Agenda de contactes.

Alumna: **Sofía Gràcia Mascarell.**

23/05/2024

Link a GitHub: https://github.com/SofiaGracia/AgendaDeContactes

Índex

2. Introducció

El projecte "Agenda de contactes" es presenta com una ferramenta fonamental en l'àmbit de la gestió d'informació personal i professional. Desenvolupat amb l'objectiu de proporcionar una solució eficient i accessible per a l'organització de contactes, este projecte busca oferir una aplicació intuïtiva i versàtil que satisfaça les necessitats dels usuaris en termes d'emmagatzematge i gestió de contactes.

A través d'una interfície amigable i funcionalitats clau, com la creació, consulta, modificació i eliminació de contactes, l'Agenda de contactes busca simplificar el procés de mantindre i accedir a la informació de contacte rellevant. A més, la implementació d'una base de dades robusta garantix la persistència de les dades, permetent als usuaris accedir a la seua informació en qualsevol moment i des de qualsevol dispositiu.

Este projecte busca explorar i aplicar conceptes fonamentals de programació, disseny d'interfícies d'usuari i maneig de bases de dades. En resum, el projecte "Agenda de contactes" oferix una solució pràctica i educativa per a la gestió de contactes, brindant als usuaris una ferramenta útil i efectiva per a organitzar la seua xarxa de contactes de manera eficient i sistemàtica.

3. Ferramentes i mètodes

- **Descripció de les ferramentes i mètodes utilizades en el projecte:**

En primer lloc, es va optar per utilitzar **Visual Studio Code** com a entorn de desenvolupament integrat (IDE) a causa de les seues funcionalitats avançades i suport per a una àmplia gamma de llenguatges de programació. L'IDE s'utilitzà en un entorn **Ubuntu LTS** Jammy Jellyfish **24.04.3** que garantix l'estabilitat de l'entorn de desenvolupament i la compatibilitat amb les ferramentes utilitzades.

Per a la planificació i disseny del projecte, es va fer ús de la ferramenta **Umbrello**, que proporciona capacitats de modelatge visual per a la

representació de diagrames de classes, casos d'ús i altres artefactes de disseny. Esta ferramenta permet realitzar una organització prèvia i estructurada de les classes i casos d'ús que conformarien l'aplicació, facilitant així la comprensió i comunicació del disseny del programari.

Quant a les biblioteques i dependències utilitzades en el projecte, es va optar per integrar **biblioteques estàndard de Python** juntament amb aquelles específiques per al desenvolupament d'interfícies gràfiques d'usuari (GUI) i l'accés a bases de dades:

- pip (gestor de paquets a Python).
- Python 3.
- Tkinter (paquet GUI per a Python).
- pymongo (client de Mongo DB per a Python).

Pel que fa a la base de dades, es situa en un contenedor docker amb imatge de mongodb, conectat al port 27017 en el host: El port 27017 de la meua máquina host està amb connexió amb el port 27017 del servidor de MongoDB. Per tant també ha sigut necessari instalar:

- Docker
- Docker-compose
- Imatge: mongo:latest.

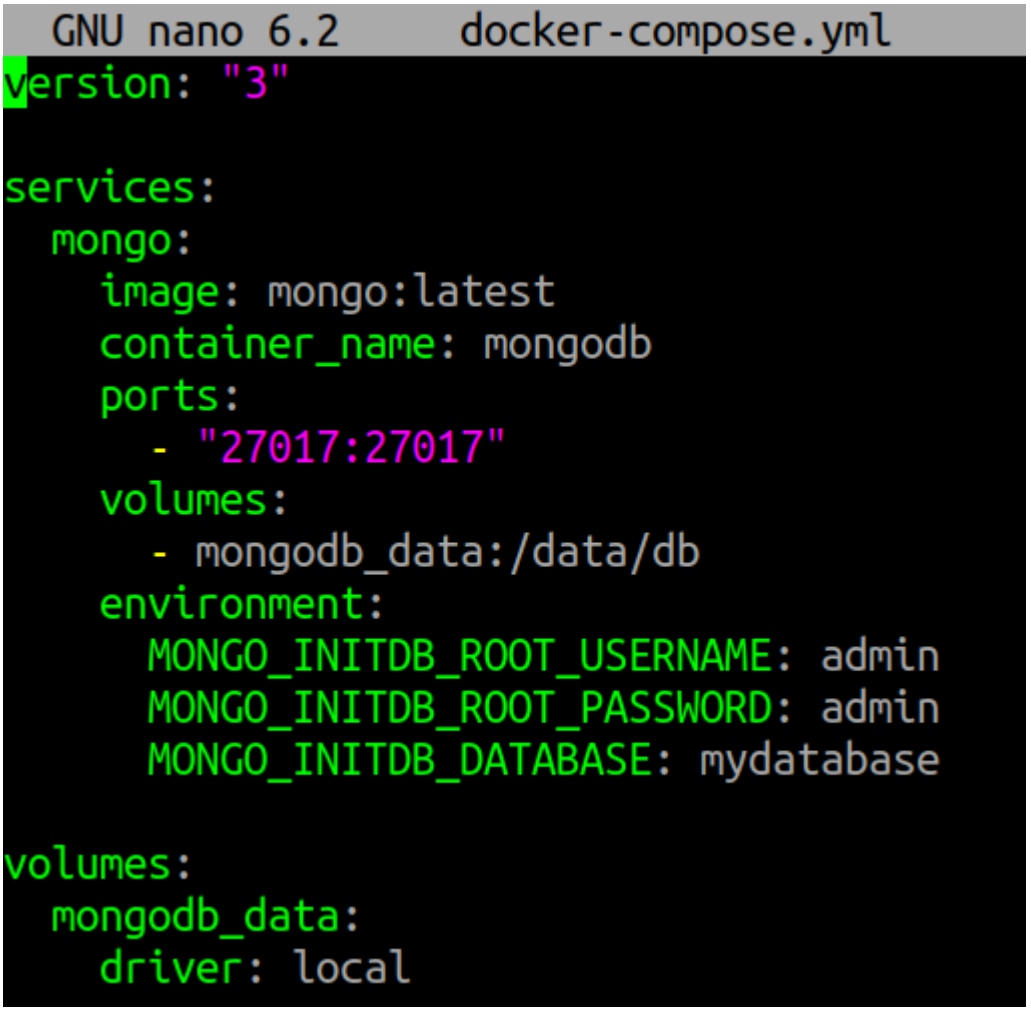

Fig. 1. Contingut del fitxer docker-compose en yaml que ens facilita l'arrancada d'un contenidor amb MongoDB.

Així mateix també s'han fet ús de ferramentes gráfiques per a gestionar la base de dades com MongoDB Compass.

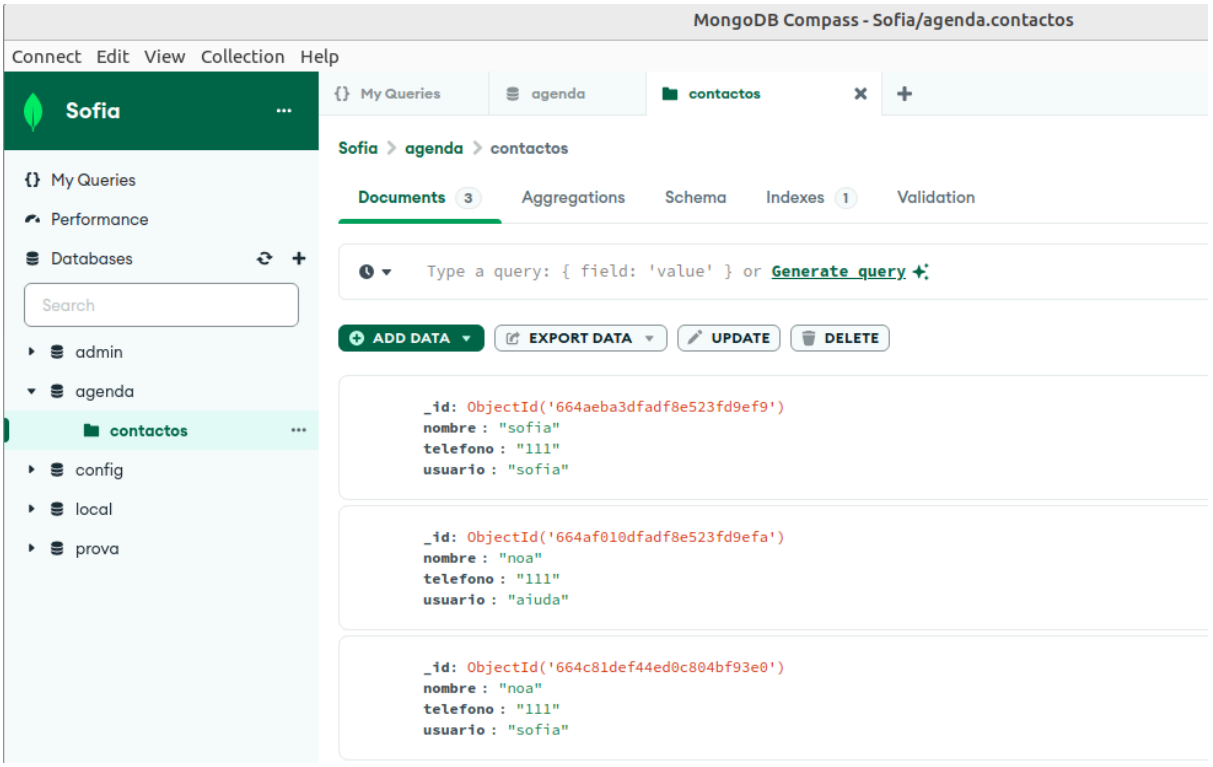

Fig. 2. L'ús de ferramentes gràfiques faciliten la gestió de la base de dades.

Finalment, però no menys important, es va fer ús de **ChatGPT**, una ferramenta d'intel·ligència artificial desenvolupada per OpenAI, que va proporcionar assistència en la generació de text i la resolució d'alguns dubtes durant el procés de desenvolupament i documentació del projecte.

4. Perspectiva Estàtica

4.1. E/R (Entitat-Relació)

Diagrama que representa l'estructura de les entitats i les seues relacions:

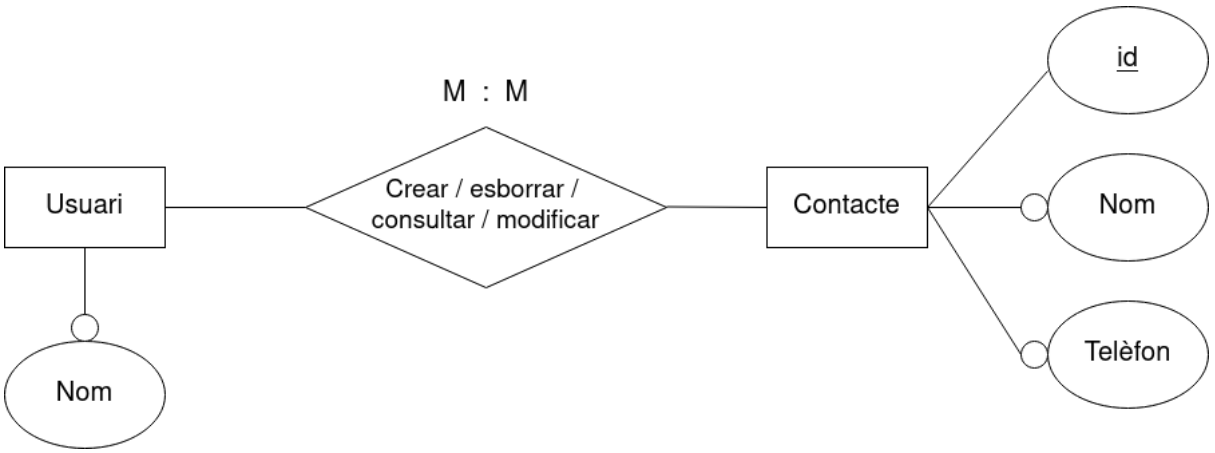

Fig. 3. Model E/R de Agenda de Contactes.

4.2. Pas a taules

USUARI = nom

VNN: nom

CONTACTE = id + nom + telèfon

VNN: id, nom, telèfon

CREAR_CONTACTE = contacteID + usuari + nom + telèfon

C. Ali: contacteID -> CONTACTE (id)

usuari -> USUARI (nom)

4.3. CRUD: Operacions bàsiques per a la gestió

A MongoDB no tindrem taules, sino una col·lecció amb documents clau-valor que tindran un identificador:

```
agenda> db.contactos.findOne({usuario:"sofia"})
{
  _id: ObjectId('664aeba3dfadf8e523fd9ef9'),
  nombre: 'sofia',
  telefono: '111',
  usuario: 'sofia'
}
agenda> db.contactos.findOne({telefono:"111"})
{
```

```
  _id: ObjectId('664aeba3dfadf8e523fd9ef9'),
  nombre: 'sofia',
  telefono: '111',
  usuario: 'sofia'
}
```

4.3.1. DDL (Data Definition Language)

S'utilitza el següent codi per a poder conectar-se i utilitzar la base de dades "agenda":

```
self.cliente = MongoClient("localhost", 27017, username="usuario",
password="password_seguro", authSource="agenda")
```

Definim la base de dades que utilitzarem:

```
self.db = self.cliente["agenda"]
```

I la col·lecció que modificarem i consultarem:

```
self.collection = self.db["contactos"]
```

El codi que modifica la base de dades al inserir un nou contacte és el següent:

```
self.collection.insert_one({"nombre": nombre, "telefono": telefono,
"usuario": self.nombre_usuario})
```

4.3.2. DML (Data Manipulation Language)

Exemples de codi que manipulen les dades en la base de dades.

4.3.2.1. Borrar un contacte

El mètode **borrar_contacto()** executa el següent codi per tal de borrar un document de la col·lecció:

```
self.collection.delete_one({"nombre": nombre, "usuario":
self.nombre_usuario})
```

4.3.2.2. Modificació del telèfon d'un contacte

En el mètode **modificar_telefono()** utilitza updateOne(), per a modificar i actualitzar els canvis d'un contacte:

```
self.collection.update_one({"nombre": nombre, "usuario":
self.nombre_usuario}, {"$set": {"telefono": nuevo_telefono}})
```

4.4. DQL (Data Query Language)

Amb findOne() podem consultar un document pel criteri que especifiquem, l'utilitzen totes les funcions per tal de consultar si un contacte existeix abans de crear-lo o modificar-lo:

```
contacto_existente = self.collection.find_one({"nombre": nombre, "usuario":
self.nombre_usuario})
```

4.5. DCL (Data Control Language)

A diferència de sistemes de bases de dades relacionals tradicionals com SQL, MongoDB utilitza un enfocament diferent per a la gestió de la seguretat i el control d'accés. Per tal que els usuaris puguen accedir a la base de dades hem creat un usuari amb permisos d'escritura i lectura:

```
users: [
      {
      _id: 'agenda.usuario',
      userId: UUID('724dd574-50c8-42e5-9dcf-d50e0c3f00b1'),
      user: 'usuario',
      db: 'agenda',
      roles: [ { role: 'readWrite', db: 'agenda' } ],
      mechanisms: [ 'SCRAM-SHA-1', 'SCRAM-SHA-256' ]
      }
```

```
],
ok: 1
```

5. Perspectiva Dinàmica

5.1. Sketch

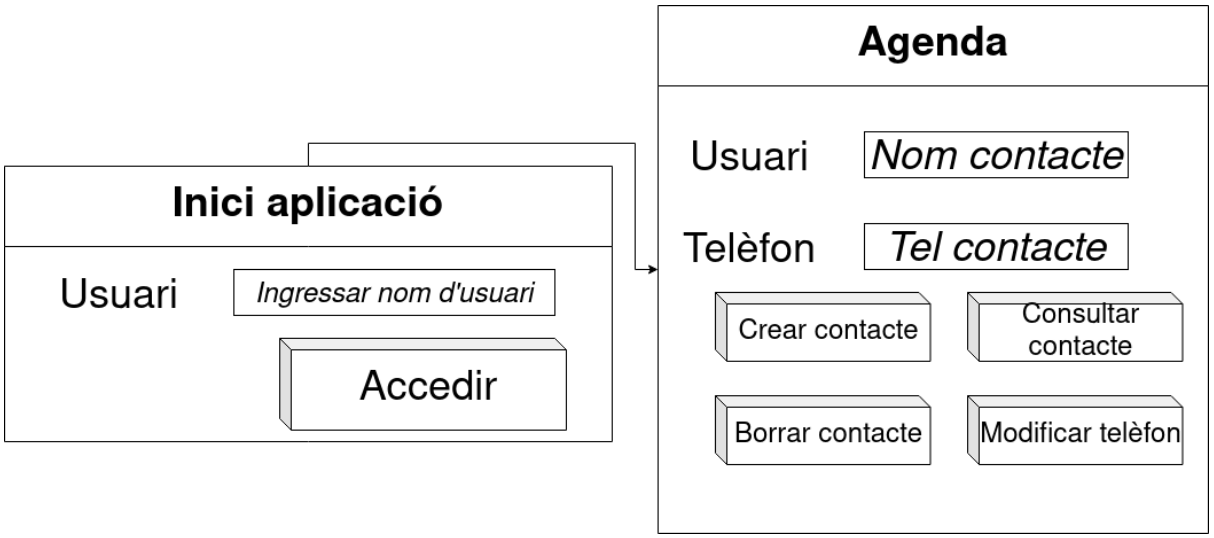

Fig. 4. Sketch inicial de l'aplicació.

5.2. Casos d'ús (Mètodes)

Descripció dels casos d'ús i com s'implementen.

Cas d'ús	Ingressar a l'aplicació.	**Identificador:** 1
Actors	Usuari	
Tipus	Primari	
Referències	Per executar l'aplicació necessitarem: - Python - Mòdul Tkinter - SQLite	
Precondició	Executar l'aplicació	
Postcondició	Ingresar el nom de l'usuari a utilitzar l'aplicació	
Descripció	L'usuari pot iniciar sessió en l'aplicació donant el seu nom	
Resum	Si l'usuari ingressa un nom d'usuari vàlid, s'obrirà la finestra de l'agenda.	

	Si l'usuari intenta ingressar sense proporcionar un nom d'usuari, l'aplicació mostrarà un missatge d'error i sol·licitarà a l'usuari que ingresse un nom d'usuari vàlid.

Curs normal 1: (Actor: Usuari)

1. L'usuari obri l'aplicació.
2. L'usuari ingressa el seu nom d'usuari.
3. L'usuari fa clic en el botó "Ingressar".
4. El sistema verifica que s'haja ingressat un nom d'usuari.
5. Si el nom d'usuari és vàlid, s'obri la finestra de l'agenda.

Curs altern 1 : (Actor: Usuari)

1. L'usuari obri l'aplicació.
2. L'usuari no ingressa el seu nom d'usuari.
3. L'usuari fa clic en el botó "Ingressar".
4. El sistema verifica que s'haja ingressat un nom d'usuari.
5. Es mostra el missatge d'error.

Cas d'ús	Crear contacto	**Identificador:** 2
Actors	Usuari	
Tipus	Primari	
Referències	Per executar l'aplicació necessitarem: - Python - Mòdul Tkinter - SQLite	
Precondició	- Ingressar a l'aplicació - Introduir el nom i el telèfon del contacte.	
Postcondició	Prémer opció per a gestionar contactes o tancar agenda.	
Descripció	Permet a l'usuari realitzar accions relacionades amb la gestió de contactes, com crear, consultar, modificar i eliminar contactes.	
Resum	L'opció permet a l'usuari registrar nom i telèfon d'un contacte. Si l'usuari intenta crear un contacte sense proporcionar informació obligatòria, com el nom, l'aplicació mostra un missatge d'error i sol·licita la informació que manca.	

Curs normal 2: (Actor: Usuari)

1. L'usuari obri la finestra de l'agenda.
2. L'usuari proporciona la informació necessària per a crear el contacte (nom i telèfon).
3. El sistema realitza l'acció corresponent i es registren les dades en la base de dades.

5. El sistema mostra un missatge indicant el resultat de l'acció.

Curs altern 2.1 : No es proporciona les dades (Actor: Usuari)

1. L'usuari obri la finestra de l'agenda.
2. L'usuari no proporciona nom i teléfon.
3.L'usuari selecciona una acció entre crear, consultar, modificar o eliminar contacte.
4. El sistema mostra el missatge d'error corresponent resultant de l'acció fallida.

Curs altern 2.2 : Es proporcionen dades existents (Actor: Usuari)

1. L'usuari obri la finestra de l'agenda.
2. L'usuari no proporciona nom i teléfon.
3.L'usuari selecciona l'opció de crear contacte.
4. El sistema mostra el missatge de que ja existeix el contacte.

Cas d'ús	Consultar contacte	**Identificador:** 3
Actors	Usuari	
Tipus	Primari	
Referències	Per executar l'aplicació necessitarem: - Python - Mòdul Tkinter - SQLite	
Precondició	- Ingressar a l'aplicació - Introduir el nom del contacte.	
Postcondició	Prémer opció per a gestionar contactes o tancar agenda.	
Descripció	Permet a l'usuari realitzar accions relacionades amb la gestió de contactes, com crear, consultar, modificar i eliminar contactes.	
Resum	L'opció permet consultar el telèfon d'un contacte a partir del nom introduit.	

Curs normal 3: (Actor: Usuari)
1. L'usuari obri la finestra de l'agenda.
2. L'usuari proporciona el nom del contacte..
3. El sistema mostra el número de telèfon del contacte.

Curs altern 3.1 : No es proporciona les dades (Actor: Usuari)

1. L'usuari obri la finestra de l'agenda.
2. L'usuari no proporciona nom.

4. El sistema mostra el missatge d'error corresponent resultant de l'acció fallida.

Curs altern 3.2 : Es proporciones dades inexistents (Actor: Usuari)

1. L'usuari obri la finestra de l'agenda.
2. L'usuari proporciona el nom d'un contacte que no existeix.
4. El sistema mostra el missatge d'error corresponent resultant de l'acció fallida.

Cas d'ús	Borrar contacte	**Identificador:** 4
Actors	Usuari	
Tipus	Primari	
Referències	Per executar l'aplicació necessitarem: - Python - Mòdul Tkinter - SQLite	
Precondició	Ingressar a l'aplicació	
Postcondició	Prémer opció per a gestionar contactes o tancar agenda.	
Descripció	Permet a l'usuari realitzar accions relacionades amb la gestió de contactes, com crear, consultar, modificar i eliminar contactes.	
Resum	L'opció permet borrar el contacte i el seu telèfon de la base de dades.	

Curs normal 4: (Actor: Usuari)
1. L'usuari obri la finestra de l'agenda.
2. L'usuari proporciona el nom del contacte.
3.L'usuari selecciona "borrrar contacte".
4. El sistema realitza l'acció corresponent en la base de dades.
5. El sistema mostra un missatge indicant el resultat de l'acció.

Curs altern 4.1 : No es proporciona les dades (Actor: Usuari)

1. L'usuari obri la finestra de l'agenda.
2. L'usuari no proporciona nom i teléfon.
3.L'usuari selecciona l'opció "borrar contacte".
4. El sistema mostra el missatge d'error corresponent resultant de l'acció fallida.

Curs altern 4.2 : Es proporciones dades inexistents (Actor: Usuari)

1. L'usuari obri la finestra de l'agenda.
2. L'usuari proporciona el nom d'un contacte que no existeix.

4. El sistema mostra el missatge d'error corresponent resultant de l'acció fallida.

Cas d'ús	Modificar contacto	**Identificador:** 5
Actors	Usuari	
Tipus	Primari	
Referències	Per executar l'aplicació necessitarem: - Python - Mòdul Tkinter - SQLite	
Precondició	Ingressar a l'aplicació	
Postcondició	Prémer opció per a gestionar contactes o tancar agenda.	
Descripció	Permet a l'usuari realitzar accions relacionades amb la gestió de contactes, com crear, consultar, modificar i eliminar contactes.	
Resum	L'opció permet modificar el telèfon del contacte de la base de dades.	

Curs normal 5: (Actor: Usuari)
1. L'usuari obri la finestra de l'agenda.
2. L'usuari proporciona el nom i el nou telèfon del contacte.
3.L'usuari selecciona l'acció de modificar el contacte.
4. El sistema realitza l'acció corresponent en la base de dades.
5. El sistema mostra un missatge indicant el resultat de l'acció.

Curs altern 5.1 : No es proporciona les dades (Actor: Usuari)

1. L'usuari obri la finestra de l'agenda.
2. L'usuari no proporciona nom i teléfon.
3.L'usuari selecciona l'acció de modificar el contacte.
4. El sistema mostra el missatge d'error corresponent resultant de l'acció fallida.

Cas d'ús	Sortir de l'aplicació	**Identificador:** 6
Actors	Usuari	
Tipus	Primari	
Referències	Per executar l'aplicació necessitarem: - Python - Mòdul Tkinter - SQLite	
Precondició	Ingressar a l'aplicació.	
Postcondició	L'aplicació es tanca correctament	
Descripció	Permet a l'usuari tancar l'aplicació de manera segura.	

Resum	No s'identifiquen excepcions específiques en este cas d'ús, ja que el tancament de l'aplicació es considera una acció estàndard i no està subjecte a condicions especials.

Curs normal 6: (Actor: Usuari)

1. L'usuari fa clic en el botó de tancar finestra.
2. El sistema tanca la connexió a la base de dades.
3. El sistema tanca la finestra i finalitza l'aplicació.

Curs altern 6.1: (Actor: Usuari)

1. L'usuari no fa clic en el botó de tancar finestra.
2. El sistema no tanca la connexió a la base de dades.
3. No finalitza l'aplicació.

6. Conclusions

En resum, el projecte ha demostrat la importància d'explorar i aprofitar les biblioteques i ferramentes disponibles en el desenvolupament de programari. Si bé l'aplicació desenvolupada no és perfecta, ha sigut una oportunitat valuosa per a aplicar i combinar coneixements en programació i bases de dades.

A través de l'experiència pràctica d'este projecte, s'ha evidenciat que la incorporació de biblioteques específiques pot millorar significativament la funcionalitat i eficiència de les aplicacions desenvolupades. En este cas, l'ús de biblioteques com Tkinter per a la interfície gràfica i pymongo per a la gestió de la base de dades ha permés crear una segona pràctica de creació d'una aplicació.

Encara que sabem que el projecte coixeja en molts aspectes, és un començament d'un viatge d'aprenentatge continu en el camp de la informàtica, on la pràctica i l'experimentació són clau per al creixement professional.

7. Annex

Imatge de l'aplicació en execució:

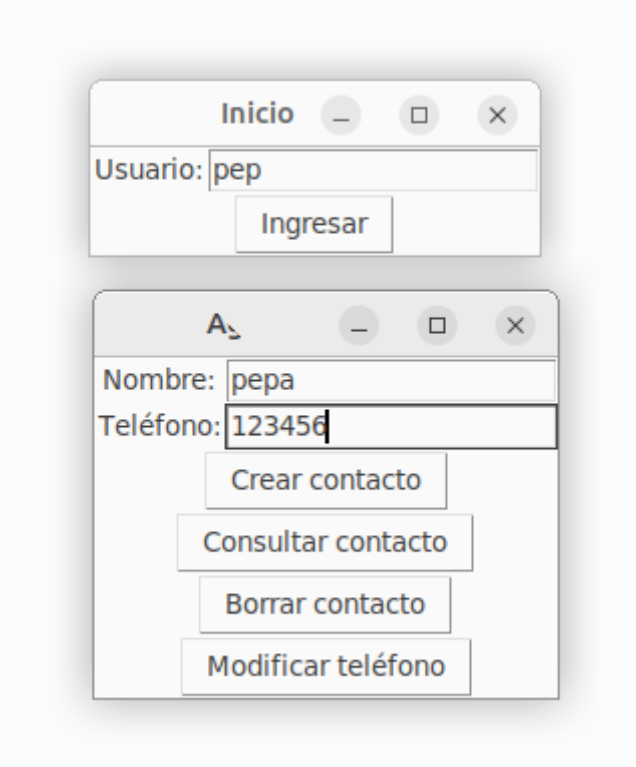

Fig. 5. Pantalla principal de l'aplicació.

8. Webgrafia

Webgrafia:

1. Documentació oficial de Python: https://docs.python.org/
2. Documentació oficial de Tkinter: https://docs.python.org/3/library/tkinter.html
3. Documentació oficial de Docker Compose: Docker Docs
4. Documentació oficial de MongoDB: MongoDB: The Developer Data Platform | MongoDB
5. Documentació oficial de MongoDB Compass: Download and Install Compass - MongoDB
6. Documentació oficial de Visual Studio Code: Visual Studio: IDE y Editor de código para desarrolladores de ...
7. Umbrello UML Modeller: https://umbrello.kde.org/
8. Link del projecte a GitHub: https://github.com/SofiaGracia/AgendaDeContactes