AUTOMATIZACIÓN DE REGISTRO DE HORAS EN LA APLICACIÓN MULTI TIMESHEET

FRANCISCO JAVIER RAMIREZ BARRETO

UNIVERSIDAD SANTO TOMAS
FACULTAD DE INGENIERIAS
INGENIERÍA DE TELECOMUNICACIONES
BOGOTÁ
2024

AUTOMATIZACIÓN DE REGISTRO DE HORAS EN LA APLICACIÓN MULTI TIMESHEET

FRANCISCO JAVIER RAMIREZ BARRETO

MONOGRAFÍA DE LA PASANTIA EN KEYRUS

DIRECTOR:
VICTOR MANUEL CASTRO RAMIREZ
INGENIERO ELECTRONICO

UNIVERSIDAD SANTO TOMAS
FACULTAD DE INGENIERIAS
INGENIERÍA DE TELECOMUNICACIONES
BOGOTÁ
2024

Nota de Aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Bogotá, Colombia, 09/12/2024

CONTENIDO

1. LISTA DE IMAGENES

2. RESUMEN

Este documento presenta el desarrollo de un proyecto realizado durante las prácticas del segundo semestre de 2024 en Keyrus, una consultora multinacional especializada en datos. Inicialmente, se planeó un proyecto de Machine Learning vinculado a las herramientas internas de la empresa, pero debido a limitaciones de acceso y de datos, no fue posible llevarlo a cabo. Posteriormente, se propuso crear un algoritmo de Machine Learning para la predicción de valores numéricos, pero este también se vio restringido por la falta de datos adecuados para evaluar su desempeño. Finalmente, el proyecto surgió de la idea de automatizar un proceso tedioso dentro de la empresa, lo que llevó al desarrollo de un programa para optimizar el registro de horas trabajadas. Este documento detalla el planteamiento, objetivos, justificación, marco teórico y desarrollo de dicho proyecto.

PALABRAS CLAVE: Automatización, Python, PyQT, Keyrus.

3. INTRODUCCIÓN

Este documento presenta el desarrollo de un proyecto de monografía realizado durante las prácticas del segundo semestre de 2024 en la empresa Keyrus. Keyrus es una consultora multinacional especializada en el área de datos, con presencia en los cinco continentes y una amplia base de clientes. La empresa ofrece soluciones en diversas áreas, como data, marketing, experiencia del cliente y EPM (Enterprise Performance Management). Este último fue el área en la que realicé mis prácticas.

Inicialmente, tenía planeado desarrollar un proyecto de Machine Learning vinculado a una de las herramientas que utilizamos a diario. Sin embargo, tras un tiempo de trabajo, me di cuenta de que este proyecto no sería viable debido a que dependía de una sub-herramienta interna, la cual estaba vinculada a la herramienta principal. El problema radicaba en que no se tenía acceso a esta sub-herramienta.

En un segundo intento, decidí crear un algoritmo de Machine Learning de forma independiente. El objetivo de este algoritmo era ayudar a la predicción de valores numéricos en el tiempo. Sin embargo, este proyecto también se vio limitado, ya que no se podían utilizar datos de clientes y, en caso de usar datos internos, estos no resultaban adecuados para medir el desempeño del algoritmo debido a que no eran datos naturales ni se actualizaban de manera continua.

Finalmente, el proyecto surgió de la idea de automatizar algún proceso tedioso dentro de la empresa. En un principio, se consideró la creación de un programa para automatizar algún proceso en una de las herramientas que utilizábamos a diario. No obstante, tras una conversación con un compañero del equipo sobre mi propuesta, él sugirió que desarrollara un programa similar, pero enfocado en una de las herramientas utilizadas para registrar las horas trabajadas. Así fue como nació este proyecto.

Este documento describe el desarrollo completo del proyecto, comenzando con el planteamiento de la pregunta problema, la formulación de los objetivos generales y específicos, la justificación del proyecto, el marco teórico y el proceso de implementación.

4. PLANTEAMIENTO DEL PROBLEMA

4.1 DEFINICIÓN DEL PROBLEMA

En el contexto actual de la empresa, la gestión del tiempo de trabajo se ha vuelto crucial, especialmente debido al aumento del trabajo remoto. Para llevar un control preciso de las horas trabajadas, se utilizan diversas aplicaciones, siendo dos de las más relevantes para el área de EPM en Latinoamérica: Multi Timesheet y Nexonia. Ambas plataformas permiten registrar horas por proyecto, aunque presentan diferencias significativas en su funcionamiento.

Nexonia ofrece un proceso simplificado, donde los usuarios pueden ingresar las horas trabajadas por día sin necesidad de especificar los horarios. En contraste, Multi Timesheet exige que se registren tanto los horarios de entrada y salida como las horas efectivamente trabajadas. Aunque Multi Timesheet automatiza el cálculo de horas trabajadas, esta obligación de ingresar información adicional puede complicar el proceso y aumentar el riesgo de errores.

Además, es importante mencionar que Multi Timesheet presenta limitaciones cuando se utiliza en un idioma diferente al portugués, lo que dificulta ciertas acciones que podrían facilitar el registro de horas. Cabe destacar que, mientras Multi Timesheet es la herramienta obligatoria para toda la rama de EPM en Latinoamérica, Nexonia solo es necesaria para proyectos de implementación o soporte con empresas de Estados Unidos.

Otro aspecto preocupante es la frecuencia con la que los empleados olvidan registrar sus horas, lo que no solo afecta su propia carga horaria, sino que puede repercutir en otras áreas de la empresa, generando atrasos en la facturación y en el cumplimiento de compromisos con los clientes.

Con base en este contexto, la pregunta problema que se abordará en esta monografía es: ¿Qué posibilidades existen para automatizar el proceso de carga de horas en la aplicación Multi Timesheet, y cuáles serían las implicaciones de dicha automatización en el registro y facturación de horas trabajadas?

5. OBJETIVOS

5.1 OBJETIVO GENERAL

- Desarrollar un software que automatice el registro de horas en la aplicación Multi Timesheet, minimizando la posibilidad de errores en el proceso.

5.2 OBJETIVOS ESPECÍFICOS

- Analizar el proceso actual de registro de horas en la aplicación Multidados para identificar ineficiencias y áreas de mejora.
- Definir y documentar los requisitos funcionales y no funcionales del software para automatizar el registro de horas en la aplicación Multidados.
- Desarrollar y validar la lógica del software, asegurando su funcionalidad y precisión en el registro de horas.
- Diseñar y desarrollar una interfaz gráfica intuitiva para el usuario final, facilitando el uso del software y mejorando la experiencia del usuario.

6. JUSTIFICACIÓN

6.1 JUSTIFICACIÓN

El registro de horas en la aplicación Multi Timesheet presenta varios desafíos que afectan la experiencia de los empleados. Este problema debe ser abordado para facilitar el proceso de ingreso de horas. La automatización de este proceso permitirá que los empleados se concentren en sus tareas sin la carga del registro manual.

Además, es importante tener en cuenta que los empleados que trabajan en proyectos en Estados Unidos deben registrar sus horas en dos plataformas: Multi Timesheet y Nexonia. En comparación con Multi Timesheet, Nexonia ofrece un sistema de registro más sencillo. En Nexonia, los empleados solo necesitan ingresar la cantidad de horas trabajadas en un proyecto, sin necesidad de especificar los horarios exactos. Esta plataforma también permite copiar y pegar datos de un día a otro, lo que simplifica aún más el proceso.

Por otro lado, el registro en Multi Timesheet requiere que los empleados introduzcan los horarios específicos en los que trabajaron, y la aplicación calcula automáticamente las horas totales. Sin embargo, esta plataforma presenta algunas limitaciones, como la falta de la opción de copiar y pegar datos, a menos que se utilice el idioma portugués. Esta restricción puede resultar confusa, ya que, al registrar horas, los usuarios se encuentran con dos botones: "Guardar" y "Guardar y continuar". El botón "Guardar" almacena los datos del día seleccionado, siempre y cuando no haya errores en las horas registradas y todos los campos estén completos. En contraste, el botón "Guardar y continuar" debería permitir la opción de seguir registrando horas, pero esta funcionalidad solo se activa cuando la aplicación está configurada en portugués; en otros idiomas, su funcionamiento es idéntico al botón "Guardar".

Por estas razones, es fundamental desarrollar un software que automatice y simplifique el proceso de registro de horas en Multi Timesheet, permitiendo una experiencia más fluida y menos propensa a errores.

7. MARCO TEÓRICO

Sistemas automatizados:

Izaguirre Castellanos describe los sistemas de automatizados de la siguiente manera:

> Un sistema automatizado es el conjunto de elementos (equipamiento, sistema de información, y procedimientos) interrelacionados funcionalmente entre sí que conforman una estructura jerárquicamente expandida cuya función es garantizar el desempeño independiente del proceso a través de operaciones de control y supervisión total del sistema, bajo las técnicas más modernas y cumpliendo los requisitos establecidos de acuerdo al tipo de planta. (Izaguirre Castellanos, 2012, pág. 10)

De acuerdo con el texto anterior se infiere que un sistema automatizado es un conjunto de elementos que tienen una jerarquía y que están comunicados entre sí, con el objetivo de que el proceso se realice de manera independiente y que un ser humano solo tenga que intervenir cuando haya errores o para supervisar el desempeño del sistema automatizado.
Todo sistema automatizado tiene intervención humana en algún punto ya que las tecnologías de hoy en día no son suficientes para mantenerse por sí solas, y se sabe que los errores pueden suceder así la probabilidad de estos sea muy baja, es por esto que todos los procesos de automatización contienen operaciones de control y supervisión total del sistema, además de que algunos de sistemas cuentan con reportes de errores, es decir que algunos sistemas automatizados continúan operando así allá un error y lo que hacen es reportar los errores en algún archivo para que después un humano los revise y empiece un proceso para saber por qué se originó el error y como solucionarlo.

Multi Timesheet es un servicio que ofrece la empresa Multidados TI, por consiguiente, se describirá primero que es la empresa Multidados TI y después se describirá el servicio de Multi Timesheet, ambas descripciones son recolectadas directamente de la página de Multidados TI, después de documentar la empresa y el servicio se describirá como es el proceso que se realiza dentro del servicio de Multi Timesheet.

Multidados TI

Multidados TI describe a su empresa de la siguiente manera:

> Multidados IT se especializa en soluciones de ventas, atención al cliente y gestión empresarial, aumentando significativamente la eficiencia de estas áreas. Con 38 años de experiencia, tenemos un historial de trabajo con las principales empresas del mercado, comprometidos a llevar la innovación a nuestros clientes, ofreciendo siempre las mejores soluciones tecnológicas. (Traducción propia)

Referencia original en portugués: (Multidados TI, 2024)

El texto anterior fue traducido del idioma portugués, ya que la página del Multidados TI está completamente en portugués. Del texto citado se percibe que Multidados TI es una empresa con 38 años de experiencia en ofrecer soluciones de ventas, atención al cliente y de gestión empresarial utilizando las mejores soluciones tecnológicas, y estas soluciones que ofrecen tienen el fin de aumentar significativamente a la eficiencia de esas áreas nombradas.

El área EPM de Keyrus en Latinoamérica utiliza un servicio de Multidados TI, y este tipo de servicio que debe estar utilizando de Multidados TI es un servicio de gestión empresarial, ya que es un servicio para registrar las horas trabajadas, este objetivo no tiene nada que ver con ventas o atención al cliente, es por esto por lo que se determina que el servicio de Multi Timesheet es un servicio de gestión empresarial.

Multi Timesheet:

Multidados TI describe el sistema de gestión de horas de la siguiente manera:

> Multi Timesheet va más allá del control horario: puedes controlar las horas facturables y no facturables, la aprobación multinivel, los gastos, la tarificación de las horas y los informes inteligentes para tomar decisiones más asertivas. Todo online de forma amigable, sencilla y sin complicaciones, ya sea vía desktop o App. (Traducción propia)

Referencia original en portugués: (Multidados TI, 2024)

Esta cita al igual que la anterior fue traducida del idioma portugués, ya que la página de Multidados está completamente en portugués. Como se puede ver en el texto citado el servicio Multi Timesheet que ofrece Multidados TI es una

herramienta que permite registrar las horas trabajadas por parte de los empleados, pero este servicio también tiene la habilidad de controlar las horas facturables y no facturables, la habilidad de aprobar estas horas por varios niveles, la habilidad de reportar gastos, la habilidad de hacer una tarificación por las horas reportadas o planificadas y se tiene también la habilidad de generar informes.

Con esto se puede apreciar que Keyrus puede estar utilizando casi todas las habilidades del servicio de Multi Timesheet, ya que dentro de la aplicación en la página web se puede reportar las horas en diferentes proyectos, al reportar en proyectos internos de Keyrus todo parece normal, pero si se reporta en un proyecto de algún cliente este aparece en el calendario con un símbolo de dinero, por lo tanto se puede creer que esto pueden ser horas facturables, además cada semana o cada fin de mes un superior debe aprobar estas horas registradas y esto se refleja en el calendario con un color verde, en la parte de reporte de gastos Keyrus tiene diversas actividades durante el año y planifica un presupuesto para todo el año y que se pueden utilizar en estas actividades, por lo tanto los empleados solo deben registrar estos gastos y Keyrus después devolverá el dinero gastado por el empleado.
Por la parte de generar tarifas por horas planificadas o trabajadas no se sabe si se esté utilizando, además tampoco se sabe si se está utilizando la habilidad de generar informes.

Python:

Python es un lenguaje de programación usado en muchas aplicaciones y en diversos sectores. Python es considerado por muchas personas como un lenguaje de más fácil aprendizaje, es decir que estas personas consideran que la curva de aprendizaje es mucho más rápida que otros lenguajes de programación, ahora veremos cómo es descrito Python desde su propia página web, más específicamente desde su wiki.

Python menciona que, “Python es un gran lenguaje de programación orientado a objetos, interpretado e interactivo” (Traducción propia) Referencia original en inglés: (Python, 2024).Para definir que es un leguaje orientado a objetos se tiene que saber primero que es la programación orientada a objetos (POO).

Según el autor Efraín Manuel Oviedo Regino en su libro Lógica de programación orientada a objetos, un lenguaje orientado a objetos debe cumplir con los siguientes conceptos:

- Abstracción
- Encapsulamiento
- Ocultamiento de información
- Sobrecarga
- Polimorfismo
- Herencia
- Reutilización

En los siguientes párrafos se verán cada uno de estos conceptos y se dará una interpretación propia de dichos conceptos, pero antes de pasar a ver estos conceptos se revisará que es una clase y un objeto, ya que son términos usados dentro de estos conceptos.

Clase:

Oviedo Regino explica el concepto de clase de la siguiente manera:

> Una clase es una plantilla para crear objetos que constituye una abstracción del mundo real, como la clase Burro, la clase persona, la clase cuenta, etcétera. Las clases se nombran como sustantivos en singular y poseen variables que definen la información que se desea almacenar y métodos que definen las acciones que se desean realizar. Las variables se nombran como sustantivos y los métodos como verbos en infinitivo. (Oviedo Regino, 2015, pág. 206)

Según el texto anterior se puede ver que una clase es una especie de plantilla o molde que sirve para crear objetos que simulan ser del mundo real, dentro de estas clases hay variables que definen la información que se va a almacenar y métodos que definen las acciones que se pueden realizar. Por lo tanto, para las clases que se nombraban en este texto como burro o persona se puede tener una variable que sea nombre y esta variable almacenara el nombre del burro o de la persona, y como métodos se pueden tener acciones como caminar, la única diferencia en este caso es que el burro camina a 4 patas mientras que la persona camina a 2 pies.

Objeto: Según Oviedo Regino (2015) “Un objeto es una instanciación de una clase, es decir, la materialización de la clase. Cuando se instancia un objeto se asignan datos a las variables de la clase y se pueden ejecutar los métodos.” (Oviedo Regino, 2015, pág. 207)

Del texto anterior se evidencia que un objeto es llevar una clase a la materialización o mejor dicho a crear un objeto de acuerdo con esa plantilla o molde. Además, se evidencia que todo objeto creado a partir de una clase contiene las variables de la clase, pero con datos y estos objetos poseen también los métodos de las clases por lo que un objeto puede ejecutar las acciones que estén declaradas en los métodos.
Siguiendo con el ejemplo anterior si se instancia un objeto de la clase burro se le puede dar un nombre y además este tiene la acción de que puede caminar.

Ya con la definición de clase y objeto se continua con la definición e interpretación propia de los conceptos.

Abstracción: Oviedo Regino señala que “La abstracción es la característica más básica de la POO y puede definirse como la acción de identificar las cualidades y acciones que un objeto puede realizar, lo que permite diferenciar los conceptos de clase y objeto.” (Oviedo Regino, 2015, pág. 208)

Según el texto anterior la abstracción es la característica más básica de la programación orientada a objetos y la definen como las acciones y cualidades que tiene un objeto, además esto permite diferenciar una clase de un objeto, que ya fue explicado anteriormente. Por lo tanto, la abstracción es lo que nos permite identificar un objeto por sus acciones y cualidades, para poner un ejemplo del mundo real podríamos tomar cualquier objeto, en este caso tomaremos un objeto básico y este objeto será un vaso, el vaso puede tener diferentes cualidades como material, dimensiones como la altura, lo ancho y lo profundo, la geometría de este ya que un vaso puede ser cuadrado, cilíndrico, triangular, entre muchas otras geometrías. Estas son algunas de las cualidades que puede tener el objeto vaso, ahora las acciones que este vaso podría tener son llenar, vaciar, verter, entre muchas otras, llenar ya que el vaso es un recipiente el cual puede almacenar principalmente líquidos, pero todos saben que un vaso también puede almacenar objetos de dimensiones más pequeñas dentro de él. Vaciar y verter se podría pensar como una misma acción, pero esto no es tan cierto, ya que para vaciar un recipiente puedo desechar el contenido de este en algún otro lugar y es casi de manera directa, es decir que este proceso no toma mucho tiempo, mientras que verter un líquido puede verse como transferir ese liquido de un recipiente X a un recipiente Y o sencillamente desechar ese contenido, pero de una manera más lenta, es decir que este proceso toma un poco más de tiempo que el de vaciar.

Encapsulamiento:

Oviedo Regino también señala que:

> El encapsulamiento es la propiedad que tienen las clases de agrupar las características y las acciones relacionadas con una abstracción bajo una misma unidad de programación. Con la encapsulación, las clases son vistas como cajas negras donde se conocen las características y las acciones (variables y métodos) pero no sus detalles internos, es decir que se conoce lo que hace la clase, pero no la forma en que lo hace. (Oviedo Regino, 2015, pág. 211)

Del texto anterior se puede decir que la encapsulación es una propiedad que tienen todas las clases y esta propiedad permite que la clase agrupe todas sus variables y métodos en algún lugar. Además, la encapsulación cambia la perspectiva de las clases haciéndolas ver como sistemas de caja negra, en donde se conocen las variables y los métodos per no sus detalles más internos, por lo que se puede conocer como que hace una clase, pero no como lo hace.

Siguiendo con el ejemplo anterior se puede decir que conoceremos las variables del vaso, como su geometría, su altura y su el material del que está compuesto el vaso y además se conocen sus métodos como llenar, vaciar y verter, lo que no sabemos es cómo funcionan estas variables y métodos, lo único que sabemos es que cumplen con su objetivo.

Ocultamiento de la información:

Según Oviedo Regino:

> El ocultamiento de la información protege los objetos restringiendo y controlando el acceso en la clase que los define. Esto permite al programador definir exactamente cuáles variables y métodos serán visibles para otras clases, asegurándose de que el estado interno de un objeto no pueda ser cambiado de forma inesperada por una entidad externa. (Oviedo Regino, 2015, pág. 211)

Con el texto anterior se puede ver que el ocultamiento de la información es con fines de seguridad, ya que el programador elige que variables y métodos se van a mostrar y que variables y métodos no se van a mostrar a otras clases y esto es para que un tipo de clase no pueda cambiar otra ya sea accidental o intencionalmente.

Si se mira con un ejemplo, se puede regresar al ejemplo de la clase del burro y la persona, ambas clases tienen la misma variable que es el nombre y el mismo

método que es caminar, si se tienen dos objetos, uno de clase burro y otro de clase persona y se llama al objeto persona y se requiere que camine, lo que se hará es pasarle el método caminar a este objeto y lo que deberá pasar es que el objeto persona empezara a caminar, en vez de que el objeto burro empiece a caminar y esto pasa porque yo primero llamo el objeto y después llamo su método, aunque estos dos tengan el mismo método primero estoy llamando al objeto persona.

Sobrecarga:

Oviedo Regino dice que:

> La sobrecarga permite a una misma clase tener varios métodos con el mismo nombre, inclusive con el mismo valor de retorno, pero con diferentes parámetros. De este modo se puede redefinir un mismo método las veces que sea necesario dentro de una misma clase variando la cantidad de parámetros o el tipo de dato de estos. (Oviedo Regino, 2015, pág. 239)

De acuerdo con el texto citado anteriormente una clase puede tener métodos que se llaman de la misma manera y pueden tener el mismo valor de retorno. Estos métodos solo se diferencian en la cantidad de parámetros que reciben y el tipo de dato que reciben.
Si esto se lleva a un ejemplo se puede volver a la clase del burro o de la persona, en este caso se escogió la clase persona, recordemos que esta clase tenía una sola variable que es el nombre y un solo método que es caminar, siguiendo el concepto de sobrecarga podemos modificar el método caminar y como parámetro de entrada tendrá la cantidad de pasos que la persona dará, este parámetro deberá ser de tipo entero y positivo, este método devolverá la cantidad de pasos que dará la persona que será un valor entero positivo.
Mientras que habrá otro método que se llamara igual “caminar” y este tendrá como parámetros de entrada la cantidad de pasos que la persona dará, este parámetro será de tipo entero y positivo igual que el anterior método y este método tendrá otro parámetro que será la dirección y este será de tipo texto y el usuario podrá ingresar una dirección, que podría ser adelante, atrás, izquierda y derecha, este método retornara la cantidad un texto indicando la cantidad de pasos que dará la persona y la dirección en la que dará estos pasos.
Por lo tanto, si un usuario inicializa un objeto de clase persona y llama al método caminar y le da solo la cantidad de pasos, se llamará al primer método caminar y arrojara como resultado la cantidad de pasos que dio la persona, mientras que, si

llama al método caminar y le pasa la cantidad de pasos y una dirección, se llamará el segundo método y este arrojará un texto indicando la cantidad de pasos dados y la dirección en la que se dieron estos pasos.

Herencia y Reutilización:

Para Oviedo Regino:

> Al igual que los hijos heredamos características de nuestros padres, en la POO podemos obtener clases hijas con el mismo contenido de su clase padre. De esta manera se obtiene una reutilización de las clases, la cual es una de las propiedades más deseadas en la programación ya que conlleva un ahorro de tiempo y de líneas de código. (Oviedo Regino, 2015, pág. 243)

Con el texto citado anteriormente se infiere que hay clases que pueden ser hijas de otras clases y estas clases hijas heredan las variables y métodos de las clases padres. De aquí se percibe la reutilización de código, ya que, si se quiere programar algo parecido a algo que ya se ha programado antes, pero con unas ligeras diferencias o agregándole algunas cosas nuevas, se puede utilizar este concepto de herencia y crear nuevas plantillas o moldes a partir de las existentes.
Si se ve esto en un ejemplo se puede retornar a la clase persona que será utilizada como clase padre para crear una subclase o clase hija, esta clase hija se llamara “adulto” y esta clase tendrá como nueva variable la edad que en este caso solo indicara que es mayor de 18 años, y tendrá un nuevo método que será trabajar, además de la variable edad y el método trabajar la clase hija “adulto” hereda la variable nombre y el método caminar, por lo que la clase adulto puede utilizar el método caminar y se le puede asignar un nombre cuando se instancie un objeto de clase adulto.

Polimorfismo: Según Oviedo Regino “El concepto de polimorfismo se aplica a varios objetos que comparten un método con el mismo nombre. Cuando este método es invocado depende del objeto que lo contiene; por lo tanto, tendrá un comportamiento diferente para cada instancia.” (Oviedo Regino, 2015, pág. 257)

De acuerdo con el texto citado anteriormente se percibe que el polimorfismo es cuando dos o más clases tienen métodos con el mismo nombre y estos se diferencian porque para cada clase implementa de manera distinta el método. En este caso se puede ver este concepto si agregamos una nueva clase hija llamada bebe a la clase padre persona, al utilizar el método caminar tanto en la clase bebe

y en la clase adulto, se recibirían resultados diferentes ya que el adulto si podrá dar los pasos mientras que un bebe por sí solo no puede caminar en sus primeros años de vida.

Ya con los conceptos explicados se puede inferir que Python aplica cada uno de estos tanto internamente en su código propio como en códigos externos pero construidos con su lenguaje.

Ahora bien, se nombra que Python es un lenguaje interpretado y esto lo que significa es que Python utiliza un intérprete y no compila el código antes de ejecutarlo. Lo que hace el intérprete es leer línea por línea y ejecutar cada línea, mientras que un lenguaje que compila coge el script y lee todo su código, para después si ejecutarlo. El utilizar un intérprete hace que el desarrollo de aplicaciones sea mucho más rápido, ya que no se requiere compilar todo el script cuando solo se hizo un simple cambio, pero el hecho de utilizar un intérprete hace que el tiempo de ejecución de un script de sea más lento ya que tiene que ver línea por línea en vivo, mientras que un lenguaje que compila el script antes ya sabe lo que va a ocurrir y su tiempo de ejecución es mucho más rápido.
Python también es un lenguaje interactivo ya que gracias a que utiliza un intérprete y este lee línea por línea se puede experimentar que está realizando cada línea en el momento especifico, el ser interactivo también permite probar solo un fragmento de código de todo el script.

Selenium WebDriver:

Selenium (2024) describe el WebDriver de la siguiente manera:

> WebDriver maneja un navegador de forma nativa, como lo haría un usuario, ya sea localmente o en una máquina remota utilizando el servidor Selenium. Supone un salto adelante en términos de automatización de navegadores. Selenium WebDriver se refiere tanto a los enlaces de lenguaje como a las implementaciones del código individual de control del navegador. Esto se conoce comúnmente como sólo WebDriver. (Traducción propia)

Referencia original en inglés: (Selenium, 2024)

El texto citado anteriormente ha sido traducido del idioma inglés a español. De este texto se puede ver que la función principal de Selenium WebDriver es poder automatizar procesos dentro de cualquier navegador como si se tratase de un

usuario común y corriente. Este tipo de herramientas permite que un programa tome por un momento el control del computador y ejecute las tareas requeridas dentro de un navegador, algo importante a recalcar es que este tipo de programas hace todo en tiempo real por lo que si se requiere iniciar sesión en alguna página web y sacar información de esta página web o hacer algún proceso con la sesión iniciada se puede hacer sin ningún problema.

Tkinter y PyQT:

Python describe tkinter como:

> El paquete tkinter («interfaz Tk») es la interfaz por defecto de Python para el kit de herramientas de GUI Tk. Tanto Tk como tkinter están disponibles en la mayoría de las plataformas Unix, así como en sistemas Windows (Tk en sí no es parte de Python, es mantenido por ActiveState). (Python, 2024)

Del texto anteriormente citado se puede inferir que Tkinter es una interfaz de herramientas GUI por defecto implementada en Python, lo que quiere decir que esta herramienta permite el diseño de una interfaz gráfica para el usuario o mejor conocido como frontend.

pythonpyqt.com explica que:

> PyQt es el puente que integra a la perfección el robusto framework multiplataforma Qt C++ con el flexible lenguaje de programación Python, sirviendo principalmente como un potente módulo GUI. Qt va más allá de ser un simple conjunto de herramientas GUI. Abarca una amplia gama de funcionalidades y ofrece abstracciones para elementos como sockets de red, hilos, Unicode, SQL, bases de datos, SVG, OpenGL y XML. Además, incorpora un navegador web y un sofisticado sistema de servicios, respaldados por una amplia colección de widgets de interfaz gráfica. (Traducción propia)

Referencia original en inglés: (pythonpyqt.com, 2024)

El texto anterior fue traducido del inglés al español. En el texto anterior se lee que QT es un framework del lenguaje de programación C++ y PyQT es la integración de este framework con el lenguaje de programación Python. Además, se lee que PyQT sirve como un potente módulo de GUI, pero a la vez se dice que QT no solo es una herramienta de interfaces gráficas de usuarios, sino que también tiene

otras funcionalidades para elementos como sockets de red, hilos, Unicode, SQL, bases de datos, SVG, OpenGL y XML, además que también cuenta con un navegador.

8. DESARROLLO DEL PROYECTO

8.1 ANÁLISIS DEL PROCESO ACTUAL DE REGISTRO DE HORAS EN MULTI TIMESHEET

Las imágenes de la aplicación Multi Timesheet tendrán algunos elementos difuminados ya que no se tiene permitido el mostrar nombres tanto de la aplicación como de la empresa y así mismos nombres de usuarios, proyectos y clientes.

El empleado que debe registrar sus horas trabajadas en la aplicación Multi Timesheet debe ingresar a la aplicación con sus credenciales, las credenciales se componen de un usuario y una contraseña, la siguiente imagen muestra como la página donde el usuario debe ingresar sus credenciales.

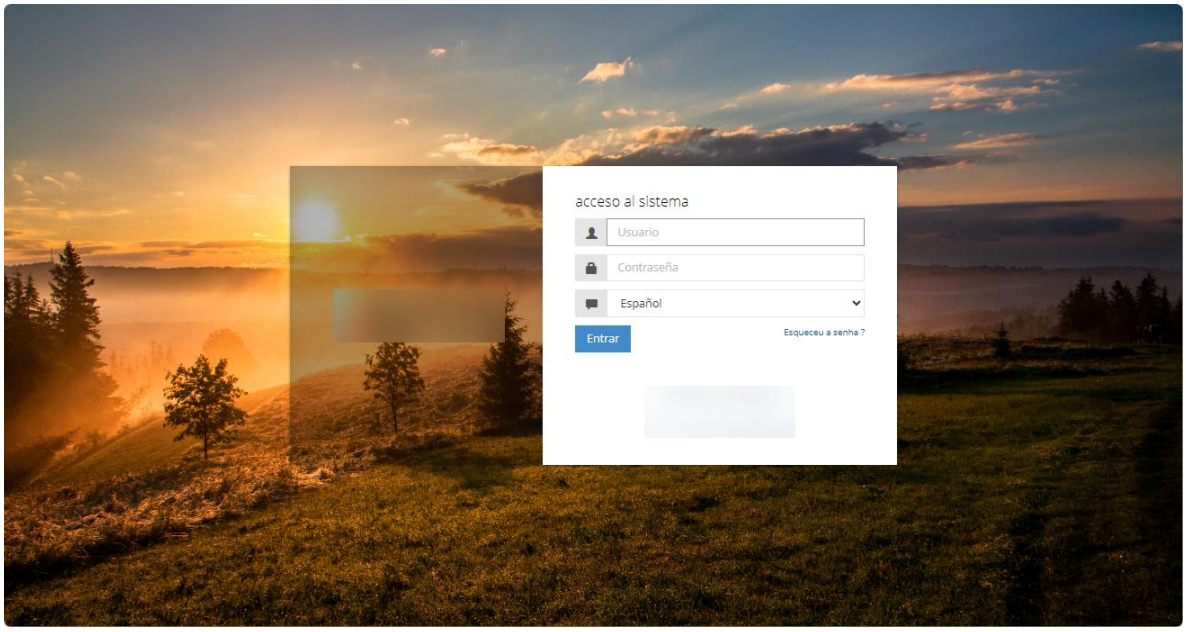


Imagen 1 *Pagina de ingreso a Multi Timesheet*

En la imagen se aprecia dónde deben ingresarse las credenciales y además hay marcadores de posición (placeholder) para indicar que parte de la credencial debe ir en que parte. En la imagen también se aprecia un campo que es de tipo lista, ya

que este se puede desplegar y esto se percibe por su flecha en la parte de la derecha, este campo lo que permite es cambiar el idioma de la aplicación, las opciones de este campo son “Español”, “Inglés” y “Portugués”.
Después de que el empleado registra sus credenciales y da clic en entrar se encuentra con una nueva interfaz, esta interfaz ya es la aplicación de Multi Timesheet, en esta ya puede registrar las horas que trabajo, para la empresa no es necesario que el empleado reporte las horas trabajadas día a día, para la empresa es fundamental que a fin de semana y a fin de mes estén registradas todas las horas trabajadas por el empleado, sin embargo si un empleado no registra las horas que trabajo en un día la misma aplicación de Multi Timesheet enviara un correo a dicho empleado informando que no ha registrado las horas del día anterior, este correo lo envía la aplicación al día siguiente, es decir si el empleado no registro horas el día lunes, el martes a primera hora la aplicación enviara el correo.

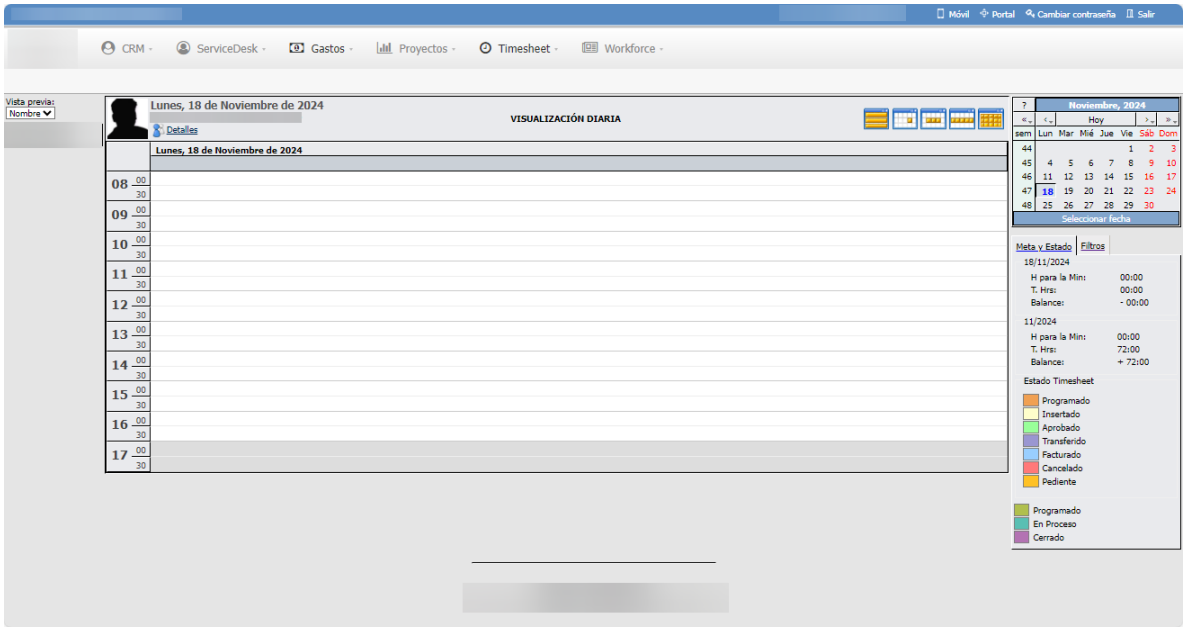


Imagen 2 *Página principal de Multi Timesheet*

La anterior imagen muestra la página inicial de la aplicación Multi Timesheet, en esta se ve el día actual, unos botones para cambiar la visualización, ya sea que se desee ver solo un día, la semana laboral (lunes a viernes), toda le semana de lunes a domingo o el mes completo, por defecto cuando se entra en la aplicación se muestra solo el día actual. Además de esos botones de visualización se observa que hay un calendario a la derecha y desde este se puede seleccionar el

día que uno desea ver, y en la parte superior también hay unos botones de los cuales no se tienen acceso a todos por limitaciones de la página.
Para el registro de horas se puede hacer desde cualquiera de estas visualizaciones, normalmente se realizan desde la visualización de mes, por lo tanto, se observará como es el proceso desde esta visualización.

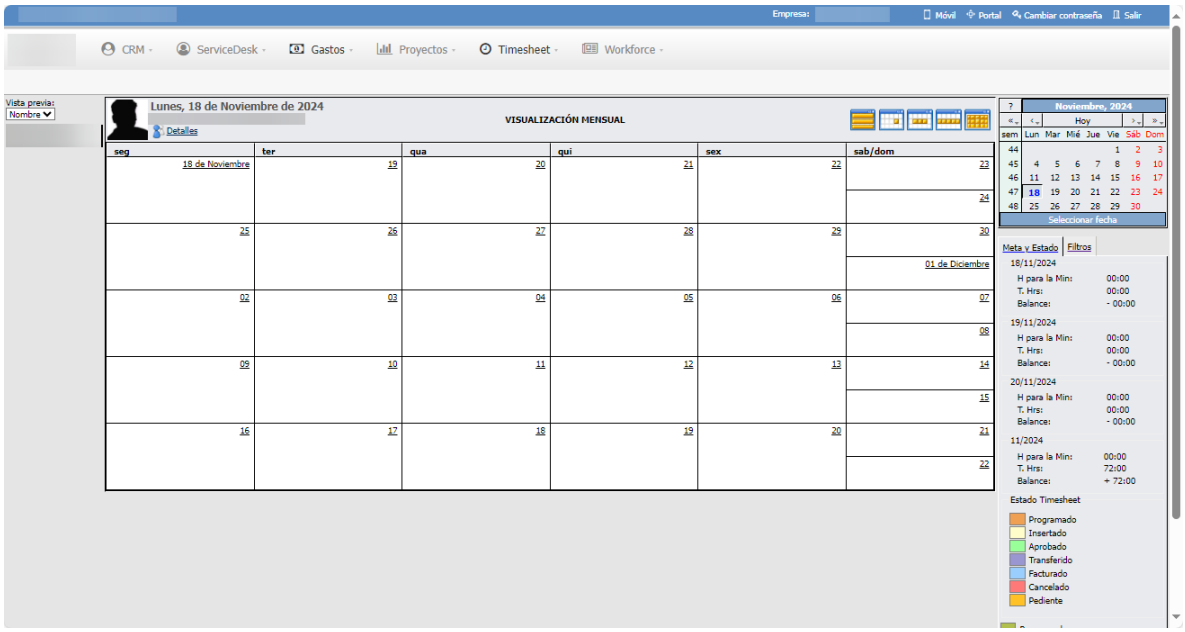


Imagen 3 *Visualización mensual Multi Timesheet*

Como se aprecia en la imagen anterior se puede ver todo un mes desde la fecha que se tiene seleccionada en el calendario, lo que se puede apreciar es que no solo se va a ver el mes en el que esta la fecha actualmente, sino que también se pueden ver fechas del mes siguiente, ya que esta visualización lo que hace es mostrar 5 semanas desde la fecha seleccionada, es por eso por lo que en el caso de la imagen se logra apreciar desde el 18 de noviembre hasta el 22 de diciembre.
Para registrar las horas en esta visualización se debe dar clic en el día que se desean registrar las horas, con esto se desplegara una ventana en la cual se registraran los datos pertinentes, la siguiente imagen muestra esta nueva ventana.

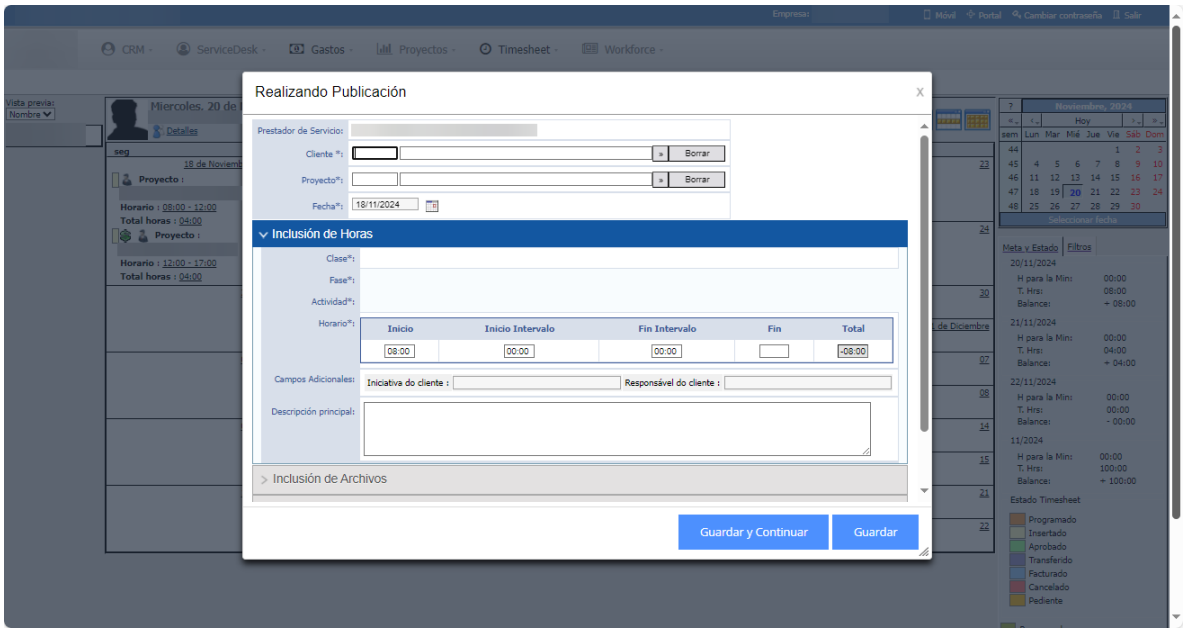


Imagen 4 *Ventana de registro de horas en Multi Timesheet*

En la anterior imagen se aprecia que existe un campo para el cliente, un campo para el proyecto, un campo para la fecha que en este caso tiene la fecha del día en el que se dio clic anteriormente, al ingresar estos campos anteriores se registran por defecto unos valores en los campos “Clase”, “Fase” y “Actividad”.

Después de haber registrado el cliente, el proyecto y la fecha, el usuario deberá ingresar el horario en el que trabajo para ese cliente y para ese proyecto, esto en el campo correspondiente que es en el de “Horario”, en este campo hay 4 subcampos que son “Inicio”, “Inicio Intervalo”, “Fin Intervalo”, “Fin” y “Total”. Dentro de los primeros 4 campos se deberán registrar las horas en las que se trabajó para ese cliente y para ese proyecto, en el campo de “Inicio” se indica la hora en la que se inició a trabajar, en el campo de “Inicio Intervalo” se indica la hora en la que se inició el descanso, en el campo de “Fin Intervalo” se indica la hora en la que finalizo el descanso, y en el campo de “Fin” se indica la hora en la que se finalizó el trabajo con ese cliente en ese proyecto. El campo “Total” se completa por sí solo, este solo sirve para indicar el total de horas trabajadas, es decir que este campo toma el total de horas desde “Inicio” hasta “Fin” y se le resta el total de horas que halla entre el “Inicio Intervalo” y “Fin Intervalo”.

Con esto ya solo queda completar un campo que es el de descripción principal, ya que, los campos de la parte de campos adicionales no deben ser ingresados. En el campo de descripción principal se deben poner las reuniones que se tuvieron con el cliente o que corresponden al proyecto en el que se trabajó.

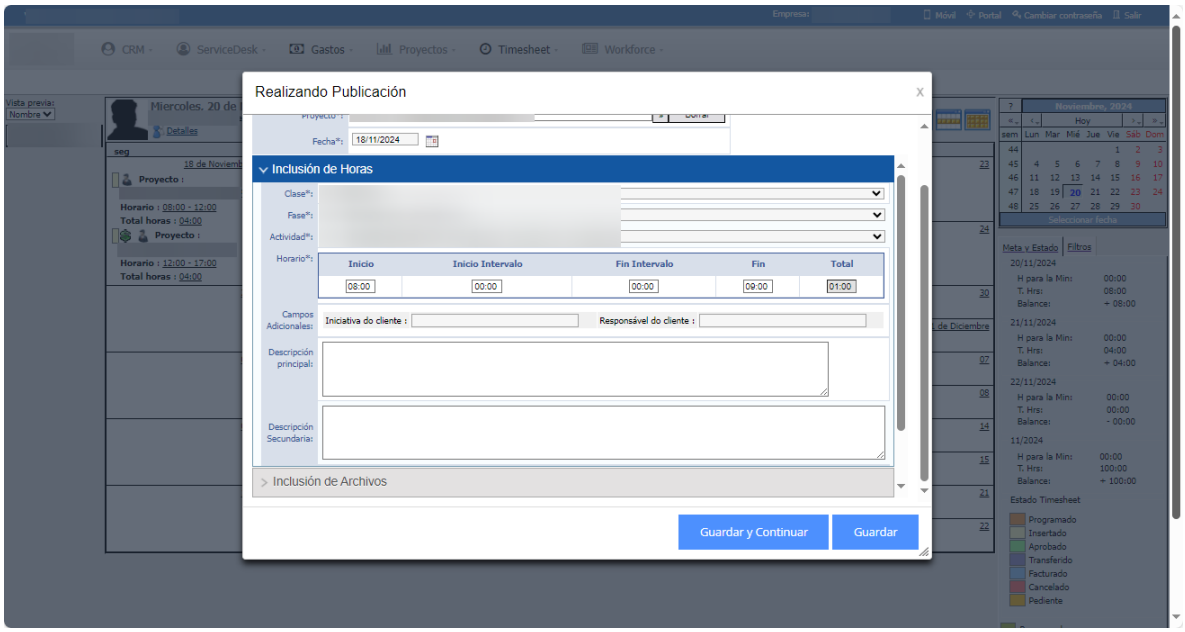


Imagen 5 *Campo de Descripción Secundaria en Multi Timesheet*

Algo que hay que recalcar es que como cliente se puede tener a la misma empresa Keyrus, y este es muy usado para reportar únicamente las horas en las que uno está trabajando, pero no está realizando algo para un cliente, esto pasa ya que para cada cliente hay un contrato en el cual se define cuantas horas de soporte habrá a la semana. Cuando se está registrando las horas trabajadas para un proyecto de un cliente se habilita este campo “Descripción Secundaria”, mientras que cuando se está trabajando en algo interno para Keyrus no se habilita este campo. El campo de “Descripción Secundaria” se utiliza para describir en pocas palabras que se hizo durante esas horas trabajadas en el proyecto del cliente.

Otro tema importante es que el campo de “Descripción” no puede estar vacío, este siempre debe contener algo y si se intentan registrar horas con este campo vacío la aplicación arrojara una ventana de error explicando el error y no permitirá registrar esas horas hasta que se llene el campo de “Descripción”. Otro tema es que si no se tiene un intervalo de descanso se debe dejar el campo “Inicio Intervalo” y “Fin Intervalo” en 00:00, además estos campos solo se utilizan para reportar la hora de descanso o de almuerzo que uno tiene en el trabajo.

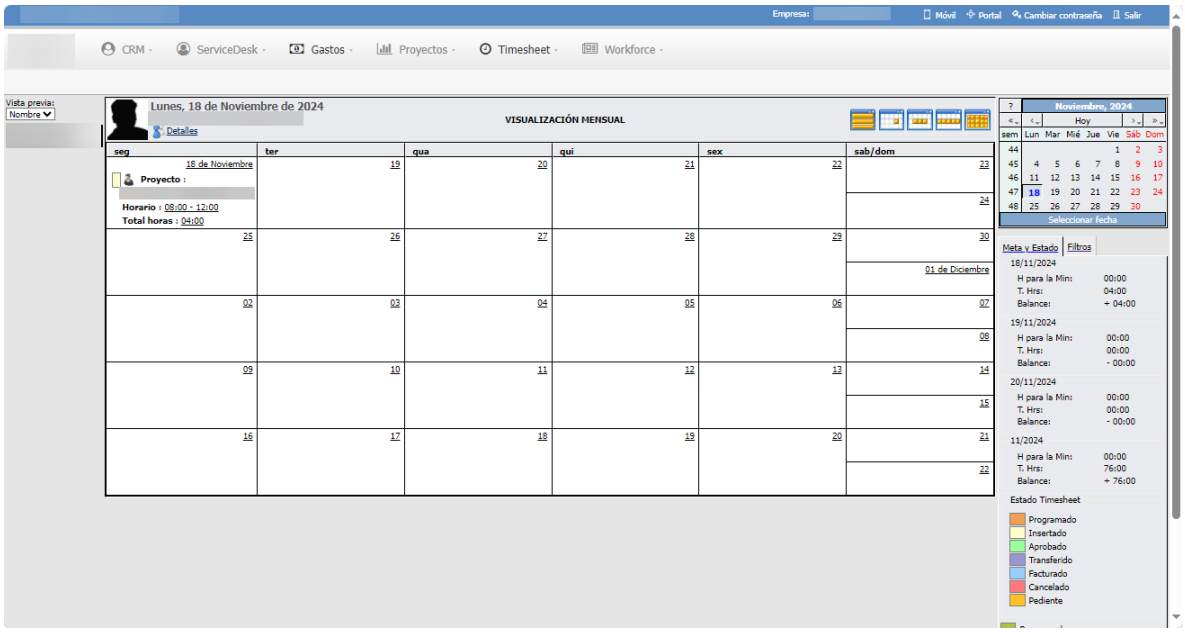


Imagen 6 *Resumen registro de horas en Multi Timesheet*

En la imagen anterior se puede notar que hay un nuevo registro en el día 18 de noviembre, este fue el registro que se hizo, este registro muestra las horas trabajadas, el intervalo en el que se trabajó y el proyecto en el que se trabajó, además de esto hay un cuadro con un color, lo que significa cada cuadro de color está en la esquina inferior derecha, normalmente solo se utilizan los cuadros de "Insertado" y "Aprobado".

Una de las reglas que tiene la aplicación es que no se pueden reportar más de las 8 horas de trabajo, si se necesitan reportar más en un día se debe escalar la situación para que otorguen ese acceso o permiso en el día especifico. Si se reportan más de las 8 horas y no se tiene este permiso, la aplicación desplegara una ventana de error, donde explica cuál es el error y no permitirá registrar las horas hasta resolver el error.

8.2 REQUISITOS FUNCIONALES Y NO FUNCIONALES

8.2.1 REQUISITOS FUNCIONALES

1. **RF001 - Autenticación de usuario:**
    a. El sistema debe permitir que el usuario ingrese su usuario y contraseña para autenticarse en la aplicación Multidados.

b. El sistema debe validar que los campos de usuario y contraseña no estén vacíos. Si los campos están vacíos, el sistema debe mostrar un mensaje de advertencia y no puede subir los registros a la aplicación Multi Timesheet.

2. **RF002 - Ingreso de horas trabajadas:**
    a. El sistema debe permitir que el usuario ingrese la fecha en la que desea registrar las horas trabajadas.
    b. El sistema debe permitir seleccionar el código del proyecto al que se desea asignar el registro de horas.
    c. El sistema debe permitir al usuario ingresar las siguientes horas:
        i. Hora de entrada
        ii. Hora de inicio del descanso
        iii. Hora de finalización del descanso
        iv. Hora de salida
    d. El sistema debe permitir ingresar una descripción de las reuniones o actividades realizadas durante el día. La descripción no puede estar vacía; si el campo de descripción está vacío, el sistema debe mostrar un mensaje de advertencia y no permitir guardar el registro hasta que se ingrese una descripción válida.

3. **RF003 - Guardar y borrar registros:**
    a. El sistema debe permitir al usuario guardar los registros de horas ingresados sin enviarlos a Multidados para su posterior revisión.
    b. El sistema debe permitir al usuario borrar los registros seleccionados si lo desea.

4. **RF004 - Guardar múltiples registros:**
    a. El sistema debe ser capaz de manejar y guardar múltiples registros de horas en una misma sesión. Cada registro debe incluir los campos de fecha, código de proyecto, hora de entrada, hora de inicio y fin del descanso, hora de salida y descripción.

5. **RF005 - Confirmación y subida de registros:**
    a. El sistema debe permitir al usuario revisar todos los registros guardados antes de decidir subirlos a la plataforma Multidados.
    b. Si el usuario está seguro de que los registros son correctos, el sistema debe permitir subir todos los registros guardados a la aplicación Multidados utilizando la librería Selenium.

c. El sistema debe gestionar la subida de registros de forma ordenada y automática, sin necesidad de intervención manual por parte del usuario, para cada registro de forma individual.

6. **RF006 - Visualización de registros:**
    a. El sistema debe mostrar todos los registros guardados en una vista de tabla, donde se muestren los campos: fecha, código del proyecto, hora de entrada, hora de salida, hora de inicio del descanso, hora de fin del descanso y descripción. Los campos usuario y contraseña no deben ser visibles en la tabla.
    b. El sistema debe permitir seleccionar cualquier registro de la tabla para después decidir si eliminarlo.

7. **RF007 - Filtrado de registros por fecha:**
    a. El sistema debe permitir filtrar los registros de la tabla a través de un calendario.
    b. Al seleccionar una fecha en el calendario, la tabla debe mostrar solo los registros correspondientes a esa fecha.

8.2.2 REQUISITOS NO FUNCIONALES

1. **RNF001 - Rendimiento:**
    a. El sistema debe ser capaz de manejar registros de horas de manera eficiente, incluso si un usuario ingresa múltiples registros a la vez. El sistema debe garantizar que la visualización, almacenamiento y subida de los registros se realice sin errores ni bloqueos, independientemente de la cantidad de registros.

2. **RNF002 - Escalabilidad:**
    a. El sistema debe ser escalable, lo que significa que debe poder adaptarse a un aumento en el número de usuarios sin que el rendimiento se vea afectado. Esto incluye la capacidad de agregar nuevos registros de horas o nuevos usuarios sin problemas a medida que la aplicación crece.

3. **RNF003 - Usabilidad:**

a. La interfaz de usuario debe ser intuitiva y fácil de usar. Los usuarios deben poder ingresar, guardar, revisar y subir sus registros de horas de manera clara y sencilla, sin necesidad de formación previa.
b. El sistema debe tener una interfaz de usuario consistente que permita a los usuarios navegar fácilmente entre las distintas funcionalidades (como ingreso de horas, visualización de registros, y subida de los mismos).

4. **RNF004 - Integración con Selenium:**
    a. El sistema debe tener una integración estable y confiable con la librería Selenium para realizar la carga de los registros en la aplicación web Multidados. La integración debe ser capaz de manejar múltiples registros de forma secuencial y garantizar que los datos se suban correctamente.

8.3 DESARROLLO Y VALIDACIÓN DE LA LÓGICA DEL SOFTWARE

Para el desarrollo de la lógica del software se dividió en tres archivos (scripts), dos de estos son para la parte de la automatización de registro de horas dentro de la aplicación Multi Timesheet y el tercero es para la integración de todos los elementos del diseño en QT Designer y el funcionamiento de la aplicación. En este segmento del documento se documentarán o explicaran los dos primeros scripts, los scripts que contienen la lógica para realizar la automatización del registro de horas en Multi Timesheet. Algo que recalcar antes de pasar a la documentación de los scripts es que se utilizó la versión 3.9.13 de Python, que es la última que se puede descargar para la versión 3.9 de Python y no se utilizó una versión más actual de Python porque la librería PyQT6 que es la herramienta con la que se va a hacer el diseño de la interfaz gráfica no tiene compatibilidad con versiones mayores a la 3.9.

8.3.1 PRIMER SCRIPT

El primer script tiene el nombre de “full_process” y en este archivo se encuentra parte de todo el proceso para registrar las horas de manera satisfactoria, en este archivo solo se encuentra una función la cual recibe tres argumentos y contiene todo el proceso de automatización, la siguiente imagen muestra las librerías o dependencias que se utilizaron dentro de este script.

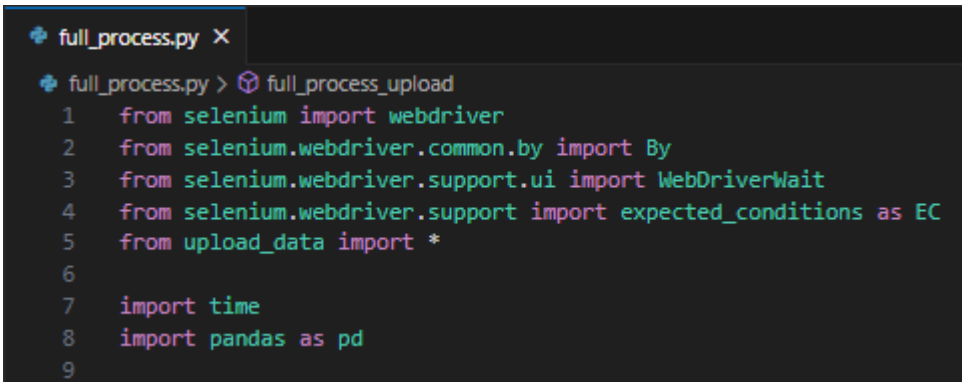
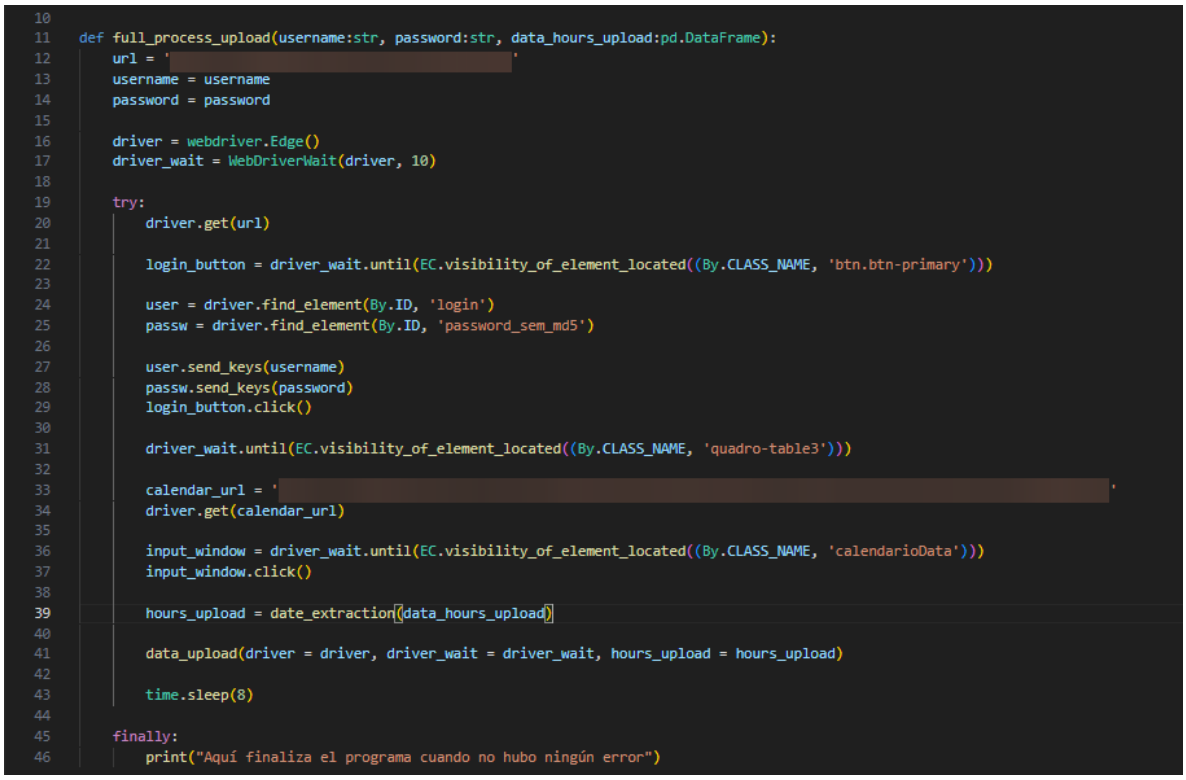


Imagen 7 *Librerías script full_process*

De la imagen anterior se aprecian las librerías de Selenium, time y pandas. El importe desde “upload_data” es el importe de todas las funciones que tiene el segundo script. Desde la librería de Selenium se importan las clases “webdriver”, “By”, “WebDriverWait” y “expected_conditions”, este último importe se deja le asigna la palabra “EC” para no tener que digitar siempre “expected_conditions”.
Más abajo en la línea 7 vemos el importe de la librería llamada “time”, esta es una librería nativa de Python, esto quiere decir que apenas uno descarga Python uno tiene ciertas librerías que ya vienen integradas, una de estas es la librería “time”.
Por último, se tiene la librería de pandas y se le asigna la palabra “pd” para llamar cualquier método o clase de la librería de pandas. Esta librería de pandas es utilizada en estos dos primeros scripts para poder leer el Dataframe creado y consultar los registros hechos por el usuario desde la interfaz gráfica.



```python

def full_process_upload(username:str, password:str, data_hours_upload:pd.DataFrame):
    url = '                                                    '
    username = username
    password = password

    driver = webdriver.Edge()
    driver_wait = WebDriverWait(driver, 10)

    try:
        driver.get(url)

        login_button = driver_wait.until(EC.visibility_of_element_located((By.CLASS_NAME, 'btn.btn-primary')))

        user = driver.find_element(By.ID, 'login')
        passw = driver.find_element(By.ID, 'password_sem_md5')

        user.send_keys(username)
        passw.send_keys(password)
        login_button.click()

        driver_wait.until(EC.visibility_of_element_located((By.CLASS_NAME, 'quadro-table3')))

        calendar_url = '                                                                                                    '
        driver.get(calendar_url)

        input_window = driver_wait.until(EC.visibility_of_element_located((By.CLASS_NAME, 'calendarioData')))
        input_window.click()

        hours_upload = date_extraction(data_hours_upload)

        data_upload(driver = driver, driver_wait = driver_wait, hours_upload = hours_upload)

        time.sleep(8)

    finally:
        print("Aquí finaliza el programa cuando no hubo ningún error")
```

Imagen 8 *Función full_process_upload*

En la anterior imagen se logra apreciar toda la función llamada “full_process_upload”, como se nombró anteriormente, esta función recibe tres parámetros de entrada, los cuales son el usuario, la contraseña y el Dataframe con los registros del usuario. El usuario y la contraseña deben ser variables de tipo texto.

Al inicio del script se tiene una variable que contiene el enlace donde el usuario debería registrar sus credenciales, pero en este caso el usuario no hará nada de esto, el programa es el que se encargara de hacer todo sin intervención del ser humano. Después se encuentran dos variables, estas dos variables reciben los parámetros de usuario y contraseña. En la línea 16 y 17 se aprecian dos variables que son importantes, la primera es la que permite trabajar con el navegador que se haya descargado, algo importante que aclarar aquí es que para poder utilizar Selenium se debe descargar el WebDriver del navegador que se va a utilizar, en este caso se descargó el de Microsoft Edge, la segunda variable es como un temporizador y este se utilizará a lo largo de los scripts para saber si una página o algún elemento de la página ya cargo completamente, esta clase que se utiliza para crear esta variable recibe en este caso como argumentos, el driver, es decir

una variable que sea de tipo WebDriver y un número de punto flotante, es decir un número decimal, este número se utiliza como la cantidad de tiempo máxima que se puede esperar a que un elemento cargue completamente, este tiempo es dado en segundos, por lo que para este caso se le han dado 10 segundos.

```
18
19  try:
20      driver.get(url)
21
22      login_button = driver_wait.until(EC.visibility_of_element_located((By.CLASS_NAME, 'btn.btn-primary')))
23
24      user = driver.find_element(By.ID, 'login')
25      passw = driver.find_element(By.ID, 'password_sem_md5')
26
27      user.send_keys(username)
28      passw.send_keys(password)
29      login_button.click()
30
31      driver_wait.until(EC.visibility_of_element_located((By.CLASS_NAME, 'quadro-table3')))
32
33      calendar_url = '                                                                                                    '
34      driver.get(calendar_url)
35
36      input_window = driver_wait.until(EC.visibility_of_element_located((By.CLASS_NAME, 'calendarioData')))
37      input_window.click()
38
39      hours_upload = date_extraction(data_hours_upload)
40
41      data_upload(driver = driver, driver_wait = driver_wait, hours_upload = hours_upload)
42
43      time.sleep(8)
44
45  finally:
46      print("Aquí finaliza el programa cuando no hubo ningún error")
```

Imagen 9 *Parte "try" de la función full_process_upload*

De la anterior imagen se evidencia que se utiliza la estructura “try” pero no se utiliza la parte del “catch” en vez se utiliza la parte de “finally”. Dentro del “try”, en la línea 20 se utiliza el objeto del WebDriver creado para ir a la URL donde el usuario inicia sesión en la aplicación Multi Timesheet. En la siguiente línea se espera a que el botón de Inicio de sesión cargue completamente, algo que aclarar es que si se le asigna el temporizador a una variable este no solo hace la parte del temporizador sino que también se queda con el elemento que está siendo utilizado dentro del temporizador, otra cosa es que dentro del temporizador se utiliza la librería “expected_conditions”, con esta librería se puede esperar a que un elemento tenga un cierto estado, en este caso se está esperando a que el botón de inicio de sesión sea visible.

En las líneas 24 y 25 se asignan unas variables a los elementos donde se ingresan el usuario y la contraseña, esto se hace utilizando el método “find_element” y como parámetros recibe atributos de HTML y el nombre del atributo de HMTL, estos atributos pueden ser el ID, la clase, entre otros. En las líneas 27 y 28 se utilizan estas variables ya creadas para el usuario y la

contraseña y lo que se hace es utilizar el método “send_keys” para que el programa digite el usuario y contraseña que el usuario registro en la aplicación creada y después en la línea 29 se pulsa el botón de inicio de sesión con el método “click”.

En la línea 31 se utiliza el temporizador para esperar a que cargue la página principal de la aplicación Multi Timesheet, esta página se puede ver en la imagen 2 de este documento. En la línea 33 se crea una variable con la URL de la visualización mensual y en la línea 34 se redirecciona el navegador a esta página. Cuando ya la página de la visualización mensual ha cargado se crea una variable con el temporizador, esta variable espera a que el número del primer día de la semana sea visible, para después pulsar esta fecha y así poder abrir la ventana de registro de horas.

En la línea 39 se crea una variable con una función del segundo script, este script será documentado más adelante, lo necesario por ahora es que esta función recibe como parámetros el Dataframe en el que están los registros de horas trabajadas por el usuario, esta función retornara el mismo Dataframe, pero con unas columnas nuevas. En la línea 41 se utiliza la otra función del segundo script y esta función es la encargada de ingresar los datos en los campos específicos y de guardar dichos registros en la aplicación de Multi Timesheet, esta función tiene dos procesos internamente que dependen de si es el primer registro o no del Dataframe. Más adelante se explicarán estas dos funciones del segundo script, por el momento a esta segunda función le entran como parámetros el WebDriver que se está utilizando, el temporizador y el Dataframe que se editó en la anterior línea de código.

Al final de este primer script hay un temporizador que espera 8 segundos, para que el usuario pueda ver como quedo el registro de los datos que el ingreso en la aplicación creada, después de este tiempo el navegador se cierra y el usuario queda en la aplicación creada.

8.3.2 SEGUNDO SCRIPT

En el segundo script se tiene las mismas librerías que en el primer script por lo que no se nombraran, pero si se muestran en la siguiente imagen.

upload_data.py

```python
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

import time
import pandas as pd

```

Imagen 10 *Librerías script upload_data*

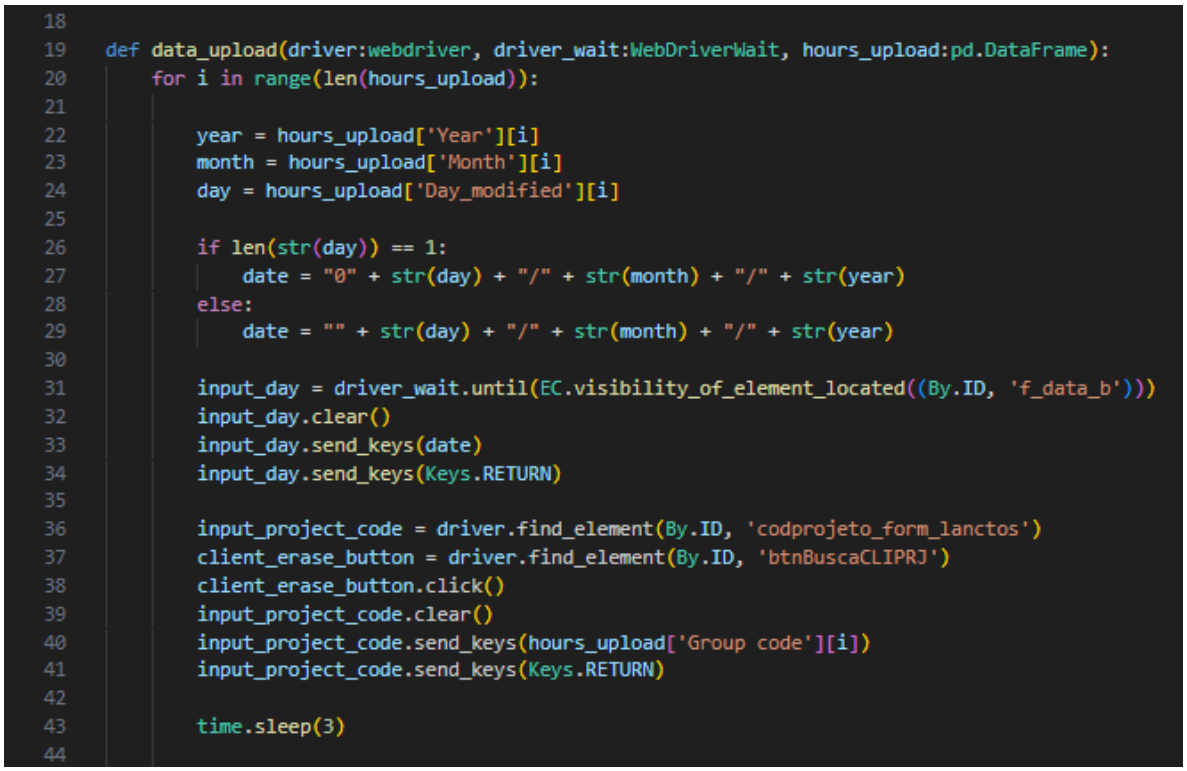
En la siguiente imagen se muestra la primera función de este segundo script, esta función tiene como nombre “date_extraction”, esta función recibe como parámetros el Dataframe que contiene los registros hechos por el usuario en la aplicación creada, al final del proceso de esta aplicación se retorna el mismo Dataframe, pero con 3 columnas nuevas y los datos originales no son modificados en ningún momento.

```python

def date_extraction(hours_upload:pd.DataFrame) -> pd.DataFrame:
    hours_upload['Year'] = hours_upload['Day'].apply(lambda x: x.year)
    hours_upload['Month'] = hours_upload['Day'].apply(lambda x: x.month)
    hours_upload['Day_modified'] = hours_upload['Day'].apply(lambda x: x.day)

    return hours_upload

```

Imagen 11 *Función date_extraction del script upload_data*

Como se aprecia en la imagen, la función “date_extraction” crea tres columnas nuevas, una columna para el año, una columna para el mes y una columna para el día. Estas tres columnas nuevas se añadieron para poder leer solamente el Dataframe y traer la información que se necesita para ingresarla en la aplicación de Multi Timesheet, si no se hace este proceso antes se tendría que hacer cuando se van a subir las horas y esto implica más gasto en tiempo y en recursos que hacerlo antes, es por esto por lo que se dividen los procesos para no saturar un proceso con varias tareas.

La siguiente función es la responsable de registrar todos los datos por cada registro de horas trabajadas que hizo el usuario en la aplicación final. Como esta es una función muy larga se documentará en dos partes.



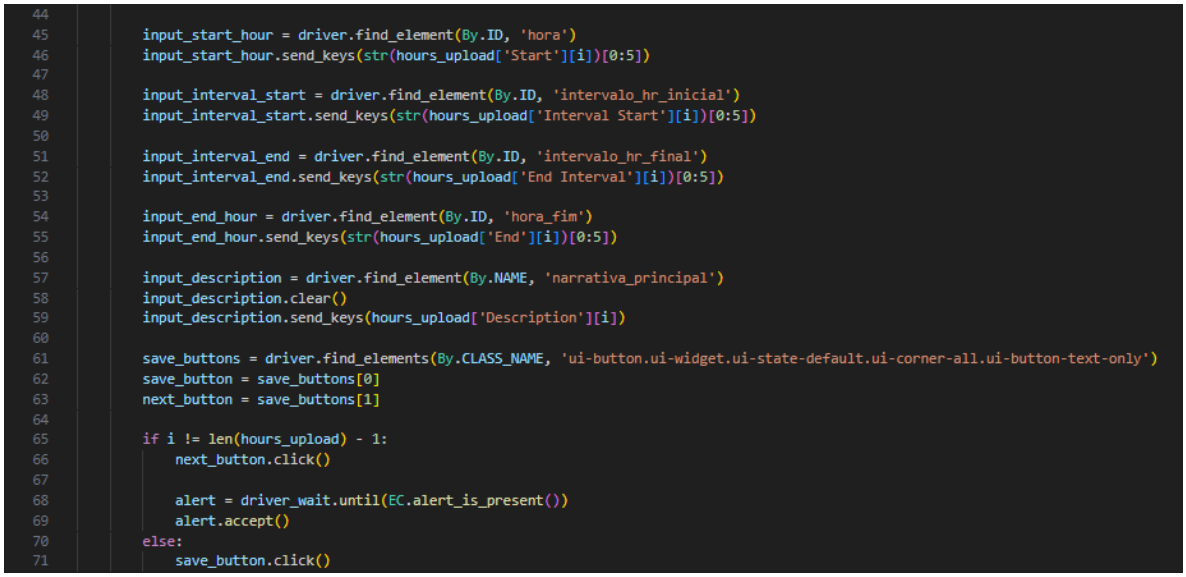
```python

def data_upload(driver:webdriver, driver_wait:WebDriverWait, hours_upload:pd.DataFrame):
    for i in range(len(hours_upload)):

        year = hours_upload['Year'][i]
        month = hours_upload['Month'][i]
        day = hours_upload['Day_modified'][i]

        if len(str(day)) == 1:
            date = "0" + str(day) + "/" + str(month) + "/" + str(year)
        else:
            date = "" + str(day) + "/" + str(month) + "/" + str(year)

        input_day = driver_wait.until(EC.visibility_of_element_located((By.ID, 'f_data_b')))
        input_day.clear()
        input_day.send_keys(date)
        input_day.send_keys(Keys.RETURN)

        input_project_code = driver.find_element(By.ID, 'codprojeto_form_lanctos')
        client_erase_button = driver.find_element(By.ID, 'btnBuscaCLIPRJ')
        client_erase_button.click()
        input_project_code.clear()
        input_project_code.send_keys(hours_upload['Group code'][i])
        input_project_code.send_keys(Keys.RETURN)

        time.sleep(3)

```

Imagen 12 *1ra parte función data_upload*

Como se aprecia en la imagen el nombre de la función es “data_upload” y esta función recibe como parámetros el WebDriver, el temporizador y el Dataframe resultante de la función “date_extraction”. Al inicio de esta función se observa un bucle, que en este caso es un bucle “for” y este buble recorrerá cada registro o fila del Dataframe.

En las siguientes líneas se declaran tres variables, una para el año, otra para el mes y una última para el día, estas variables al final se convierten variables de tipo texto y se concatenan. Al momento de concatenar estas variables se mira si la variable día tiene una longitud igual a 1 o no, si la longitud es igual a 1 se le agrega un 0 antes de la variable día y si la longitud es mayor a 1 no se le añade nada antes. Con el proceso de concatenación hecho se crea una variable con el temporizador, en este caso el temporizador esperara a la visibilidad del campo donde se cambia la fecha, cuando este ya sea visible se realizarán 3 pasos, primero se limpiará este campo, segundo se digitarán la fecha correspondiente y por último se pulsará la tecla “Enter”.

Después se busca el campo donde se digitará el código del proyecto y también se busca el botón de borrar el contenido del cliente, se busca el campo para digitar el código del proyecto más adelante y se busca el botón de borrar el contenido del cliente, para que cuando se digite otro código de proyecto se pueda encontrar el código, ya que cada cliente tiene sus propios códigos de proyectos. Los próximos pasos son dar click en el botón de borrar el cliente, después limpiar el campo donde se digita el código del proyecto, seguido de esto se digitará el código del proyecto y para finalizar se dará pulsará la tecla “Enter”. Antes de seguir con la otra parte del código se deja un temporizador de 3 segundos para que la aplicación Multi Timesheet reconozca el código del proyecto y rellene automáticamente el campo del cliente.



```
44
45          input_start_hour = driver.find_element(By.ID, 'hora')
46          input_start_hour.send_keys(str(hours_upload['Start'][i])[0:5])
47
48          input_interval_start = driver.find_element(By.ID, 'intervalo_hr_inicial')
49          input_interval_start.send_keys(str(hours_upload['Interval Start'][i])[0:5])
50
51          input_interval_end = driver.find_element(By.ID, 'intervalo_hr_final')
52          input_interval_end.send_keys(str(hours_upload['End Interval'][i])[0:5])
53
54          input_end_hour = driver.find_element(By.ID, 'hora_fim')
55          input_end_hour.send_keys(str(hours_upload['End'][i])[0:5])
56
57          input_description = driver.find_element(By.NAME, 'narrativa_principal')
58          input_description.clear()
59          input_description.send_keys(hours_upload['Description'][i])
60
61          save_buttons = driver.find_elements(By.CLASS_NAME, 'ui-button.ui-widget.ui-state-default.ui-corner-all.ui-button-text-only')
62          save_button = save_buttons[0]
63          next_button = save_buttons[1]
64
65          if i != len(hours_upload) - 1:
66              next_button.click()
67
68              alert = driver_wait.until(EC.alert_is_present())
69              alert.accept()
70          else:
71              save_button.click()
```

Imagen 13 *2da parte función data_upload*

En la imagen se aprecia que desde la línea de código 45 hasta la 55 se parecen mucho, esto es porque se están ingresando los horarios de entrada, de inicio de intervalo, de fin de intervalo y des salida. Para cada campo de los horarios se halla el campo y después se digitan los horarios. Después de digitar los horarios se busca el campo de descripción, este se busca por el atributo nombre, después de encontrarlo se limpia el campo y después se envía la descripción que el usuario haya ingresado en la aplicación creada. Por último, se buscan los botones de guardar y de guardar y continuar, esto se hace utilizando un método parecido al de encontrar un solo elemento, en este caso el método “find_elements” devuelve todos los elementos que tengan el nombre del atributo que se le pase al método

como parámetro, para buscar los botones se busca la clase y se dividen ambos botones en dos variables.

Al final hay un condicional el cual revisa si el valor de la variable iteradora “i” es diferente al total de registros del Dataframe menos 1, si esta condición es verdadera el programa dará click en el botón de guardar y continuar, después de dar click aparecerá una alerta, ha esta alerta se le dará aceptar para continuar con el siguiente registro. Si el condicional no es verdadero quiere decir que la variable iteradora “i” ya es igual al total de registros del Dataframe menos 1, y si esto es así el programa dará click en el botón de guardar y se terminará el proceso de registro de horas trabajadas.

8.4 DISEÑO Y DESARROLLO DE UNA INTERFAZ GRÁFICA

8.4.1 TERCER SCRIPT Y DISEÑO DE LA APLICACIÓN

Este tercer script realiza la integración del diseño de la aplicación con Python y con los dos scripts anteriores, cabe aclarar que solo se necesitaría la integración con el script “full_process”, ya que este contiene todo el proceso e integrado el script “upload_data”. Para el diseño de la interfaz gráfica se utilizó la aplicación “designer” que se instala cuando se instala la librería de PyQT6.

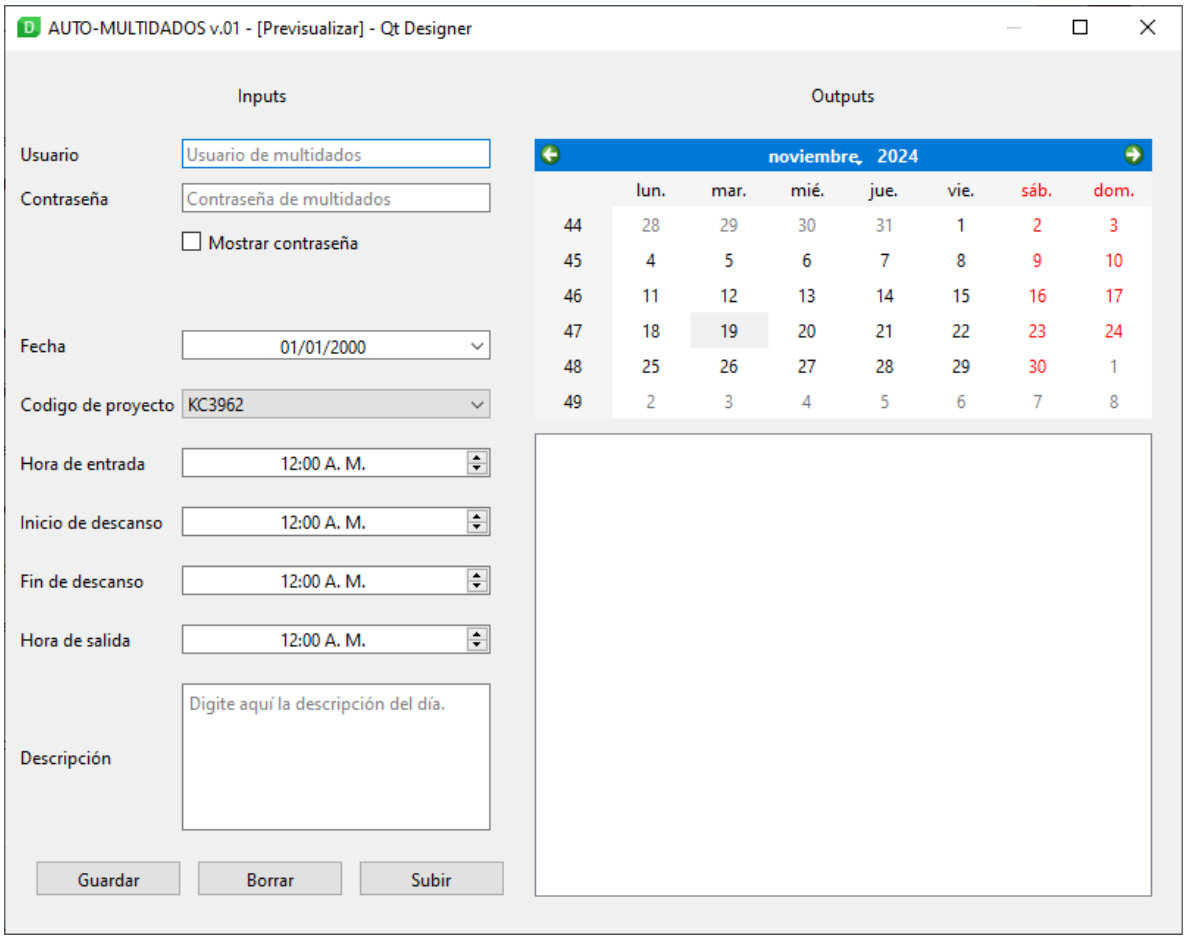


Imagen 14 *Interfaz gráfica de la aplicación creada*

En la anterior imagen se aprecia la interfaz gráfica final, en esta se aprecian dos columnas, una para los ingresos que debe hacer el usuario y una para mostrar las salidas de la aplicación. Al inicio de la columna de ingresos hay dos campos editables, uno para el usuario de la aplicación de Multi Timesheet y otro para la contraseña de la aplicación Multi Timesheet, además hay una casilla de selección para decidir si se podrá ver la contraseña o no, por defecto la contraseña solo muestra puntos negros, para que nadie más pueda verla.

En la misma columna de ingresos hay un campo para ingresar la fecha, ya sea de manera manual o seleccionando por el calendario que se despliega, más abajo hay una lista desplegable donde deberán estar todos los códigos de los proyectos que se tengan dentro de la empresa. Después de la selección del código del proyecto se debe seleccionar los horarios de ingreso, de inicio de descanso, de fin de descaso y de salida, por último, hay un campo editable para la descripción. Al

final de todos los campos donde el usuario debe seleccionar e ingresar datos están los botones, estos botones permiten guardar un registro, borrar un registro previamente seleccionado en la tabla de la columna de salidas y subir todos los registros guardados.

En la columna de salidas hay un calendario y una tabla, esta tabla esta por el momento vacía, pero al iniciar la aplicación por el script de integración se puede percibir las columnas que componen esta tabla. Esta tabla almacenara todos los registros guardados, seleccionar el registro que se desea eliminar y mostrar los registros filtrados por el calendario. El calendario esta para poder seleccionar una fecha y poder filtrar los registros de acuerdo con la fecha seleccionada.

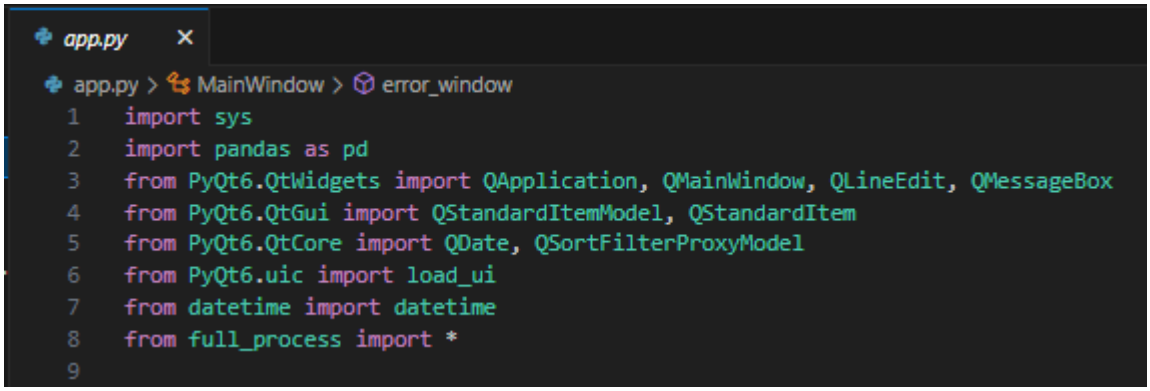


Imagen 15 *Librerías script app*

En la imagen anterior se aprecian las librerías que se usaron dentro del tercer script. La librería de “sys” es utilizada para correr la aplicación en una ventana, la librería de pandas es para la creación del Dataframe, las librerías de PyQT6 son utilizadas por si se deben crear algún otro elemento y para cargar la interfaz gráfica ya creada, la librería de datetime es utilizada para convertir una variable de tipo texto en tipo fecha, y por último se importa el primer script que contiene todo el proceso de automatización.

```
9
10  class MainWindow(QMainWindow):
11      def __init__(self):
12          super().__init__()
13          load_ui.loadUi('appui.ui', self)
14
15          self.model = QStandardItemModel()
16          self.model.setHorizontalHeaderLabels(['Fecha', 'Proyecto', 'Entrada', 'Inicio descanso', 'Fin descanso', 'Salida', 'Descripción'])
17
18          self.filter_model = QSortFilterProxyModel(self)
19          self.filter_model.setSourceModel(self.model)
20          self.tableView.setModel(self.filter_model)
21
22          self.dateEdit.setDate(QDate.currentDate())
23
24          self.checkBox_passw.stateChanged.connect(self.toggle_show_password)
25          self.btn_save.clicked.connect(self.push_save_button)
26          self.calendarWidget.selectionChanged.connect(self.table_filter)
27          self.btn_erase.clicked.connect(self.push_erase_button)
28          self.btn_upload.clicked.connect(self.push_upload_button)
29
```

Imagen 16 *Clase MainWindow y constructor*

En la anterior imagen se aprecia la creación de una clase heredando los atributos y métodos de la clase “QMainWindow”. Dentro del constructor se inicia con traer el constructor de la clase “QMainWindow” y cargando la interfaz gráfica, para esto solo se debe pasar la ruta donde se guardó el archivo de la interfaz gráfica, en este caso se pasa solo el nombre porque este archivo está en la misma carpeta que el tercer script.

Después se crea un modelo estándar de ítems, este es el que se pasara a la tabla para poder ver los registros guardados por el usuario. Después de haber creado el modelo estándar de ítems se crean los nombres de las columnas que van a existir en la tabla, estos son los mismos nombres que los campos que debe seleccionar o digitar el usuario. Después se crea un modelo para filtrar la tabla, al crearlo se le debe pasar el modelo estándar de ítems que será la fuente que el nuevo modelo filtrara, al final se pasa este modelo de filtrado como el modelo que se mostrara en la tabla de la aplicación.

En la próxima línea se modifica la fecha del campo donde el usuario digitara la fecha y se deja la fecha actual, esto con ayuda de la clase “QDate”. Ya para finalizar el constructor se enlazan los botones, cambios de estado y cambios de selección con los métodos creados posteriormente. Para este caso se enlazan los botones guardar, borrar, y subir, además se enlaza la casilla de selección, pero esta esta enlazada cuando cambia de estado y por último se enlaza la selección de fecha en el calendario.

```
30
31      def toggle_show_password(self):
32          if self.checkBox_passw.isChecked():
33              self.lineEdit_password.setEchoMode(QLineEdit.EchoMode.Normal)
34          else:
35              self.lineEdit_password.setEchoMode(QLineEdit.EchoMode.Password)
36
```

Imagen 17 *Método toggle_show_password*

En la imagen anterior se evidencia el método “toggle_show_password”, este método revisa si la casilla de selección esta seleccionada o no y si esta seleccionada cambia el modo de vista del campo editable de la contraseña a normal, donde se podría ver la contraseña, y si no está seleccionada cambia al modo de contraseña, donde el campo editable no deja ver texto si no puntos negros, simbolizando los caracteres de la contraseña.

```
37
38      def table_filter(self):
39          filter_text = self.calendarWidget.selectedDate().toString("dd/MM/yyyy")
40          self.filter_model.setFilterRegularExpression(filter_text)
41          self.filter_model.setFilterKeyColumn(0)
42
```

Imagen 18 *Método table_filter*

La imagen anterior muestra el método que permite filtrar la tabla por la fecha seleccionada. Primero se trae la fecha seleccionada en formato texto y después se utiliza esta fecha en formato texto para filtrar por expresión regular y sobre la columna que se quiere filtrar, en este caso se filtra por primera columna que es la que contiene las fechas.

```
43
44      def data_transformation(self) -> list:
45          date = datetime.strptime(self.dateEdit.date().toString("dd/MM/yyyy"), "%d/%m/%Y")
46          project_code = self.comboBox_project_code.currentText()
47          time_entry = self.timeEdit_entry_hour.time().toString("HH:mm")
48          time_start_break = self.timeEdit_start_break.time().toString("HH:mm")
49          time_end_break = self.timeEdit_end_break.time().toString("HH:mm")
50          time_leaving = self.timeEdit_leaving_hour.time().toString("HH:mm")
51          description = self.plainTextEdit_description.toPlainText()
52
53          return [date, project_code, time_entry, time_start_break, time_end_break, time_leaving, description]
54
```

Imagen 19 *Método data_transformartion*

En este método lo que se hace es transformar el resultado del campo fecha que ingresa el usuario, se va a pasar de un formato de “QDate” a un formato de

“datetime”. En el resto del método lo que se hace es traer el código del proyecto como texto, los horarios como texto y la descripción como texto. Al final esta función retorna una lista con todos estos campos transformados.

```
55
56      def push_save_button(self):
57          inputs = self.data_transformation()
58
59          date = inputs[0].strftime("%d/%m/%Y")
60          project_code = inputs[1]
61          time_entry = inputs[2]
62          time_start_break = inputs[3]
63          time_end_break = inputs[4]
64          time_leaving = inputs[5]
65          description = inputs[6]
66
67          if len(description) > 0:
68              self.model.appendRow([QStandardItem(date),
69                                    QStandardItem(project_code),
70                                    QStandardItem(time_entry),
71                                    QStandardItem(time_start_break),
72                                    QStandardItem(time_end_break),
73                                    QStandardItem(time_leaving),
74                                    QStandardItem(description)])
75          else:
76              self.error_window("CAMPO DE DESCRIPCIÓN VACIO", "Llenar el campo descripción, ya que no puede estar vacio")
77
```

Imagen 20 *Método push_save_button*

Este método es el que se utiliza cuando se pulsa el botón de guardar en la aplicación creada. Lo primero que hace este botón es reutilizar el método anterior, es decir se invoca el método anterior y se guarda la lista en una variable. Después se extrae cada valor de la lista en una variable diferente y se pasa a un condicional el cual revisa si la longitud de la descripción es mayor a cero, si esto es así se ingresara un nuevo registro en la tabla, si la condición es falsa, es decir que la longitud de la descripción es cero, entonces se invoca un método llamado “error_window” y este lo que hace es desplegar una ventana emergente, mostrando el error que tiene el registro y lo que debe hacer el usuario, estas dos cosas son que muestra la ventana emergente son los parámetros que ingresan al método.

```
78
79      def push_erase_button(self):
80          selected_indexes = self.tableView.selectedIndexes()
81
82          if selected_indexes:
83              table_index = selected_indexes[0]
84              table_filter_index = self.filter_model.mapToSource(table_index)
85              row_delete = table_filter_index.row()
86
87              self.model.removeRow(row_delete)
88
```

Imagen 21 *Método push_erase_button*

Este método es accionado cuando se pulsa el botón borrar en la aplicación creada. Lo primero que hace este método es revisar cual es el índice de la tabla que ha sido seleccionado y se guarda en una variable, después se pasa a un condicional que revisa si hay algo en esa nueva variable creada, si hay algo entonces se toma el índice y se revisa cual sería el índice en el modelo estándar de ítems, esto debe realizarse ya que cuando hay filtros los índices de la tabla pueden variar, cuando ya se tiene el índice verdadero según el modelo estándar de ítems se extrae la fila de ese índice y se elimina.

```
89
90      def push_upload_button(self):
91          if self.credentials_validation(self.lineEdit_user.text(), self.lineEdit_password.text()):
92              data_upload = pd.DataFrame()
93
94              rows = self.model.rowCount()
95
96              for row in range(rows):
97                  long = len(data_upload)
98                  data_upload.loc[long, 'Day'] = datetime.strptime(self.model.item(row, 0).text(), "%d/%m/%Y")
99                  data_upload.loc[long, 'Grooup code'] = self.model.item(row, 1).text()
100                 data_upload.loc[long, 'Start'] = self.model.item(row, 2).text()
101                 data_upload.loc[long, 'Interval Start'] = self.model.item(row, 3).text()
102                 data_upload.loc[long, 'End Interval'] = self.model.item(row, 4).text()
103                 data_upload.loc[long, 'End'] = self.model.item(row, 5).text()
104                 data_upload.loc[long, 'Description'] = self.model.item(row, 6).text()
105
106             data_upload.columns = ['Day', 'Group code', 'Start', 'Interval Start', 'End Interval', 'End', 'Description']
107
108             full_process_upload(self.lineEdit_user.text(), self.lineEdit_password.text(), data_upload)
109
```

Imagen 22 *Método push_upload_button*

Este método es invocado cuando se pulsa el botón de subir en la aplicación creada. Lo primero que realiza este método es revisar si las credenciales son válidas, esto lo hace invocando otro método llamado “credentials_validation”, los parámetros de este método son las credenciales en formato texto. Si el método “credentials_validation” devuelve el valor verdadero, significa que las credenciales

son válidas y se puede seguir, en caso de que no sean válidas el mismo método de “credentials_validation” tiene un mensaje de error. Después dentro del condicional se crea un Dataframe, que es donde se almacenaran los registros, y se saca la cantidad de registros o filas que tiene la tabla. Con estos datos se pasa a un bucle “for” donde se pasarán por todos los registros que contiene la tabla. Dentro del bucle siempre se crea una variable con el total de registros que tiene le Dataframe, después se ingresa cada dato que el usuario ingreso en el registro en una columna especifica, es decir la fecha va a ir a la columna de “Day” en el Dataframe y así consecutivamente por cada parte del registro. Al finalizar el bucle se le da el nombre correspondiente a cada columna. Para finalizar se llama la función “full_process_upload” del primer script y se le pasa como argumento, el usuario, la contraseña y el Dataframe resultante.

```
110
111     def error_window(self, error_text:str, inf_text:str):
112         msg = QMessageBox(self)
113         msg.setIcon(QMessageBox.Icon.Critical)
114         msg.setWindowTitle("Error")
115         msg.setText(error_text)
116         msg.setInformativeText(inf_text)
117         msg.setStandardButtons(QMessageBox.StandardButton.Close)
118         msg.exec()
119
```

Imagen 23 *Método error_window*

Este método es usado cuando suceda algún error que puede ha sido detectado con anterioridad dentro de la aplicación creada. Este método tiene como parámetros de entrada el texto del error y el texto de información, el texto de error esclarece cual es el error y el texto de información le indica al usuario que debe realizar para que no aparezca la ventana de error.

```
120
121     def credentials_validation(self, username:str, password:str) -> bool:
122         credentials_validated = False
123
124         if len(username) > 0 and len(password) > 0:
125             credentials_validated = True
126         else:
127             self.error_window("USUARIO O CONTRASEÑA NO VALIDO", "Por favor ingresar un usuario y una contraseña")
128
129         return credentials_validated
```

Imagen 24 *Método credentials_validation*

Este método valida que el usuario y contraseña ingresados por el usuario sean válidos, por el momento no se tiene una base de datos con los usuarios y

contraseñas de cada empleado que utiliza la aplicación de Multi Timesheet, por lo que el programa valida es que el usuario haya puesto algo así sea erróneo, tanto en el usuario como en la contraseña. Si el usuario o la contraseña no tienen ningún carácter el programa invocara el método “error_window” y le indicara al usuario que el usuario o contraseña no son válidos. Al final el método retorna un booleano indicando verdadero si el usuario y contraseña son válidos y falso si no lo son.

```
130
131 if __name__ == '__main__':
132     app = QApplication(sys.argv)
133     gui = MainWindow()
134     gui.show()
135     sys.exit(app.exec())
```

Imagen 25 *main del script app*

En la imagen anterior se muestra la parte que permite que la aplicación se inicie y corra. Al inicio se crea un objeto QApplication, después se crea un objeto MainWindow, se utiliza un método heredado para mostrar la interfaz gráfica y al final se ejecuta la aplicación.

8.5 VISUALIZCIÓN DEL PRODCUTO FINAL

En esta parte del documento se mostrarán imágenes de como quedo el producto final, todas las ventanas emergentes y un ejemplo de cómo se ve el resultado final en la aplicación Multi Timesheet.

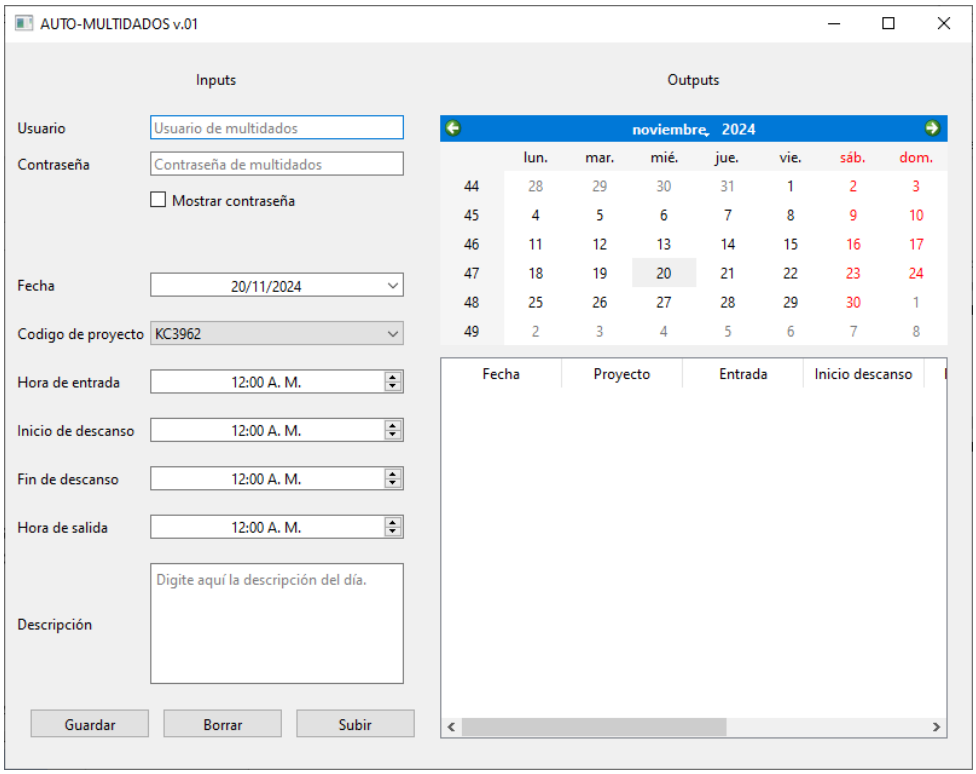


Imagen 26 *Pantalla inicial de la aplicación creada*

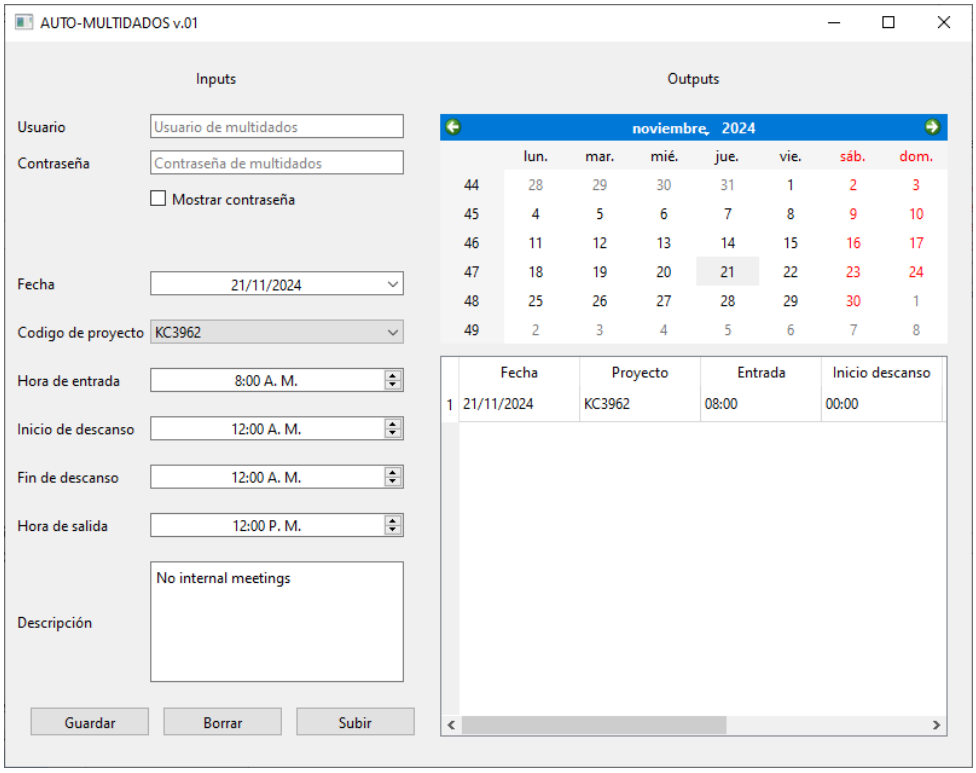


Imagen 27 *Ejemplo de registro de datos*

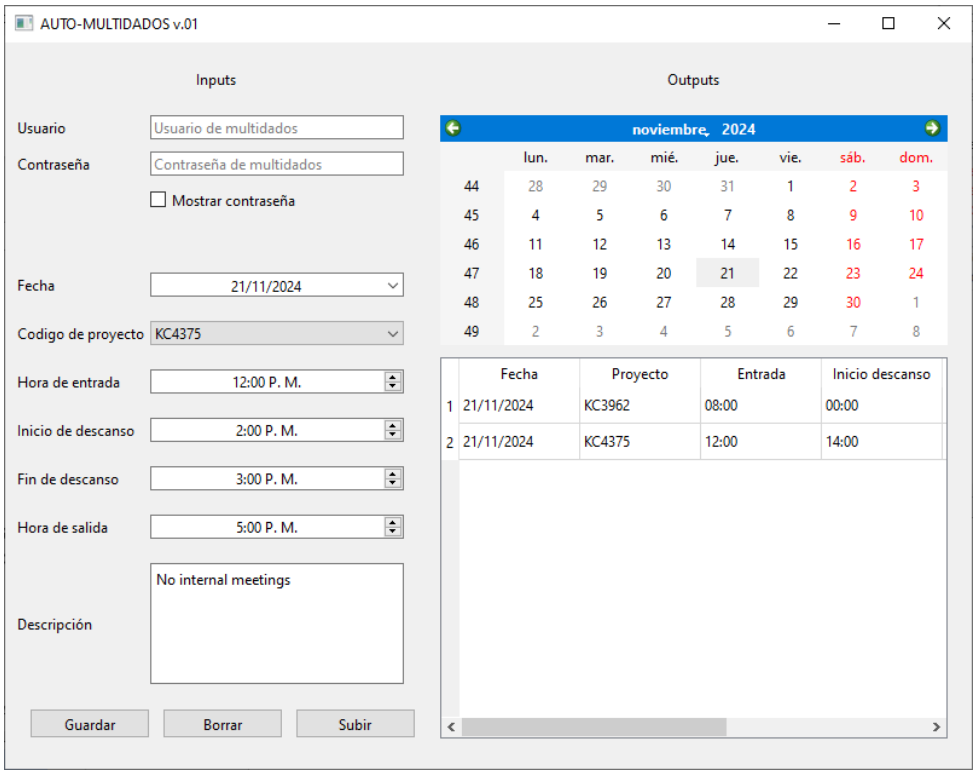


Imagen 28 *Ejemplo varios registros de datos*

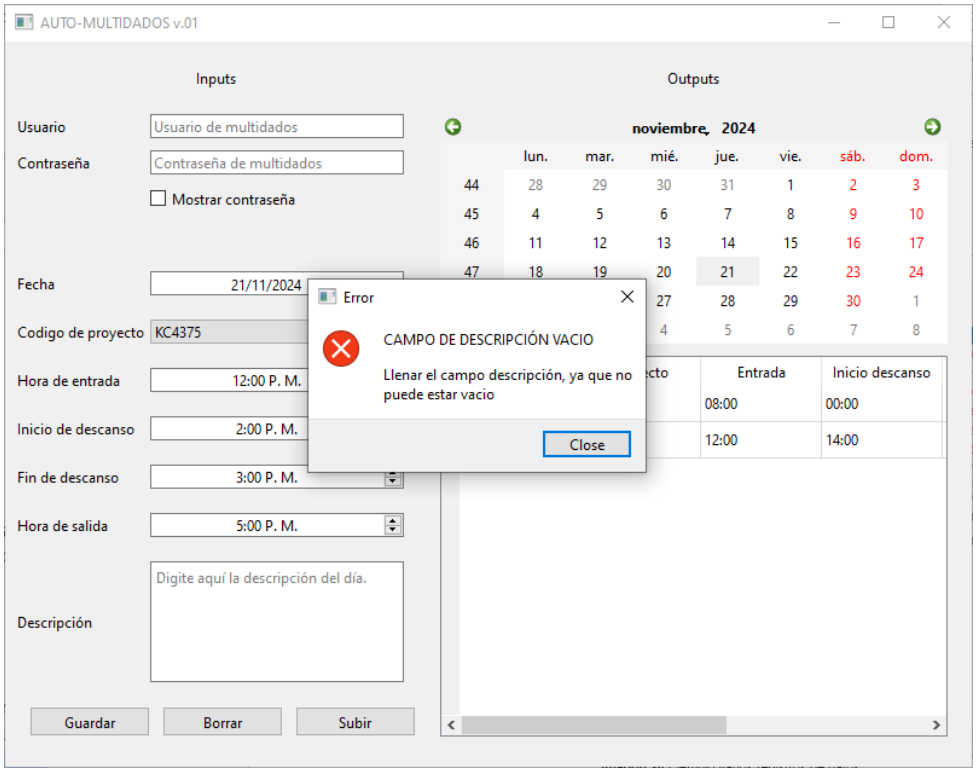


Imagen 29 *Ejemplo error en el campo descripción*

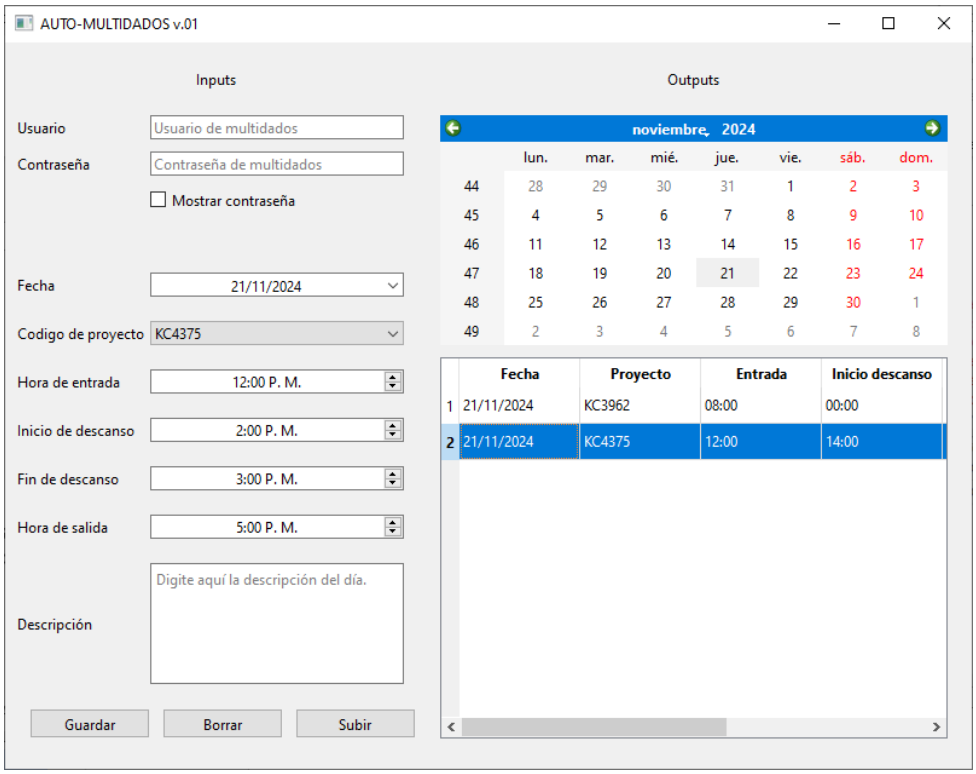


Imagen 30 *Ejemplo de selección de registro*

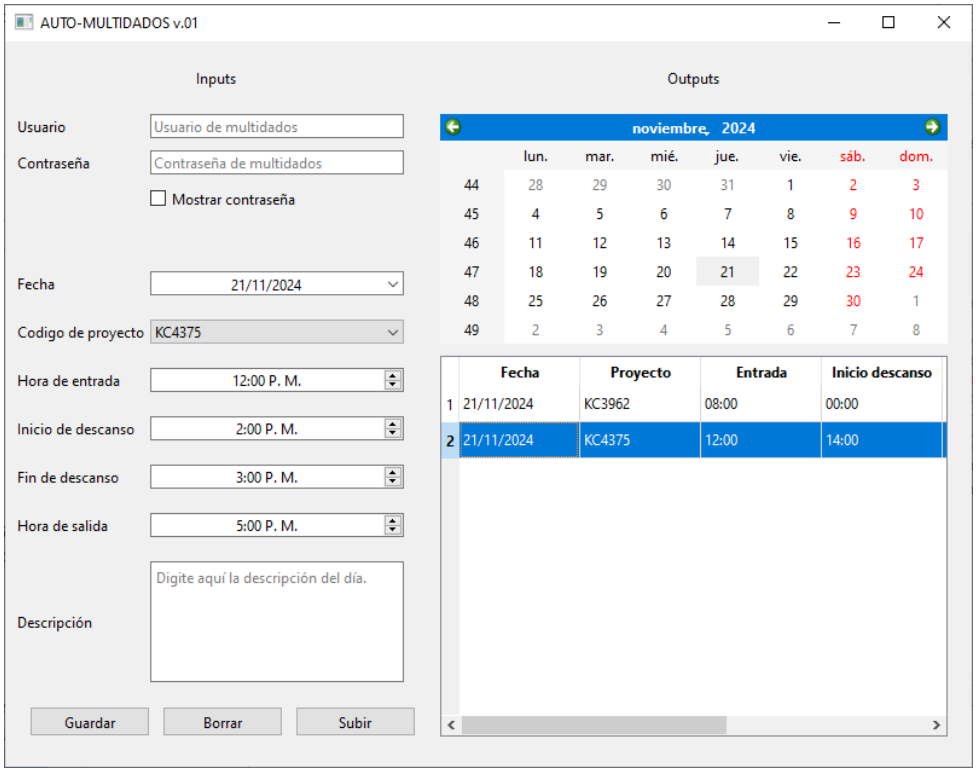


Imagen 31 *Ejemplo de borrado de registro*

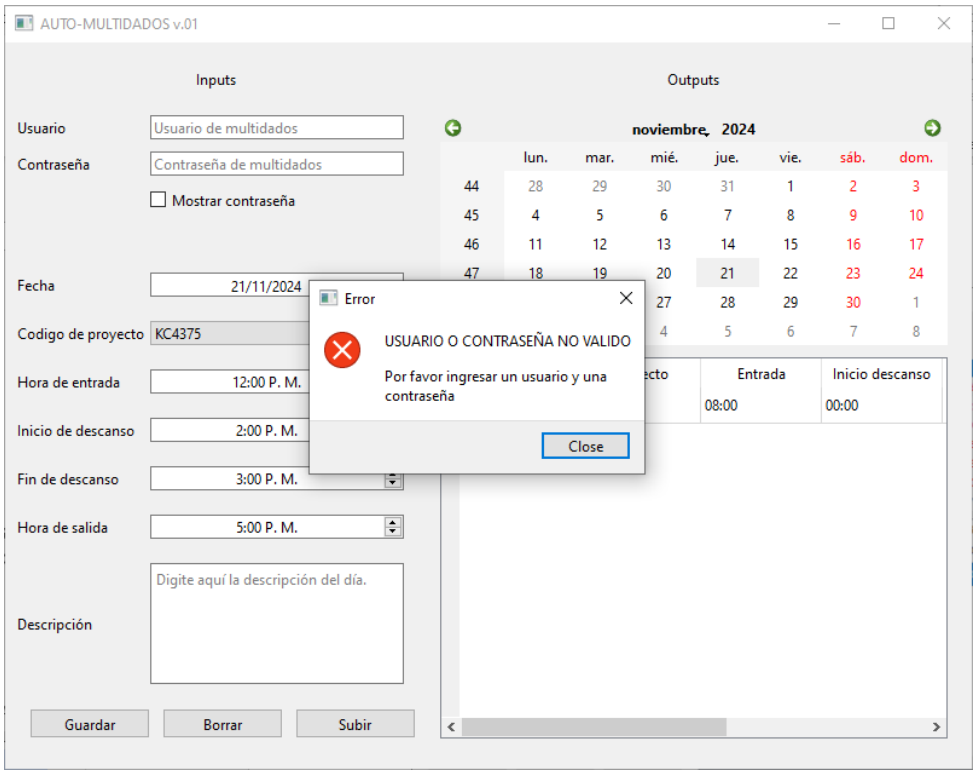


Imagen 32 *Ejemplo de error en credenciales*

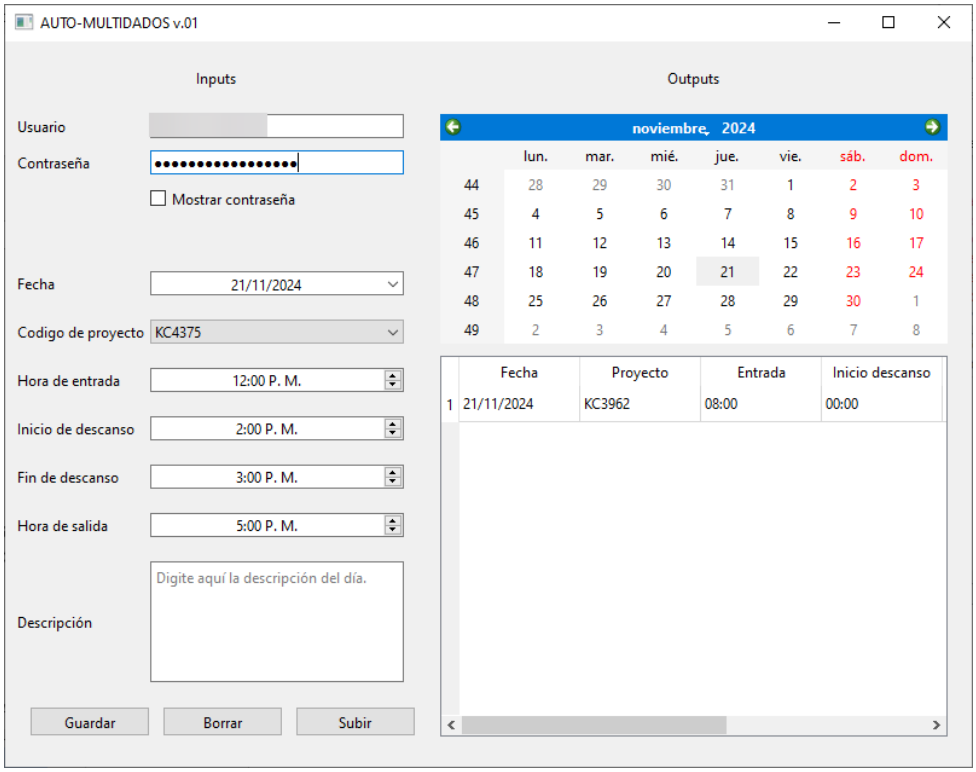


Imagen 33 *Ejemplo de credenciales válidas*

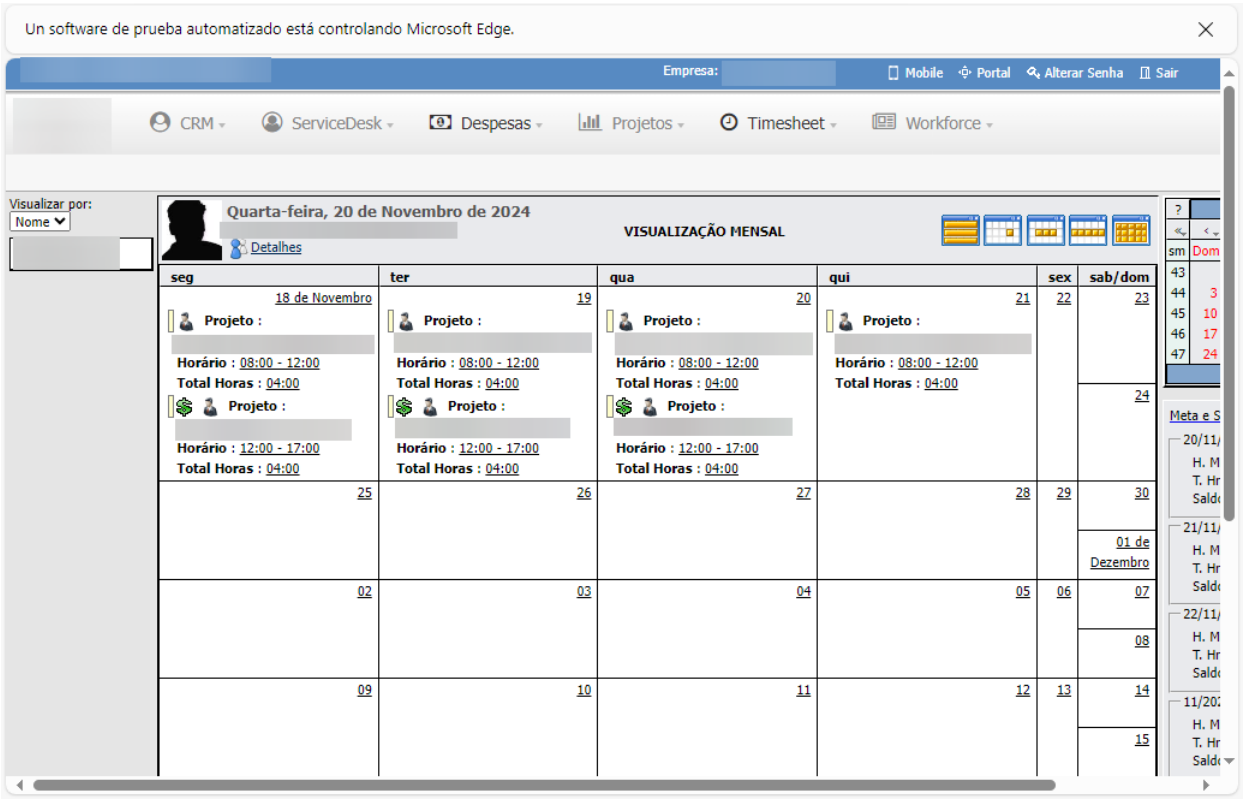


Imagen 34 *Ejemplo de subida de datos satisfactoria*

CONCLUSIONES

- El proceso de registrar las horas trabajadas en la aplicación Multi Timesheet es tedioso y consume mucho tiempo. Aunque no es complicado, requiere varios pasos. Con la nueva aplicación, el tiempo de registro se reduce significativamente, ya que, al ingresar la misma información en ambas plataformas, solo será necesario un clic para que el programa registre todas las horas, siendo mucho más rápido que un ser humano.
- Los requisitos funcionales del programa se cumplieron satisfactoriamente. El usuario puede autenticarse de cierta manera, registrar las horas trabajadas, guardar y eliminar registros, y guardar múltiples registros, aunque uno a la vez. Además, puede visualizar los registros, confirmarlos y subirlos a la aplicación Multi Timesheet. También tiene la opción de filtrar los registros por fecha.
- En cuanto a los requisitos no funcionales, el programa cumple con las especificaciones, ya que actualmente la aplicación debe utilizarse desde un sistema local. Es decir, cada usuario tendría su propia instalación del programa, sin conexión a internet ni integración con otras aplicaciones. Por lo tanto, el rendimiento de la aplicación dependerá de las características de la máquina que ejecute el programa. A pesar de esto, el programa tiene un alto potencial de escalabilidad, ya que se pueden mejorar muchas de sus funcionalidades. Además, la aplicación logró integrar de manera exitosa todas las tecnologías utilizadas.
- Con esta primera versión del programa se ha logrado automatizar el proceso de registro de horas en la aplicación Multi Timesheet, siempre y cuando el usuario ingrese correctamente sus credenciales, la fecha, los horarios y la descripción de las actividades realizadas.
- Aunque no se realizaron pruebas con los usuarios finales de la aplicación, se percibe una mejora en los tiempos de registro de horas y en la eficiencia del proceso de registro.
- Durante el análisis del proceso de registro de horas en Multi Timesheet, se observó que el botón "Guardar y continuar" funciona correctamente en portugués, pero no en otros idiomas. Por ello, se decidió que la automatización para cargar las horas a la aplicación mantenga el idioma por defecto, que es el portugués.
- Los primeros cinco requisitos funcionales se cumplen adecuadamente. La aplicación permite una autenticación funcional (aunque no óptima), y los usuarios pueden ingresar la fecha, código de proyecto, horarios y descripción. Además, pueden guardar, eliminar y revisar los registros de

horas, y estos se suben automáticamente mediante Selenium sin intervención humana. Los registros deben guardarse uno por uno, pero el sistema permite gestionar múltiples registros en una sesión.

- El sexto requisito funcional también se cumple, ya que la tabla muestra los campos necesarios fecha, código de proyecto, horarios y descripción, además los usuarios pueden eliminar cualquier registro de la tabla.
- El séptimo requisito funcional se cumple, ya que es posible filtrar los registros de la tabla por una fecha seleccionada en el calendario.
- El primer requisito no funcional se cumple parcialmente. Aunque se verificó que la aplicación guarda, visualiza y sube registros correctamente, no se pudo determinar la capacidad máxima de registros que puede manejar, ya que las pruebas realizadas no fueron lo suficientemente exhaustivas.
- El segundo y tercer requisito no funcional no se cumplen completamente. La aplicación aún no está alojada en la nube, por lo que se ejecuta de manera individual en cada computador. Además, no se realizaron pruebas con usuarios, por lo que la facilidad de uso y la intuición de la aplicación deben mejorarse en futuras versiones.
- El cuarto requisito se cumple completamente, ya que la aplicación está integrada con la librería Selenium, lo que permite automatizar el proceso de registro de horas de manera secuencial.
- El documento muestra que la aplicación carga correctamente los horarios registrados en Multi Timesheet, lo que indica que los scripts funcionan adecuadamente.

Referencias

Izaguirre Castellanos, E. (2012). *Sistemas de automatización.* Editorial Feijóo.

Multidados TI. (2024, Noviembre 09). *Conheça a MultidadosTI*. Retrieved Noviembre 09, 2024, from MULTIDADOS TI: https://multidadosti.com.br/sobre/

Multidados TI. (9 de Noviembre de 2024). *Gestão de horas trabalhadas Multi Timesheet*. Recuperado el 9 de Noviembre de 2024, de MULTIDADOS TI: https://multidadosti.com.br/multi-timesheet/

Oviedo Regino, E. M. (2015). *Lógica de programación orientada a objetos.* Ecoe Ediciones.

Python. (9 de Noviembre de 2024). *The Python Wiki*. Recuperado el 9 de Noviembre de 2024, de FrontPage - Python Wiki: https://wiki.python.org/moin/

Python. (9 de Noviembre de 2024). *tkinter*. Recuperado el 9 de Noviembre de 2024, de tkinter — Python interface to Tcl/Tk: https://docs.python.org/es/3/library/tkinter.html

pythonpyqt.com. (9 de Noviembre de 2024). *What is PyQt?* Recuperado el 9 de Noviembre de 2024, de Learn Python PyQt: https://pythonpyqt.com/what-is-pyqt/

Selenium. (9 de Noviembre de 2024). *WebDriver*. Recuperado el 9 de Noviembre de 2024, de WebDriver | Selenium: https://www.selenium.dev/documentation/webdriver/