AGENDA TELEFÓNICA DE CONTACTOS

Hugo Díaz Casado 1º DAW

ÍNDICE

1. Análisis:

1.1 INTRODUCCIÓN

Necesitamos desarrollar un programa de una agenda telefónica de contactos, donde el usuario podrá hacer movimientos como agregar y borrar contactos entre otros.

1.2 REQUERIMIENTOS DE LA APLICACIÓN

1.2.1 Requisitos funcionales

Con esta agenda telefónica, el usuario podrá usar diferentes opciones del programa, las cuales son:

a. **AÑADIR CONTACTOS**: El usuario podrá añadir contactos (siempre siguiendo un patrón correcto por cada campo), por ejemplo no podrá introducir un nombre que contenga valores numéricos. Los contactos se añadirán en caso de no existir ya en la propia agenda, no podremos tener en nuestra agenda dos contactos con el mismo nombre. A la hora de añadir contactos el usuario deberá de seleccionar si quiere que sea un contacto personal o de empresa, ya que tendrán diferentes campos.

b. **BORRAR CONTACTOS**: el usuario podrá eliminar un contacto mediante el nombre, en caso de no encontrar un contacto con dicho nombre nos saldrá un mensaje de alerta avisándonos de que ese contacto no se encuentra en la agenda, si lo borra nos avisará de que ha sido eliminado con éxito.

c. **LISTAR CONTACTOS:** el usuario también tendrá la opción de listar contactos en la cuál podrá visualizar toda la información de sus contactos, además vendrá reflejado el tipo de contacto de cada uno de los contactos de la agenda.

d. **BUSCAR CONTACTOS:** El usuario tendrá la opción que mediante su nombre sea capaz de encontrar el contacto, en caso de estar guardado en la agenda, lo reflejará en la pantalla, si no lo está dirá que no existe ningún contacto con dicho nombre.

e. **EXISTE CONTACTO:** Siguiendo la misma metodología de la opción anterior, mediante el nombre de un contacto el programa nos comunicará si el contacto existe o no.

1.2.2 Requisitos no funcionales

Las diferentes características que debe de tener nuestro programa son las siguientes:

a. **USABILIDAD:** Nuestra aplicación debe de ser bastante simple y sencilla para que cualquier usuario la pueda usar sin ningún problema (debe de estar preparada para cualquier entrada por parte del usuario), y debe de tener una interfaz lo más visual y atractiva posible.

b. **RENDIMIENTO:** El programa debe de ser lo más efectivo y rápido posible, debe de tener lo justo y necesario para no forzar errores, y que nuestra aplicación ejecutae el recorrido más corto posible para una opción de la agenda (como por ejemplo, eliminar contactos), eso sí, debiendo de pasar por todos los caminos implementados (todos los casos posibles).

c. **ESCALABIDAD:** La aplicación debe de soportar una gran cantidad de datos sin que ello afecte a la rapidez de la aplicación.

d. **COMPATIBILIDAD:** El programa debe de ser compatible con distintos S.O. como por ejemplo Linux, Windows, MacOS…

e. **SEGURIDAD:** Debemos de proporcionar seguridad al usuario, y que su información enviada vaya protegida y segura, para garantizar la privacidad.

f. **DOCUMENTACIÓN:** La aplicación debe de estar documentada de una manera clara y legible, para que cualquier desarrollador mediante esta documentación pueda seguir manteniendo de la forma menos compleja posible dicha app.

g. **DISPONIBILIDAD:** El programa debe de poder usarse en todo momento, que no genere caídas y que no tenga posea de inactividad.

1.3 REQUERIMIENTOS SOFTWARE Y HARDWARE

Para poder soportar esta agenda telefónica necesitamos los siguientes requerimientos:

1.3.1 Hardware:

a. **DISPOSITIVO:** debemos tener un dispositivo compatible, que puede hacer uso de la aplicación, en este caso un ordenador o un portátil.

b. **PROCESADOR:** Un procesador que tenga la suficiente potencia y velocidad como para ejecutar la aplicación y funcione de una forma dinámica.

c. **RAM:** Se recomienda un Memoria RAM que pueda mantener de una forma óptima el funcionamiento de la aplicación.

d. **ESPACIO DE ALMACENAMIENTO:** Un espacio de almacenamiento para que se puedan guardar todo los datos de la agenda telefónica.

1.3.2 Software:

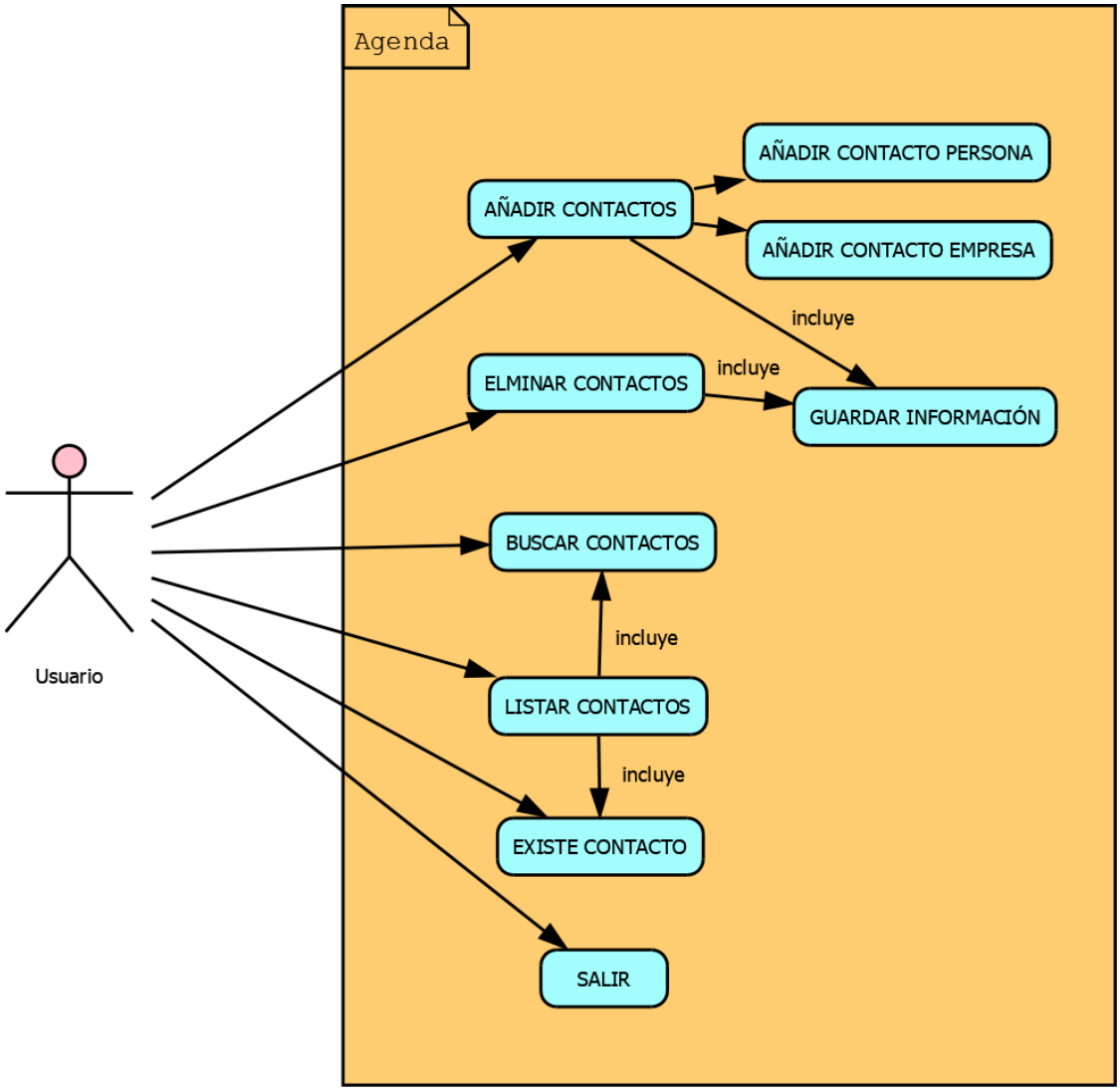
a. **SISTEMA OPERATIVO:** La aplicación es compatible con Windows, Linux y MacOS

b. **VERSION DEL S.O.:** La aplicación requiere una versión mínima de S.O. para poder poner en funcionamiento la agenda telefónica.

c. **BASES DE DATOS:** La agenda utiliza un SGBD para almacenar toda la información, en concreto MySQL

d. **ENTORNO DE DESARROLLO:** Como mínimo usaremos la consola de Python para poner en funcionamiento la aplicación, pero es recomendable un entorno de desarrollo como por ejemplo PyCharm o Visual Studio Code.

1.4 CASOS DE USO



Tenemos un actor en esta aplicación:

USUARIO:
El usuario entrará en un menú principal con 6 opciones a elegir:

```
================================================================================
                                   MI AGENDA
================================================================================
                                   OPCIONES
================================================================================
1. AÑADIR UN CONTACTO
2. ELIMINAR UN CONTACTO
3. LISTAR MIS CONTACTOS
4. BUSCAR UN CONTACTO
5. EXISTE UN CONTACTO?
6. SALIR?
================================================================================
¿QUÉ DESEAS HACER?:
```

La primera opción le permite añadir un contacto, al seleccionar esa opción se la abrirá un submenú que le permitirá decidir si el contacto que quiere añadir es personal o de empresa, dependiendo de el que elija debe de rellenar unos campos los cuales algunos requerirán un patrón especifico, cuando el usuario ya no necesite añadir más contactos tendrá una 3ª opción para volver al menú principal cuando lo desee.

```
================================================================================
                                 AÑADIR CONTACTO
================================================================================
1. CONTACTO PERSONAL
2. CONTACTO EMPRESA
3. VOLVER?
================================================================================
¿QUÉ TIPO DE CONTACTO DESEAS AÑADIR?:
```

CONTACTO PERSONAL:

```
¿QUÉ TIPO DE CONTACTO DESEAS AÑADIR?: 1
TELÉFONO: 678
EL Nº DE TELÉFONO DEBE TENER 9 DÍGITOS
TELÉFONO: 689432103
NOMBRE COMPLETO: PEDRO PÉREZ
FECHA DE NACIMIENTO(YYYY-MM-DD): 13-01-2023
FORMATO INCORRECTO
FECHA DE NACIMIENTO(YYYY-MM-DD): 2023-01-10
```

CONTACTO DE EMPRESA:

```
¿QUÉ TIPO DE CONTACTO DESEAS AÑADIR?: 2
TELÉFONO: 956
EL Nº DE TELÉFONO DEBE TENER 9 DÍGITOS
TELÉFONO: 954302145
NOMBRE COMPLETO: Bosch
DESCRIPCIÓN DE LA EMPRESA: Empresa de Electrodomésticos
PAGINA WEB: bosch.es
```

Estas dos opciones a su vez permiten modificar la información ya que cualquiera de estas modificaciones se guardan de forma dinámica en el programa.

Como segunda opción el usuario podrá eliminar un contacto al elegir esta función, el programa solicitará el nombre del contacto que desea eliminar, el programa le informará si ha podido eliminar ese contacto o no, debido a que no existía un contacto con dicho nombre.

CONTACTO ELIMINADO ✓

```
================================================================================
                                    MI AGENDA
================================================================================
                                    OPCIONES
================================================================================
1. AÑADIR UN CONTACTO
2. ELIMINAR UN CONTACTO
3. LISTAR MIS CONTACTOS
4. BUSCAR UN CONTACTO
5. EXISTE UN CONTACTO?
6. SALIR?
================================================================================
¿QUÉ DESEAS HACER?: 2
CONTACTO A BORRAR: Pepe Navarro Rosas
CONTACTO ELIMINADO
```

NO EXISTE EL CONTACTO ❌

```
==========================================================================================
                                        MI AGENDA
==========================================================================================
                                        OPCIONES
==========================================================================================
1. AÑADIR UN CONTACTO
2. ELIMINAR UN CONTACTO
3. LISTAR MIS CONTACTOS
4. BUSCAR UN CONTACTO
5. EXISTE UN CONTACTO?
6. SALIR?
==========================================================================================
¿QUÉ DESEAS HACER?: 2
CONTACTO A BORRAR: Pepe Viyuelas
NO EXISTE EL CONTACTO
```

Otra opción del menú (concretamente la tercera) permite al usuario visualizar su agenda y ver todos los contactos que tiene registrados, podremos ver todas y cada una de sus características de una forma clara y ordenada.

```
==========================================================================================
TIPO: EMPRESA
TELÉFONO: 906334500
NOMBRE: Perfumerías Avenida
FECHA DE INGRESO: 2023-05-18 08:07:31
DESCRIPCIPCIÓN DE LA EMPRESA: Productos de cosmética, maquillaje e higiene
PÁGINA WEB: perfumeriasavenida.es
==========================================================================================
TIPO: PERSONAL
TELÉFONO: 602403924
NOMBRE: Pedro Horonorato Otero
FECHA DE INGRESO: 2023-05-18 08:11:08
FECHA DE NACIMIENTO: 1966-01-25
==========================================================================================
TIPO: EMPRESA
TELÉFONO: 901239430
NOMBRE: AndorraTelecom
FECHA DE INGRESO: 2023-05-18 08:12:09
DESCRIPCIPCIÓN DE LA EMPRESA: Empresa de telecomunicaciones de Andorra
PÁGINA WEB: andorratelecom.ad
==========================================================================================
TIPO: PERSONAL
TELÉFONO: 683201456
NOMBRE: Pepe Lunas Pérez
FECHA DE INGRESO: 2023-05-20 14:14:45
FECHA DE NACIMIENTO: 2000-01-24
==========================================================================================
TIPO: EMPRESA
TELÉFONO: 903220001
NOMBRE: DXRacer
FECHA DE INGRESO: 2023-05-20 14:17:47
DESCRIPCIPCIÓN DE LA EMPRESA: Empresa fabricadora de material gaming
PÁGINA WEB: dxracer.fr
==========================================================================================
```

Como cuarta opción el usuario podrá buscar un contacto, al seleccionarla le pedirá al usuario un nombre del usuario a buscar, en caso de estar almacenada en su agenda de contactos, el programa reflejará el contacto en la pantalla con cada uno de sus campos en el mismo formato que en el listado de contactos, en caso de no encontrar ningún contacto con ese nombre la aplicación avisará de que no lo ha podido encontrar.

CONTACTO ENCONTRADO ✓

```
================================================================================
                                   MI AGENDA
================================================================================
                                   OPCIONES
================================================================================
1. AÑADIR UN CONTACTO
2. ELIMINAR UN CONTACTO
3. LISTAR MIS CONTACTOS
4. BUSCAR UN CONTACTO
5. EXISTE UN CONTACTO?
6. SALIR?
================================================================================
¿QUÉ DESEAS HACER?: 4
ESCRIBEME EL NOMBRE DE UN CONTACTO: Pepe Navarro Rosas
================================================================================
                                CONTACTO ENCONTRADO
================================================================================
TIPO: PERSONAL
TELÉFONO: 983450234
NOMBRE: Pepe Navarro Rosas
FECHA DE INGRESO: 2023-05-10 15:48:25
FECHA DE NACIMIENTO: 1977-04-04
================================================================================
```

CONTACTO NO ENCONTRADO ✗

```
================================================================================
                                   MI AGENDA
================================================================================
                                   OPCIONES
================================================================================
1. AÑADIR UN CONTACTO
2. ELIMINAR UN CONTACTO
3. LISTAR MIS CONTACTOS
4. BUSCAR UN CONTACTO
5. EXISTE UN CONTACTO?
6. SALIR?
================================================================================
¿QUÉ DESEAS HACER?: 4
ESCRIBEME EL NOMBRE DE UN CONTACTO: Pepe Viyuelas
CONTACTO NO ENCONTRADO
```

La 5ª opción permite al usuario saber si existe o no un contacto, mediante un nombre, la metodología que sigue la aplicación para detectar si existe es idéntica a la anterior opción, pero en este caso en caso de existir ese contacto reflejará un pantalla un mensaje que dice que ese contacto existe, en caso de existir mostrará un mensaje diciendo lo contrario.

CONTACTO EXISTE ✓

```
================================================================================
                                    MI AGENDA
================================================================================
                                    OPCIONES
================================================================================
1. AÑADIR UN CONTACTO
2. ELIMINAR UN CONTACTO
3. LISTAR MIS CONTACTOS
4. BUSCAR UN CONTACTO
5. EXISTE UN CONTACTO?
6. SALIR?
================================================================================
¿QUÉ DESEAS HACER?: 5
ESCRIBEME EL NOMBRE DE UN CONTACTO: Pepe Navarro Rosas
EL CONTACTO EXISTE
```

CONTACTO NO EXISTE ✗

```
================================================================================
                                    MI AGENDA
================================================================================
                                    OPCIONES
================================================================================
1. AÑADIR UN CONTACTO
2. ELIMINAR UN CONTACTO
3. LISTAR MIS CONTACTOS
4. BUSCAR UN CONTACTO
5. EXISTE UN CONTACTO?
6. SALIR?
================================================================================
¿QUÉ DESEAS HACER?: 5
ESCRIBEME EL NOMBRE DE UN CONTACTO: Pepe Viyuelas
EL CONTACTO NO EXISTE
```

Por último, el usuario podrá salir de la aplicación en cualquier momento, siempre y cuando esté en el menú de inicio de la agenda, cuando seleccione esta opción el programa se detendrá automáticamente, y todos los registros de almacenarán en una Base de datos.

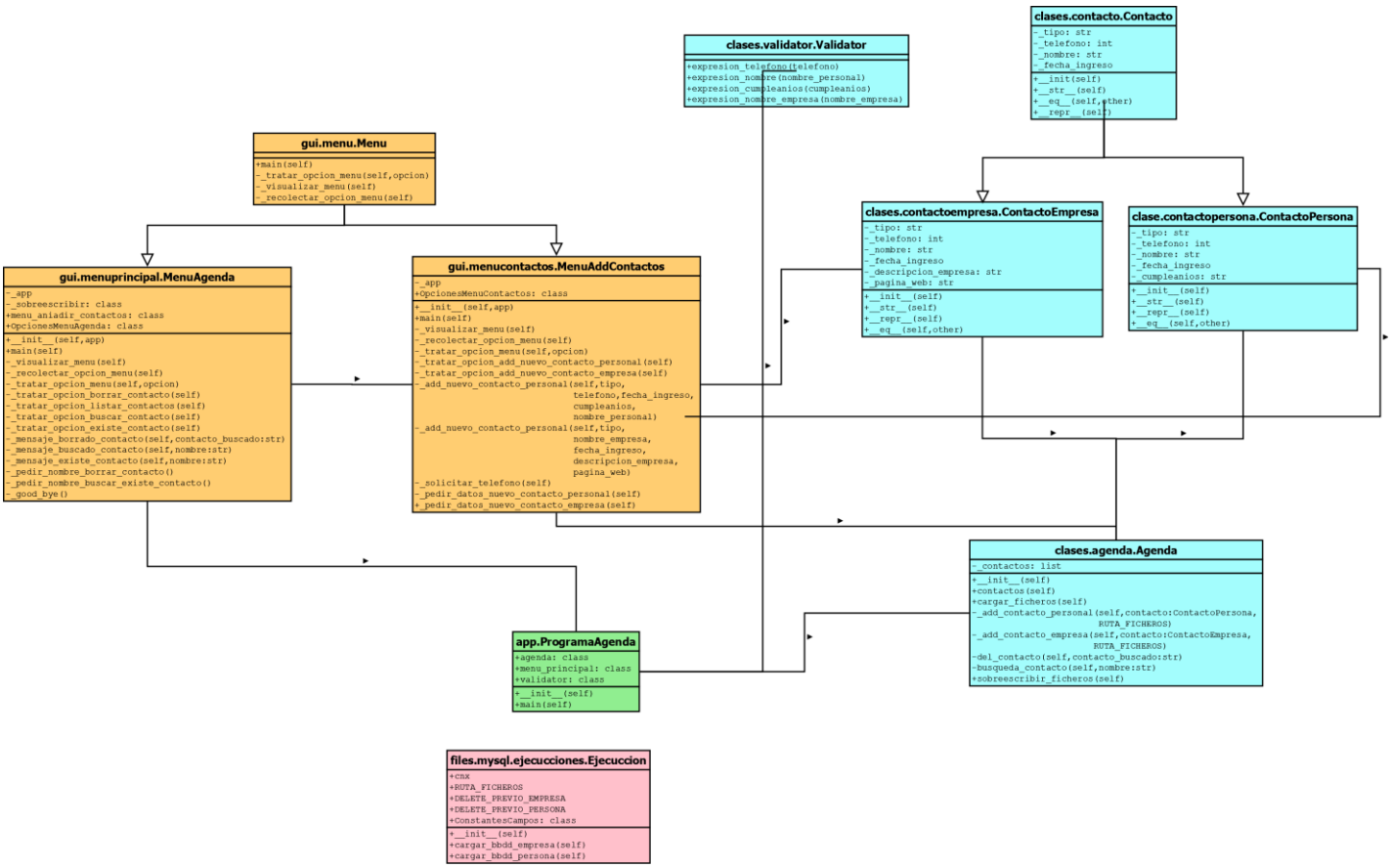
```
================================================================================
                                    MI AGENDA
================================================================================
                                    OPCIONES
================================================================================
1. AÑADIR UN CONTACTO
2. ELIMINAR UN CONTACTO
3. LISTAR MIS CONTACTOS
4. BUSCAR UN CONTACTO
5. EXISTE UN CONTACTO?
6. SALIR?
================================================================================
¿QUÉ DESEAS HACER?: 6

HASTA LA PRÓXIMA
```

2. Diseño:

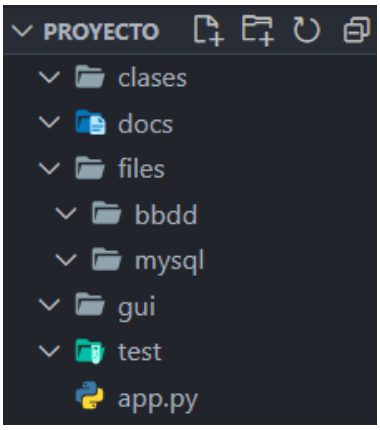
La implementación utilizada para la agenda telefónica, es la Programación Orientada a Objetos (POO), donde haremos una distinción entre clases pertenecientes a la interfaz gráfica y las clases que almacenen y generen contactos para la agenda.

2.1 DIAGRAMA DE CLASES



2.2 ESTRUCTURA DE ALMACENAMIENTO

Para realizar este proyecto y que todos los ficheros y datos del mismo queden de forma organizada y clara vamos a usar la siguiente distribución de directorios:



En el primer directorio almacenaremos todas las **clases** principales (los contactos, la agenda de contactos y un validador que nos valide cada uno de los campos de los usuarios).

En el directorio llamado **docs** guardaremos toda la documentación externa del proyecto, casos de uso, diagrama UML, la propia documentación de la app, y la finalidad de la aplicación...

En la carpeta **files** guardaremos los ficheros que almacenan todos los datos de la app, los emplearemos primeramente para cargar los datos de la aplicación al iniciarla, y cuando paremos la ejecución, se producirán las modificaciones convenientes, además dentro de esta carpeta guardaremos dos más, una llamada **mysql** en la que guardaremos los ficheros ".py" que nos permitirán guardar todos los datos de nuestros ficheros en una base de datos, para ello usaremos el SGBD MySQL. Y por último tenemos una carpeta **bbdd** que guardará un fichero de extensión ".sql" donde quedará guardada toda la información de nuestra bbdd (create tables e inserts).

En el directorio llamado **gui** guardaremos toda la parte de la interfaz es decir la parte visible para al usuario de nuestra aplicación.

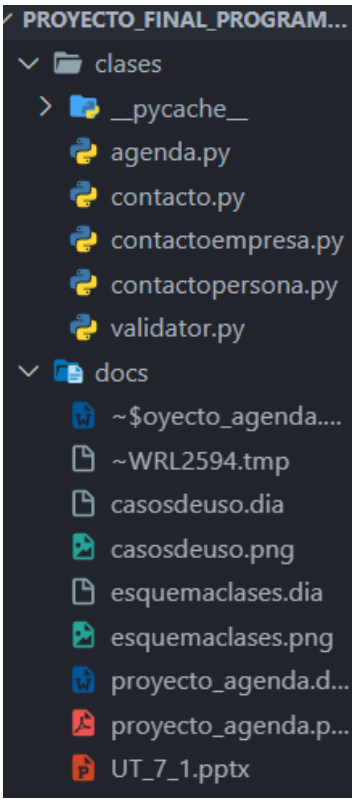
Por último, otro directorio que también tendrá nuestra aplicación será **test** allí almacenaremos todos los test que validarán el correcto funcionamiento de nuestra aplicación mediante casos de prueba, para comprobar todos los caminos que puede tomar nuestro programa en base a los datos enviados por el usuario a nuestra aplicación.

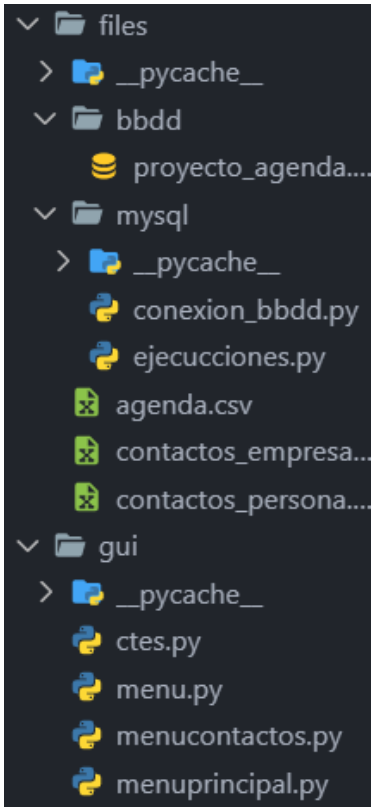
Por último, tendremos nuestro fichero **app.py** con el que pondremos la aplicación en marcha.

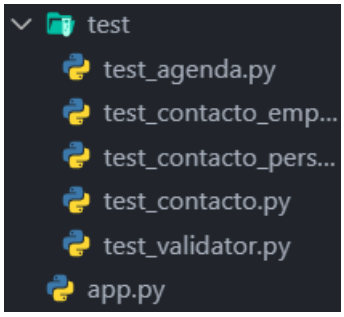
3. Codificación:

Para la codificación utilizaremos un entorno de desarrollo, en este caso Visual Studio Code para facilitar el desarrollo de la aplicación, y utilizaremos el lenguaje de programación Python con el que gestionaremos el funcionamiento de la app, creando ficheros ".csv" para guardar los datos y conectándonos en localhost a un SGBD, en este caso MySQL. Para codificar como hemos mencionado anteriormente usaremos la POO.

Tras finalizar todo el proceso de codificación, nos quedó la siguiente estructura de directorios y ficheros:







4. Pruebas:

Para generar las pruebas de nuestra aplicación importamos el módulo que incluye Python en su librería estándar llamado **unittest** con el que programamos las pruebas de la agenda telefónica. Solo realizamos las pruebas a las clases de almacenamiento y modificación de los contactos, a los propios contactos y al validador de las características de nuestros contactos, necesitamos probar todos los casos que podrían ser introducidos por el usuario en nuestra aplicación.

Algunas de las pruebas que hemos realizado son las siguientes:

```
def test_no_igualdad(self):
        self.assertFalse(
ContactoEmpresa("EMPRESA", 678444653, "2023-03-07",  "Mario.com",
"Empresa con 500.000 trabajadores", "www.mariosl.com")  ==
ContactoEmpresa("EMPRESA", 678444653, "2023-01-01", "Mario S.L.",
"Empresa con 100.000 trabajadores", "www.mariosl.com"))
```

En esta prueba comparamos dos contactos de empresa donde podemos visualizar cada uno de sus campos, pero vemos que no son iguales ya que el campo nombre (en este caso el 4º campo), vemos que es diferente en los dos contactos (Mario S.L. y Mario.com) por tanto el segundo contacto podría ser añadido en la agenda con éxito.

```
def test_igualdad(self):
        self.assertTrue(
ContactoEmpresa("EMPRESA", 678444653, "2023-03-07", "Mario S.L.",
"Empresa con 500.000 trabajadores", "www.mariosl.com") ==
ContactoEmpresa("EMPRESA", 678444653, "2023-01-01", "Mario S.L.",
"Empresa con 100.000 trabajadores","www.mariosl.com"))
```

En este caso sucede todo lo contrario ya que tienen el mismo nombre por tanto, el segundo no podría ser añadido al sistema.

Con los contactos de persona sucede lo mismo, pero con distintos campos:

```
def test_no_igualdad(self):
        self.assertFalse(
ContactoPersona("PERSONAL", 678444653, "2023-01-01", "Mario",
"2000-01-01") ==
ContactoPersona("PERSONAL", 678444653, "2023-01-01", "Pepe",
"2000-01-01"))
```

En este caso tienen distinto nombre por tanto, el segundo contacto podría añadirse sin ningún problema al programa.

```
def test_igualdad(self):
        self.assertTrue(
ContactoPersona("PERSONAL", 678444653, "2023-01-01", "Mario",
"2000-01-01") ==
ContactoPersona("PERSONAL", 666666666, "2023-07-04", "Mario",
"2000-01-01"))
```

Al tener el mismo nombre no el segundo contacto no podrá ser añadido (a pesar de tener algún campo distinto).

Algunas pruebas que le hemos hecho al validador de la aplicación son los siguientes:

```
def test_expresion_telefono(self):
        telefono = 949494949
        self.assertTrue(telefono)
```

Teléfono correcto.

```
def test_no_expresion_telefono(self):
    telefono1 = 949494
    telefono2 = "b"
    self.assertFalse(telefono1)
    self.assertFalse(telefono2)
```

Teléfono incorrecto debido a que un número no puede tener menos de 9 dígitos ni letras

```
def test_expresion_nombre(self):
    nombre1 = "MARIO PÉREZ"
    nombre2 = "MARTA SÁNCHEZ-ARJONA"
    self.assertTrue(nombre1)
    self.assertTrue(nombre2)
```

Nombres correctos (un nombre puede contener guiones).

```
def test_no_expresion_nombre(self):
    nombre1 = 9878695
    nombre2 = "MARIO.GLD"
    self.assertFalse(nombre1)
    self.assertFalse(nombre2)
```

Nombre incorrecto, un nombre no puede tener números ni punto.

```
def test_expresion_cumpleanios(self):
    cumpleanios1 = "2000-01-01"
    self.assertTrue(cumpleanios1)
```

Formato correcto (YYYY-MM-DD).

```
def test_no_expresion_cumpleanios(self):
    cumpleanios1 = "02-01-01"
    cumpleanios2 = "2000/01/01"
    cumpleanios3 = 3469858
    self.assertFalse(cumpleanios1)
    self.assertFalse(cumpleanios2)
    self.assertFalse(cumpleanios3)
```

Formatos incorrectos de fecha.

```
def test_expresion_nombre_empresa(self):
    nombre = "ejemplo.com"
    self.assertTrue(nombre)
```

Nombre de empresa correcto.

```
def test_no_expresion_nombre_empresa(self):
    nombre1 = "MA380757"
    nombre2 = 343563
    self.assertFalse(nombre1)
    self.assertFalse(nombre2)
```

Nombre de empresa incorrecto, ya que no puede contener números.

Por último, algunas de las pruebas que hemos hecho a la clase agenda son:

```
def test_add_contacto_personal(self):
    contacto1 = ContactoPersona("PERSONAL", 678444653, "2023-01-01",
    "Mario", "2000-01-01")
    contacto2 = ContactoPersona("PERSONAL", 678444653, "2023-01-01",
    "Pedro", "2000-01-01")
    contacto3 = ContactoPersona("PERSONAL", 678444653, "2023-01-01",
    "Pepe", "2000-01-01")
    contacto4 = ContactoPersona("PERSONAL", 678444653, "2023-01-01",
    "Marta", "2000-01-01")

    arraycontactos = [contacto1, contacto2, contacto3]
    self.assertEqual(contacto1 in arraycontactos,
    [contacto1, contacto2, contacto3])
    self.asssertEqual(contacto4 in arraycontactos,
    [contacto1, contacto2, contacto3, contacto4])
```

Si el contacto personal ya está en el sistema (tiene un nombre igual a uno de la agenda), no lo añade, si no está, lo añade al final de la agenda.

```
def test_add_contacto_empresa(self):
    contacto1 = ContactoEmpresa(“EMPRESA”, 678444653, “2023-03-07”,
    “Mario S.L.”, “Empresa con 500.000 trabajadores”,
“www.mariosl.com”)
    contacto2 = ContactoEmpresa(“EMPRESA”, 678444653, “2023-03-07”,
    “Pepe S.L.”, “Empresa con 500.000 trabajadores”,
“www.mariosl.com”)
    contacto3 = ContactoEmpresa(“EMPRESA”, 678444653, “2023-03-07”,
    “Juan S.L.”, “Empresa con 500.000 trabajadores”,
“www.mariosl.com”)
    contacto4 = ContactoEmpresa(“EMPRESA”, 678444653, “2023-03-07”,
    “Marta S.L.”, “Empresa con 500.000 trabajadores”,
“www.mariosl.com”)

    arraycontactos = [contacto1, contacto2, contacto3]
    self.assertEqual(contacto1 in arraycontactos,
    [contacto1, contacto2, contacto3])
    self.asssertEqual(contacto4 in arraycontactos,
    [contacto1, contacto2, contacto3, contacto4])
```

Lo mismo que el caso anterior, pero contactos de empresa.

```
def test_del_busqueda_contacto_empresa(self):
    '''
    Vale para borrar, buscar y existe
    '''
    contacto1 = ContactoEmpresa("EMPRESA", 678444653, "2023-03-07",
    "Mario S.L.", "Empresa con 500.000 trabajadores",
"www.mariosl.com")
    contacto2 = ContactoEmpresa("EMPRESA", 678444653, "2023-03-07",
```

```
    "Pepe S.L.", "Empresa con 500.000 trabajadores",
"www.mariosl.com")
    contacto3 = ContactoEmpresa("EMPRESA", 678444653, "2023-03-07",
    "Juan S.L.", "Empresa con 500.000 trabajadores",
"www.mariosl.com")
    contacto4 = ContactoEmpresa("EMPRESA", 678444653, "2023-03-07",
    "Marta S.L.", "Empresa con 500.000 trabajadores",
"www.mariosl.com")
    contacto_buscado1 = "Mario S.L"
    contacto_buscado2 = "Mario"

    arraycontacto_buscado = [contacto1, contacto2, contacto3,
contacto4]
    self.asssertTrue(contacto_buscado1 in arraydefecto)
    self.assertFalse(contacto_buscado2 in arraydefecto)
```

En caso de que el contacto buscado esté en la agenda (que tenga el mismo nombre que el introducido por el usuario) lo borra, en caso contrario la agenda se queda tal y como estaba.