PROJECTE II BASE DE DADES “XIRINGUITO DB”

Autor: Cristian Espinosa Vicens

Data:23/05/2024

1DAM

https://github.com/Crisespvic/Microprojecte-II-XiringuitoDB

ÍNDEX

1.Introducció:

En context, aquest projecte de base de dades es crea per la necessitat i voluntat del xiringuito “Xupitet i becaeta” de diferenciar-se de la resta de xiringuitos que existeixen al voltant, donant un servici molt més ràpid, personal i de qualitat al seu client.

En aquest document, es presenta el desenvolupament d'un sistema de gestió per al xiringuito que facilite el control de les tumbones disponibles i la informació dels clients que les utilitzen. Aquest sistema està dissenyat per facilitar i optimitzar les operacions diàries del negoci i millorar l'experiència dels clients.

El projecte es centra en el disseny, la creació i implementació d'una base de dades relacional que permet la gestió eficient de les tumbones, així com l’accés a la informació dels clients associats al lloguer de cada tumbona.

2.Eines i mètodes

Aquest projecte comença amb el dibuix del diagrama Entitat- Relació (ER), seguit del pseudocodi per convertir-ho en col.leccións i posterior transformació del pseudocodi a codi Bynary-JSON.

Per a la part de transformació a codi, creació de la base de dades i implementació de l’interfície s’ha optat per usar el sistema operatiu Linux en la versió Ubuntu 20.04.

La base de dades està creada en un servidor de MongoDB instal.lat en un contenedor en docker.

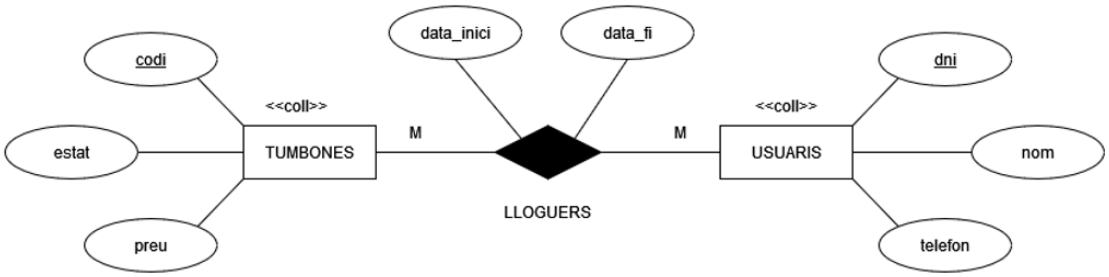
Entre els llenguatges per als quals MongoDB ofereix connectors oficials es troba Python, que es el que s’ha utilitzat tant per crear la base de dades, com per crear una interfície gràfica amb la que poder interactuar amb ella. L’IDE utilitzat per a l’implementació de tot el codi Python es Visual Studio Code.

A més, cal fer ús de les llibreries TKinter de Python que és la que permet construir les diferents finestres i botóns de l’interfície gràfica y pymongo que facilita l’interacció amb MongoDB.

3. Perspectiva estàtica

3.1. ER (Entitat-Relació)

A continuació es mostra el diagrama ER (Entitat-Relació) on es pot apreciar la relació N:M (molts a molts) entre les col.leccions TUMBONES i USUARIS. Aquesta relació acaba creant una col.lecció extra en la base de dades anomenada LLOGUERS.



3.2. Patró de document d'associació (association document pattern)

Aquest patró permet una major flexibilitat i escalabilitat en comparació amb les bases de dades relacionals tradicionals, ja que es poden afegir noves relacions sense necessitat de modificar l'esquema de la base de dades.
S'ha optat per utilitzar aquest patró ja que la base de dades necessita d'una col.lecció extra per enmagatzemar l'asociació dels registres entre les tumbones i els usuaris.
A continuació podem veure l'interpretació en pseudocodi del diagrama ER.

Implementació en pseudocodi

Entitats:

- **TUMBONES**
  - Atributs: _id, preu, estat
- **USUARIS**
  - Atributs: _id, nom, telefon

Relació:

- **TUMBONES-USUARIS**
  - Atributs: _id, tumbona_id, usuari_id, data_inici, data_fi

4. CRUD

4.1. CREATE O DDL (Data Definition Language)

En MongoDB les bases de dades es creen de forma ímplicita quan es crea una col.lecció dins d'elles. DDL es refereix al codi utilitzat per definir y gestionar l'estructura de la base de dades, com crear la base de dades, eliminar-la, crear col.leccions i configurar índex.
A continuació es pot veure el codi utilitzat per aquest fi.

```
# Codi per crear les coleccions
    usuaris = db["usuaris"]
    tumbones = db["tumbones"]
    lloguers = db["lloguers"]
    users = db["users"]

# Codi per insertar les tumbones per defecte
tumbones_inicials = [{"_id": i, "Estat": "Lliure", "Preu": 40.00} for i in range(1, 21)]
```

4.2. READ O DQL (Data Query Language)

Consultar beneficis:

L'aplicació té l'opció (només per l'usuari administrador) de consultar els beneficis obtinguts. Per obtindre aquesta informació calen les següents intruccions.

```
# Es fa el recompte de tumbones que tenen registrada una data_fi

tumbones_cobrades = lloguers.count_documents({'Data_Fi': {'$ne': None}})


# Es selecciona el preu de la tumbona (Totes valen igual)

preu_tumbona = tumbones.find_one({}, {'Preu': 1})


# Calcula el benefici

benefici = tumbones_cobrades * preu_tumbona['Preu']
```

Consultar la quantitat de tumbones lliures i el codi de les que estan lliures i ocupades:

```
# Es fa el recompte de les tumbones que estan lliures
total_lliures = tumbones.count_documents({'Estat': 'Lliure'})

# Es selecciona el codi de les tumbones amb estat Lliure
codi_lliures = tumbones.find({'Estat': 'Lliure'}, {'_id': 1})
codisL = ", ".join(str(codi['_id']) for codi in codi_lliures)

# Es selecciona el codi de les tumbones amb estat Ocupada
codi_ocupades = tumbones.find({'Estat': 'Ocupada'}, {'_id': 1})
codisO = ", ".join(str(codi['_id']) for codi in codi_ocupades)
```

Consultar les dades de l'usuari:

```
# Es selecciona el codi de les tumbones que no tenen data_fi
lloguer = lloguers.find_one({'Tumbona': codi, 'Data_Fi': None})

# Es selecciona el client que té llogada la tumbona
client_info = usuaris.find_one({'dni': lloguer['Usuari']})

# Es mostra l'informació del client
info_text = (
    f"Informació del client:\n"
    f"DNI: {client_info['dni']}\n"
    f"Nom: {client_info['nom']}\n"
    f"Telèfon: {client_info['telefon']}\n"
)
```

4.3. UPDATE O DML (Data Manipulation Language)

Llogar una tumbona:

```
# Si l'estat de la tumbona es Lliure
if tumbona['Estat'] == "Lliure":

# Es registren les dades del client
usuaris.update_one({'dni': dni}, {'$set': {'nom': nom, 'telefon': telefon}}, upsert=True)

# S'actualitza l'estat de la tumbona a Ocupada
tumbones.update_one({'_id': int(codi)}, {'$set': {'Estat': "Ocupada"}})

# I es registra la data en la que s'ha llogat la tumbona
lloguers.insert_one({'Usuari': dni, 'Tumbona': codi, 'Data_Inici': data_actual, 'Data_Fi': None})
```

Cobrar lloguer:

```
# Es busca el lloguer d'una tumbona determinada i el client que l'ocupa
lloguer = lloguers.find_one({'Tumbona': codi, 'Usuari': dni, 'Data_Fi': None})

# S'actualitza el document del lloguer registrant la data_fi
lloguers.update_one({'_id': lloguer['_id']}, {'$set': {'Data_Fi': data_fi}})

# S'actualitza l'estat de la tumbona a Lliure
tumbones.update_one({'_id': int(codi)}, {'$set': {'Estat': "Lliure"}})
```

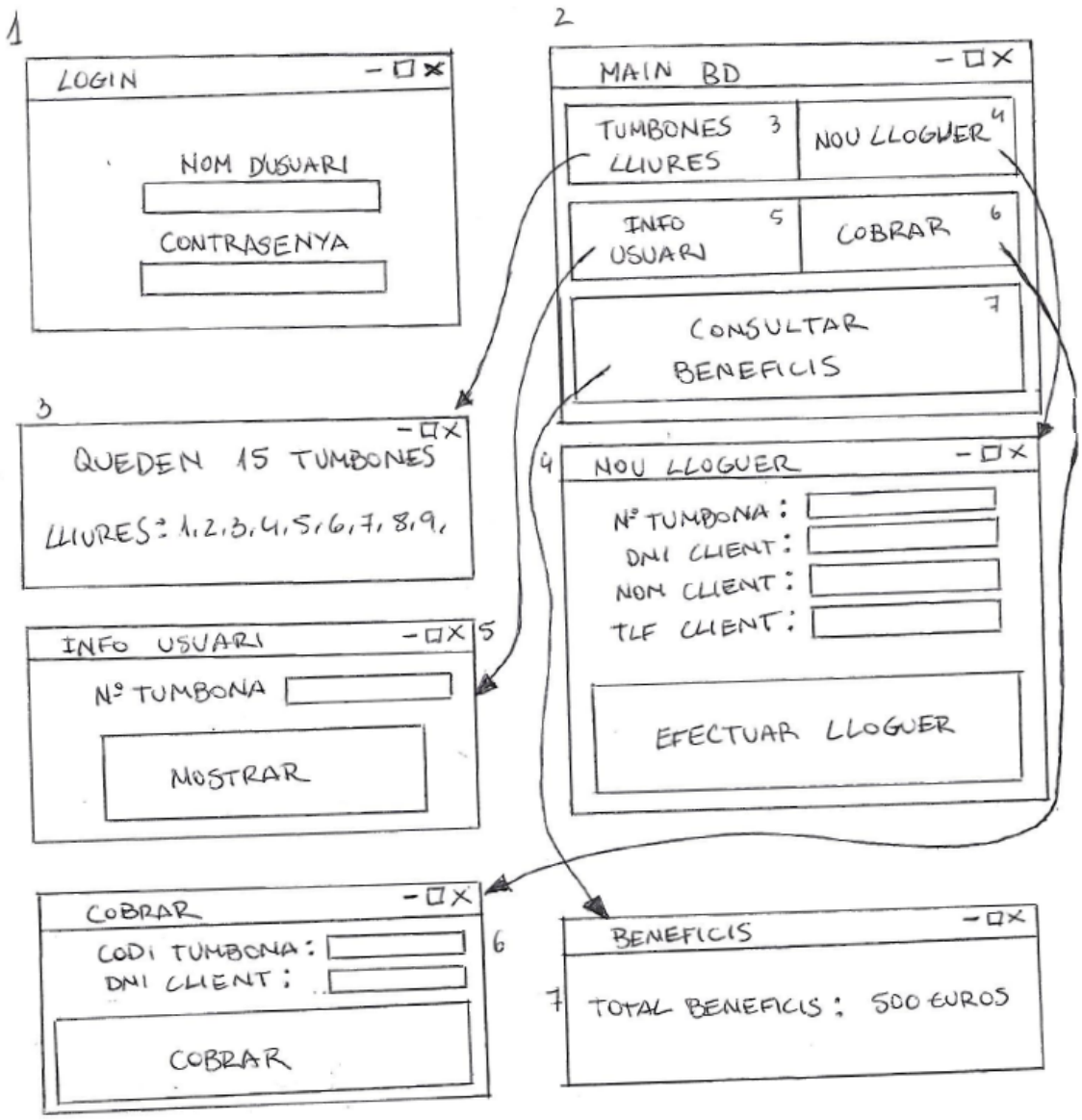
4.4. DCL (Data Control Language)

MongoDB no disposa d'un sistema de control d'accés integrat, però si permet configurar autenticació i autorització per controlar l'accés a la base de dades.
Per controlar l'accés a la base de dades s'ha implementat aquesta funcionalitat mitjançant codi Python i la consulta d'una col.lecció extra existent a la base de dades, que guarda els diferents usuaris amb contrasenya i els seus permissos.
Així l'aplicació també inclou un control d'usuaris amb contrasenya a l'hora d'accedir i uns permissos que permeten fer ús d'unes opcións o d'altres dins de l'aplicació depenent si eres usuari "admin" o "user". L'usuari "**admin**" amb contrasenya **1234** per defecte pot accedir a totes les opcións, mentre que l'usuari "**user**" amb contrasenya **0000** només pot fer ús de les funcions per consultar les tumbones lliures i fer nous lloguers, però no tenen permís per cobrar, consultar informació dels clients ni tampoc consultar els beneficis.

```
# Codi per insertar els users per accedir al login per defecte
 users_inicials = [
       {"_id": 1, "NomUsuari": "admin", "Contrasenya": "1234",
"Permissos": "Administrador"},
       {"_id": 2, "NomUsuari": "user", "Contrasenya": "0000",
"Permissos": "Usuari"}
    ]
```

5. Perspectiva dinámica

5.1. Sketch complet:



5.2. Escenes, estats i casos d'ús de l'aplicació:

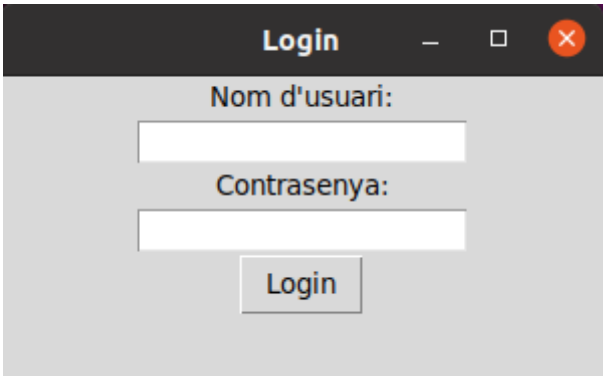


Fig.1 Login

Abans de poder accedir a l'aplicació, s'ha d'insertar el nom d'usuari i la contrasenya corresponent en la pantalla de Login. Hi han dos tipus d'usuaris creats:

admin amb contrasenya **1234,** té permís d'accés i ús a tota l'aplicació.

user amb contrasenya **0000,** només té permís d'accés i ús per consultar les tumbones lliures i fer nous lloguers.

Si l'usuari o la contrasenya insertada no són les correctes no es pot accedir, però es pot intentar introduïr les credencials totes les vegades que es vullga.

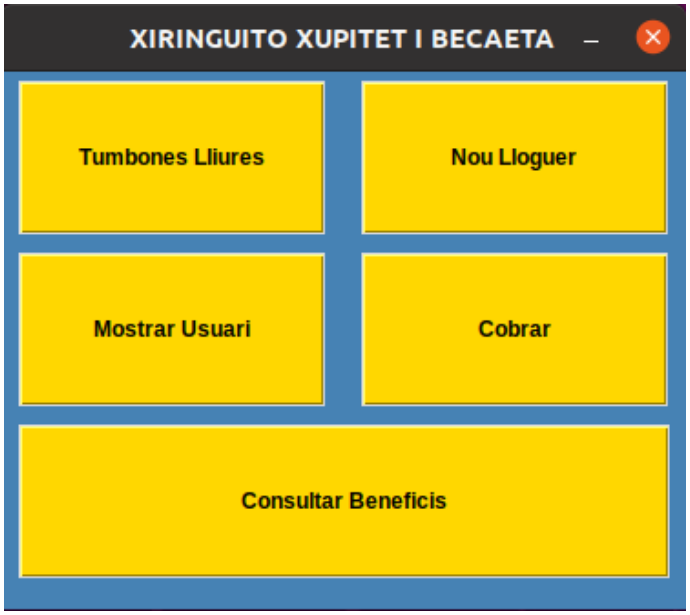


Fig.2 Menú principal

Aquesta és la finestra principal de l'aplicació des d'on es pot accedir a totes les funcionalitats.

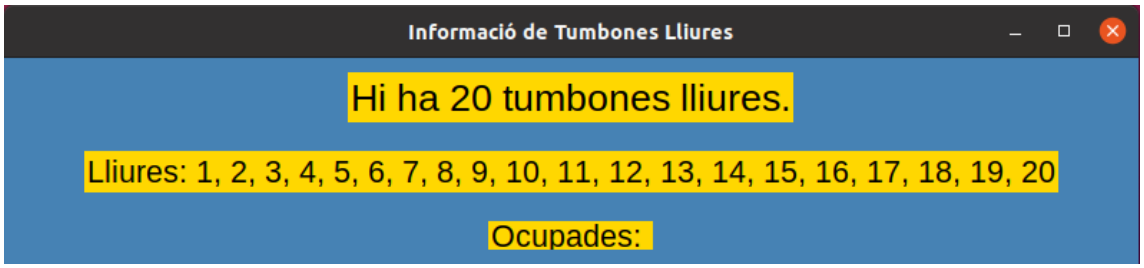


Fig.3 Finestra d'informació de tumbones lliures

La primera de les funcionalitats de l'aplicació, ens mostra la quantitat de tumbones lliures, els codis de les que estàn lliures i els codis de les que estàn ocupades.

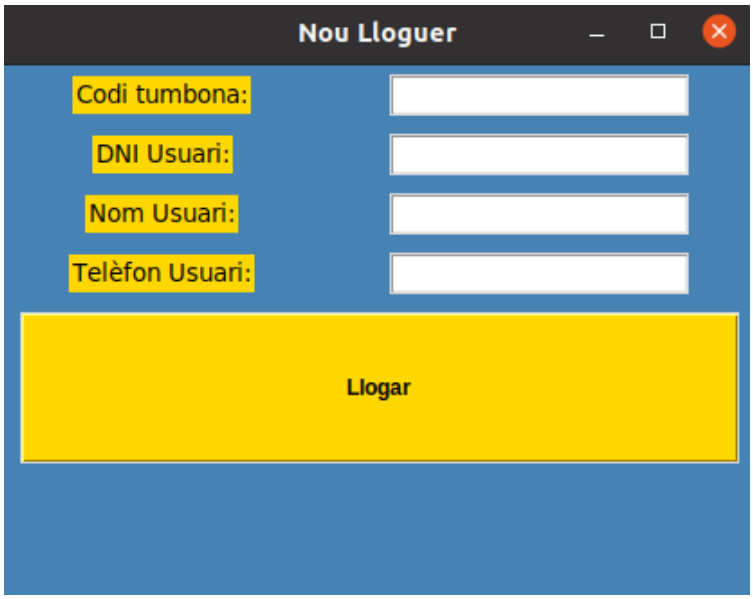


Fig.4 Finestra per fer nous lloguers

La segona funció permet realitzar el lloguer d'una tumbona, enregistrant les dades del client i la data i hora del lloguer.

Fig.5 Finestra per mostrar informació del client

La tercera funció permet (a l'usuari admin) consultar les dades del client que ocupa una tumbona determinada introduïnt el codi de la tumbona.

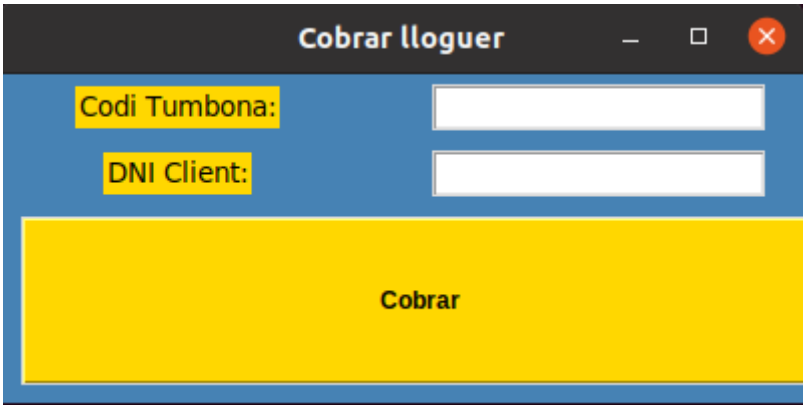


Fig.6 Finestra per cobrar

La quarta funció de l'aplicació permet (a l'usuari admin) efectuar el cobrament d'una tumbona introduïnt el codi de la tumbona i el dni del client.

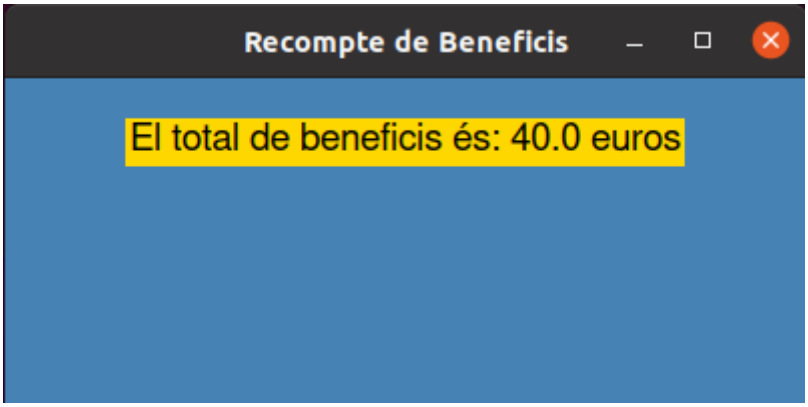


Fig.7 Finestra per mostrar els beneficis

La quinta i última funció mostra (a l'usuari admin) el benefici acumulat.

5.3. Especificació de casos d'ús

1. [Login]

Cas d'ús	Login	**Identificador:** CU01
Actors	Usuari (user i admin)	
Tipus	Primari	
Precondició	Executar l'aplicació	
Postcondició	Accés al Menú principal	
Descripció	L'usuari introdueix el nom d'usuari i contrasenya. Si aquests són correctes s'accedirà a l'aplicació amb distints permisos. Si no botarà un error.	

2. **[Consultar tumbones lliures]**

Cas d'ús	Consultar tumbones lliures	**Identificador:** CU02
Actors	Usuari (user i admin)	
Tipus	Primari	
Precondició	Login correcte	
Postcondició	Ninguna	
Descripció	Es mostrarà la quantitat de tumbones lliures i el codi de cadascuna d'elles.	

3. **[Llogar tumbona]**

Cas d'ús	Llogar tumbona	**Identificador:** CU03
Actors	Usuari (user i admin)	
Tipus	Primari	
Precondició	Login correcte	
Postcondició	Ninguna	
Descripció	S'introdueix el codi de la tumbona i les dades del client i es crea el registre en la base de dades. També s'actualitza l'estat de la tumbona. Si no s'introdueix el dni o el codi botarà un error.	

4. **[Mostrar dades usuari]**

Cas d'ús	Mostrar dades usuari	**Identificador:** CU04
Actors	Usuari (admin)	
Tipus	Primari	
Precondició	Login correcte amb usuari "admin"	
Postcondició	Ninguna	
Descripció	S'introdueix el codi de la tumbona i es mostren les dades del client que l'ocupa.	

5. **[Cobrar lloguer]**

Cas d'ús	Cobrar lloguer	**Identificador:** CU05
Actors	Usuari (admin)	
Tipus	Primari	
Precondició	Login correcte amb usuari "admin". Ha d'existir un lloguer fet i no cobrat.	
Postcondició	Ninguna	
Descripció	S'introdueix el codi de la tumbona i el dni del client, aleshores s'efectua el pagament, es registra la data i s'actualitza l'estat de la tumbona. Si les dades introduides són incorrectes botarà un error.	

6. **[Consultar beneficis]**

Cas d'ús	Consultar beneficis	**Identificador:** CU06
Actors	Usuari (admin)	
Tipus	Primari	
Precondició	Login correcte amb usuari "admin"	
Postcondició	Ninguna	
Descripció	Es mostra el benefici acumulat en euros.	

6. Conclusións

El projecte "XIRINGUITO DB" és una iniciativa per millorar la gestió i experiència dels clients del xiringuito "Xupitet i becaeta" mitjançant una base de dades relacional. A continuació, es presenten els resultats i possibles millores d'aquest projecte:

Resultats obtinguts:

1. **Disseny i Implementació de la Base de Dades:**
   - S'ha transformat l'estructura del sistema de gestió de tumbones amb una base de dades relacional en una base de dades dinàmica amb MongoDB.
   - Es va començar amb un diagrama Entitat-Relació (ER) i es va transformar a un pseudocodi anterior al codi Python i B-JSON.
   - La base de dades s'ha creat amb MongoDB utilitzant Python com a llenguatge de programació.
   - S'ha dissenyat una interfície gràfica amb TKinter per interactuar amb la base de dades.

2. **Funcionalitats Implementades:**
   - Llogar una tumbona: El sistema permet registrar lloguers associant clients a tumbones i actualitzant l'estat de les tumbones a "Ocupada".
   - Cobrar un lloguer: Es pot registrar el pagament d'una tumbona i actualitzar l'estat a "Lliure".
   - Consultar beneficis: L'usuari administrador pot obtenir informació sobre els beneficis acumulats mitjançant consultes a la base de dades.
   - Consultar tumbones lliures: Es pot verificar quines tumbones estan disponibles en temps real.
   - Ara també es pot consultar quines tumbones són les que estàn ocupades.
3. **Control d'Usuaris i Permissos:**
   - S'ha implementat un sistema d'autenticació amb control d'usuaris i permisos.
   - L'aplicació diferencia entre usuaris administradors i usuaris estàndard amb funcionalitats limitades.

Possibles millores:

- L'interfície gràfica és básica i molt millorable.
- Es podrien afegir més funcions, com per exemple registrar i cobrar les consumissions dels clients o poder insertar i eliminar els usuaris dels empleats.
- Ampliar les opcións de consulta en relació al que es consumeix o per controlar quin empleat ha fet un determinat servici.
- Poder reiniciar la quantitat de beneficis desde l'aplicació.

7. Bibliografia i webgrafia

- Apunts per a la introducció a MongoDB.
- Informació per a crear interfícies i botons en Python: https://docs.python.org/es/3/library/tkinter.html
- Com crear aplicacions en Python, mysql i tkinter: https://programacionfacil.org/blog/como-crear-aplicaciones-con-python-mysql-y-tkinter-1/
- Youtube
- ChatGpt