I.E.S LAS SALINAS

PROYECTO FINAL FIN DE GRADO

CURSO 22-23

Ticket Manager

CICLO FORMATIVO GRADO SUPERIOR

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Autor: Daniel Muscalu

El proyecto Ticket Manager tiene como objetivo abordar la problemática que conlleva en una empresa llevar los registros de fichaje del conjunto de trabajadores que operan en ella. Este sistema automatiza el proceso de registro y la gestión de la asistencia del personal, garantizando una solución eficiente y confiable.

Ticket Manager es una aplicación de escritorio donde los usuarios dados de alta en el sistema por un administrador podrán registrar sus horas de entrada, de salida, sus pausas. En la propia base de datos podemos realizar todas las consultas convenientes y modificaciones, sin embargo si necesitamos una solución más visual y amigable, se puede adquirir un módulo adicional de administrador. Sin embargo es un complemento a la solución ERP que se ofrece aquí.

La implementación nos permitirá realizar seguimientos directamente sobre la asistencia, reduciendo los esfuerzos y minimizando errores y mejorando la gestión de recursos permitiendo un mejor cumplimiento de las regulaciones laborales.

Este proyecto se desarrollará utilizando tecnología como Python, Ubuntu Server y CSS y se realizará siguiendo las mejores prácticas de desarrollo de software.

The project Ticket Manager aims to deal with the great problem it means for a business to manage the clock ins/outs of their workers.

This system automats the register process and the management of the personal attendance, guaranteeing an efficient and reliable solution.

Ticket Manager is a desktop application where users registered on the system by an administrator can register their ins/outs and breaks. In the database you can make your check queries and modifications, but if needed there is an administrator module that offers a graphic and more friendly environment. It’s important to highlight that this is just an add-on to the original solution offered.

The implementation will grant us to follow closely the personal attendance, reducing time, effort and errors and letting us make better compliance with laws and job regulations.

This project will be developed using Python, Ubuntu Server and CSS using the best software development practices.

1. Justificación del proyecto

Este proyecto está enfocado en la necesidad de una empresa de recoger los fichajes de sus trabajadores. A continuación explicaré de manera más detallada los motivos:

1. **Automatización:** El sistema permite automatizar el proceso de fichajes de la empresa, eliminando la necesidad de realizarlos de manera manual y así, ahorrando tiempo y esfuerzo al personal encargado de estos.
2. **Exactitud:** Al contar con un sistema digitalizado, se minimizan los posibles errores humanos. Además al estar digitalizado y automatizado permite recopilar los registros de manera mucho más precisa y confiable.
3. **Facilidad de acceso y consulta:** El sistema permite almacenar y organizar de manera centralizada todos los registros de los trabajadores. Eso facilita cualquier posible consulta de la información por parte de los propios trabajadores o el personal encargado de manejarlos.
4. **Mejora en la gestión de recursos humanos:** Contar con un sistema de este estilo permite una gran mejoría en la gestión de recursos humanos en cuanto a la consulta de los datos respecto a la puntualidad, asistencia, etc. Lo que facilitará la toma de decisiones y la implementación de medidas para mejorar la eficiencia y productividad.
5. **Cumplimiento legal y normativo:** El registro de los fichajes de los trabajadores es un requisito legal en España. Este proyecto asegura que los horarios de los trabajadores cumplan con las regulaciones laborales vigentes, evitando sanciones o problemas legales derivados de un incumplimiento.

2. Introducción al proyecto. Origen e inspiración.

Durante todo el tiempo que he trabajado, me he dado cuenta de la importancia que tiene un sistema eficaz de fichajes, y es esencial tener en cuenta que si el sistema no es eficiente, habrá problemas muy importantes para la empresa y para el trabajador. Tanto lógicos cómo legales o financieros. Imagina que al finalizar tu jornada, no te han cogido bien la huella. Ese día o no cuenta o alguien de recursos humanos te puede llegar a llamar por una ausencia injustificada.

En el primer lugar donde trabajé el sistema era uno con huella, era muy rápido para los cientos de trabajadores que había. Sin embargo, una mala interfaz o poco intuitiva te podía llegar a causar un error en el que se duplican dos ficheros y ese trabajador si fichaba no se registraba, vamos que los

de recursos humanos sin darse cuenta podrían cometer un error grave, y el que tenía que dar luego explicaciones eres tú.

Recientemente en mis prácticas, observé otros sistemas que habían. Uno con una interfaz muy mala, dos botones, muchos registros en todo el medio,era intuitiva porque no tenía casi nada, pero era muy poco amigable e incluso ambigua. Y para los chicos de prácticas no había método de fichaje. Escribíamos un mensaje al entrar y al salir, y además mandábamos nuestros datos al finalizar la jornada y las horas en un formulario que repetimos todos los días, no se tardaba mucho tiempo, pero no había un método de fichaje como tal y lo que había estaba muy poco automatizado, además de que podía dar lugar a muchísimos errores humanos, cosa que de hecho, hacía.

En ese momento fue cuando pensé, ¿y si realizo mi propio sistema de fichajes?

2.1. **Organización y Limitaciones de recursos**

La empresa en la que se desarrolla el proyecto es RR Solv. Una empresa dedicada a proporcionar soluciones tecnológicas para la gestión empresarial.Cuenta con empleados de diversos sectores conformando un total de 100. Abajo pongo cuántos hay de cada uno y sus rangos salariales

- Personal administrativo: 20 empleados(1,500€-2,000€)
- Técnicos: 30 empleados(2,000€-3,500€)
- Personal de ventas: 15 empleados(1,500€-3,000€)
- Personal de producción: 25 empleados(1,200€-2,500€)
- Directivos: 10 empleados(3,000€-6,000€)

Hay que tener en cuenta que estos valores son solo sus respectivos rangos, cada empleado cobra en función de proyectos, características de su trabajo o responsabilidades.

Los costes mensuales de una empresa de este tipo:

- El total de los salarios sale a un estimado de 400,000€
- Otros costes adicionales:
- Alquiler del local: 10,000€
- Equipos y tecnologías: 15,000€
- Gastos operativos: 5,000 €

- Marketing y publicidad: 3,000€

Problemas a resolver:

- Gestionar de manera eficiente los fichajes y la asistencia del personal
- Automatizar el proceso de registro y pausas
- Garantizar un sistema confiable y preciso
- Proporcionar una interfaz intuitiva y amigable

Requisitos de la aplicación:

- Sistema de registro y pausas
- Administración de usuarios y permisos
- Base de datos para almacenar los registros y otros datos
- Capacidades de consulta
- Interfaz sencilla de usar

Presupuesto

-Los costos de implementación de la aplicación serán gratuitos, sin embargo el módulo habrá que abonar de manera mensual como se mencionó anteriormente(35 euros en total)

-Los requisitos mínimos del sistema son bajos, es una aplicación que ocupa poco y pretende correr en cualquier ordenador que cumpla con un windows 10

-Los costos de hardware dependen del servidor que se vaya a instalar, una empresa con muchos equipos necesitará servidores más potentes debido a la cantidad de peticiones, pero cualquier PYME puede llegar a gastar entre 100-300 euros en un servidor para alojar la base de datos.

Teniendo en cuenta estos datos, y que en principio este programa piensa alojarse en intranets, la instalación del servidor, cableado y configuración puede llegar a ascender entre los 2,000 y 3,000 euros para una PYME. En empresas de mayor tamaño dependerá de la infraestructura que haga falta.

3. Desarrollo

3.1. Sistemas Operativos y aplicaciones

A la hora de elegir el sistema operativo, he optado por dos soluciones clásicas, sencillas pero eficaces. Para empezar para el servidor que tendré donde se almacenará la base de datos se utilizará un Ubuntu 18.04 en su versión con solo texto.

La decisión es muy clara, requisitos mínimos, ahorro en hardware de un gran servidor carísimo, configuración sencilla, soporte y documentación en línea a mi disposición y actualizaciones.

Esta máquina que he creado tiene unos requisitos sencillos:

- 4GB de RAM
- Disco SATA de 40GB
- Procesador intel i5 de 2,4GHz y gráficos integrados

Hablando ahora de los equipos de mi organización estos dispondrán en su totalidad de un sistema operativo Windows 10. En cuanto a las aplicaciones que estarán en los equipos se utilizarán las siguientes.

- Ticket Manager: La principal aplicación desarrollada para este proyecto.
- Microsoft Office Suite: Son altamente utilizadas y en un entorno ofimático es fundamental tenerlas
- Navegadores web
- Aplicaciones de seguridad: Antivirus, Programas de copias de seguridad y restauración de equipos, programas para cifrado de discos y adblock.
- Aplicaciones de comunicación: Como Microsoft Teams o Skype para facilitar y mejorar la comunicación en el entorno laboral
- Las respectivas herramientas de desarrollo como Visual Studio Code(para múltiples lenguajes de programación)
- Conector ODBC y las librerías de Python respectivas para el funcionamiento de Ticket Manager

3.2 Fase de recogida de información

Una vez que conozco mis requisitos, las aplicaciones que voy a tener, es cuando empezó el verdadero planteamiento sobre la aplicación, el decir, ¿cómo voy a hacerla?

Y ahí es cuando comienza una fase de investigación, de búsqueda de información, de inspiración y es cuando uno llega y pregunta a la gente de su entorno, ¿Qué falla?, ¿Qué se necesita?

Lo primero siempre es preguntarnos ¿qué necesidad hay que cubrir?, y no iba a comenzar por el tejado. Tras hablar con la gente y haber experimentado el sistema que se implementó de fichajes, me hice una idea de lo que hacía falta.
Simplicidad, eficacia, y acortar procesos. Aplicando todo esto junto evitamos la mayor cantidad de errores y hacemos una cosa fundamental para los usuarios, simplificarles la vida.

Lo que necesitaban es una aplicación, nada de formularios donde hay que introducir todos los mismos datos, todos los días, nada de informar de cuando se inicia y se termina la jornada, estos procesos debían terminarse. Tras navegar por las redes sociales, vi un vídeo que me hizo tener una pequeña idea sobre la interfaz, partía sobre la base de que quería dividirla interfaz en dos, y hacerla intentando adaptar una vista moderna. Y ahí es donde comencé mi primer proceso en el desarrollo, el diseño.

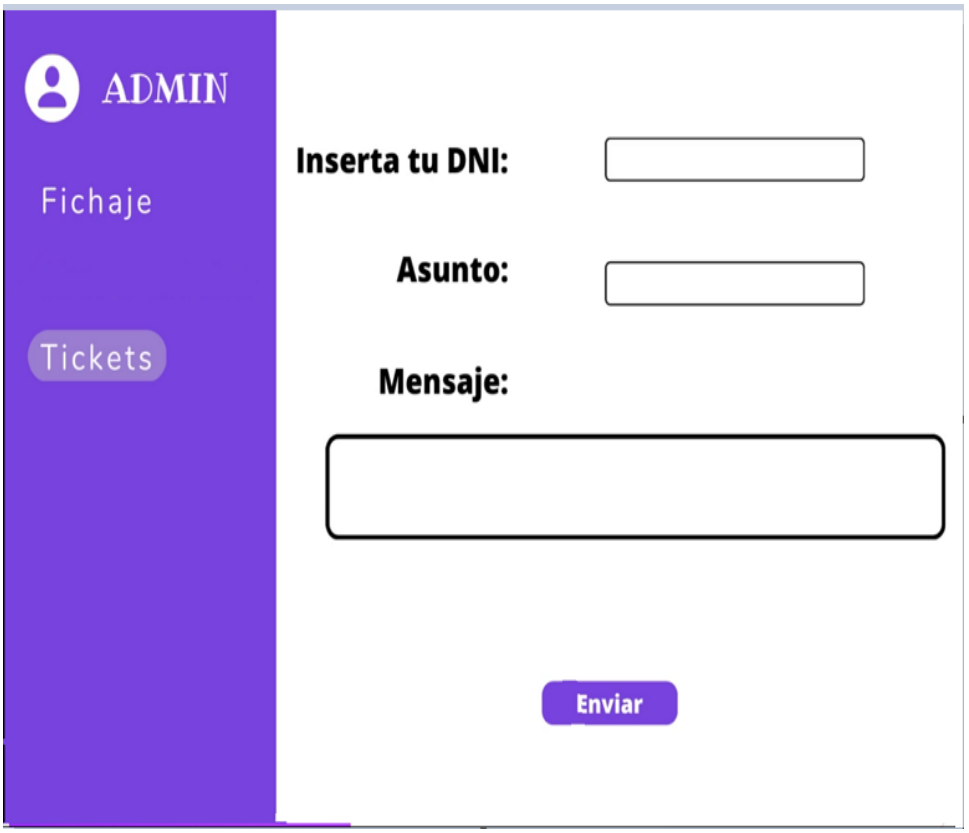
3.3 Diseño de la interfaz

Yo ya tenía en mente como quería empezar mi aplicación, y como quería darle vida a mi marca y demostrar mi logo, y es donde comencé a realizar los primeros bocetos.

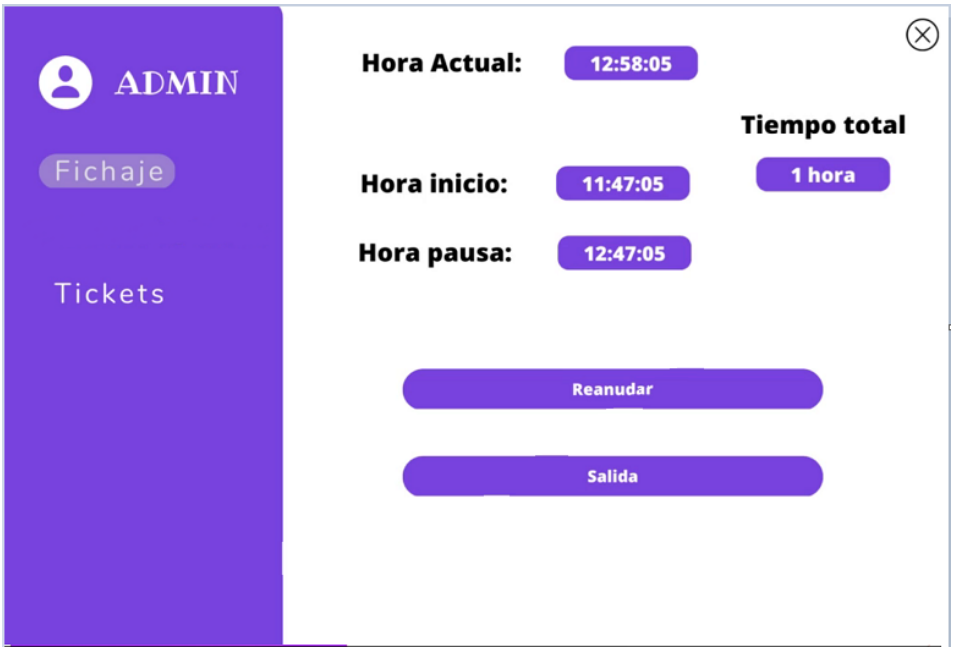
Boceto de inicio de sesión en canva

El proceso para el diseño se realizó utilizando herramientas como Canva y Paint.
Una vez diseñada la ventana de login, fundamental para una aplicación de esta índole, surge la duda de qué apartados debo incluir en mi aplicación.

Tras un tiempo de pensarlo incluí el apartado de fichajes, el apartado perfil, donde muestro toda la información, nombre, dni, apellido, y además permito desde ahí cambiar la contraseña, además incluye un apartado de tickets, donde se pueden realizar comunicaciones importantes o se puede informar de errores.



Boceto de tickets en canva



Boceto de fichaje en canva

Todas las decisiones relacionadas con la interfaz se realizaron pensando en el usuario, se trata de dar una vista estética, una interfaz útil/sencilla que busca la funcionalidad, a la que te acostumbras rápido y es intuitiva y a darle una credibilidad al proyecto, una razón de ser y un motivo para que los usuarios a los que lo destinamos en mi empresa confíen en nuestro trabajo.

Una vez realizados los bocetos iniciales, y sabiendo más o menos lo que quiero/necesito, comienzo con los dos siguientes procesos

3.4. Bases de datos y acceso a datos

En este proyecto se ha utilizado MySQL para almacenar y gestionar la información relacionada con los fichajes y otras actividades. ¿Por qué lo elegí?

- Requisitos funcionales: MySQL cuenta con la capacidad que necesitaba para el proyecto
- Escalabilidad: En términos de volúmen de datos MySQL es otra buena opción
- Rendimiento: A la hora de ejecutar consultas es muy rápido
- Seguridad: Las contraseñas se pueden almacenar de manera segura en la base de datos debido al cifrado SHA que proporciona
- Facilidad de uso: Es una herramienta sencilla de utilizar y dispone de mucha documentación online y un soporte rápido y eficaz

3.4.1. Cómo está estructurada la base de datos

La estructura se compone de tres tablas (horas, tickets, y users)

En la tabla de usuarios tenemos varios campos id, dni, nombre, apellido, departamento y contrasena. El campo que identifica a cada usuario es el id, debido a que por extraño que pueda parecer, puede haber una ínfima posibilidad de existir dos con el mismo DNI en la misma empresa.

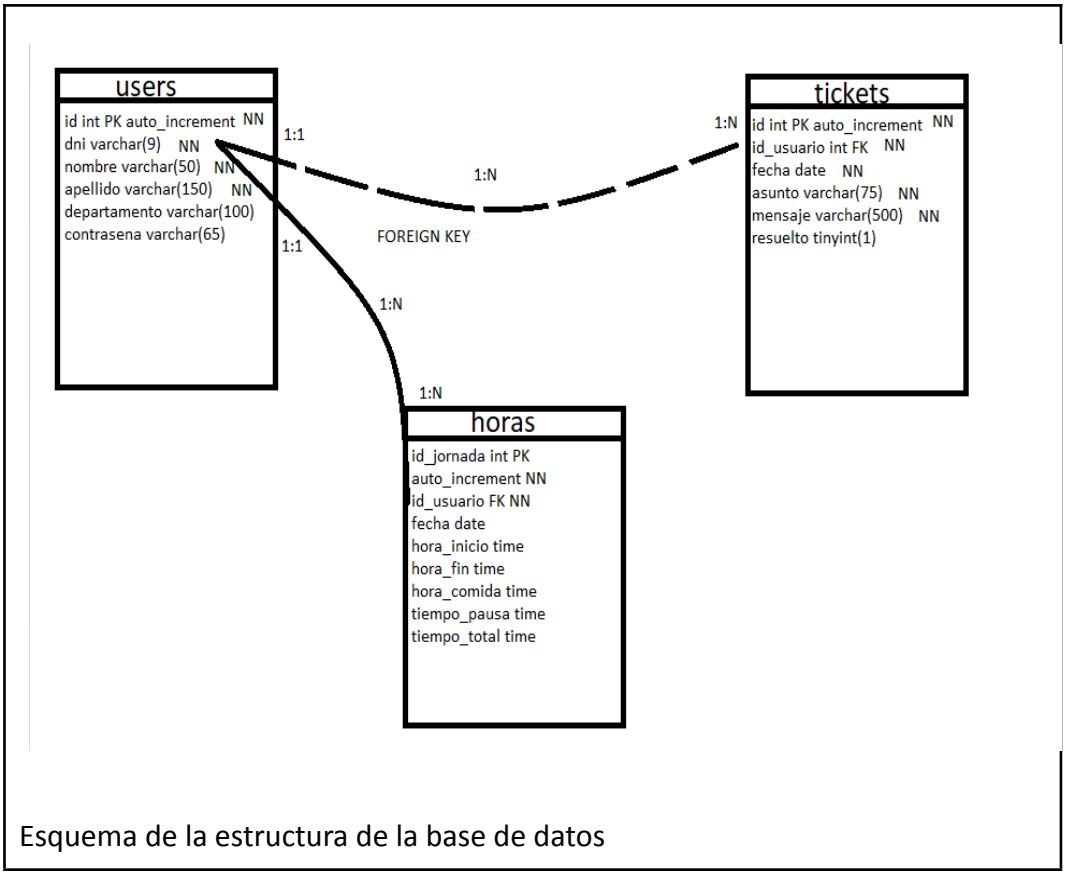
El DNI es un campo de texto que permite 9 caracteres, nombre y apellido son dos campos de texto que permiten 50 y 150 caracteres en ese orden.

Departamento es otro campo de tipo texto que permite 100 caracteres, y contrasena es un campo de texto que admite hasta 65 caracteres, debido a que cuando se cifran las contraseñas, arrojan una cierta cantidad de caracteres dependiendo del estándar que se utilice a la hora de cifrar.

En la tabla de tickets tenemos varios campos, id, id_usuario, fecha, asunto, mensaje, resuelto. Id es el identificador de cada incidencia o mensaje, id_usuario es el identificador de cada usuario y es el campo que vincula a users con esta tabla. Fecha es para especificar la fecha en la que ocurrió la incidencia. Asunto es para indicar el asunto de la consulta o incidencia, mensaje es una cadena larga de texto que permite hasta 500 caracteres y resuelto es un boolean, que sirve para que los administradores puedan filtrar entre los mensajes resueltos.

En la tabla horas tenemos id_jornada para identificar de manera única cada campo, id_usuario que es el campo que vincula esta tabla con la de users, fecha para indicar la fecha en la que se realiza el fichaje, hora_inicio para indicar la hora a la que empezó la jornada, la hora_fin para indicar a que hora terminó, la hora_comida, que es cuando se realiza la primera pausa, tiempo_pausa hace referencia al tiempo total que se ha estado descansando, independientemente de la comida, es decir, se hace un sumatorio entre los periodos de descanso y la comida y por último tiempo total, que recoge cuanto tiempo se ha trabajado en la jornada de hoy.

La estructura de la base de datos quedaría de la siguiente manera:



Esquema de la estructura de la base de datos

3.4.2. Porque está estructurada de esta manera

La respuesta es realmente sencilla, lo fundamental es todo lo que muestro. Ni más ni menos. Pero que me lleva a desarrollar este entramado es la cuestión, parece una base de datos sencilla, pero realmente tuvo su complejidad.

Lo primero de todo fue concretar que mi necesidad principal son los usuarios, y de ahí puedo estirar el hilo. Pero los usuarios porque están identificados por id y no por DNI. Como bien mencione antes la posibilidad es mucho menor de que toque la lotería y sin embargo puede pasar que coincida alguien con el mismo DNI en tu empresa, por eso por si las moscas mejor poner un ID autonumérico y nos quitamos de problemas. El resto de datos de esta misma tabla se sacan solos y los números de caracteres igual. Pero en cuanto a horas y tickets vienen otras cuestiones.

Los tickets deben identificarse igual que los usuarios, por id autonumérico, para que un mismo usuario pueda tener varios a pesar de que el id_usuario se repita, y no por fecha, porque en la misma fecha pueden haber varios. El asunto y el mensaje es evidente el porqué de su existencia,

pero no tanto el de resuelto. Resuelto es un campo boolean, es decir 0 y 1, dependiendo de su valor, podemos filtrar en las consultas por ejemplo por fecha y por resuelto 0 para poder ver todas las incidencias o preguntas de ese día que estén sin resolver. Es una solución sencilla pero muy útil.

En horas, ocurre lo mismo se identifica igual por id autonumérico, porque en un mismo día dni, fecha o ambos se pueden repetir. Tras concretar todo esto y poner las horas que necesitamos registrar tenemos el almacenamiento preparado. Pero aún así en el servidor quedan por concretar algunas cosas.

3.5. El servidor

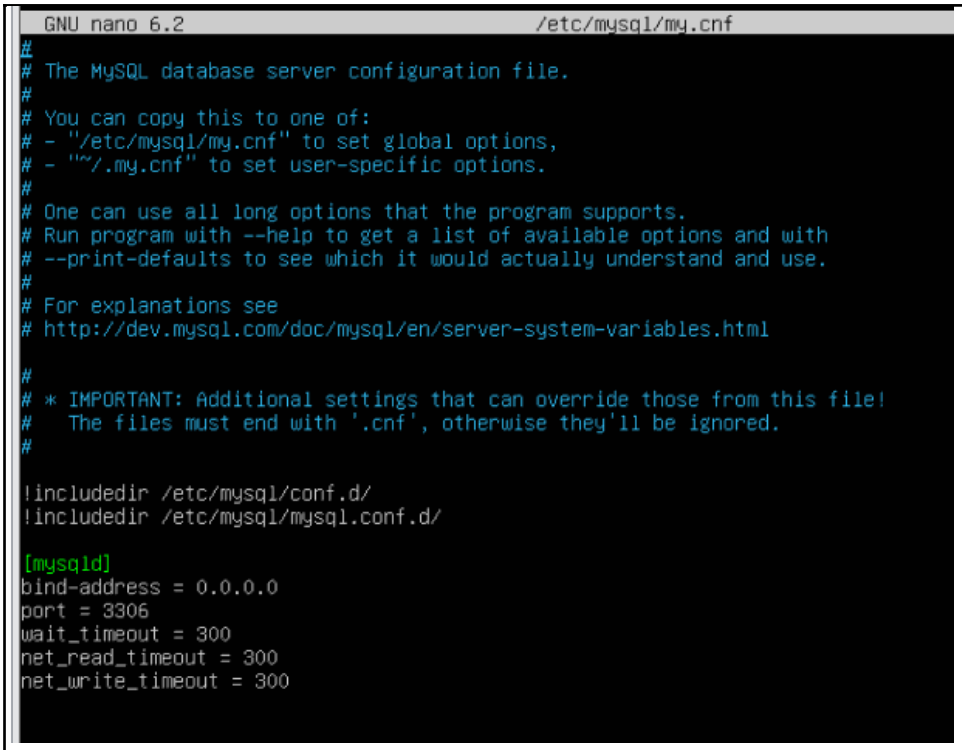
Ya se ha mencionado varias veces al servidor, pero todavía no se ha explicado cómo funciona. En primer lugar este servidor tiene un servidor DNS instalado, identificado con la marca que tienen mi departamento de proveedores (adrs).
La configuración del DNS es sencilla y se realizó en otros años. Es por esto que la configuración la adjuntaré en la bibliografía(incluiré la guía que seguí)
Una vez configurado el servidor, instalé mysql.
La primera vez que inicie sesión puedo hacerlo con mysql -u root -p y tras esto asignarle una contraseña con :

mysql> ALTER USER 'root'@'localhost' IDENTIFIED BY '*new_password*';

Ahora es super importante abrir los puertos, 3306 y habilitar el firewall con los comandos:

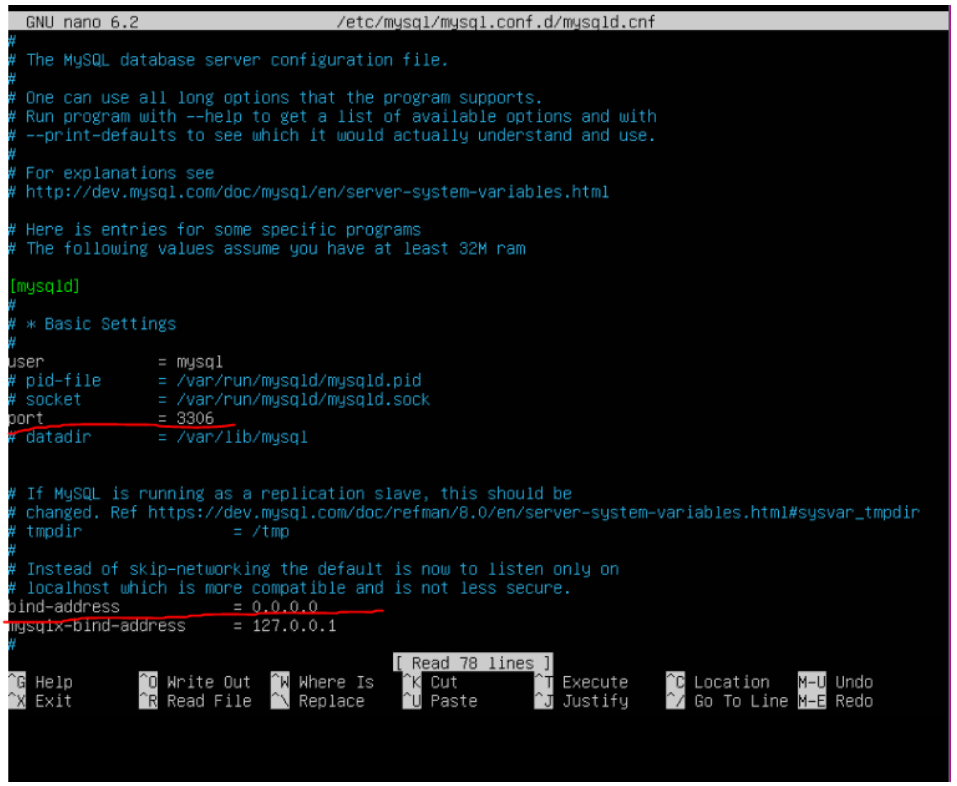
- sudo ufw allow 3306
- sudo ufw allow 3306/tcp
- sudo ufw enable

Ahora hay que configurar el servidor para que escuche cualquier ip para ello en el fichero /etc/mysql/my.cnf añadiré lo siguiente



Archivo 1 de configuración de mysql

Son el puerto que va a escuchar, la dirección 0.0.0.0 para que escuche cualquier ip y el tiempo de espera del servidor. También en el fichero /etc/mysql/mysqld.conf.d/mysqld.cnf añadiré esto:

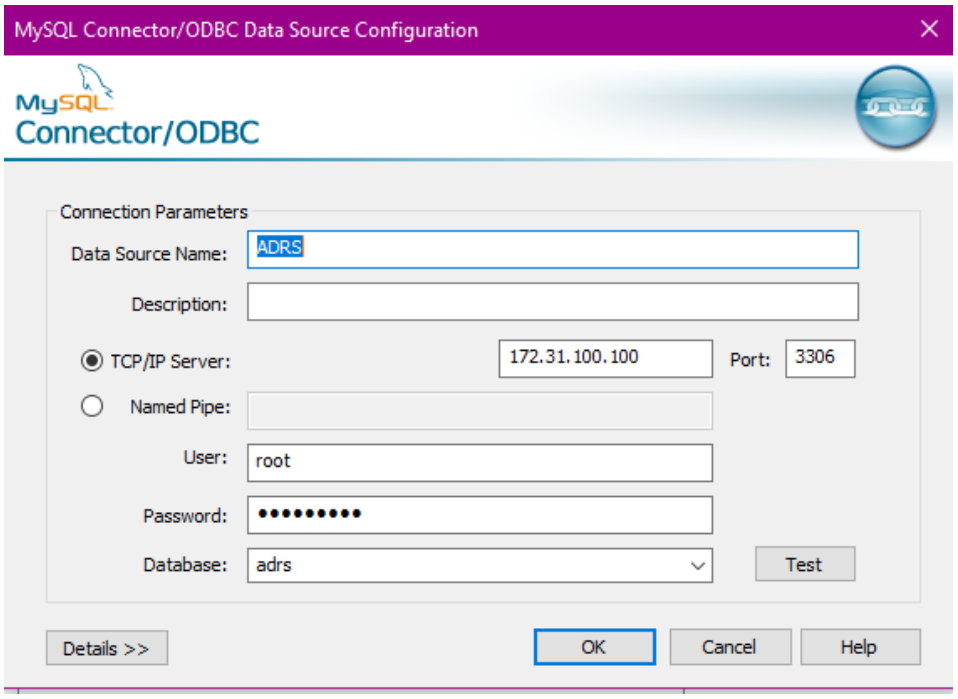


Archivo 2 de configuración de mysql

Después un reinicio al servicio mysql, loguear en mysql y cambiar el usuario que tengas para la conexión, en mi caso uso root, con los comandos

- USE mysql;
- UPDATE user SET Host='%' WHERE user='root';

Tras esto, en la máquina real es necesario instalar el controlador ODBC y configurarlo para que apunte al servidor y a la base de datos, debería quedar algo como esto:



Configuración de ODBC

3.6. El esqueleto de la aplicación

Una vez terminados los bocetos y la base de datos, toca comenzar con los primeros pasos del desarrollo.

Para la aplicación voy a utilizar la herramienta QT Designer(y las librerías de Qt y Pyside 6), ya que me permite agilizar mucho los procesos de desarrollo. Además esta herramienta trabaja con python y de hecho el comando:

- python -m PyQt6.uic.pyuic .\archivo.ui -o archivo.py -x

Me permite convertirlo a un archivo python, pero deja todo muy desordenado. El tema es que QT Designer es bastante intuitivo, además python me gusta mucho como se utiliza para el tema visual, y es un lenguaje de programación bastante completo y que trabaja bien para lo que quiero realizar, eso y que además cuando quieres personalizar el como se ve por ejemplo un botón solo tienes que escribir CSS. Otra cosa buena es que con Visual Studio Code este lenguaje también se trabaja bastante bien, y tiene una gran cantidad de plugins muy útiles.

Cuando iba terminando y convirtiendo los archivos al formato de python, el código lo ordenaba, para obtener una estructura mucho más limpia y clara, y trataba de mejorarlo. Porque la verdad es que la conversión es horrenda si quieres mantener un código limpio y ordenado.

La estructura de la aplicación está dividida en 3 archivos, la interfaz, con toda la interfaz gráfica y los métodos que operan y calculan datos que funcionan directamente en la interfaz(tratando de realizar un MVC). El conector, que se ocupa de recoger los datos que le mandemos, estructurar la consulta, y devolver un resultado, y el último es conexión, aquí se ejecutan las consultas y se realiza la conexión.

3.7. Implementación de la base de datos

Cuando ya tenemos el ODBC instalado y configurado, y no tenemos problemas de puertos. Hay que crear la conexión. En el archivo de conexión tengo esto

```python
import pyodbc
import mysql.connector as conector
from mysql.connector import *

#Datos para la conexión
server = '172.31.100.100'
database = 'adrs'
username = 'root'
password = 'Password@'
driver = '{MySQL ODBC 8.0 Unicode Driver}'


#Método para establecer la conexión
def conectarConBBDD():
    try:
        conexion = pyodbc.connect(f"DRIVER={driver};SERVER={server};DATABASE={database};UID={username};PWD={password}")
        return conexion
    except Error as e:
        print("No se ha podido realizar la conexión")
        return None
```

Conexión a la base de datos en archivo Conexión

Cada vez que quiera realizar una consulta no tengo más que mandar a los métodos creados más abajo, la consulta y la conexión.

```python
#Método para consultar 1 dato
def consultaDNI(conexion, ordenSQL):
    try:
        cursor = conexion.cursor()
        cursor.execute(ordenSQL)
        resultado = cursor.fetchone()
        return resultado
    except Error as e:
        print("No se pudo realizar la consulta")

#Método para consultar varios datos
def consultaTicket(conexion, ordenSQL):
    try:
        cursor = conexion.cursor()
        cursor.execute(ordenSQL)
        conexion.commit()  # Confirmar los cambios en la base de datos
        return True
    except Error as e:
        conexion.rollback()  # Revertir los cambios en caso de error
        print("No se pudo realizar la consulta")
```

Los métodos que ejecutan las consultas en archivo Conexión

Un ejemplo de cómo funciona una consulta es por ejemplo el inicio de sesión, que cuando le das a iniciar, realiza unas comprobaciones previas en el texto y si todo está bien manda el dni y la contraseña en forma de cifrado y compara las dos cadenas de texto de 64 caracteres.

```python
#Metodo para inicar sesion
def iniciarSesion(self):
    dni = self.user.text()
    password = self.password.text()

    if self.password.text() == "" and self.user.text() == "":
        self.passNone.setText("Debes ingresar la contraseña")
        self.DNI_NONE.setText("Debes ingresar un DNI")
    elif self.user.text() == "":
        self.DNI_NONE.setText("Debes ingresar un DNI")
        self.passNone.setText("")
    elif self.password.text() == "":
        self.passNone.setText("Debes ingresar la contraseña")
        self.DNI_NONE.setText("")
    else:
        resultado = Conector.iniciarSesion(dni,password)
        if resultado == None or resultado == False:
            self.DNI_NONE.setText("No existe ningun usuario relacionado o está mal ingresada la contraseña")
            self.passNone.setText("")
        else:
            self.DNI_NONE.setText("")
            self.passNone.setText("")
            self.w = VentanaMain(dni)
            self.w.show()
            self.hide()
```

Método donde se recogen los datos y se hacen las comprobaciones previas

```python
#Metodo para inicar sesion
def iniciarSesion(self):
    dni = self.user.text()
    password = self.password.text()

    if self.password.text() == "" and self.user.text() == "":
        self.passNone.setText("Debes ingresar la contraseña")
        self.DNI_NONE.setText("Debes ingresar un DNI")
    elif self.user.text() == "":
```

```python
def iniciarSesion(dni, password):
    conexion = Conexion.conectarConBBDD()
    if conexion == None:
        print("No se ha conectado")
        return False
    else:
        try:
            resultado = Conexion.consultaDNI(conexion, "SELECT dni,contrasena FROM users WHERE dni="+"'" + dni + "'")
            if resultado:
                resSeparado = resultado.__str__().split(",")
                cifrado = hashlib.sha256()
                cifrado.update(password.encode('utf-8'))
                passwordCifrado = cifrado.hexdigest()
                passwordRecogido = resSeparado[1].lstrip().translate(str.maketrans("", "", "',()"))
                if passwordRecogido == passwordCifrado:
                    return True
                else:
                    return False
            else:
                print("No se encontro ningun resultado")
        except Conexion.Error as e:
            print("")
```

Método dentro del conector que recoge los datos y prepara la consulta

```python
#Método para consultar 1 dato
def consultaDNI(conexion, ordenSQL):
    try:
        cursor = conexion.cursor()
        cursor.execute(ordenSQL)
        resultado = cursor.fetchone()
        return resultado
    except Error as e:
        print("No se pudo realizar la consulta")
```

Método que ejecuta la consulta

3.9. Fase de conclusiones y posible escalado

Tras la implementación del apartado gráfico a la aplicación, construir la funcionalidad de este y vincularlo con el servidor(además de configurar este mismo) vienen los planteamientos de después de terminar la aplicación. ¿Estoy realmente cumpliendo con lo que necesitan los usuarios?,¿Soy solo una solución más?

Responder a estas preguntas no es cosa sencilla, pero puede serlo, las consultas a otros usuarios, o realizar pruebas en empresas para que los usuarios puedan reportar un feedback es muy importante y puede allanar bastante el camino. Después de observar las críticas y recomendaciones de los usuarios, llegue a la conclusión de que mi solución es lo bastante

amigable e intuitiva, la comunicación es importante resaltar para el usuario que se usa para bugs/problemas/o comunicaciones importantes para evitar que se use por cualquier motivo como si fuese un para mensajería. Esto puede llegar a saturar al administrador o administradores que estén trabajando en ese momento.

¿En qué me distingo de otras soluciones similares?
Esta es otra muy buena pregunta, y la verdad es que esta es fácil de responder, mi aplicación es una solución moderna, sencilla y útil, no pierde tiempo en interfaces complicadas, datos innecesarios o redundantes, vas a lo que vas. La solución presentada en este proyecto para el usuario, permite ahorrar tiempo y errores. Porque por extraño que parezca una tarea sencilla como esta puede ocasionar muchos errores, tanto en la administración, como en los propios usuarios. Un mal sistema o unos malos registros pueden directamente afectar a la legalidad de una empresa en cuanto a las jornadas de los trabajadores, sin registros, ¿cómo se sabe que has trabajado?, ¿cómo se sabe que has hecho más o menos horas?, ¿estás dado de alta en la empresa pero no trabajas?. Estas cuestiones pueden dar muchos dolores de cabeza y los de RRHH lo saben.

En cuanto al posible escalado, la aplicación hoy por hoy no es perfecta, el módulo de usuario está bastante bien y no carece de nada, pero como en todo siempre se puede mejorar o añadir cosas. Hablando con trabajadores de la empresa, me he dado cuenta de que hay mucho trámite a la hora de pedir un día libre, un apartado en la aplicación con calendario que nos permita seleccionar días y motivo, permitiría en un solo click solicitar un día/periodo de vacaciones o asuntos propios. Que se pueda añadir en el de usuario hoy por hoy es la última solución que se podría añadir ya que es bastante completa.

Otra cosa que se menciona alguna vez es el módulo de administrador. No es el proyecto principal y es una solución aparte, pero es importante hablar de ello. Este panel facilita el trabajo a las personas administradoras de la base de datos, permitiendo generar informes, y tener todo en un apartado gráfico y sencillo. Como al final todo se puede realizar conectandote al servidor, este módulo es un plus que a la empresa podría interesar contratar, ya que como se hará todo de manera gráfica ahorras mucho tiempo/recursos y puedes hacer que la persona encargada de revisar todo sea cualquier usuario promedio de RRHH.

Volviendo a la aplicación original, otro asunto interesante que se podría incluir, es la personalización, en un futuro se pretende incluir temas sencillos para poder hacer la aplicación mucho más visual y poder conectar un poco más a los trabajadores con ella. Ya que lo vas a usar todos los días que menos que poder ponerlo verde, ¿no?

3.10. Medidas de seguridad de la aplicación y de la empresa

Tras entender cómo funciona la aplicación, ahora debemos hablar sobre la seguridad, la aplicación al fin y al cabo se instala en nuestros equipos, pero lo verdaderamente importante es proteger los datos, proteger los accesos, y proteger el servidor.

El servidor depende mucho de la extensión a la que pueda alcanzar el proyecto, en cualquier PYME, el servidor suele estar en una oficina, dentro de una cabina a la que solo tiene acceso por llave el de mantenimiento o el encargado de reiniciarlo si es que hubiera problemas. Pero en una empresa grande podemos hablar de otras problemáticas.

3.10.1 Seguridad física y lógica de los servidores

Tanto en mi empresa como en la empresa donde se aplicará esta solución ERP, no son empresas digamos pequeñas, o al menos su sistema no lo es.

- Es importante llevar un registro de control de acceso a todo el personal que entre en las instalaciones donde se encuentran los servidores, y que sólo se permita su entrada a personal autorizado.
- Es importante que la zona donde se encuentran los servidores, sea una zona bien organizada en cuanto al cableado, bien ventilada y con los servidores bien colocados para poder evitar accidentes, sobrecalentamientos y sobre todo aprovechar al máximo la temperatura que se generará y que todo funcione en su condición más óptima.
- Otros detalles también importantes es incluir un sistema de alarmas y de cámaras, para evitar todos los accesos no autorizados o posibles robos.
- Una persona de seguridad que trabaje vigilando la zona tampoco está de más

Todo esto es hablando de la seguridad física, pero la seguridad lógica es también muy importante, y es aún más importante tener todo muy controlado.

- Para evitar que los datos se puedan robar disco es importante que los discos estén cifrados

- Un buen antivirus que realice chequeos periódicos y detecte amenazas alivia muchos problemas
- Un adblock en los equipos de los trabajadores evita que los pop ups, o ventanas emergentes, junto con otros anuncios que puedan contener JS y a la hora de clickar o simplemente aparecer ejecutarlo y meter un posible virus. Es una solución muy sencilla pero muy eficaz.
- Realizar copias de seguridad periódicas es también importante.
- Los accesos están restringidos a un máximo de intentos y los pings están bloqueados, para evitar que los servidores puedan tener accesos a la fuerza o tener caídas inesperadas
- Es importante incluir un registro de auditoría, para saber quién está realizando cambios y evitar problemas graves.

Hablando de otros temas legales, protección de datos, consentimiento y términos y condiciones.Se pretende añadir en la ventana de bienvenida un pequeño apartado desplegable con todos los datos que incluyan en estos, aún quedan por determinar los términos de la empresa en las que se va a desplegar esta solución, pero es importante que los usuarios puedan consultarlos en todo momento.
En estos términos se debe incluir un apartado que explique cómo se realizará el tratamiento de sus datos, quienes están involucrados.
Si se tratan datos de carácter especial es importante indicarlo, de hecho si realmente este tipo de datos se incluyen se debe realizar un registro obligatorio de actividades de tratamiento. Para saber que tipos de datos son estos hay que mirar el apartado 9 RGPD.
También deberemos incluir este registro si se tratan estos datos:
- Los datos que se tratan pueden entrañar un riesgo para los derechos y libertades de los interesados.
- Los datos tratados son relativos a condenas e infracciones penales.
- Se tratan datos de manera sistemática a gran escala.

Deberemos también incluir otro apartado relacionado a la propia política de privacidad.
Se deberá informar de manera clara y concisa de todo y a todos los usuarios.

Remontándonos a la aplicación, esta tiene un tratamiento cuidadoso con las contraseñas, en ningún momento se pueden revelar, a los usuarios les aparecerá como asteriscos, para evitar que sí están realizando un seguimiento de su monitor no se puedan filtrar, y a la hora de realizar las comprobaciones de las contraseñas, en ningún momento se utilizan como texto, se comparan dos cadenas de string cifradas con SHA para saber si son iguales, pero nunca se devuelve la contraseña real.

4. Uso de la aplicación

Para el uso de la aplicación incluiré junto a este proyecto el archivo “README.MD” que explique lo mismo que haré aquí.

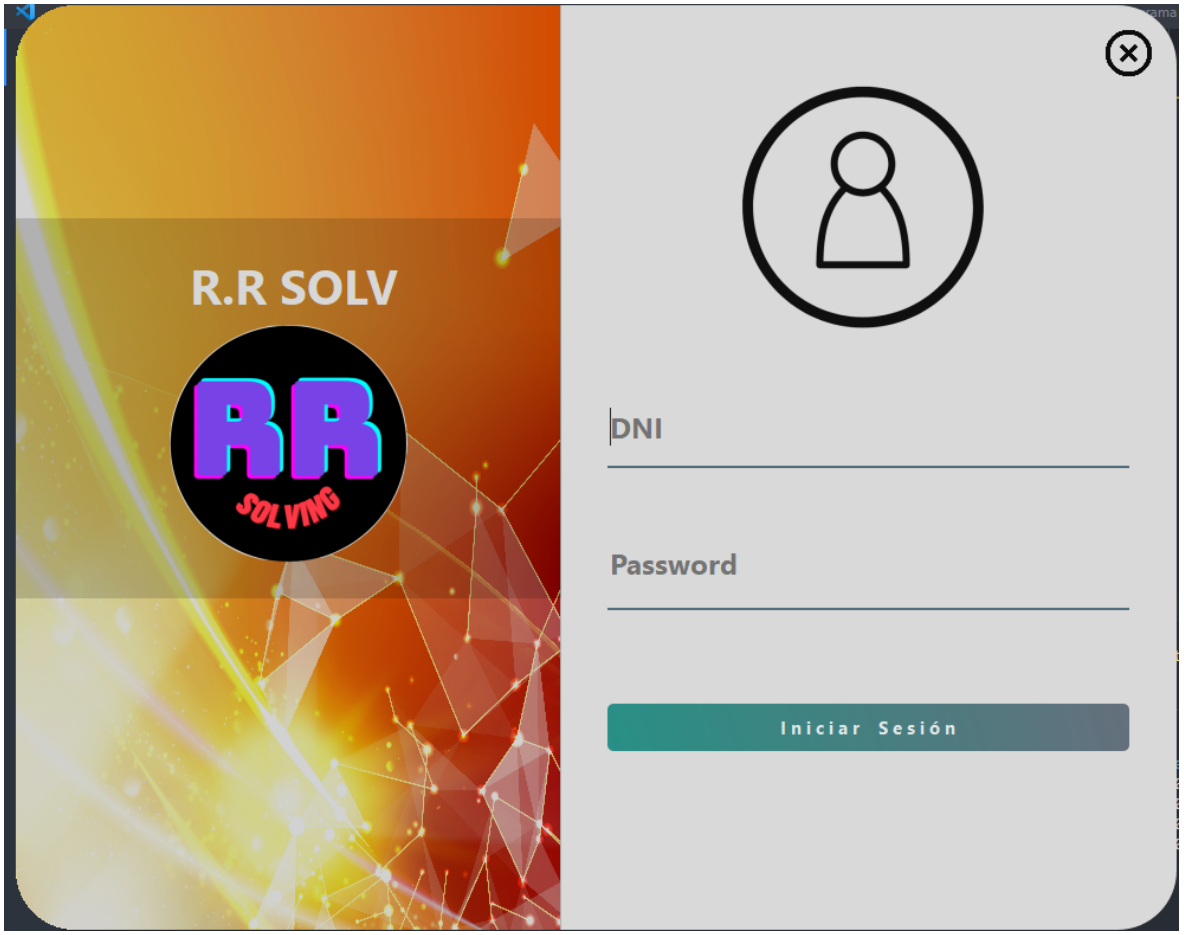
4.1. Instalación

Para la instalación de este proyecto se requerirá de las librerías de python correspondientes, estarán ubicadas dentro de un fichero de texto llamado requirements.txt
La versión de python que se utilizó fue la 3.11.3

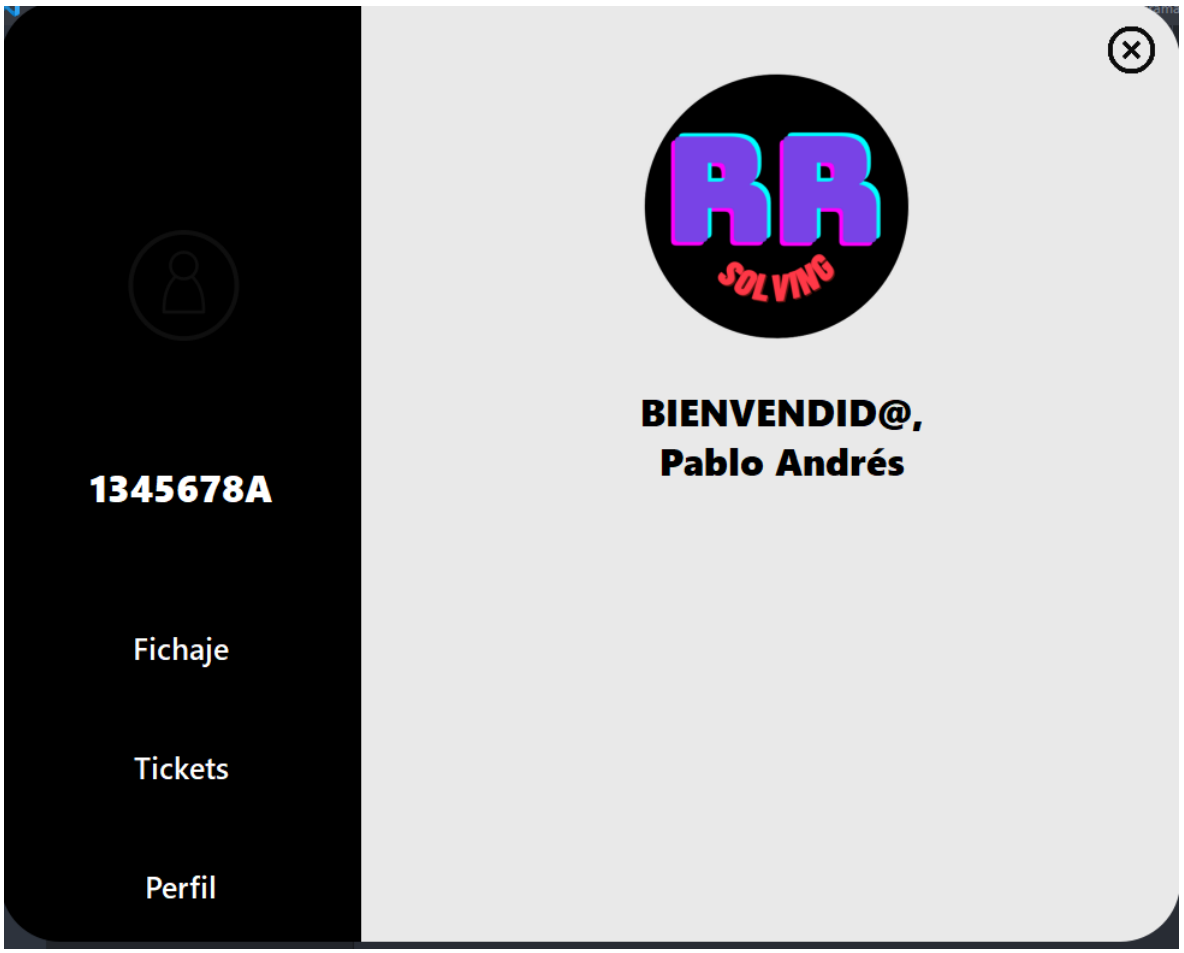
Se necesitará el controlador ODBC en la versión 8.0.33 para Windows en versión x86,x64

4.2. Uso de la aplicación

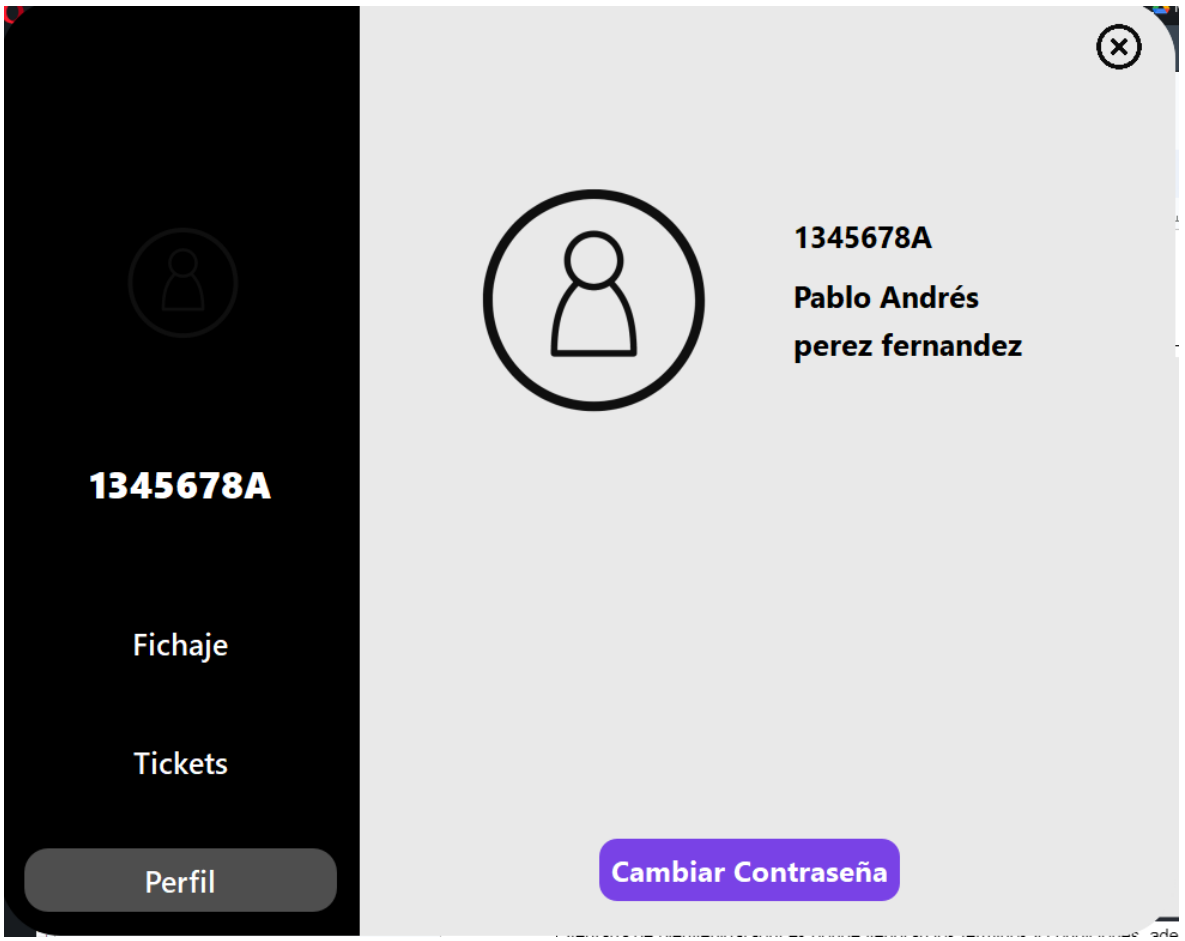
Para el uso de la aplicación un usuario tendrá que utilizar la ventana de login e introducir su DNI y contraseña. Los usuarios deben estar dados de alta por un administrador y este a su vez debe concederles una contraseña, la cuál deberán cambiar en el apartado perfil



Ventana de inicio de sesión



Ventana de bienvenida(aquí es donde vendrán los términos y condiciones, además de la declaración de privacidad una vez que estén redactados por la empresa)



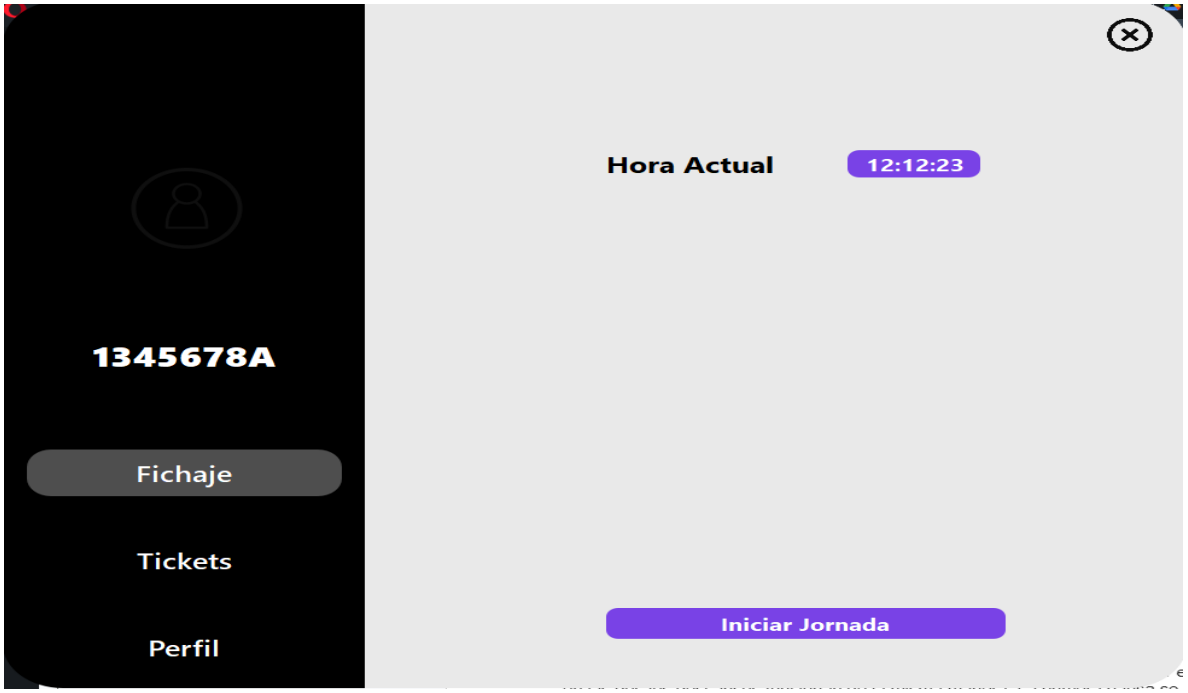
Ventana de Perfil

En esta ventana el usuario le deberá dar a cambiar Contraseña, introducir la que hay actualmente, darle al botón de verificar, y tras esto si es correcta aparecerán dos campos adicionales para introducir y confirmar la contraseña. Cabe destacar que para que sea lo más segura posible, admite solo contraseñas de entre 9-12 caracteres y con una mayúscula, minúscula, un número y un carácter especial.

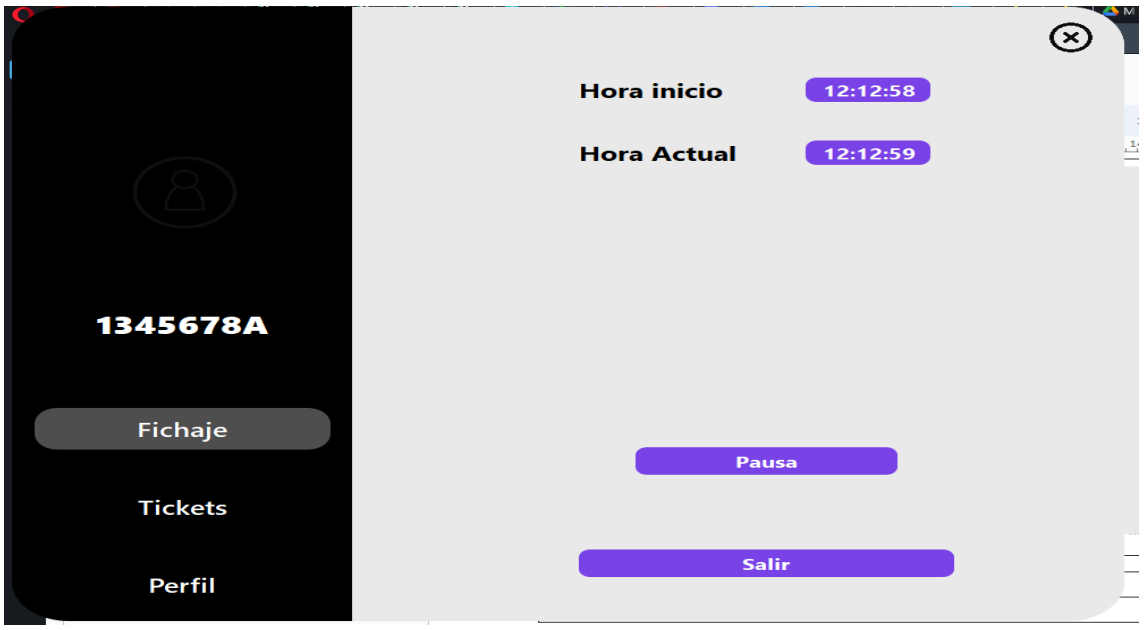
Ventana de perfil para confirmar contraseña actual

Ventana de Perfil para cambiar contraseña después de confirmar

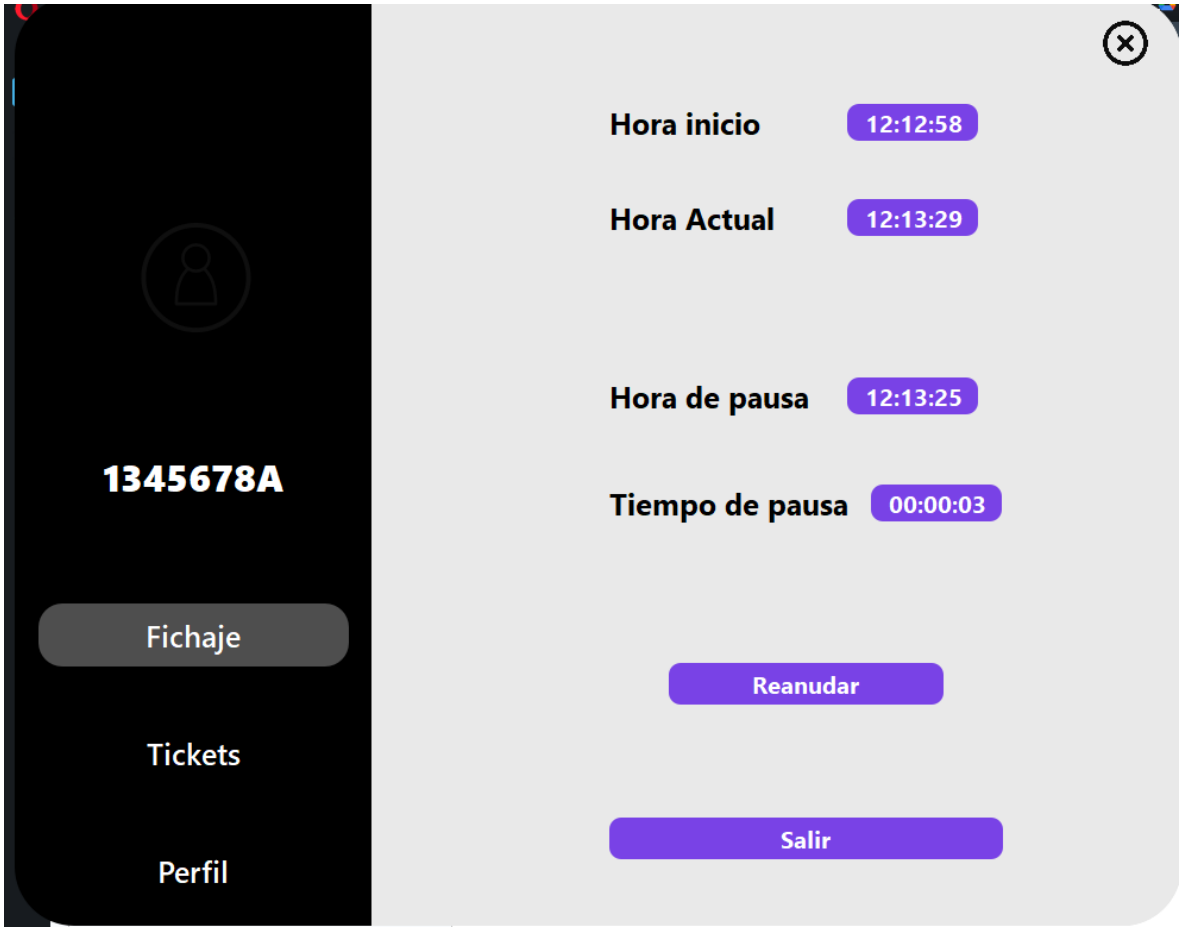
Una vez que haya cambiado la contraseña si va a empezar a trabajar le daría a fichaje. Dentro de fichaje hay un reloj que indica la hora, cuando le das a iniciar jornada te marca la hora a la que has empezado. En mi empresa y en la que se va a implementar este proyecto, en las dos los descansos funcionan de la misma manera. La primera pausa se registra a la hora de la comida y la guarda, y a partir de ahí puedes seguir luego realizando otras pausas de entre 5-10 min. Una vez finalizas tu jornada le darías a salir y habrías terminado.



Ventana de fichaje - Iniciar jornada



Ventana de fichaje - Fichaje iniciado



Ventana de pausa - Descanso iniciado

Si ocurre cualquier incidencia o comunicación urgente, iríamos a tickets, el usuario debe indicar un asunto claro y explicar en la longitud del mensaje lo más detalladamente la incidencia

Ventana de tickets

5. Bibliografía

Para el código y buscar problemas similares:

- https://stackoverflow.com
- https://ciphertrick.com
- https://www.pythonguis.com/search/?q=PySide6
- https://doc.qt.io/qtforpython-6/index.html

Para MySQL y sus consultas:

- https://dev.mysql.com/doc/mysql-getting-started/en/
- https://www.w3schools.com/mysql/mysql_alter.asp

Para diseño de logos y el fondo:

- https://www.freepik.es
- https://www.canva.com

Eliminar fondos y retocar la imagen:

- https://www.remove.bg/es/upload

Consultas de CSS y colores rgb/hexadecimal:

- https://htmlcolorcodes.com/es/
- https://www.w3schools.com/css/

- https://developer.mozilla.org/en-US/docs/Learn/CSS

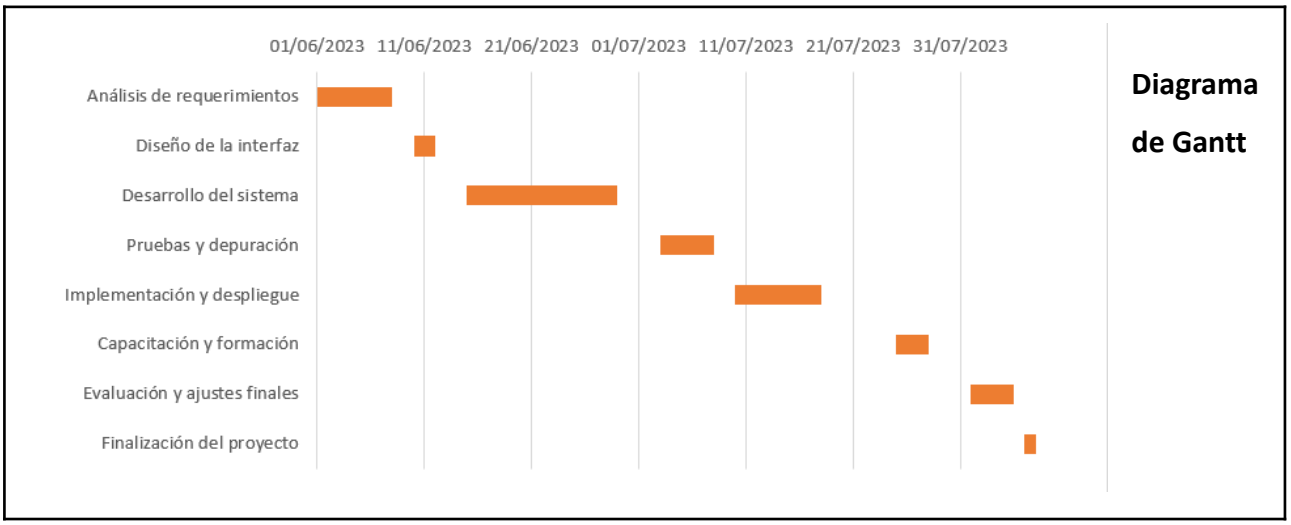
Para el DNS del servidor y la configuración de MySQL:

- https://ubuntu.com/server/docs/service-domain-name-service-dns
- https://www.makeuseof.com/install-configure-mysql-ubuntu/
- https://www.digitalocean.com/community/tutorials/how-to-allow-remote-access-to-mysql

Mi repositorio de github:

- https://github.com/copper-san/TicketManager

Anexo I - Diagrama de Gantt



En este diagrama se pueden apreciar los distintos periodos que habrán en este proyecto. Primero habrá un periodo de análisis donde se tratará de encontrar los requisitos mínimos que necesitamos para un proyecto como este para tratar de tener un mejor aprovechamiento de los equipos en los que vaya a utilizarse.

Tras esto comenzará la etapa de diseño, es una etapa corta puesto que pondremos a varios diseñadores a realizar cada ventana y a realizar los primeros bocetos, estos bocetos y los propios diseñadores, acompañarán al proceso para realizar las mejoras convenientes.

La etapa de desarrollo se dividirá entre varios trabajadores y tras esta vendrá la parte de pruebas continuada por despliegue. El apartado de pruebas consiste en probar las funciones del programa tratando de encontrar errores o de realizar pruebas unitarias a las distintas funciones que componen el programa.

El despliegue consistirá en realizar las instalaciones, tendremos a varios técnicos trabajando en esto, ya que hay bastantes equipos y un servidor por configurar.

En la capacitación, se instruirá a los empleados de una explicación breve de cómo funciona la aplicación y se les proveerá de un pequeño folleto con instrucciones sencillas para que puedan tener en todo momento. El proceso durará un tiempo más porque la persona administradora debe conocer cómo funciona la base de datos y el manejo de la información implícita en estos.
Tras esto vendrán los últimos retoques si hace falta y finalmente la aplicación ya estaría en pleno funcionamiento.

Anexo II - Contenido de README.MD

Ticket Manager

· Ticket Manager es una aplicación para agilizar los procesos de fichaje y mejorar los procesos de la empresa

Tutorial

La aplicación es sencilla de utilizar, debes hacer login poniendo tu usuario/contraseña. El usuario que hay creado de pruebas es:

DNI: 1345678A

Contraseña: 1Password@?

Tras hacer login puedes ir a fichajes, al accionar el botón de iniciar jornada marcará la hora de inicio y el reloj continuará, cuando quieras hacer la pausa de la comida tendrás que darle al botón "Pausa" y comenzará un contador y además se marcará la hora de comienzo. Tras esta pausa puedes seguir haciendo pausas. Para terminar la jornada deberás darle al botón de salir, asegurate que el botón está en Pausa y que no sigues en el descanso. Si quieres subir un reporte deberás ir a tickets e introducir los datos que se pidan. Si quieres cambiar la contraseña deberás darle al botón de cambiar contraseña en el perfil, poner la actual y darle al botón confirmar. Si la contraseña es correcta te aparecerán dos nuevos campos para introducir la nueva contraseña y confirmar. Tras esto dale al botón de actualizar y se cambiará sola.

Enlaces necesarios para el funcionamiento de la aplicación

Servidor:

https://drive.google.com/file/d/1u8gbIqPmF6Ic8OTHAhQv9bGpZYhIvWq-/view?usp=sharing

VMware:

https://drive.google.com/file/d/1FW5gfHYU8fNSB0mEM6GbcFVmwrc0Sae7/view?usp=drive_link

ODBC:

https://drive.google.com/file/d/1twzeoiLueuJkmKKhyMUQ5CWq1JGZU6-I/view?usp=sharing

Python:

https://drive.google.com/file/d/1P5Rej05WpZEUXxluBW2E_b_W3gZbIKwt/view?usp=sharing

Ordenador utilizado para las pruebas:

Procesador: Intel(R) Core(TM) i7-10700 KF CPU @ 3.80GHz 3.79 GHz

RAM: 32,0 GB

Monitor: 1920x1080 24"

Sistema Operativo: Windows 10 Pro

Ordenador utilizado para el servidor

RAM: 4GB

Disco: 40 GB HDD

Sistema Operativo: Ubuntu 18.04 command line version

Versiones utilizadas en el programa

Python 3.11.3

ODBC 8.0.33

PySide 6.5.0

PyQT 6 6.4.2

pyodbc 4.0.39

NOTA: TODOS LAS LIBRERÍAS SE ENCUENTRAN EN REQUIREMENTS.TXT

Anexo III - Instalación y ejecución

Para realizar la instalación del programa comencemos primero con la instalación de VMware, para ello tendremos que descargarlo.

Si alguien no dispone de las claves para utilizarlo en esta página de GitHub podemos encontrar unas cuantas: https://gist.github.com/gopalindians/94c2c8617028cfe7a5788f760e036fd2

Para instalar VMware y el controlador ODBC, es tan sencillo como dejar todas las opciones por defecto.

Para la instalación de Python, hay que ejecutar el instalador y abajo hay una casilla para añadir python al Path de Windows, es importante antes de instalar nada seleccionarlo.

Una vez instalado lo esencial, vienen las librerías de python, lo primero que debemos hacer es ubicar nuestro archivo requirements.txt, dentro del explorador de archivos, arriba en archivo, seleccionamos PowerShell y ejecutamos el comando de:

- pip install -r ./requirements.txt

Una vez instalado todo, procederemos a importar la máquina virtual.

Dentro de VMware Archivo -> Abrir -> Seleccionar el archivo server.ova y esperar a que finalice.

Una vez importada hay que asegurarnos que la primera tarjeta de red esté en NAT y que la segunda en bridged y con la casilla de replicar tu conexión física.

Ejecutamos la máquina e iniciamos sesión con las credenciales de:

- Usuario: root
- Contraseña: Password@

Con el servidor corriendo y todo instalado, iremos al ODBC, y pondremos la configuración de la foto. Es importante saber que la ip que usaremos será la de la tarjeta ens33, para conocer la ip, debemos en el servidor escribir 'ifconfig'.

Escribimos las credenciales de inicio de sesión de mysql que son las mismas que están aquí mencionadas y le damos a Test. Si funciona todo bien pondrá "Conectado correctamente".

Ahora que también tenemos el ODBC configurado y la conexión funcionando, iniciaremos el programa, para ello descomprimimos el archivo 'ProyectoDefinitivo.zip' y sin modificar la ruta ejecutaremos ./Programa/Interfaz_Definitiva.py y ya se podrá proceder a utilizar el programa.