión.
- Reducirá y/o suprimirá los errores manuales.
- Reduce el esfuerzo manual y el tiempo para ingresar o extraer datos.

Se puede concluir que la automatización de procesos efectivamente será de utilidad para reducir las brechas encontradas y resolver los problemas encontrados en la empresa, ya que varios de los problemas identificados son respondidos por alguno de los beneficios señalados.

4.2) Tecnologías

Existen diferentes tipos de automatización, sin embargo, los relevantes para este proyecto son dos; automatización programable y RPA (automatización robótica de procesos). La primera se refiere a las automatizaciones realizadas mediante el uso de un lenguaje de programación. Por otro lado, RPA es un tipo de software que tiene como objetivo automatizar un proceso, optimizar el tiempo de ejecución, minimizar los errores, todo esto en un conjunto de costo – beneficio que es rentable para el usuario (Bermúdez, 2021). En términos de RPA, cuando se habla de robot no se hace referencia a un robot físico, si no a un software que lleva a cabo tareas. La gran diferencia entre estos dos tipos de automatización es que RPA implica la utilización de un software desarrollado por terceros, con una interfaz gráfica para el

desarrollo de las soluciones automatizadas, por lo que no es un requisito tener conocimientos en programación. Mientras que de la otra manera la automatización se hace de forma “manual”. Existen una multitud de empresas que ofrecen software de RPA con el uso de una licencia, sin embargo, este proyecto no va a utilizar estas soluciones, se dedicará a la automatización mediante programación. Una de las principales razones para esto es que RPA puede ser más simple debido a la utilización de una interfaz gráfica, pero puede complicarse mucho si en algún punto se desea hacer algo que no está disponible nativamente como una funcionalidad del software.

Una elección de lenguaje de programación muy popular mundialmente es Python, por las siguientes razones (Sayeeth, et al. 2019):

- Simplicidad.
- Eficiencia.
- Utilidad para Big Data, Machine Learning y Desarrollo Web.
- Comunidad Activa.

Python tiene una utilización muy amplia, incluyendo grandes empresas del sector tecnológico como: IBM, Google, HP, Dropbox, y Cisco. Python es también una buena elección para desarrollos de automatización. Debido a que sus interfaces internas permiten trabajar con servicios del sistema operativo, haciéndole un lenguaje útil para la programación de sistemas. La librería estándar de Python puede interactuar con una variedad de plataformas, sistemas operativos y contiene herramientas para trabajar con recursos como variables de entorno, archivos, sockets, procesos, línea de comandos, etc. (Srinath, 2019).

Una tecnología relativamente nueva que está volviéndose cada vez más popular es el concepto de Serverless, un modelo de cloud computing que permite construir y correr aplicaciones sin la necesidad de proveer o administrar un servidor. El servicio serverless que provee AWS se denomina Lambda, en el que se destacan características como: ejecución de funciones en la nube, potencial de automatización dentro de AWS y desarrollo de flujos de trabajo basados en eventos (Sbarski & Kroonenburg, 2017). Combinando este servicio con AWS EventBridge, se pueden desarrollar funciones de código que pueden ser programadas para ser ejecutadas luego de detectarse un evento determinado, como una periodicidad definida arbitrariamente (ej. semanalmente) e incluso eventos producidos por otros servicios de AWS.

5.1) Propuestas de Solución

5.1.1) Sistema Automatizado de Reportes.

Este sistema busca ayudar tanto a los trabajadores del área de Operaciones como a los de TI que están involucrados en el proceso de generar, revisar y enviar diversos reportes. Lo que se busca es reducir al mínimo la intervención humana en el flujo de los reportes. Para esto se proponen 2 soluciones.

Antes del comienzo de este proyecto, los reportes son ejecutados de manera manual a partir de un Jupyter Notebook, un entorno interactivo que organiza código en celdas, esto permite la ejecución de código y visualización de resultados por partes, utilizando el lenguaje Python. Además de ser ejecutado manualmente, se deben modificar las fechas de inicio y término del reporte, indicando el margen de datos que debe ser incluido. Luego, la persona debe esperar a que se ejecute la generación del reporte, para después descargar el archivo Excel, revisarlo, formatearlo y volver a subirlo al servidor de Jupyter para ejecutar código que envía el reporte por correo. La persona que lo recibe debe revisar manualmente que los datos estén dentro de rangos “normales” establecidos, marcando anomalías. El proceso descrito se repite para varios reportes. Como solución, se propone descartar la utilización de Jupyter Notebooks e implementar Python en forma de scripts tradicionales, esto permite ejecutar el código de manera arbitraria y, además, se hará tanto el formateo del archivo como la revisión de datos programáticamente, evitando así que tenga que ser realizada por una persona.

La gran diferencia entre las soluciones propuestas es su infraestructura. Se propone lo siguiente:

1. **AWS Lambda:** Servicio serverless de AWS. La tecnología serverless permite ejecutar código sin la necesidad de que este viva en un servidor. El código ejecutado puede estar en una variedad de lenguajes, entre ellos Python, el estándar de la empresa. Se crearía una función Lambda para cada uno de los reportes, las cuales pueden ser ejecutadas de manera programada utilizando AWS EventBridge. Una de las limitantes de Lambda es que, debido a no tener un servidor, no posee memoria persistente, esto quiere decir que cada ejecución de la función es completamente independiente, y no se puede acceder a información generada por otras ejecuciones de la misma función.

2. **Máquina Virtual:** La segunda propuesta implica el uso de una máquina virtual AWS EC2, en la cual estarán todos los scripts que generan reportes, centralizándolos en un mismo lugar. Luego se utilizaría el crontab del sistema para programar la ejecución de reportes en sus horarios y fechas indicadas. Crontab es una herramienta incluida en sistemas basados en UNIX que permite programar temporalmente la ejecución de comandos y scripts de bash. A diferencia de AWS Lambda, una máquina virtual sí tiene persistencia de memoria, lo que permite generar y modificar archivos que pueden compartirse entre ejecuciones de creación de un reporte.

5.1.2) Uptime y Desconexiones de Dispositivos.

Se busca que la empresa pase de tener un soporte reactivo, a un soporte proactivo. Actualmente sucede que la empresa se entera de que dispositivos están desconectados cuando clientes los contactan para preguntar por ellos al notar inconsistencias en sus datos, esto sucede porque actualmente no existe una manera simple de identificar dispositivos desconectados. Se deben revisar uno por uno para luego tomar acción sobre estos, lo que muchas veces es tan simple como enviar al dispositivo un comando de reinicio.
La primera solución propuesta es la siguiente:

1. Utilización de AWS Lambda para hacer un chequeo periódico de la base de datos y así identificar los dispositivos que llevan una determinada cantidad de tiempo sin transmitir datos. Con esto se puede enviar un correo a la persona encargada para notificar la desconexión del dispositivo.

La segunda propuesta surge luego de haber tenido conversaciones con los trabajadores del área de Soporte:

2. Notificaciones de desconexiones a la plataforma Freshdesk, la plataforma web usada por Soporte para el manejo de tickets. En esta solución, en lugar de notificar al trabajador directamente a su correo, se registrarán las desconexiones en forma de tickets, donde luego pueden ser procesados de manera simple usando la experiencia que ya tiene usando la plataforma. Además, en lugar de usar AWS Lambda, se puede aprovechar un módulo de reglas que ya existe en el software de la empresa.

5.1.3) Recolección de Información de Salidas a Terreno.

Actualmente, la información de salidas a terreno (por ejemplo, órdenes de trabajo) se hace de manera manual. Los trabajadores imprimen y rellenan hojas de papel con la información necesaria. Esto no es consistente con la imagen tecnológica que quiere transmitir la empresa, además de costarle tiempo a los empleados.

1. La primera propuesta de solución es pasar del papel y lápiz a un software de terceros que permita la edición directa de los documentos. Entre las posibilidades están Adobe Acrobat Pro o SmallPDF.
2. La segunda propuesta involucra desarrollar una forma de recolectar la información. Esta solución constaría de 2 principales partes:
    a. **Obtención de la información:** Desarrollo de una plataforma donde se pueda ingresar toda la información en un solo lugar. Esta parte se desarrollaría usando una plataforma low/no-code, debido al tiempo limitado que se tiene para el proyecto.
    b. **Procesamiento y Llenado de Plantillas:** Una vez hecho el ingreso de datos, estos serán manejados con Python para rellenar plantillas estandarizadas para cada documento, los cuales luego serán enviados a las personas necesarias, se almacenarán de manera que los trabajadores puedan tener acceso a ellos y poder hacer un seguimiento de las actividades realizadas en terreno.

5.2) Soluciones Escogidas

5.2.1) Sistema Automatizado de Reportes.

Se decide implementar la solución basada en una máquina virtual. Se opta por ésta porque la empresa ya tiene una máquina virtual disponible con usos varios, por lo que no habría un costo nuevo para la empresa. Un problema que se podría tener tiene que ver con los recursos computacionales que usa la generación de reportes, sin embargo, puede solucionarse programando la ejecución de reportes para horas en las que la máquina no hace nada. Otra razón para elegir esta solución es la memoria persistente que no se tiene si se utiliza Lambda. Algunos reportes requieren incluir datos históricos, sin memoria persistente se tendría que obtener nuevamente esa información cada vez que se genera un reporte. De la manera elegida se pueden generar archivos reutilizables que contengan esta información.

5.2.2) Uptime y Desconexiones de Dispositivos.

Se elige la solución que hace notificaciones mediante tickets a Freshdesk. Tiene más sentido utilizar esta plataforma donde la información puede estar centralizada en lugar de estar constantemente enviando correos al trabajador. Además, la plataforma es accesible por múltiples personas, por lo que no sería un problema que el trabajador típicamente responsable de esta tarea no esté disponible.

5.2.3) Recolección de Información de Salidas a Terreno.

Para esta tarea se implementa la solución que implica el desarrollo de una aplicación, utilizando Power Apps, ya que la empresa ya posee licencias para las herramientas de Microsoft, y se adapta muy bien a lo que se necesita para la recolección de información, especialmente porque una de las cosas que se incluyen en los documentos generados es la firma, cosa que puede lograrse fácilmente con dicho software. Hay problemas con la otra propuesta de solución que la hacen más difícil de justificar. Una de las dificultades que tiene el proceso de recolección de datos es que una vez generados los documentos no existe una manera estandarizada de almacenarlos en la que fácilmente se pueda recuperar información de salidas a terreno pasadas. La solución elegida nos permite implementar una organización de los documentos automática una vez que estos sean generados con la información ingresada a la plataforma por el usuario.

6) Evaluación Económica

La evaluación económica del proyecto fue realizada con las siguientes consideraciones:

- Se usa una tasa de descuento de 9,5%.
- Los costos fijos son: la remuneración recibida, y una licencia que permite la publicación de la aplicación. Las otras licencias necesarias ya se tenían en la empresa, por lo que no incurren un costo adicional asociado al proyecto. La máquina virtual en AWS utilizada ya existe como parte de la infraestructura del proyecto, por lo que tampoco genera un costo adicional.
- Como inversión se considera una cotización por una Tablet, lápiz y teclado para utilización de la aplicación de generación de documentos.
- Para el cálculo del beneficio en HH ahorradas se consideró un sueldo promedio de $1.500.000 CLP.
- El beneficio económico generado por la automatización de reportes se cuenta a partir de su implementación en el mes 3. Mientras que el beneficio debido a las automatizaciones relacionadas a las desconexiones de dispositivos y la generación de documentos se suman a partir de su implementación realizada en el mes 4.

- La solución a desconexiones de dispositivos permite a la empresa responder a requerimientos crecientes sin la obligación de contratar a un nuevo Técnico en Soporte, generando un beneficio en el ahorro de $786.000 CLP mensuales, remuneración que recibiría esa persona.

	Mes	1	2	3	4	5
Remuneración	$300,000	$300,000	$300,000	$300,000	$300,000	
Tablet + Lápiz + Teclado	$329,990	$329,990				
Licencia Power Apps (20 USD)					$17,418	$17,418
Beneficio (HH ahorradas)				$122,093	$313,953	$313,953
Ahorro de Costos (Sueldo de Técnico)					$786,000	$786,000
Flujo		$629,990	$300,000	$177,907	$782,535	$1,082,535
Tasa de descuento	0.095					

Tabla 2: Flujo de Caja

Luego de un análisis económico del proyecto, se determina que este empieza a tener un VAN positivo a partir del mes 5, momento en el que empieza a ser económicamente rentable, con una TIR de 19%.

7) Metodologías

Para el cumplimiento de los objetivos específicos mencionados, se proponen las siguientes metodologías respectivamente:

1. El cumplimiento del primer objetivo requiere un periodo de investigación. Hoy en día existen múltiples maneras de abordar la automatización, variados softwares para generarla, o se puede desarrollar individualmente usando programación. Además, se debe hacer un análisis de cómo abordar la automatización, donde 'vive' el proceso a automatizar, computadores de los empleados, software propietario de la empresa, servidor, etc. Al planificar la implementación, se deben considerar estos factores.
2. Para el segundo objetivo, lo más importante es la planificación y organización de cómo trabajar el proyecto en el tiempo dedicado a la pasantía. Para lograrlo se utilizará una metodología de desarrollo ágil, en particular, Kanban. La agilidad se basa en hacer pequeñas entregas, frecuentes e incrementales, recibiendo validación y feedback de parte de las personas que se verán impactadas por el proyecto, permitiendo hacer cambios pertinentes antes de que sea demasiado tarde. Se generará una lista priorizada de tareas a completar, actuando como el backlog. Uno de los beneficios de Kanban, es que fuerza a poner un límite de tareas 'activas', lo que evita una sobrecarga por tratar de abordar demasiadas tareas al mismo tiempo.

3. Para asegurarse de que los desarrollos realizados durante el proyecto sean efectivamente implementados y usados a largo plazo en la operación de la empresa, se redactará documentación tanto del funcionamiento como uso de las soluciones, acompañado de sesiones de capacitación donde se aclararán dudas que puedan surgir a raíz del uso ellas. Además de hacer monitoreo continuo, realizando los ajustes necesarios una vez se haga la implementación.

8) Medidas de Desempeño

Las medidas de desempeño que permitirán medir el éxito del proyecto son las siguientes:

1. **Horas Hombre liberadas:** Se dejará registro de cuantas horas hombres se destinan para cada tarea y tomando en cuenta su frecuencia, determinar cuántas H.H. mensuales se destinan a cada tarea. Luego, podemos hacer una comparación de antes y después del proyecto, observando cuántas horas hombre mensuales se dedican a las tareas luego de ser automatizadas.
2. **Potenciales beneficios:** Beneficios, de naturaleza tanto económica como otras, en la empresa gracias a la automatización de los procesos manuales.

9) Desarrollo

La planificación general de las diferentes soluciones del proyecto se puede ver en el Anexo 13.2. Para el transcurso del proyecto se definen los siguientes hitos:

9.1) Hitos Relevantes

A continuación, se presentan una serie de hitos relevantes para el proyecto, que permitirán tener un control sobre el progreso de este, acompañado de la/las personas de la empresa que estarán encargadas de validar su completitud:

1. **Selección de Procesos para Automatizar:** Se identifican y priorizan los procesos que serán objeto de automatización, validada con la líder de Operaciones
2. **Diseño:** Diseño detallado de los flujos de trabajo automatizados para los procesos seleccionados, validado con la persona más cercana al proceso.
3. **Primer Desarrollo:** Se genera un MVP para poder validar con el área, validado con el Supervisor de la pasantía.
4. **Desarrollo Completo**: Consideración del feedback recibido y finalización del desarrollo de un proceso automatizado, validado con la misma persona que validó el diseño de la solución.
5. **Implementación:** Puesta en marcha de un proceso automatizado en la empresa, validada con el Supervisor.

6. **Evaluación Final del Proyecto:** Evaluación del proyecto en su conjunto, incluyendo el logro de los objetivos generales y específicos, validada por 3 evaluadores cercanos al proyecto.
7. **Documentación y Entrega de Resultados:** Traspaso final de las soluciones a la empresa, acompañada de toda la documentación relevante en cuanto a su desarrollo, mantención y funcionamiento, validada con el Supervisor de la pasantía.

9.2) Sistema Automatizado de Reportes

El desarrollo del sistema automatizado de reportes inicia con la creación de un repositorio Git (Anexo 13.2.1) para control de versiones. Esta herramienta permite la gestión de cambios en código, trazabilidad y ramificaciones, facilitando el desarrollo, experimentación/testeo e integración de nuevas características a un software.

A continuación, se configuró un ambiente de desarrollo local estandarizado, lo que tiene una serie de ventajas por sobre un desarrollo sin entorno: consistencia, es reproducible, integración continua, reducción de errores, entre otros. Este entorno de desarrollo se genera utilizando la herramienta Docker, la cual nos permite crear contenedores, una forma muy básica de una máquina virtual que inicialmente posee lo mínimo necesario para funcionar, y se agregan solamente las herramientas necesarias. Teniendo en cuenta que una vez implementada la solución, ésta se va a encontrar en una máquina virtual, Docker puede utilizarse para replicar las condiciones de ese entorno para el desarrollo local, si el software funciona dentro del contendor, lo más seguro es que no haya problemas cuando se encuentre en la máquina virtual.

```
# Python Version
FROM python:3.10-slim

# Set working dir
WORKDIR /src

# Copy and install dependencies
COPY ./deps.txt /src
RUN pip install -U pip
RUN pip install -r deps.txt

# Copy files
COPY . /src
CMD ["bash"]
```

Ilustración 1: Dockerfile generado para desarrollo local.

La ilustración 1 muestra la configuración del entorno de desarrollo. Se usa como imagen base una predefinida por Docker hecha para el trabajo con la versión 3.10 de Python, la misma que está instalada en la instancia EC2 en AWS. Luego se utiliza PIP, el administrador de paquetes por defecto de Python para instalar las dependencias necesarias con su versión especificada en el archivo deps.txt.

La tarea de generar y enviar reportes no se encontraba estandarizada dentro de la empresa. Los reportes se corrían individualmente en los computadores de trabajo de los empleados, todos con entornos diferentes que, sin un ambiente estándar, debido a esto las versiones de Python y librerías utilizadas eran diferentes entre reportes, lo que requirió un trabajo de migración a Python 3.10 (la versión instalada en la máquina virtual) y fijación de versiones de librerías.

En la migración de estos reportes se encontró que en diversas partes se repetía lógica similar entre reportes, por lo que todas estas secciones se extrajeron para la creación de funciones de utilidad que pueden usarse entre los reportes existentes, haciendo el código más legible y facilitando el desarrollo de nuevos reportes cuando sea necesario (Anexo 13.2.2.1).

Una vez estaba funcionando la ejecución de reportes, llevar el código a la máquina virtual consistía simplemente en acceder a ella mediante SSH y clonar el repositorio Git. No fue necesario levantar una nueva máquina virtual para la implementación de la solución, la empresa ya contaba con una instancia AWS EC2 de poco uso en la que se podía instalar el servicio de generación de reportes.

Para la ejecución programada de la generación de reportes se aprovechó la herramienta Crontab. Ésta es parte de sistemas operativos basados en Unix y permite ejecutar tareas en momentos específicos. La ilustración 2 muestra el contenido del crontab, en él se definen tareas a ejecutarse de manera recurrente y su periodicidad. Cada línea empieza definiendo que tan seguido se debe ejecutar la tarea con 5 parámetros: minuto, hora (UTC), día del mes, mes y día de la semana, respectivamente. Por ejemplo, en la ilustración se definen 3 tareas a ejecutarse a las 9 de la mañana del horario chileno, todos los lunes.

Ilustración 2: Crontab de la máquina virtual, con partes censuradas para mantener la confidencialidad.

Finalmente, se define lo que debe ocurrir cuando se cumpla esa periodicidad, para esto se generaron scripts de bash que levantan el entorno virtual que contiene las dependencias necesarias y luego ejecuta el reporte.

```
#!/usr/bin/bash

cd /home/ubuntu/reportes-automatizados/
source env/bin/activate
export $(grep -v '^#' .env | xargs -d '\r')
echo "Ejecutando Reporte Adjudicaciones"
python reporte-indicadores-adjudicacion.py
cd ~
```

Ilustración 3: Ejemplo de script bash que levanta el entorno virtual, establece variables de entorno y ejecutan el reporte.

9.2.1) Diagrama de la Solución

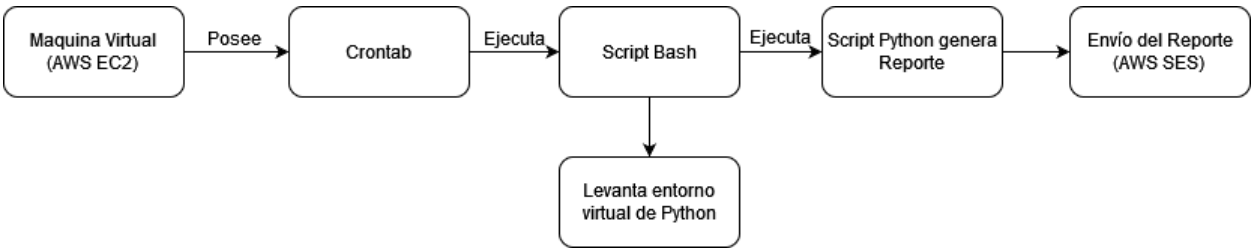

Ilustración 4: Muestra en forma de diagrama el funcionamiento del Sistema Automatizado de Generación y Entrega de Reportes.

9.3) Uptime y Desconexiones de Dispositivos

La implementación de esta solución integra diferentes servicios y funcionalidades existentes en la empresa de una manera nueva. El producto actual de software que We Techs ofrece a sus clientes ya tiene un módulo de reglas/alarmas, el cual puede aprovecharse para no tener que desarrollar manualmente una solución nueva. Este módulo permite crear reglas que observan la base de datos según los parámetros que se le definieron, por ejemplo, enfocarse en una variable como el nivel de agua de un pozo, luego se define una condición, para el ejemplo dado podría ser un valor umbral para el pozo. Finalmente, se define una acción a ejecutar cuando se cumpla esa condición, si esa acción es una notificación, puede ser por SMS, llamada o correo, cuando la acción de una regla es una notificación, se considera que es una alarma.

La otra parte de esta solución es el software utilizado por Soporte para el manejo de tickets, Freshdesk. La empresa ha configurado un correo al cual un cliente puede enviar correos, y se generará automáticamente un ticket en la plataforma con su contenido.

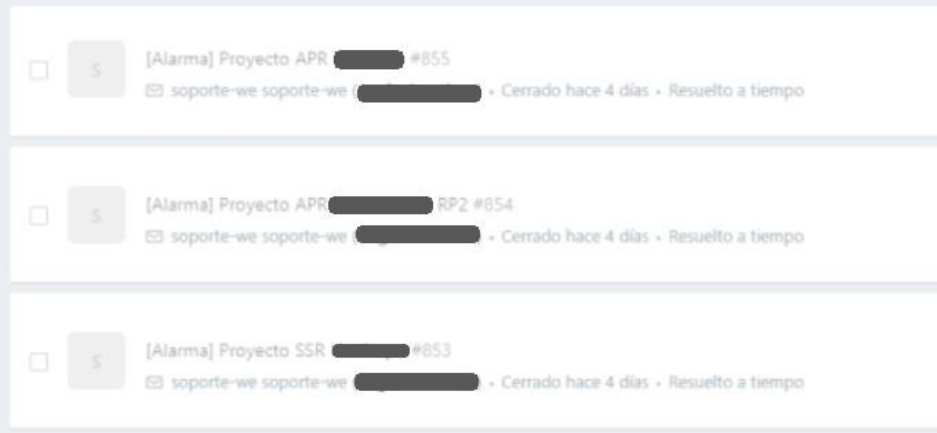

Ilustración 5: Visualización de tickets de desconexión en Freshdesk, aparecen opacos porque ya fueron marcados como resueltos.

Para la automatización de esta tarea se utilizó un script que crea e ingresa las reglas/alarmas para cada dispositivo a la BBDD, estas alarmas harán la notificación al correo electrónico vinculado con Freshdesk cuando se detecte que un dispositivo no ha transmitido datos por un periodo de 3 horas, así generando un ticket en la plataforma por cada desconexión detectada.

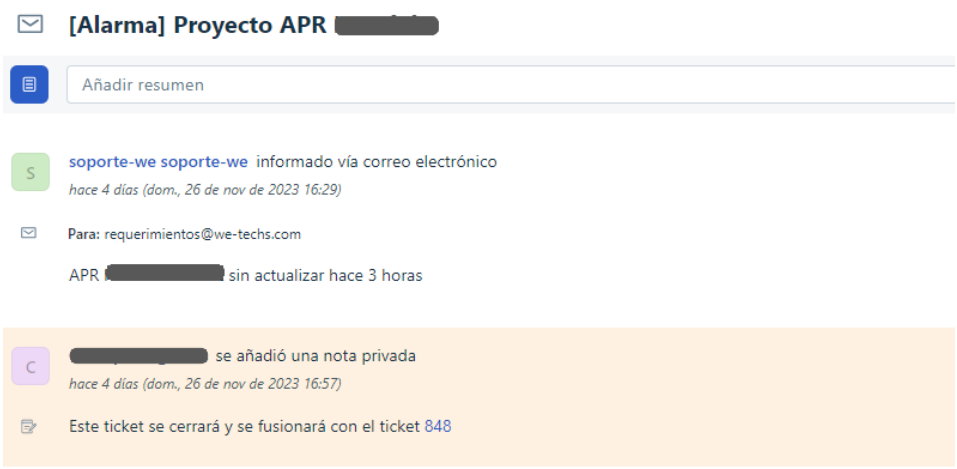

Ilustración 6: Ejemplo de ticket, da aviso de un dispositivo que lleva 3 horas sin transmitir información. Una vez solucionado, se cierra el ticket.

9.3.1) Diagrama de la Solución

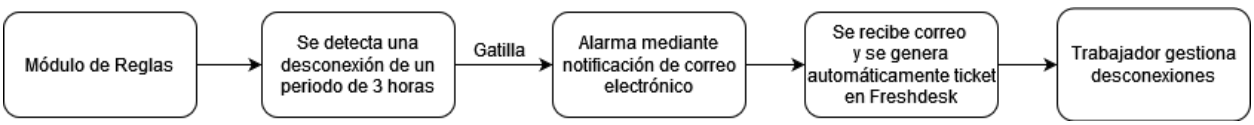

Ilustración 7: Flujo de alarmas automatizadas al detectarse un dispositivo desconectado

9.4) Recolección de Información de Salidas a Terreno

El desarrollo de esta solución se divide en dos partes: Frontend y Backend. El primero se refiere a la parte del software que está de cara al usuario y con la que este puede interactuar directamente. El segundo se refiere a todo lo que el usuario no ve directamente.

9.4.1) Frontend

Para el desarrollo de esta parte de la solución se decidió utilizar una plataforma low-code, particularmente, Microsoft Power Apps, por 3 principales razones: la primera es la restricción de tiempo en el marco de la pasantía, desarrollar una interfaz de manera tradicional hubiese requerido más tiempo del que se disponía para la realización del proyecto, la segunda es debido a su integración con las otras herramientas de Microsoft que son de uso estándar en la empresa, como Teams y SharePoint, por último, las soluciones Power Apps son automáticamente multiplataforma, utilizables desde la web, dispositivos Android, Apple y más. Para asegurar un buen desarrollo y resultado se validó tanto con la Diseñadora UI/UX como con las personas que serían usuarios del software de manera regular.

Ilustración 8: Primera vista al abrir la aplicación. El usuario elige qué documento desea crear.

9.4.2) Backend

El desarrollo del backend comienza con la creación de un repositorio Git que contendrá el código de las funciones AWS Lambda y las utilidades/dependencias comunes entre ellas (Anexo 13.2.1). El siguiente paso constó de configurar un entorno local de desarrollo, para esto se aprovechó nuevamente la herramienta Docker, de manera que el código se ejecute en un ambiente aislado y controlado. A esto se sumó la plataforma Serverless Framework, que simplifica el desarrollo y despliegue de aplicaciones

serverless, esto se refiere a aplicaciones donde el desarrollador no está involucrado en la gestión de la infraestructura. La combinación de estos 2 software permite simular localmente la ejecución de funciones Lambda y un servicio AWS API Gateway. Este último es un servicio de AWS para la creación, publicación y más de APIs. Podemos combinarlo con AWS Lambda para ejecutar funciones a partir de peticiones HTTP. Este entorno de desarrollo permite ejecutar las funciones de manera muy similar al entorno de producción.

En paralelo, se organizaron reuniones con las personas de Operaciones involucradas en el proceso, para la identificación de los documentos que se generan habitualmente, luego se hizo una priorización, eligiendo los siguientes tres documentos para ser incluidos en la solución generada en el plazo de la pasantía:

Documento	Función
Orden de Trabajo	Autorizar, describir, y gestionar trabajos realizados en salidas a terreno. Proporciona detalles sobre la labor a realizar, recursos necesarios, resoluciones y otros.
Checklist de Camionetas	Registro del estado físico y la valides de los documentos legales de las camionetas usadas en salidas a terreno.
Análisis de Riesgos en el Trabajo (ART)	Dar cuenta de todos los posibles riesgos al trabajador en una salida a terreno, en conjunto con las medidas a tomar para su prevención.

Tabla 3: Describe la función de los documentos seleccionados para implementación

Luego de configurar el entorno de desarrollo se desarrollan las funciones Lambda. La planificación contempla el desarrollo de una función Lambda individual para cada documento generado. Con el objetivo de generar un MVP se empezó con la función que generaría documentos de Órdenes de Trabajo.

Para la creación de documentos PDF se creó una plantilla a partir del documento que se usa actualmente, se configuraron campos de input utilizando el software LibreOffice Draw, un editor de documentos/gráficos de código abierto. A cada campo se le asigna un tipo de contenido: número, fecha, hora, etc. Una vez creada esta plantilla a partir de la cual se generarán los documentos, se programa la función que rellena la plantilla a partir de un input: una estructura llave-valor, donde la llave es el nombre del campo a rellenar y el valor es el dato correspondiente a dicho campo.

Para la integración entre frontend y backend se utiliza Microsoft Power Automate. Una plataforma de automatización que permite crear y administrar la automatización de procesos de manera sencilla. Funciona con gatillantes y acciones, los gatillantes ejecutan acciones en el momento que se cumplan las condiciones definidas en el flujo. La herramienta se utiliza para ejecutar una petición HTTP al enlace definido por API Gateway cada vez que un usuario ingrese datos para un documento. Con esto se logra la ejecución de la función lambda que genera y envía el documento.

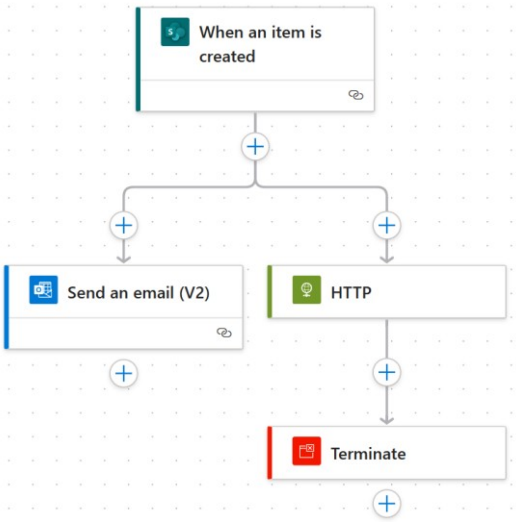

Ilustración 9: Flujo de Power Automate. El primer nodo define la condición para que se corra el flujo definido, la creación de un ítem (ingreso de información de un documento). Los nodos siguientes definen las acciones a realizar. Se envía un correo notificando el ingreso de información, y se hace una petición HTTP a la URL definida por AWS API Gateway, que ejecuta la función lambda correspondiente.

Finalmente se estableció un CI/CD (Integración y Despliegue continuos), su función es desplegar automáticamente a AWS cuando se hacen cambios a la rama de producción del repositorio.

```
deploy:prod:
  image: node:lts-alpine
  stage: deploy
  variables:
    AWS_REGION_NAME: $AWS_REGION_PROD
  script:
    - echo "Installing system level dependencies"
    - apk add --no-cache aws-cli python3 build-base py-pip python3-dev libffi-dev
    - npm install -g serverless
    - cd $CI_BUILDS_DIR/$CI_PROJECT_PATH/
    - cd $CI_BUILDS_DIR/$CI_PROJECT_PATH/functions
    - pip install -r requirements.txt -t ../layer/python
    - echo "Deploying app..."
    - serverless deploy --region $AWS_REGION_PROD --stage prod
  only:
    - main
```

Ilustración 10: Definición de instrucciones a realizar por el pipeline CI/CD, instala todas las dependencias necesarias.

9.4.3) Diagrama de la Solución

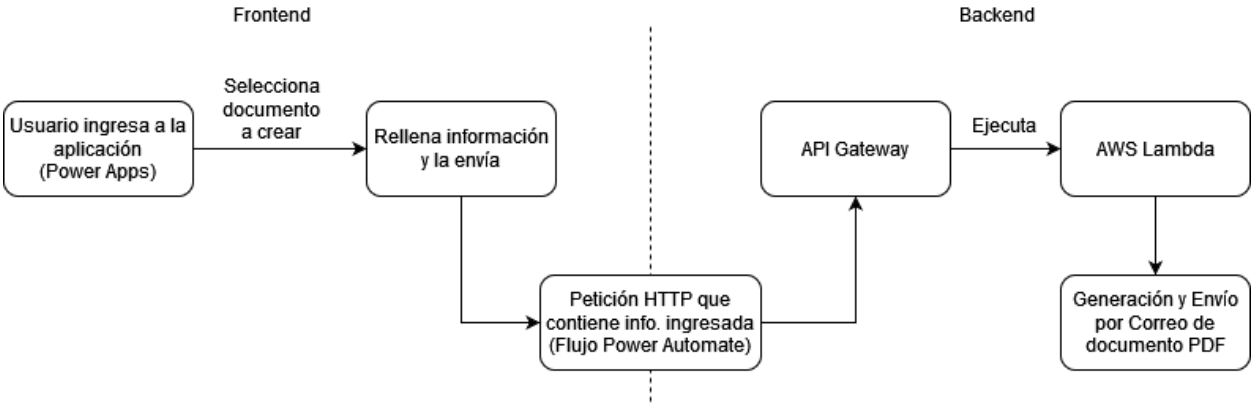

Ilustración 11: Flujo desde la entrada del usuario a la aplicación móvil hasta que se envía el documento generado.

10) Resultados

En el plazo destinado a la realización del proyecto de pasantía, quedaron implementadas las 3 soluciones de automatización propuestas.

El Sistema de Reportes Automatizado se implementó al final del segundo mes del proyecto y actualmente se encarga de la generación y envío de 6 reportes, tanto de uso interno como de envío a clientes. En este último tipo se observa el mayor ahorro en Horas Hombre. Uno de los reportes se enviaba a uno de los clientes más importantes de We Techs, éste se entrega 2 veces a la semana, lunes y jueves. Parte del flujo de ese reporte implica una revisión de los datos, repartidos en 27 columnas, y verificar que los datos estuvieran dentro de sus rangos normales. Esta tarea tomaba entre 1 y 2 horas cada vez que se creaba el reporte (2 veces a la semana). Como parte de la solución se automatizó esa labor y al momento de recibir el reporte el trabajador recibe información acerca de las variables que presentaron valores fuera del rango esperado. Con la solución implementada, está labor de medición tomaba entre 15 y 30 minutos como máximo, reduciendo el tiempo invertido en esta labor en un 75%. En total se estima que, en su forma actual, el Sistema de Reportes Automatizado libera 14 horas al mes.

La solución al problema de Uptime y Desconexiones, por decisión de la empresa, se implementó solamente para el cliente más importante, pero es fácilmente implementable al resto. La tarea de identificar manualmente los dispositivos desconectados originalmente demoraba más de una hora cada vez que debía realizarse esta investigación. Una vez implementada la solución, ya no es necesario hacer la investigación y el descubrimiento de desconexiones es instantáneo, apareciendo directamente en la plataforma utilizada por Soporte para el manejo de tickets. Gracias a esto la empresa pasa de tener un Soporte Reactivo, que tomaba acción cuando el problema era identificado por el cliente, a un Soporte

Proactivo, en el que los problemas pueden solucionarse antes de que cliente se dé cuenta de que sucedió algo. Se estima que esta tarea libera 12 horas mensuales al trabajador involucrado, las cuales se utilizan para llevar a cabo el soporte proactivo. Adicionalmente, antes del proyecto se planeaba contratar un nuevo Técnico en Soporte debido a los crecientes requerimientos el cliente en cuestión, su principal labor habría sido gestionar las desconexiones, pero gracias a la solución implementada, se pueden cumplir estos requerimientos con el personal actual de la empresa, generando el ahorro de lo que hubiese sido el sueldo de ese técnico, evaluado en $786.000 CLP mensuales.

Finalmente, la implementación de la aplicación para generación de documentos se estima que libera 22 horas mensuales. Sin embargo, esta solución entrega más beneficios que el ahorro de HH. La transición a un dispositivo electrónico en lugar de papel aporta a la imagen de empresa tecnológica y consciente del medio ambiente que We Techs busca transmitir. Además, a consecuencia de tener documentos digitales, la empresa ahora posee un lugar centralizado para encontrar la información de salidas a terreno, accesible por todos, facilitando el seguimiento de proyectos que involucran varias visitas a clientes.

Solución	**Descripción**	**HH liberadas (Mensuales)**
Sistema de Reportes Automatizado	Instancia de AWS EC2, que ejecuta reportes en el momento adecuado con la configuración de Crontab.	14
Alarmas de Desconexión a plataforma Freshdesk	Cada vez que se detecta una desconexión se genera un ticket en la plataforma de Soporte Freshdesk.	12
Aplicación para registro de Salidas a Terreno	Aplicación con un Tablet destinado para su uso. Usuarios ingresan información para correspondiente a documentos de Salidas a Terreno, los cuales luego se generan digitalmente como archivos PDF.	22

Tabla 4: Tabla resumen de las soluciones implementadas.

11) Conclusiones

En síntesis, el proyecto generó 3 soluciones a problemas dentro de la empresa que actualmente liberan un total aproximado de 48 horas hombre. Las soluciones implementadas fueron diseñadas para ser fácilmente extensibles. Gracias a esto será fácil agregar nuevos reportes al sistema, incluir más clientes a las alarmas de desconexiones y agregar nuevos documentos a la aplicación. Los trabajadores beneficiados por las soluciones implementadas hicieron comentarios positivos en términos de sus nuevos flujos de trabajo, contentos porque ya no deben realizar las partes más tediosas de su trabajo. La empresa quedó satisfecha con los resultados del proyecto y ya demostró interés en extender las soluciones entregadas.

Se utilizaron herramientas y servicios con los que no se tenía experiencia anteriormente, como AWS Lambda. Ésta demostró ser muy útil, entregando una manera de desplegar funciones pequeñas a bajo costo, la implementación usada en este proyecto tiene un costo mensual menor a 1 USD, sin tener que administrar un servidor nuevo. Por otro lado, la experiencia obtenida con las herramientas de Microsoft Power Platform, Power Apps y Power Automate, mostraron el valor que pueden aportar herramientas low-code, especialmente en un contexto de tiempo limitado para entregar una solución. Para ser herramientas gráficas, tienen mucha libertad en términos de las cosas que pueden integrarse, permitiendo conectar estas herramientas de Microsoft con los servicios Cloud de AWS.

El proyecto implementado se mantendrá a futuro en We Techs, gracias a la documentación generada y las introducciones realizadas con los trabajadores para que conozcan todos los aspectos de las nuevas herramientas que tienen a su disposición. Además, la empresa ya se ha propuesto desafíos para la extensión de las herramientas, particularmente la integración de la Aplicación de Salidas a Terreno con el módulo de Archivos existente, que permitirá a los clientes acceder automáticamente a los documentos relacionados a ellos.

12) Referencias

1. Aguirre, S. & Rodríguez, A. (2017). Automation of a Business Process Using Robotic Process Automation (RPA): A Case Study. Bogotá: Pontificia Universidad Javerian.
2. Consensus. (2020) Con el RPA se extrae la parte robot de los humanos y les da más valor a las personas. https://consensussap.co/con-el-rpa-se-extrae-la-parte-robot-de-los-humanos-y-les-da-mas-valor-a-las-personas/. 2020.
3. Bermúdez Irreño, C. (2021). RPA - AUTOMATIZACIÓN ROBÓTICA DE PROCESOS: UNA REVISIÓN DE LA LITERATURA, *RIMCI*, vol. 8, n.° 15, pp. 111-122.
4. Sayeeth Saabith, A. L., Fareez, M., Vinothraj, T. (2019) PYTHON CURRENT TREND APPLICATIONS-AN OVERVIEW. *International Journal of Advance Engineering and Research Development*, *Volume 6*(Issue 10), e23484470.
5. Srinath, K. R. (2017) Python – The Fastest Growing Programming Language. *International Research Journal of Engineering and Technology*, *Volume 4(Issue 12)*, *e23950056*.
6. Sbarski, P., & Kroonenburg, S. (2017). *Serverless Architectures on AWS: With examples using AWS Lambda*. Simon and Schuster.

13) Anexos

13.1) Planificación

13.1.1) Carta Gantt

	Agosto				Septiembre					Octubre				Noviembre			
Actividades	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Inicio de la Pasantía																	
Inducción a la Empresa																	
Definir Proyecto																	
Identificar Procesos Clave																	
Selección de Herramientas																	
Priorizar Tareas Elegidas																	
Diseño																	
Desarrollo																	
Prueba																	
Validación																	
Implementación																	
Evaluación del Proyecto																	

Tabla 5: Carta Gantt: Denota las principales actividades que se llevarán a cabo en el transcurso del proyecto, con sus fechas y plazos estimados.

13.1.2) Recursos

1. **Tecnología**: Todo lo que se necesita para el desarrollo de la solución, por ejemplo: diversos softwares, lenguajes de programación, servidores / infraestructura cloud, etc.
    a. **Git**: Control de versiones.
    b. **Python**: Uno de los 2 lenguajes de programación estándar en la empresa.
    c. **SQL**: Lenguaje para comunicarse con la base de datos.
    d. **Bash**: Línea de Comandos utilizada en Linux.
    e. **Docker**: Permite correr aplicaciones en contenedores, asegurando que el ambiente de ejecución sea el mismo sin importar el sistema operativo u otras condiciones externas al programa en sí mismo.
    f. **AWS:** Es el estándar en la empresa y toda en está toda la infraestructura cloud de la empresa.

2. **Datos**: Información sobre los procesos actuales, datos para probar los flujos automatizados.
3. **Tiempo**: Tiempo destinado para el desarrollo del proyecto, organizado mediante la Carta Gantt y metodologías ágiles.
4. **Documentación**: Documentación generada a partir del desarrollo de las automatizaciones, que debe quedar en la empresa una vez finalizado el proyecto.
5. **Recursos Humanos:** Supervisor que se encargará de proporcionar orientación y apoyo durante el transcurso del proyecto. Usuarios finales, las personas que se verán beneficiadas al ser automatizadas algunas de sus tareas, serán muy importantes al entregar feedback y opiniones en cuanto a las entregas que se harán, logrando así una solución lo más adecuada posible.

13.1.3) Plan de Implementación

Sistema Automatizado de Reportes.

1. Crear un repositorio Git para el proyecto.
2. Crear ambiente local de desarrollo utilizando Docker para imitar las condiciones de la máquina virtual en la nube y evitar errores al implementarla.
3. Hacer modificaciones a los reportes existentes para que se adapten a la nueva forma de ejecutarse.
4. Probar Localmente.
5. Validar con las personas que reciben los reportes.
6. Desarrollar lógica para ejecución programada de reportes. El crontab ejecutará scripts de bash, uno por reporte, que levantarán el ambiente y llamarán a los scripts de Python que generan, revisan y envían los reportes.
7. Monitorear primeras ejecuciones.

Uptime y Desconexiones de Dispositivos.

1. Generar template de Alarmas.
2. Agregar a la Base de Datos.
3. Verificar que la carga haya sido exitosa.
4. Desarrollar sistema de reinicio automático.
5. Probar con un dispositivo de prueba.
6. Implementar funcionalidad en el ambiente de producción.

Recolección de Información de Salidas a Terreno.

1. Desarrollo de parte frontend de la solución con Microsoft Power Apps.
2. Validación con las personas correspondientes.
3. Desarrollo de parte backend que toma los datos ingresados el usuario y rellena las plantillas para generar los documentos PDF.
4. Pruebas de integración de ambas partes de la solución en ambiente local.
5. Validar funcionamiento de la solución en su totalidad con las personas que la usarán.
6. Publicar la aplicación para que pueda ser utilizada.

13.1.4) Matriz de Riesgos

		Gravedad				
		1	2	3	4	5
Probabilidad	5			(f) Cambios en los requisitos durante el desarrollo		
	4		(b) Fallos en el desarrollo de soluciones			
	3	(a) Caídas en el servicio de proveedores clave		(c) Error de Comprensión del Proceso		
	2			(d) Incompatibilidad entre sistemas		(g) Problemas de Escalabilidad de las soluciones
	1				(e) Resistencia al cambio de parte de los trabajadores	

Tabla 6: Matriz de Riesgos: Riesgos ordenados por probabilidad y gravedad entre 1 – 5. 1 es el menor valor y 5 el mayor.

13.1.4.1) Mitigaciones

a) Identificar posibles proveedores alternativos y tener planes de contingencia.
b) Realizar pruebas frecuentes y revisiones de código regulares.
c) Capacitaciones y contacto constante con el Equipo de Operaciones.
d) Realizar un análisis detallado de integración con sistemas existentes, como por ejemplo compatibilidad entre versiones de software. Planificar migraciones de ser necesario.
e) Mantener al día a los trabajadores involucrados en los procesos a automatizar y comunicar los beneficios de la solución.
f) Mantener contacto constante con el cliente interno
g) Planificar y diseñar soluciones nativamente escalables en lugar de esperar a que surja un problema.

13.2) Detalles del Desarrollo

13.2.1) Repositorios

Los repositorios creados para el proyecto se ubican en la plataforma GitLab, el proveedor de control de versión utilizado por defecto en We Techs. Las prácticas de la empresa dicen que la rama principal (main, master, prod y equivalentes) de todos los repositorios debe estar protegida, y solo pueden hacerse cambios autorizados por alguien con permisos elevados. Esto significó que siempre antes de hacer cambios a las ramas que contienen código de producción se pasaba por un proceso de aprobación por un desarrollador senior/superior. Para el cumplimiento de esta regla se trabajó con dos ramas principales: 'main/master' y 'dev', con la utilización de ramas temporales para desarrollos de nuevas funcionalidades y arreglos. El flujo del desarrollo era el siguiente: primero la creación de una rama a partir de 'dev', con un nombre descriptivo de lo que se desarrollaría en la rama, acompañada de un prefijo para determinar la naturaleza de la rama rápidamente, 'ft/nombre-de-rama' para nuevas funcionalidades (features) o 'fix/nombre-de-rama' para arreglos. Una vez terminado el desarrollo pertinente a la rama se hace merge a la rama 'dev'. Los merges a la rama 'main' se hacían una vez a la semana, más seguido solamente si era necesario.

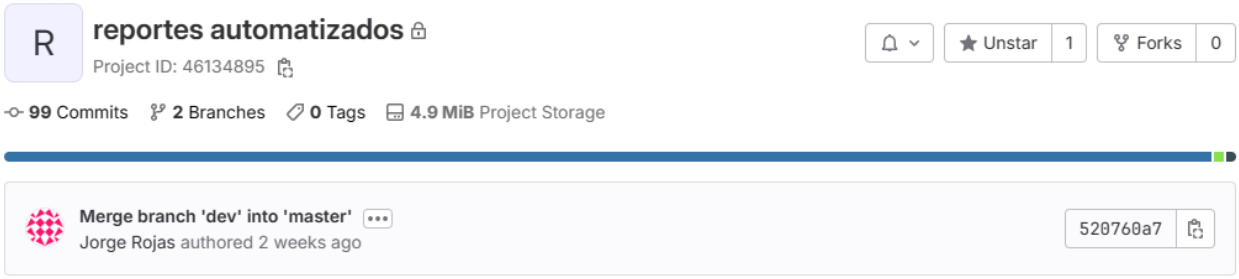

Ilustración 12: Repositorio que contiene el código asociado al Sistema Automatizado de Reportes

Ilustración 13: Repositorio que contiene el código asociado a la función AWS Lambda que genera documentos PDF respecto de salidas a terreno

13.2.2) Sistema Automatizado de Reportes

```
├── BD_Reporte_A.xlsx
├── BD_Uptime.xlsx
├── checks.json
├── deps.txt
├── Dockerfile
├── format.py
├── funcs.py
├── README.md
├── run-report-nombre_del_reporte.sh
├── reporte_nombre_del_reporte.py
└── validation.py
```

Ilustración 14: Estructura de archivos del repositorio. No se muestran todos los archivos ni sus nombres reales por confidencialidad

13.2.2.1) Funciones de Utilidad

Toda la lógica que se usaba frecuentemente entre reportes se extrajo con el objetivo de producir un código más legible y reutilizable. La ilustración 6 muestra archivos que forman parte del repositorio, de ellos 3, funcs.py, format.py y validation.py contienen funciones comunes usadas en otros lugares.

Las ilustraciones 8 y 9 muestran un ejemplo de lógica que se usa en todos los scripts del repositorio. La generación de todos los reportes implicaba extraer información de la base de datos y enviarle consultas SQL. Para no tener que reprogramar esa lógica para cada reporte se definieron las funciones que se muestran en las ilustraciones.

```python
def connect_to_we(source: str):
    """
    Establishes connection to We data source.

    :param str source: Indicates which source to connect, DB or TM for transmision.
    :return: Psycopg2 connection to specified We database.
    """
    return psycopg2.connect(user=os.environ[f'{source}_USER'],
                            host=os.environ[f'{source}_HOST'],
                            port=os.environ[f'{source}_PORT'],
                            password=os.environ[f'{source}_PASS'],
                            database=os.environ[f'{source}_NAME'])
```

Ilustración 15: Función que entrega una conexión a una fuente de datos de We Techs.

```python
def execute_query(connection, query: str) -> list:
    """
    Executes and SQL query on a given database connection.

    :param connection connection: Database connection to execute query with.
    :param str query: Query to execute.
    :return: Raw database response.
    """
    with connection.cursor() as cursor:
        cursor.execute(query)
        response_data = [row for row in cursor.fetchall()]
        cursor.close()

    return response_data
```

Ilustración 16: Función que recibe una conexión a una base de datos y una consulta SQL. Ejecuta la consulta y retorna la respuesta.

Además, en ambas ilustraciones se puede ver una práctica que es replicada en todo el código generado durante el desarrollo del proyecto. Las funciones están debidamente documentadas siguiendo una estructura común en todo el código generado. Esto permite a terceros que se vean involucrados en el futuro una comprensión más rápida y fácil del código.

13.2.3) Documentos de Salidas a Terreno

13.2.3.1) Interfaz de Usuario

A continuación, se muestran una serie de imágenes que muestran como ejemplo la interfaz desarrollada para la generación de una Orden de Trabajo.

Ilustración 17: Primera vista de la generación del documento, se ingresan datos generales de la salida a terreno.

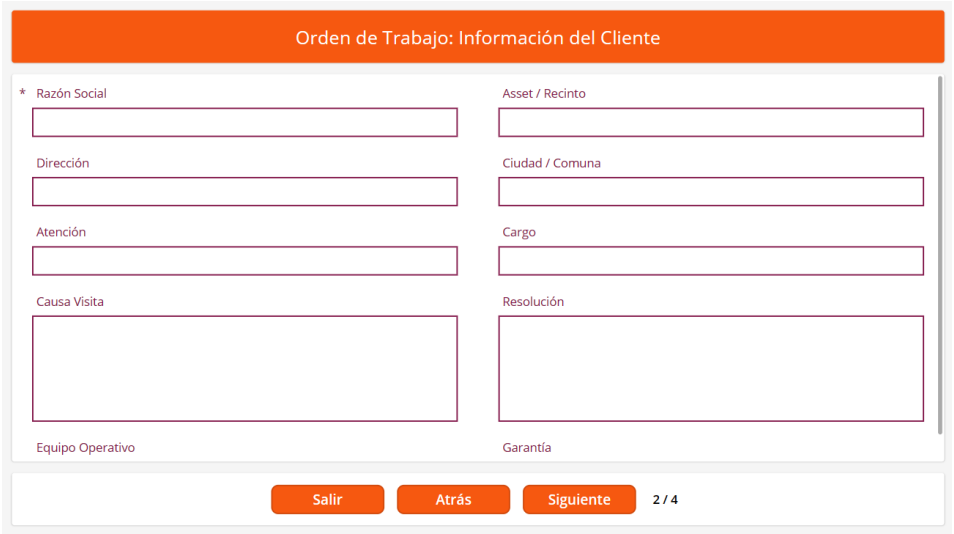

Ilustración 18: Segunda vista de la generación del documento, se ingresa la información del cliente.

Ilustración 19: Tercera vista, se ingresan los productos instalados y sus cantidades.

Ilustración 20: Cuarta vista, observaciones finales, se censura (cuadro gris), componente que permite seleccionar gráficamente correos que recibirán el documento.

13.2.3.2) Código

La ilustración 18 muestra la estructura de archivos del repositorio encargado de definir las funciones lambda y el servicio API Gateway. En la carpeta dependencias se incluyen dos archivos que contienen funciones de utilidad para uso en el repositorio, funcs.py que contiene utilidades generales y ses.py, que contiene la lógica para generar y enviar correos utilizando AWS SES. Cada función lambda está separada en una carpeta cuyo nombre empieza con *generador*, seguido del documento a generar. Dentro de cada una de esas carpetas hay un handler.py, en el cual se define el código de la función lambda.

```
├── docker-compose.yml
├── functions
│   ├── dependencies
│   │   ├── funcs.py
│   │   ├── __init__.py
│   │   └── ses.py
│   ├── Dockerfile
│   ├── generador*DocumentoAGenerar*
│   │   └──handler.py
│   ├── *Nombre_Documento*_template.pdf
│   ├── package.json
│   ├── requirements.txt
│   ├── serverless.local.yml
│   ├── serverless.yml
│   └── yarn.lock
└── README.md
```

Ilustración 21: Estructura de archivos de funciones lambda

Hay dos archivos serverless.yml, uno para uso local y otro para uso en producción. Estos definen la lógica necesaria para que el código pertenezca a un servicio de AWS Lambda.

```yaml
layers:
  docgen:
    path: ../layer/
    name: docgen-dependencies
    description: Dependencies for Internal PDF Generation
    compatibleRuntimes:
      - python3.11
    compatibleArchitectures:
      - x86_64

functions:
  generadorOT:
    timeout: 10
    handler: generadorOT.handler.handler
    events:
      - http:
          path: /pdfgen
          method: post
```

Ilustración 22: Parte del contenido del archivo serverless.yml, el cual se usa para entregarle a AWS la información que necesita para definir y desplegar los servicios requeridos.