

I.E.S LAS SALINAS



**PROYECTO FINAL FIN DE GRADO
CURSO 22-23**

MadridExplorer

**CICLO FORMATIVO GRADO SUPERIOR
DESARROLLO DE APLICACIONES MULTIPLATAFORMA**

Autor: David García Valverde

Tutor: Óscar Lillo Díaz

ABSTRACT

Español:

El objetivo principal de mi proyecto de fin de grado es abordar una posible necesidad que enfrentan los turistas o las personas en general que llegan a Madrid sin un plan definido de actividades o sin conocimiento de dónde alojarse, cómo divertirse, dónde comer, entre otros aspectos importantes. Con el fin de suplir esta necesidad, he desarrollado un sistema integral que ofrece múltiples posibilidades y soluciones en relación con los temas mencionados anteriormente.

Mi proyecto se enfoca en proporcionar opciones de alojamiento adecuadas para diferentes preferencias y presupuestos. Para lograr esto, he recopilado información sobre una pequeña gama de hoteles, restaurantes, museos y parques de atracciones en Madrid.

Inglés:

The main objective of my undergraduate project is to address a potential need faced by tourists or individuals in general who arrive in Madrid without a defined plan of activities or without knowledge of where to stay, how to have fun, where to eat, among other important aspects. In order to fulfill this need, I have developed a comprehensive system that offers multiple possibilities and solutions regarding the aforementioned topics.

My project focuses on providing suitable accommodation options for different preferences and budgets. To achieve this, I have gathered information about a wide range of hotels, restaurants, museums, and amusement parks in Madrid.

Índice

1)Justificación.....	4
2)Introducción.....	6
3)MDC-Madrid Developer Company	7
4)Software.....	9
5)Sistema Operativo.....	11
6)Lenguaje de Programación	13
7)Objetivos	15
8)Diagrama de clases	16
9)Diagrama de casos de uso	16
10)Diagrama de Gantt.....	17
11)Desarrollo.....	17
12)Diseño	28
13)Funcionalidades	28
14)Conclusión.....	30
15)Bibliografía	30
16)Anexos.....	31
Anexo 1	31
Anexo 2	31
Anexo 3	32

1)Justificación

El proyecto ha sido creado para suplir la necesidad de crear una guía de planes o ideas que poder realizar en la ciudad de Madrid, por la petición de un supuesto ayuntamiento de dicha localidad, que se propone darle más visibilidad tanto a grandes negocios de la zona como a negocios menores, lo que buscamos es crear oportunidades para todos estos comercios o empresas, patrocinarlos desde nuestra aplicación y poder permitirle al cliente una estancia guiada en dicha ciudad. ¿Cuáles serían las ventajas del cliente? Esto es un tema muy claro, al reservar en estos sitios mediante nuestra aplicación, les permitimos acceder a un descuento ya premeditado entre nosotros, el ayuntamiento de Madrid y las empresas. Lo que permite a los clientes no gastarse una excesiva cantidad de dinero. ¿Qué ganaríamos nosotros en todo esto? Nosotros con este plan ganamos cuatro cosas:

1- Un generoso pago proveniente del ayuntamiento por realizar su pedido y fomentar los comercios y negocios de la zona.

2- En el acuerdo realizado con dichas empresas se acordaría que aparte del descuento que ofrecen a los clientes por el uso de nuestra aplicación, también nos llevaríamos un mínimo porcentaje del pago de cada uno, por usar nuestros servicios.

3- Como tercer punto a redactar como ventajas para nosotros sería que a la vez que nosotros les patrocinamos, las propias empresas deberían como mínimo tener un escaso número de flyers o propaganda digital para ayudarnos a nosotros, esto para seguir retroalimentando el bucle y juntos poder crecer más y más.

4- Aparte del propio apoyo de las empresas hacia nosotros, el propio ayuntamiento de Madrid también nos permite fomentar nuestra aplicación en su ciudad gracias a reducir el pago de la propia aplicación, todo para generar ingresos a largo plazo y no dirigido tanto a ganar una buena suma de dinero en poco tiempo, con esto buscamos crear una empresa sostenible con el tiempo, a su vez otras ciudades podrían buscar el mismo objetivo y contratarnos para lograrlo.

¿Qué ganan las empresas en todo esto? Al tener que ceder parte del precio de sus servicios podemos pensar que realmente no ganan nada a cambio, pero ¿Esto es realmente así? No, ellos al igual que nosotros ganarían mucha visibilidad por el tema que hemos comentado antes, el bucle de publicidad, lo que, aunque a corto plazo no les vaya a generar tanto dinero, a la larga les puede ser muy útil, esto sobre lo que estoy hablando son estrategias de marketing

reales usadas a nivel mundial tanto por pequeñas, medianas o grandes empresas. Además de que sólo disminuirían sus ingresos para con los clientes que vayan a sus negocios de nuestra parte, ellos siguen disponiendo de sus propios métodos de propaganda mediante sus sitios web individuales como otros medios que usen. En esos casos seguirían percibiendo la misma cantidad de dinero.

A todo esto, ¿Qué puede generar el ayuntamiento de Madrid? Aparte de que es su propia iniciativa, también generarían más atracción a este núcleo, más turismo debido a una mayor visibilidad de buenos negocios, lo que aparte de un mayor turismo, generaría una mayor atracción de población al centro, lo que implica más puestos de trabajo y a su vez más residentes.

Pasando a hablar ahora sobre la aplicación como tal, he decidido realizarla en Python debido a que en nuestro curso se ha visto con menor intensidad que java, y aunque es cierto que en un principio estaba pensado que la hiciera con java, al final me retracté y pasé por esta opción. Aunque conozco de editores de Python que permiten realizar aplicaciones de forma más gráfica y sencilla, he decidido usar visual studio code y a su vez PySide6, en vez de PySide2 y los editores “sencillos” antes mencionados debido a que es la forma en la que lo hemos aprendido, y no quería buscar demasiadas facilidades sólo para la creación de dicha aplicación.

2)Introducción

El objetivo principal de mi proyecto de fin de grado es promover el turismo en una ciudad, en este caso, he elegido la fascinante ciudad de Madrid. Mediante el desarrollo de nuestra aplicación, buscamos estimular el incremento de la demanda de alojamiento y el uso de los servicios disponibles en esta ubicación. Como mencioné en la justificación, para la implementación de la aplicación, hemos utilizado el lenguaje de programación Python y el entorno de desarrollo Visual Studio Code. Aunque esta elección no haya facilitado el proceso de desarrollo, sí me ha brindado la ventaja de trabajar en un entorno con el cual estoy familiarizado.

En relación al manejo de los datos de los clientes, hemos planteado la hipotética utilización de una base de datos que, en este proyecto, está representada por un simple archivo de notas. Este enfoque nos permite recopilar y almacenar la información necesaria para ofrecer a los usuarios una experiencia personalizada y adaptada a sus preferencias.

MadridExplorer ha sido diseñada como una aplicación de escritorio, aunque en una versión futura, con una optimización perfecta, aspiramos a que también pueda ser utilizada en dispositivos móviles. Esto nos permitiría brindar a los usuarios una herramienta práctica y versátil, accesible en cualquier momento y lugar, para que puedan explorar y disfrutar al máximo de los encantos de Madrid.

3)MDC-Madrid Developer Company

La MDC o también conocida como Madrid Developer Company o Empresa de Desarrolladores de Madrid, es una nueva empresa que se creó hace apenas unos pocos años, y no cuenta con un personal ni muy experimentado ni muy abultado, sin embargo, siempre ha tenido contacto con el ayuntamiento de Madrid, para pequeños pedidos y trabajos que les han comunicado, todo esto gracias a que el fundador de la empresa tiene contacto con el alcalde de la ciudad. A día de hoy la empresa ha crecido bastante gracias a la etapa de covid, en la cual se buscó bastante teletrabajo y en este caso esta empresa salió beneficiada por ello. Cuenta con un plantel de casi 15 desarrolladores que les cuesta unos 39000 euros mensuales. Unas oficinas interesantes en las en el barrio de Salamanca de Madrid de unos 200m cuadrados que les cuestan unos 4500 euros el alquiler y disponen de un hardware más que suficiente de gama media que les costó unos 20000 euros aproximados, en el apartado del software solo tienen que pagar por el SO (sistema operativo) que van a usar Windows 10, tiene un coste aproximado de unos 1500-2000 euros, no se tendría que pagar por mucho más software ya que el entorno utilizado para el desarrollo de aplicaciones es Visual Studio Code y es gratuito. Como seguridad física para las oficinas se eligió el barrio de Salamanca gracias a la alta presencia policial y además de estar bien situado de cara al centro. Como seguridad lógica se tendría que tener en cuenta cambiar el txt por una base de datos real o al menos encriptar los usuarios y contraseñas.

Si MDC usara algún programa erp creo que se decantaría por Odoo, ¿Por qué Odoo? Pues se decantaría por Odoo gracias a las ventajas que posee frente a sus competidores:

Odoo posee una integración completa, puede acceder a cualquier área de la empresa, esto facilita a las empresas para no tener que usar múltiples sistemas independientes y les ayuda con la automatización y sincronización de los procesos empresariales.

Puedes además personalizar todos los módulos que la web te proporciona, por lo que tu empresa se verá y gestionará a tu gusto. Odoo es una plataforma de código abierto por lo que la hace más personalizable aún para las empresas que lo necesiten, odoo cuenta también con una comunidad enorme que sigue en crecimiento. Tiene costos reducidos comparados con sus competidores de mercado ERP y además brinda a los usuarios una interfaz muy sencilla de usar y amigable a la vez que moderna.

LOPD

La Ley Orgánica de Protección de Datos de Carácter Personal, comúnmente abreviada como LOPD, es un conjunto de normas que regulan el tratamiento y uso que pueden hacer empresas y profesionales de los datos de carácter personal (como direcciones, datos bancarios, deudas, etc.) de los que dispongan en el ejercicio de su actividad, así como los derechos de los titulares sobre ellos

En nuestra aplicación por el momento no se recogen datos de carácter personal de los usuarios, ya que hablamos de un prototipo, sin embargo, en versiones futuras la reserva sí debería pagarse, por lo que en ese momento se pensaría en añadir métodos de pago que sí recogerían esos datos.

4)Software

Visual Studio Code

Visual Studio Code (VS Code) es un entorno de desarrollo integrado (IDE) altamente popular y ampliamente utilizado. Es desarrollado por Microsoft y se ha convertido en una opción preferida para muchos desarrolladores de software debido a su enfoque en la simplicidad, la eficiencia y la flexibilidad.

La interfaz de usuario de VS Code es limpia y minimalista, lo que facilita la concentración en el código. La ventana principal está dividida en varias secciones, incluyendo una barra de menú en la parte superior, una barra lateral izquierda que muestra la estructura del proyecto y diferentes paneles como el Explorador de archivos, el Control de código fuente y las extensiones instaladas. En el área central se encuentra el editor de código, que ofrece una experiencia de escritura de código altamente personalizable y potente.

Una de las características destacadas de VS Code es su amplia gama de extensiones. La comunidad de desarrolladores ha creado una gran cantidad de extensiones que permiten ampliar la funcionalidad del editor. Estas extensiones van desde mejoras en el resaltado de sintaxis y la autocompletación, hasta integración con sistemas de control de versiones, depuradores, linters, frameworks y muchos otros aspectos del desarrollo de software. La instalación y gestión de las extensiones es sencilla, lo que facilita adaptar VS Code a las necesidades de cada desarrollador.

El editor de código en sí mismo es altamente personalizable. Los usuarios pueden ajustar la apariencia, el tema, los colores, las fuentes y los atajos de teclado según sus preferencias. También se pueden crear configuraciones específicas para proyectos, lo que permite tener diferentes entornos de desarrollo para distintos tipos de trabajo.

VS Code ofrece funciones de edición de código avanzadas, como resaltado de sintaxis para una amplia gama de lenguajes de programación, autocompletado inteligente, navegación

rápida por el código y refactorización. También proporciona herramientas integradas de depuración que permiten inspeccionar y solucionar problemas en el código paso a paso.

Además, VS Code ofrece una integración sólida con sistemas de control de versiones, como Git. Permite realizar operaciones comunes de control de versiones directamente desde el editor, como confirmar cambios, fusionar ramas y comparar diferencias entre versiones.

Otra característica notable de VS Code es su capacidad de trabajar con una amplia variedad de tecnologías y marcos de desarrollo. Ya sea que estés trabajando en un proyecto de desarrollo web, móvil o de inteligencia artificial, es probable que encuentres extensiones y herramientas adecuadas para tus necesidades específicas.

En resumen, Visual Studio Code es un entorno de desarrollo integrado versátil y altamente configurable que ofrece una experiencia de desarrollo de software eficiente y agradable. Con su amplia gama de extensiones, capacidades de personalización y soporte para una amplia variedad de tecnologías, se ha convertido en una herramienta imprescindible para muchos desarrolladores en su flujo de trabajo diario.

En mi caso yo he usado dos para poder escribir el código con Python de forma más sencilla, Python y Qt for Python son las extensiones que he usado.

5) Sistema Operativo

Windows 10

Windows 10 es un sistema operativo desarrollado por Microsoft y es la versión más reciente de la familia de sistemas operativos Windows. Fue lanzado oficialmente el 29 de julio de 2015 y ha recibido varias actualizaciones importantes desde entonces. A continuación, se proporciona una descripción detallada de las características y aspectos clave de Windows 10:

- 1) Interfaz de usuario moderna: Windows 10 presenta una interfaz de usuario moderna con un enfoque en la usabilidad y la estética. Incluye un menú de inicio rediseñado que combina elementos del clásico menú de inicio de Windows 7 y las baldosas interactivas de Windows 8. También tiene una barra de tareas mejorada con soporte para vistas de tareas y una bandeja del sistema más funcional.
- 2) Continuum: Windows 10 introduce el concepto de Continuum, que permite a los dispositivos con pantalla táctil cambiar entre los modos de tableta y escritorio de manera fluida. Esto es especialmente útil en dispositivos híbridos como tabletas con teclado desmontable o dispositivos 2 en 1.
- 3) Asistente virtual Cortana: Cortana es el asistente virtual de Windows 10 que permite a los usuarios realizar búsquedas, establecer recordatorios, obtener información del tiempo, abrir aplicaciones y realizar otras tareas utilizando comandos de voz o texto. Cortana está integrada en el sistema operativo y se puede acceder desde la barra de tareas.
- 4) Microsoft Edge: Windows 10 introduce Microsoft Edge, un nuevo navegador web que reemplaza a Internet Explorer como navegador predeterminado. Microsoft Edge ofrece una experiencia de navegación más rápida y segura, así como características como la capacidad de tomar notas en páginas web y compartir contenido directamente desde el navegador.
- 5) Aplicaciones universales: Windows 10 presenta las aplicaciones universales, que son aplicaciones diseñadas para funcionar en todos los dispositivos con Windows 10, incluyendo PC, tabletas, teléfonos y consolas Xbox. Estas aplicaciones ofrecen una

experiencia consistente en diferentes dispositivos y pueden sincronizar datos entre ellos.

- 6) Seguridad mejorada: Windows 10 incluye varias mejoras en seguridad, como Windows Hello, que permite a los usuarios iniciar sesión en sus dispositivos utilizando reconocimiento facial, escaneo de iris o huellas dactilares. También cuenta con Windows Defender, un programa antivirus integrado, y ofrece actualizaciones de seguridad regulares para proteger contra amenazas cibernéticas.
- 7) Integración con servicios en la nube: Windows 10 se integra estrechamente con los servicios en la nube de Microsoft, como OneDrive. Los usuarios pueden sincronizar automáticamente archivos y configuraciones en diferentes dispositivos y acceder a ellos desde cualquier lugar.
- 8) Centro de actividades: Windows 10 presenta el Centro de actividades, que es un panel de notificaciones que muestra notificaciones de aplicaciones, alertas de sistema y acceso rápido a configuraciones comunes. Los usuarios pueden personalizar las notificaciones que desean recibir y acceder a ellas fácilmente desde la barra de tareas.

Estos son solo algunos de los aspectos más destacados de Windows 10. El sistema operativo también ofrece una amplia gama de características y funciones adicionales, como soporte para aplicaciones de la Tienda Windows, compatibilidad con DirectX 12 para juegos, escritorios virtuales, compatibilidad con voz y escritura a mano, y más. Windows 10 ha sido diseñado para ser compatible con una amplia variedad de dispositivos y ofrece una experiencia unificada en múltiples plataformas.

6) Lenguaje de Programación

Python

Python es el lenguaje usado para este proyecto, Python es un lenguaje de programación de alto nivel, interpretado y de propósito general. Fue creado por Guido van Rossum y lanzado por primera vez en 1991. A lo largo de los años, Python se ha vuelto extremadamente popular debido a su sintaxis sencilla y legible, su amplia biblioteca estándar y su enfoque en la facilidad de uso y la productividad del programador. A continuación, se proporciona una descripción detallada de las características y aspectos clave de Python:

1. **Sintaxis clara y legible:** Python se destaca por su sintaxis limpia y legible, que hace que el código sea fácil de entender y escribir. Utiliza la indentación en lugar de los corchetes o paréntesis para definir bloques de código, lo que fomenta una estructura clara y coherente.
2. **Tipado dinámico:** Python es un lenguaje de tipado dinámico, lo que significa que las variables no están vinculadas a un tipo específico durante la declaración. Las variables pueden cambiar de tipo durante la ejecución del programa, lo que brinda flexibilidad al desarrollador.
3. **Orientado a objetos:** Python admite la programación orientada a objetos (POO) y permite la definición de clases y objetos. Proporciona características como herencia, polimorfismo y encapsulación, lo que facilita la creación de código modular y reutilizable.
4. **Biblioteca estándar amplia:** Python cuenta con una biblioteca estándar rica y extensa que cubre una amplia gama de áreas, como manipulación de archivos, redes, procesamiento de texto, matemáticas, generación de gráficos y más. Esta biblioteca estándar facilita el desarrollo de aplicaciones sin tener que depender de bibliotecas externas.
5. **Amplia comunidad y bibliotecas adicionales:** Python cuenta con una gran comunidad de desarrolladores que contribuyen con bibliotecas y módulos adicionales. Estas bibliotecas externas, como NumPy, Pandas, Matplotlib, Django y Flask, brindan funcionalidades adicionales y permiten a los desarrolladores aprovechar soluciones existentes para diferentes dominios.

6. Fácil integración: Python se puede integrar fácilmente con otros lenguajes de programación, lo que lo convierte en una opción popular para proyectos que requieren interoperabilidad. Puede interactuar con bibliotecas escritas en C, C++, Java y otros lenguajes, lo que permite aprovechar el rendimiento y las funcionalidades existentes.
7. Portabilidad: Python es un lenguaje multiplataforma, lo que significa que los programas escritos en Python se pueden ejecutar en diferentes sistemas operativos, como Windows, macOS y Linux, sin cambios significativos. Esto facilita el desarrollo de aplicaciones que son independientes de la plataforma.
8. Enfoque en la legibilidad y la productividad: Python se diseñó para ser un lenguaje fácil de leer y escribir, lo que promueve la legibilidad del código y la productividad del programador. Se enfoca en reducir la cantidad de código requerido para lograr una tarea y en proporcionar estructuras de datos y funciones integradas que facilitan la implementación de algoritmos complejos.

Estos son solo algunos de los aspectos más destacados de Python. El lenguaje se utiliza en una amplia variedad de aplicaciones, desde desarrollo web y científico hasta automatización de tareas y aprendizaje automático. La combinación de su sintaxis clara, su biblioteca estándar extensa y su comunidad activa lo convierten en una opción popular para desarrolladores de todos los niveles de experiencia.

PySide

PySide6 es la versión que he decidido usar, siendo la última que ha salido, PySide es un conjunto de enlaces Python para el framework de interfaz gráfica de usuario (GUI) Qt. Proporciona a los desarrolladores una forma de crear aplicaciones de escritorio multiplataforma utilizando el poderoso conjunto de herramientas de Qt.

Para ver más sobre la librería PySide ver anexo 3.

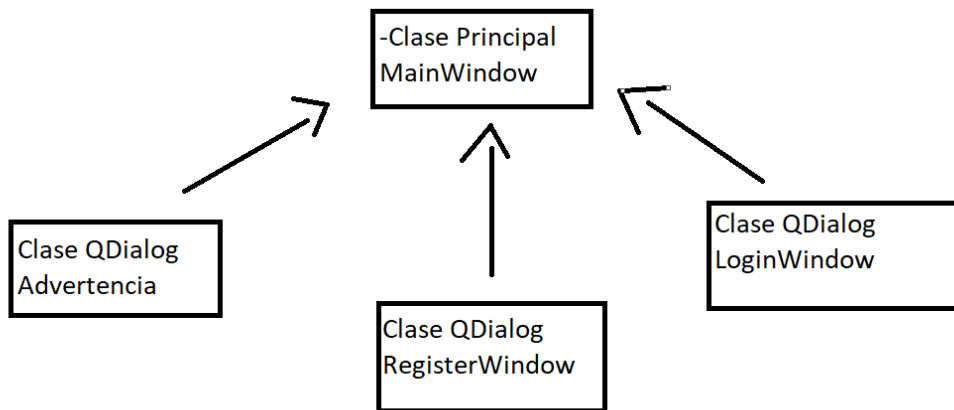
7)Objetivos

Los objetivos de mi trabajo de fin de grado son los siguientes:

1. Profundizar en el entorno utilizado para la creación de la aplicación, adquiriendo un conocimiento más detallado y completo de sus características y funcionalidades.
2. Estudiar los mecanismos de intercambio de datos mediante archivos externos a la propia aplicación, explorando las diferentes técnicas y prácticas utilizadas en este contexto.
3. Practicar el uso de webs internas en aplicaciones, familiarizándome con su implementación y comprendiendo cómo pueden mejorar la experiencia del usuario y la eficiencia del sistema.
4. Ampliar mis conocimientos de Python a nivel general, explorando diferentes aspectos del lenguaje y adquiriendo habilidades más avanzadas en su uso.
5. Explorar otros lenguajes de programación distintos a Java, con el objetivo de ampliar mi horizonte y comprender las fortalezas y debilidades de cada uno.
6. Desarrollar una sólida comprensión de los lenguajes de programación en general, profundizando en los conceptos fundamentales y adquiriendo habilidades transferibles que me permitan abordar nuevos proyectos y desafíos.

Estos objetivos me permitirán no solo realizar el desarrollo de la aplicación para mi trabajo de fin de grado, sino también expandir mis conocimientos y habilidades como desarrollador en general.

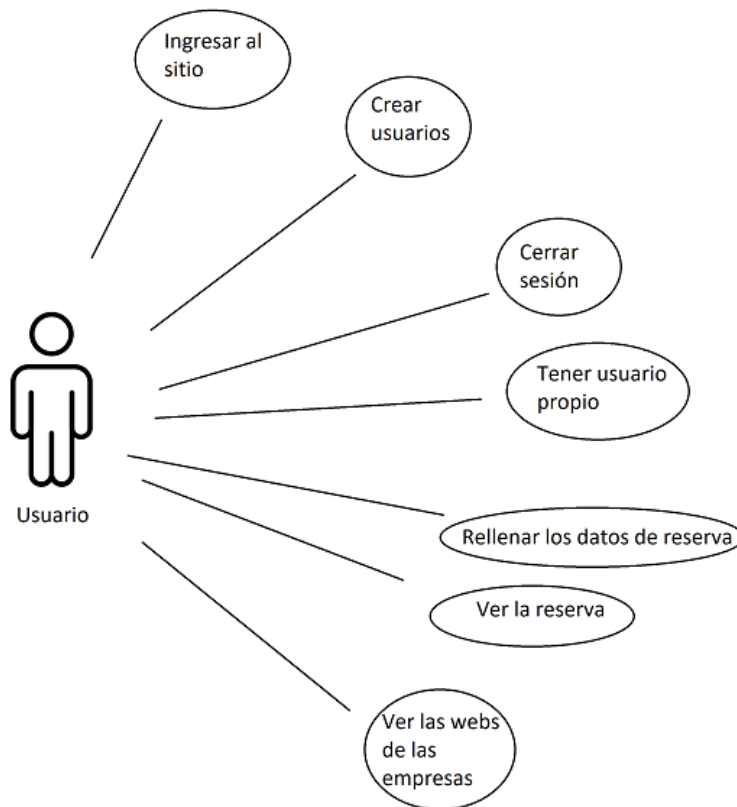
8)Diagrama de clases



(Ver más en Anexo 1)

9)Diagrama de casos de uso

En mi caso no hay perspectiva de administrador, al menos actualmente el uso de la aplicación es únicamente de parte de usuarios, por lo que solo hay una perspectiva de uso.



(Ver más en Anexo 2)

10)Diagrama de Gantt



El diagrama de Gantt ha sido creado con la web monday.com y estima los tiempos que me ha tomado pensar en la idea, empezar a desarrollarla y redactar esta memoria.

11)Desarrollo

La aplicación se ha desarrollado en Visual Studio Code utilizando Python como lenguaje de programación y un bloc de notas suplantando una hipotética base de datos

No se ha seguido una estructura de paquetes con un diseño MVC (Modelo-Vista-Controlador) debido a que con el entorno mencionado anteriormente solo necesito de un único archivo además de que no uso conexiones a bases de datos.

El desarrollo del proyecto se ha realizado utilizando el lenguaje de programación Python y el framework PySide6 para el desarrollo de la interfaz gráfica. El código se divide en varias clases que representan diferentes ventanas y funcionalidades.

Las siguientes importaciones proporcionan acceso a funcionalidades esenciales para el desarrollo de aplicaciones de escritorio con la biblioteca PySide6, como la manipulación del sistema operativo, la construcción de interfaces de usuario y la integración de contenido web, etc.

```
import os, sys, platform
from PySide6.QtCore import *
from PySide6.QtWidgets import *
from PySide6.QtGui import *
from PySide6.QtWebEngineWidgets import QWebEngineView
```

La clase **Advertencia** define una ventana de diálogo que muestra un mensaje de advertencia al usuario. Esta ventana contiene dos botones: "Cerrar" y "Abrir". La interacción del usuario con estos botones está conectada a los métodos **accept** y **reject**, respectivamente. Al pulsar en aceptar te permite ingresar en la aplicación con el usuario introducido.

Se ve de la siguiente forma en el código:

```
class Advertencia(QDialog):
    def __init__(self, parent = None):
        super().__init__(parent)

        self.setWindowTitle("Advertencia")
        botones = QDialogButtonBox.Close | QDialogButtonBox.Open

        self.caja_botones = QDialogButtonBox(botones)
        self.caja_botones.accepted.connect(self.accept)
        self.caja_botones.rejected.connect(self.reject)

        self.layout_advertencia = QVBoxLayout()
        self.layout_advertencia.addWidget(QLabel("¿Estás seguro de usar ese usuario? Si es así pulsa abrir..."))
        self.layout_advertencia.addWidget(self.caja_botones)
        self.setLayout(self.layout_advertencia)
```

La clase **LoginWindow** define una ventana de inicio de sesión. Esta ventana muestra campos de entrada para el nombre de usuario y la contraseña, y también incluye botones para iniciar sesión y cancelar. Si el usuario hace clic en el botón "Cancelar", la ventana se cierra. Si el usuario hace clic en el botón "Iniciar Sesión", el programa comprobará si los datos de inicio son correctos y entonces es cuando saltará la ventana advertencia. Se ve de la siguiente forma:

```
class LoginWindow(QDialog):
    def __init__(self):
        super().__init__()

        self.setWindowTitle("Inicio de sesión")
        #crea los campos de usuario y contraseña
        self.usuario_edit = QLineEdit()
        self.contraseña_edit = QLineEdit()
        self.contraseña_edit.setEchoMode(QLineEdit.Password)

        #crea los botones de inicio de sesión y cancelar
        self.inicio_boton = QPushButton("Iniciar Sesión")
        self.cancelar_boton = QPushButton("Cancelar")
        self.cancelar_boton.clicked.connect(self.close)

        #crea el diseño vertical para la ventana
        layout = QVBoxLayout()

        #crea el diseño horizontal para los campos de usuario y contraseña
        formulario_plantilla = QHBoxLayout()
        formulario_plantilla.addWidget(QLabel("Usuario: "))
        formulario_plantilla.addWidget(self.usuario_edit)
        formulario_plantilla.addWidget(QLabel("Contraseña: "))
        formulario_plantilla.addWidget(self.contraseña_edit)
        layout.addLayout(formulario_plantilla)

        #agrega los botones de inicio de sesión y cancelar
        boton_formulario = QHBoxLayout()
        boton_formulario.addWidget(self.inicio_boton)
        boton_formulario.addWidget(self.cancelar_boton)
        layout.addLayout(boton_formulario)

        self.setLayout(layout)
```

La clase **RegisterWindow** define una ventana de registro. Esta ventana muestra campos de entrada para el nombre de usuario y la contraseña, y también incluye botones para registrarse y cancelar. Si el usuario hace clic en el botón "Cancelar", la ventana se cierra. Si el usuario hace clic en el botón "Registrarse", se escribirá el usuario y contraseña introducidos en un bloc de notas, pero no sin antes comprobar si ese usuario introducido existe ya en el bloc de notas de "datos". El código sería el siguiente:

```
class RegisterWindow(QDialog):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("Registro")

        #campos de registro de usuario y contraseña
        self.usuario_redit = QLineEdit()
        self.contraseña_redit = QLineEdit()
        self.contraseña_redit.setEchoMode(QLineEdit.Password)

        #crear botones de registro y cancelar
        self.registro_boton = QPushButton("Registrarse")
        self.cancelar2_boton = QPushButton("Cancelar")
        self.cancelar2_boton.clicked.connect(self.close)

        #diseño del formulario
        plantilla = QVBoxLayout()
        formulario_plantilla2 = QHBoxLayout()
        formulario_plantilla2.addWidget(QLabel("Usuario: "))
        formulario_plantilla2.addWidget(self.usuario_redit)
        formulario_plantilla2.addWidget(QLabel("Contraseña: "))
        formulario_plantilla2.addWidget(self.contraseña_redit)
        plantilla.addLayout(formulario_plantilla2)

        #agregar botones
        boton_formulario2 = QHBoxLayout()
        boton_formulario2.addWidget(self.registro_boton)
        boton_formulario2.addWidget(self.cancelar2_boton)
        plantilla.addLayout(boton_formulario2)

        self.setLayout(plantilla)
```

La clase **MainWindow** define la ventana principal de la aplicación. Esta ventana muestra un menú con diferentes opciones de sitios web y un botón para ir al menú principal. También incluye un menú para cerrar sesión. El menú principal muestra diferentes opciones, como restaurantes, reservas y cerrar sesión. Cada opción está conectada a un método correspondiente.

Empecemos con el menuBar que contiene los hoteles y restaurantes, se vería de la siguiente forma:

```

#menubar
self.barra = self.menuBar()
menu = self.barra.addMenu("&Sitios Web")

##hoteles
menu2 = menu.addMenu("&Hoteles")
h1 = QAction("Ir a Four Seasons Hotel", self)
h1.triggered.connect(self.h_fourseasons_web)
h2 = QAction("Ir a Hotel Ritz", self)
h2.triggered.connect(self.h_rits_web)
h3 = QAction("Ir a Hotel Villa Magna", self)
h3.triggered.connect(self.h_villamagna_web)
menu2.addAction(h1)
menu2.addAction(h2)
menu2.addAction(h3)

##restaurantes
menu3 = menu.addMenu("&Restaurantes")
r1 = QAction("Ir a Restaurante Botín", self)
r1.triggered.connect(self.r_botin_web)
r2 = QAction("Ir a Restaurante Casa Mono", self)
r2.triggered.connect(self.r_casamono_web)
r3 = QAction("Ir a Restaurante La Vaca y la Huerta", self)
r3.triggered.connect(self.r_vacahuerta_web)
r4 = QAction("Ir a Restaurante StreetXO", self)
r4.triggered.connect(self.r_streetxo_web)
menu3.addAction(r1)
menu3.addAction(r2)
menu3.addAction(r3)
menu3.addAction(r4)

```

Siguen los museos y parques de atracciones también implementados en el menuBar:

```

##museos
menu4 = menu.addMenu("&Museos")
m1 = QAction("Ir a Museo del Prado", self)
m1.triggered.connect(self.m_prado_web)
m2 = QAction("Ir a Museo Nacional Centro de Arte Reina Sofía", self)
m2.triggered.connect(self.m_nacionalsofia_web)
m3 = QAction("Ir a Museo Thyssen-Bornemisza", self)
m3.triggered.connect(self.m_thyssen_web)
menu4.addAction(m1)
menu4.addAction(m2)
menu4.addAction(m3)

##atracciones
menu5 = menu.addMenu("&Atracciones")
a1 = QAction("Ir a Parque de Atracciones de Madrid", self)
a1.triggered.connect(self.a_parqueatracciones_web)
a2 = QAction("Ir a Parque Warner", self)
a2.triggered.connect(self.a_warner_web)
menu5.addAction(a1)
menu5.addAction(a2)

```

Y para finalizar con el menú superior tenemos las opciones de menú y cerrar sesión:

```

##ir a menú
menu6 = self.barra.addMenu("&Ir a Menú")
im = QAction("Ir a Menú", self)
im.triggered.connect(self.menu)
menu6.addAction(im)

##cerrar sesión
menu7 = self.barra.addMenu("&Cerrar Sesión")
cs = QAction("Cerrar Sesión", self)
cs.triggered.connect(self.cerrar_sesion)
menu7.addAction(cs)

```

El menú superior está deshabilitado hasta que inicias sesión, se puede ver en la siguiente línea de código:

```

self.barra.setHidden(True)

```

Ahora que el menú superior está explicado procedemos a pasar a las ventanas principales, comenzando por las ventanas de acceso al registro o inicio de sesión, además de la ventana de menú:

```

#inicio
plantilla_inicio = QFormLayout()
inicio = QWidget()
inicio
inicio.setLayout(plantilla_inicio)
registro_boton = QPushButton("Registrarse")
registro_boton.clicked.connect(self.mostrar_dialogo_de_registro)
plantilla_inicio.addRow(registro_boton)
inicio_boton = QPushButton("Iniciar Sesión")
inicio_boton.clicked.connect(self.mostrar_dialogo_de_inicio)
plantilla_inicio.addRow(inicio_boton)
salir = QPushButton("Salir")
salir.clicked.connect(self.cerrar)
plantilla_inicio.addRow(salir)

#menu
formulario_eleccionr = QFormLayout()
menu = QWidget()
menu.setLayout(formulario_eleccionr)
etiqueta1 = QLabel("MENÚ")
formulario_eleccionr.addRow(etiqueta1)
plan_boton = QPushButton("Elige tu plan")
plan_boton.clicked.connect(self.eleccion)
formulario_eleccionr.addRow(plan_boton)
reserva_boton = QPushButton("Reservas")
reserva_boton.clicked.connect(self.reservas)
formulario_eleccionr.addRow(reserva_boton)
cerrar_sesion_boton = QPushButton("Cerrar Sesión")
cerrar_sesion_boton.clicked.connect(self.cerrar_sesion)
formulario_eleccionr.addRow(cerrar_sesion_boton)

```

Seguimos con la ventana de elección, la cual nos va a permitir seleccionar qué hotel, restaurante, museo o parque de atracciones deseamos, primero decidimos que tipo de plantilla queremos usar y después procedemos con el resto del formulario:

```
#eleccion
plantilla_eleccion = QFormLayout()
eleccion = QWidget()
eleccion.setLayout(plantilla_eleccion)
```

```
eleccion_etiqueta = QLabel("Elige tu plan ideal: \n")
plantilla_eleccion.addRow(eleccion_etiqueta)

hotel_eleccion = QLabel("Hoteles: ")
self.hotel_box = QComboBox()
self.hotel_box.setFixedSize(200, 40)
plantilla_eleccion.addRow(hotel_eleccion, self.hotel_box)
self.hotel_box.insertItem(0, "")
self.hotel_box.addItems(["Hotel Four Seasons: 500€", "Hotel Ritz: 400€", "Hotel Villa Magna: 300€"])

restaurante_eleccion = QLabel("Restaurantes: ")
self.restaurante_box = QComboBox()
self.restaurante_box.setFixedSize(200, 40)
plantilla_eleccion.addRow(restaurante_eleccion, self.restaurante_box)
self.restaurante_box.insertItem(0, "")
self.restaurante_box.addItems(["Restaurante Botín: 60€", "Restaurante StreetX0: 50€", "Restaurante Casa Mono: 40€", "Restaurante La Vaca y La Huerta: 35€"])
```

```
museo_eleccion = QLabel("Museos: ")
self.museo_box = QComboBox()
self.museo_box.setFixedSize(200, 40)
plantilla_eleccion.addRow(museo_eleccion, self.museo_box)
self.museo_box.insertItem(0, "")
self.museo_box.addItems(["Museo del Prado: 15€", "Museo Thyssen-Bornemisza: 13€", "Museo Nacional Centro de Arte Reina Sofia: 10€"])

atraccion_eleccion = QLabel("Atracciones: ")
self.atracciones_box = QComboBox()
self.atracciones_box.setFixedSize(200, 40)
plantilla_eleccion.addRow(atraccion_eleccion, self.atracciones_box)
self.atracciones_box.insertItem(0, "")
self.atracciones_box.addItems(["Parque Warner: 50€", "Parque de Atracciones: 30€" ])

reserva_etiqueta = QLabel("Calcula tu plan ideal -->")
reserva_boton = QPushButton("RESERVAR")
reserva_boton.clicked.connect(self.reservar)
plantilla_eleccion.addRow(reserva_etiqueta, reserva_boton)
```

Pasamos a la ventana de reserva, donde se mostrarán los datos seleccionados anteriormente:

```

#reserva
plantilla_reserva = QFormLayout()
reserva = QWidget()
reserva.setLayout(plantilla_reserva)
datos_reserva_etiqueta = QLabel("Los Datos de la reserva son los siguientes: \n")
plantilla_reserva.addRow(datos_reserva_etiqueta)

self.hotel_etiqueta1 = QLabel("Hotel elegido: ")
self.hotel_etiqueta2 = QLabel("Ninguno")
plantilla_reserva.addRow(self.hotel_etiqueta1, self.hotel_etiqueta2)

self.restaurante_etiqueta1 = QLabel("Restaurante elegido: ")
self.restaurante_etiqueta2 = QLabel("Ninguno")
plantilla_reserva.addRow(self.restaurante_etiqueta1, self.restaurante_etiqueta2)

self.museo_etiqueta1 = QLabel("Museo elegido: ")
self.museo_etiqueta2 = QLabel("Ninguno")
plantilla_reserva.addRow(self.museo_etiqueta1, self.museo_etiqueta2)

self.atraccion_etiqueta1 = QLabel("Atracción elegida: ")
self.atraccion_etiqueta2 = QLabel("Ninguna")
plantilla_reserva.addRow(self.atraccion_etiqueta1, self.atraccion_etiqueta2)

self.total_etiqueta1 = QLabel("Total del plan: ")
self.total_etiqueta2 = QLabel("0€")
plantilla_reserva.addRow(self.total_etiqueta1, self.total_etiqueta2)

self.usuario_etiqueta1 = QLabel("Usuario en uso: ")
self.usuario_etiqueta2 = QLabel("")
plantilla_reserva.addRow(self.usuario_etiqueta1, self.usuario_etiqueta2)

```

Y después vendrían las ventanas de las webs, que solo voy a mostrar dos de las doce que hay debido a que son idénticas y rellenaría esto de imágenes iguales en las que solo cambian nombres y URLs:

```

#WEBS
##hotel cuatro estaciones
plantilla_web1 = QVBoxLayout()
fourseasons_web = QWidget()
fourseasons_web.setLayout(plantilla_web1)
fourseasons_web.setFixedSize(1000, 700)

self.vistaweb1 = QWebEngineView()
self.vistaweb1.load(QUrl("https://www.fourseasons.com/es-es/madrid/"))

self.plantilla_web1h = QHBoxLayout()

plantilla_web1.addLayout(self.plantilla_web1h)
plantilla_web1.addWidget(self.vistaweb1)

##hotel ritz
plantilla_web2 = QVBoxLayout()
ritz_web = QWidget()
ritz_web.setLayout(plantilla_web2)
ritz_web.setFixedSize(1000, 700)

self.vistaweb2 = QWebEngineView()
self.vistaweb2.load(QUrl("https://www.mandarinoriental.com/es-es/madrid/hotel-ritz"))

self.plantilla_web2h = QHBoxLayout()

plantilla_web2.addLayout(self.plantilla_web2h)
plantilla_web2.addWidget(self.vistaweb2)

```

Para finalizar con las ventanas, pasamos a añadirlas todas a una misma plantilla ordenada por capas:

```

#ventanas
self.capa = QStackedLayout()
self.capa.addWidget(inicio)
self.capa.addWidget(menu)
self.capa.addWidget(eleccion)
self.capa.addWidget(reserva)
self.capa.addWidget(fourseasons_web)
self.capa.addWidget(ritz_web)
self.capa.addWidget(villamagna_web)
self.capa.addWidget(botin_web)
self.capa.addWidget(streexo_web)
self.capa.addWidget(casamono_web)
self.capa.addWidget(vacahuerta_web)
self.capa.addWidget(prado_web)
self.capa.addWidget(thyssen_web)
self.capa.addWidget(sofia_web)
self.capa.addWidget(warner_web)
self.capa.addWidget(parque_web)
plantilla_horizontal.addLayout(self.capa)

```

Ya no hay más ventanas que mostrar, ahora pasaríamos a los métodos, empezando por el dialogo de inicio de sesión y registro del mismo:

```

def mostrar_dialogo_de_registro(self):
    #crear una instancia de la ventana de registro de sesión
    self.dialogo_registro = RegisterWindow()

    #conecta el botón de registro de sesion a la escritura
    self.dialogo_registro.registro_boton.clicked.connect(self.escribir)

    #muestra la ventana de inicio de sesion
    self.dialogo_registro.exec()

def mostrar_dialogo_de_inicio(self):
    #crea una instancia de la ventana de inicio de sesion
    self.dialogo_inicio = LoginWindow()

    #conecta el botón de inicio de sesion a la funcion de verificacion de credenciales
    self.dialogo_inicio.inicio_boton.clicked.connect(self.comprobar_credenciales)

    #muestra la ventana de inicio de sesion
    self.dialogo_inicio.exec()

```

Como se explica en el código comentado, al hacer clic en los botones, nos redireccionaríamos a otros métodos, llamados “escribir” y “comprobar_credenciales”, de los cuales hablaremos ahora.

El método escribir ya se ha mostrado en las pruebas unitarias, sin embargo, lo muestro de nuevo para mantener el orden:


```

def escribir(self):
    usuario = self.dialogo_registro.usuario_redit.text()
    contraseña = self.dialogo_registro.contraseña_redit.text()

    try:
        try:
            with open('datos.txt', 'r') as archivo:
                for linea in archivo:
                    if linea.startswith(usuario + ';'):
                        raise Exception("El usuario ya existe")
        except FileNotFoundError:
            pass

        with open('datos.txt', 'a') as archivo:
            registro_texto = usuario + ';' + contraseña + '\n'
            archivo.write(registro_texto)

        self.dialogo_registro.accept()

    except Exception as e:
        QMessageBox.critical(self, "Error de creación de usuario", "El usuario ya existe", buttons=QMessageBox.Ok, defaultButton=QMessageBox.Ok)

```

Ahora muestro el método de las credenciales:

```

def comprobar_credenciales(self):
    comprobar = 0
    registro = open('datos.txt', "r+")
    registro.seek(0)

    for linea in registro:
        user, passwd = linea.split(';')
        if self.dialogo_inicio.usuario_edit.text() == user and self.dialogo_inicio.contraseña_edit.text() + "\n" == passwd:
            comprobar = 1

    if comprobar == 1:
        translator = QTranslator(app)
        translations = QLibraryInfo.location(QLibraryInfo.TranslationsPath)
        translator.load("qt_es", translations)
        app.installTranslator(translator)
        ventana = Advertencia(self)
        resultado = ventana.exec()
        if resultado:
            self.capa.setCurrentIndex(1)
            print("Usuario de inicio correcto")
            self.capa.setCurrentIndex(1)
            self.dialogo_inicio.accept()
            self.barra.setHidden(False)
        else:
            print("")

```

Como podemos ver en la imagen, el método está incompleto, la siguiente imagen complementará a esta, esto lo he decidido hacer para poder explicar que significa y para qué es el traductor que se usa, el traductor lo uso debido a que la ventana advertencia muestra sus opciones en inglés, por lo que al usar estas líneas pasan a leerse al español.

Y paso a finalizar el método con la siguiente imagen de un critical box junto al cierre de la lectura:

```

else:
    QMessageBox.critical(self, "Error de inicio de sesión", "El usuario y/o la contraseña no son válidos", buttons = QMessageBox.Discard, defaultButton = QMessageBox.Discard)

registro.close()

```

Los siguientes métodos sirven para cerrar la aplicación, ocultar de nuevo el menú superior cuando se cierre sesión además de moverse entre ventanas y fijar el nombre de usuario para poder usarlo en la ventana de reserva.

```

def cerrar(self):
    QApplication.instance().quit()

def cerrar_sesion(self):
    self.capa.setCurrentIndex(0)
    self.barra.setHidden(True)

def menu(self):
    self.capa.setCurrentIndex(1)

def eleccion(self):
    self.capa.setCurrentIndex(2)

def reservas(self):
    self.capa.setCurrentIndex(3)
    self.usuario_etiqueta2.setText(self.dialogo_inicio.usuario_edit.text())

```

```

def h_fourseasons_web(self):
    self.capa.setCurrentIndex(4)

def h_rits_web(self):
    self.capa.setCurrentIndex(5)

def h_villamagna_web(self):
    self.capa.setCurrentIndex(6)

def r_botin_web(self):
    self.capa.setCurrentIndex(7)

def r_streetxo_web(self):
    self.capa.setCurrentIndex(8)

def r_casamono_web(self):
    self.capa.setCurrentIndex(9)

def r_vacahuerta_web(self):
    self.capa.setCurrentIndex(10)

def m_prado_web(self):
    self.capa.setCurrentIndex(11)

def m_thyssen_web(self):
    self.capa.setCurrentIndex(12)

def m_nacionalsofia_web(self):
    self.capa.setCurrentIndex(13)

def a_warner_web(self):
    self.capa.setCurrentIndex(14)

def a_parqueatracciones_web(self):
    self.capa.setCurrentIndex(15)

```

La imagen anterior es para el movimiento entre ventanas mediante acciones o botones.

Y por último tenemos el método de reserva, que nos muestra los datos seleccionados:

```

def reservar(self):
    hotel = 0
    restaurante = 0
    museo = 0
    atraccion = 0

    if self.hotel_box.currentText() == "Hotel Four Seasons: 500€":
        self.hotel_etiqueta2.setText("Four Seasons")
        hotel = 500

    elif self.hotel_box.currentText() == "Hotel Ritz: 400€":
        self.hotel_etiqueta2.setText("Ritz")
        hotel = 400
    elif self.hotel_box.currentText() == "Hotel Villa Magna: 300€":
        self.hotel_etiqueta2.setText("Villa Magna")
        hotel = 300

    if self.restaurante_box.currentText() == "Restaurante Botín: 60€":
        self.restaurante_etiqueta2.setText("Botín")
        restaurante = 60

    elif self.restaurante_box.currentText() == "Restaurante StreetX0: 50€":
        self.restaurante_etiqueta2.setText("StreetX0")
        restaurante = 50

    elif self.restaurante_box.currentText() == "Restaurante Casa Mono: 40€":
        self.restaurante_etiqueta2.setText("Casa Mono")
        restaurante = 40

    elif self.restaurante_box.currentText() == "Restaurante La Vaca y La Huerta: 35€":
        self.restaurante_etiqueta2.setText("La Vaca y La Huerta")
        restaurante = 35

    if self.museo_box.currentText() == "Museo del Prado: 15€":
        self.museo_etiqueta2.setText("Del Prado")
        museo = 15

    elif self.museo_box.currentText() == "Museo Thyssen-Bornemisza: 13€":
        self.museo_etiqueta2.setText("Thyssen-Bornemisza")
        museo = 13

    elif self.museo_box.currentText() == "Museo Nacional Centro de Arte Reina Sofia: 10€":
        self.museo_etiqueta2.setText("Nacional Centro de Arte Reina Sofia")
        museo = 10

    if self.atracciones_box.currentText() == "Parque Warner: 50€":
        self.atraccion_etiqueta2.setText("Parque Warner")
        atraccion = 50

    elif self.atracciones_box.currentText() == "Parque de Atracciones: 30€":
        self.atraccion_etiqueta2.setText("Parque de Atracciones")
        atraccion = 30

    total = hotel + restaurante + museo + atraccion
    self.total_etiqueta2.setText(str(total))
    self.capa.setCurrentIndex(3)

```

Además de las ventanas y métodos mencionados anteriormente, el código incluye diferentes widgets y diseños para organizar la interfaz gráfica de la aplicación. Se utilizan diseños verticales y horizontales para estructurar los diferentes elementos de cada ventana.

En cuanto a la interacción con la web, se utilizan componentes de la biblioteca **QtWebEngineWidgets** para mostrar sitios web dentro de la aplicación. Se crean instancias de **QWebEngineView** para cada sitio web y se cargan las URLs correspondientes.

Pruebas unitarias

Como prueba unitaria comentaré la siguiente, en mi caso mi aplicación dispone de un registro de sesión para los usuarios y guarda estos datos en un bloc de notas, pues resulta que en un principio la aplicación permitía tener varios usuarios con el mismo nombre, lo cual podía causar algún fallo de identificación.

Mi solución la explicaré con estas líneas de código:

```
def escribir(self):
    usuario = self.dialogo_registro.usuario_redit.text()
    contraseña = self.dialogo_registro.contraseña_redit.text()

    try:
        try:
            with open('datos.txt', 'r') as archivo:
                for linea in archivo:
                    if linea.startswith(usuario + ';'):
                        raise Exception("El usuario ya existe")
        except FileNotFoundError:
            pass


        with open('datos.txt', 'a') as archivo:
            registro_texto = usuario + ';' + contraseña + '\n'
            archivo.write(registro_texto)

        self.dialogo_registro.accept()

    except Exception as e:
        QMessageBox.critical(self, "Error de creación de usuario", "El usuario ya existe", buttons=QMessageBox.Ok, defaultButton=QMessageBox.Ok)
```

Lo que se puede ver en estas líneas de código del método "escribir", el cual está asociado al registro de sesión es que si el programa logra encontrar un usuario ya introducido con anterioridad pues saltará la excepción de: "El usuario ya existe".

12)Diseño

Para el diseño de la aplicación he optado por el color: 44cccc que representa este color  además, he importado los módulos QFont, para poder usar el Bold para todos los textos. A parte de eso también he cambiado el color de la letra a blanco y en los fondos a negro, además de usar los mismos colores, pero a la inversa en otras ocasiones.

13)Funcionalidades

Funcionalidad implementada: El código proporcionado contiene una aplicación de reserva de planes turísticos en Madrid. La aplicación cuenta con las siguientes funcionalidades:

Inicio de sesión y registro:

- La aplicación muestra una ventana de inicio de sesión que permite a los usuarios ingresar su nombre de usuario y contraseña.
- Si el usuario no tiene una cuenta, puede hacer clic en el botón "Registrarse" para abrir una ventana de registro donde puede crear una nueva cuenta.

Menú principal:

- Después de iniciar sesión o registrarse, se muestra un menú principal con varias opciones.
- El menú principal incluye enlaces a diferentes categorías de planes turísticos, como hoteles, restaurantes, museos y atracciones.
- También hay una opción para cerrar sesión.

Navegación por sitios web:

- Dentro de cada categoría, se muestran opciones específicas, como hoteles, restaurantes, etc.
- Al hacer clic en una opción, se abre un navegador web integrado en la aplicación que muestra el sitio web correspondiente.
- Por ejemplo, al hacer clic en "Ir a Four Seasons Hotel", se muestra el sitio web del Hotel Four Seasons en el navegador integrado.

Reserva de planes:

- La aplicación permite al usuario seleccionar diferentes opciones de planes turísticos, como hoteles, restaurantes, museos y atracciones.
- Después de seleccionar las opciones deseadas, el usuario puede hacer clic en el botón "Reservar" para confirmar la reserva.
- La aplicación muestra los detalles de la reserva, incluidos los elementos seleccionados y el costo total del plan.

Información del usuario:

- La ventana de reserva también muestra el nombre de usuario en uso.

- Esto proporciona información adicional al usuario sobre la sesión actual.

Menú de navegación adicional:

- El menú superior de la aplicación contiene opciones adicionales, como regresar al menú principal o cerrar sesión.

14) Conclusión

A la hora de arrancar el proyecto me ha resultado muy difícil elegir un tema sobre el que hacer la aplicación, tardé demasiado en encontrar un tema que me llamase la atención a la vez que fuera lo suficientemente desarrollable para el tfg, en mi caso elegí las guías debido a que encontré que podía introducir mapas interactivos con los que la aplicación hubiera ganado mucho juego, sin embargo entre una cosa y otra al final se me hizo imposible implementarlo de forma correcta, ya que los mapas pedían ser abiertos como html a través de un navegador, en su momento intenté hacerlo con los navegadores integrados que he usado para la aplicación, no obstante no conseguí que funcionaran por lo que me decanté por seguir el proyecto sin los mapas. No tengo ninguna duda de que si la app hubiera contado con los mapas que pensaba implementar no solo se hubiera visto mucho mejor decorativamente hablando además de por complejidad. Un motivo por el que seguí con el proyecto aún con esa complicación es porque ya había gastado mucho tiempo intentando que la idea funcionase, y ya tenía los links e información de las empresas que he usado, por lo que echarme para atrás a esas alturas me hubiera impedido terminar el tfg por falta de tiempo.

La aplicación no es lo que buscaba, pero sin embargo es lo que ha salido con todas las complicaciones que se me presentaron en su momento. Decidí desde el principio realizar el proyecto de forma individual sin asistencia de los profesores para poder hacer el trabajo únicamente con mi mano e ideas, podría haber salido mucho mejor si hubiera podido implementar lo antes mencionado, sin embargo, no me arrepiento de esa decisión.

15) Bibliografía

<https://stackoverflow.com>

<https://code.visualstudio.com>

<https://www.microsoft.com/es-es/>

<https://www.python.org>

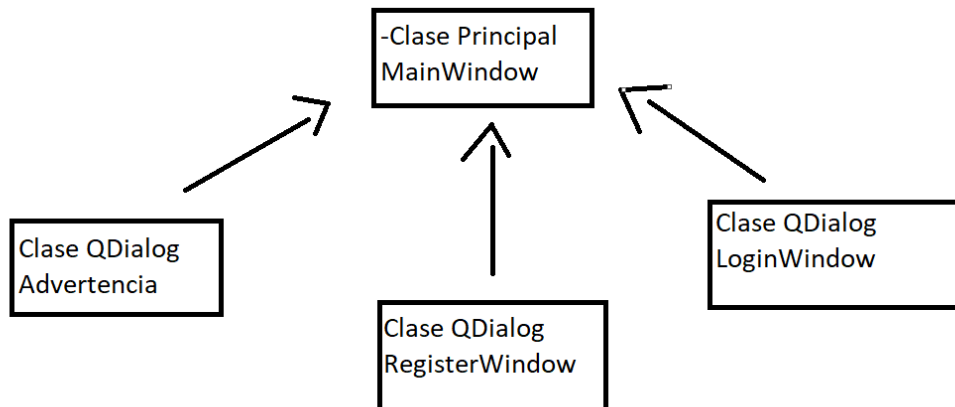
<https://pypi.org/project/PySide6/>

https://www.odoo.com/es_ES

https://monday.com/lang/es/lp/gantt?utm_medium=cpc&utm_campaign=ww-multi-prm-marketing-workos-desktop-gantt_chart-listing-core&utm_adgroup=gantt_chart&utm_source=capterra

16)Anexos

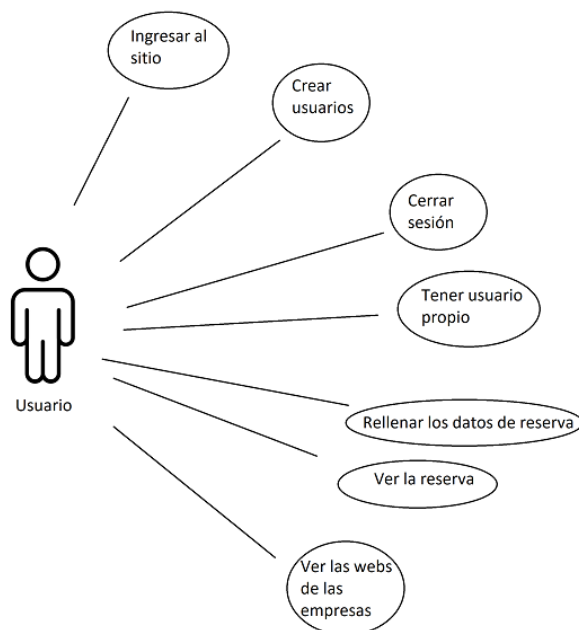
Anexo 1



En el diagrama de clases que uso para mi aplicación podemos observar que la única clase como tal y principal es la llamada “MainWindow”, por lo que las clases QDialog usadas para emerger formularios o advertencias se unirán a esta.

Anexo 2

Como ya dije en su propio apartado, el diagrama de casos de uso solo dispone de la perspectiva de usuario ya que la aplicación no está diseñada actualmente para albergar opciones a un administrador, en versiones futuras de la aplicación sí se podría integrar, ya sea para darle acciones como eliminar usuarios, añadir más sitios o categorías, etc.



Anexo 3

Librería PySide. A continuación, paso a presentar descripción detallada de las características y aspectos clave de PySide6:

1. Basado en Qt: PySide6 se basa en Qt, un popular framework de desarrollo de aplicaciones C++. Qt es conocido por su amplia gama de funcionalidades para la creación de interfaces gráficas de usuario, incluyendo controles, gráficos, estilos, animaciones y más. Al utilizar PySide6, los desarrolladores de Python pueden acceder a todas estas capacidades de Qt.
2. Interfaz gráfica de usuario moderna y atractiva: PySide6 permite crear interfaces de usuario modernas y atractivas con facilidad. Proporciona una amplia variedad de widgets y controles listos para usar, como botones, cuadros de texto, tablas, árboles y más. Estos elementos se pueden personalizar y adaptar a las necesidades específicas de la aplicación.
3. Multiplataforma: PySide6 permite desarrollar aplicaciones de escritorio que se ejecutan en múltiples plataformas, como Windows, macOS y Linux. Esto se debe a la capacidad de Qt para abstraer la capa de sistema operativo, lo que permite que las aplicaciones sean portables y funcionen de manera consistente en diferentes entornos.
4. Compatibilidad con Qt Designer: PySide6 se integra con Qt Designer, una herramienta gráfica de diseño de interfaces de usuario. Qt Designer permite crear y diseñar la interfaz gráfica de la aplicación arrastrando y soltando widgets, estableciendo propiedades y conectando señales y ranuras. La integración con PySide6 permite utilizar los diseños generados en Qt Designer directamente en la aplicación Python.
5. Señales y ranuras: PySide6 utiliza el sistema de señales y ranuras de Qt, que facilita la comunicación entre los componentes de la interfaz gráfica de usuario. Este mecanismo

permite conectar eventos y acciones a funciones específicas, lo que proporciona una forma eficiente de manejar la interacción del usuario y actualizar la interfaz en tiempo real.

6. Soporte para estilos y temas personalizados: PySide6 permite personalizar la apariencia de la aplicación utilizando hojas de estilo CSS. Esto significa que los desarrolladores pueden adaptar la interfaz gráfica a sus necesidades y crear una experiencia de usuario única. Además, PySide6 es compatible con los estilos y temas predefinidos de Qt, lo que permite una fácil adaptación a la apariencia del sistema operativo.
7. Integración con otras bibliotecas de Python: PySide6 se puede combinar con otras bibliotecas y herramientas populares de Python, lo que amplía aún más su funcionalidad. Esto incluye la integración con bibliotecas de procesamiento de datos, bibliotecas de trazado de gráficos, bases de datos y más.

En resumen, PySide6 es una biblioteca poderosa que permite a los desarrolladores de Python crear aplicaciones de escritorio modernas y atractivas con una gran variedad de funcionalidades. Con su capacidad multiplataforma y su integración con Qt, PySide6 es una opción popular para aquellos que desean desarrollar interfaces gráficas de usuario en Python. Es la alternativa que yo he escogido para no usar TKinter, además de esas dos, hay más alternativas como pueden ser PyGTK, PyQt y wxPython. Al igual que Python, Qt es de software libre y en mi caso uso su última versión, que es PySide6.