PROJECTE BASE DE DADES “XIRINGUITO DB”

Autor: Cristian Espinosa Vicens

Data:01/05/2024

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 pas a taules i posterior transformació de tot a codi SQL.

La base de dades es desenvolupa utilitzant el llenguatge SQL mitjançant SQLite i finalment es implementada amb Python per crear una interfície gràfica amb la que poder interactuar amb ella.

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 Windows 10.

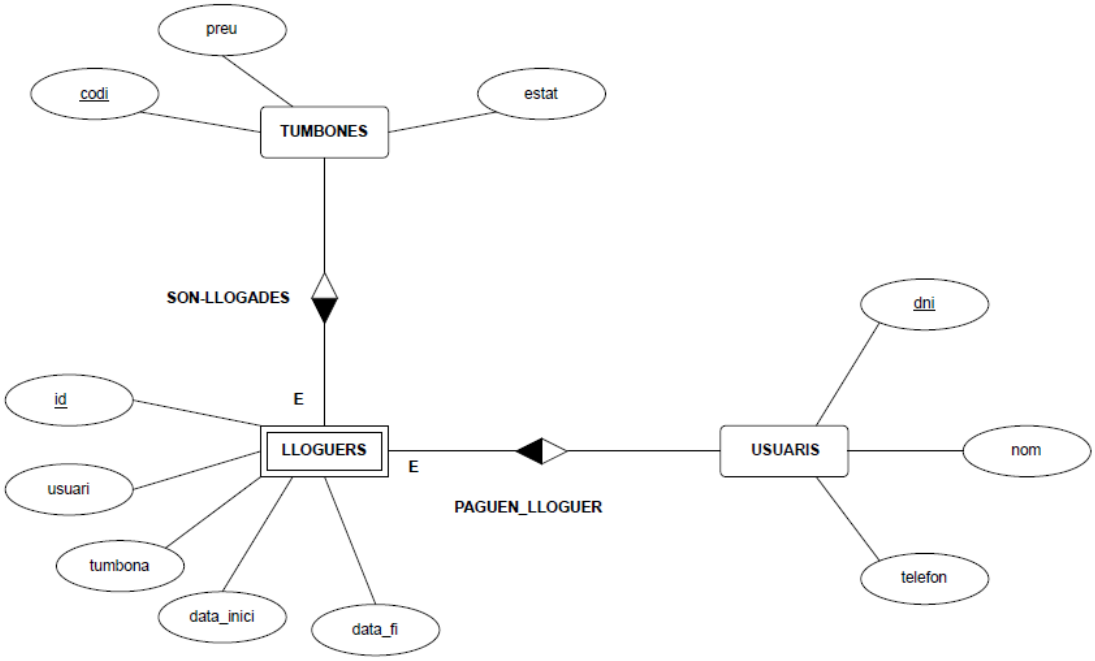
SQLite ens permet crear la petita base de dades, en llenguatge SQL, on es podrà guardar, modificar i consultar tota la informació necessària.

Per a la creació de l’interfície gràfica s’ha utilitzat l’IDE i editor de codi Visual Studio Code mitjançant el llenguatje de programació interpretat Python.

A més, cal fer ús de la llibreria TKinter de Python que és la que permet construir les diferents finestres i botóns de l’interfície gràfica.

3. Perspectiva estàtica

3.1. ER (Entitat-Relació)



3.2. Pas a taules

USUARIS = dni + nom + telefon

TUMBONES = codi + estat + preu

VNN: estat

LLOGUERS = id + usuari + tumbona + data_Inici + data_Fi

C.Ali: usuari → USUARIS(dni)

C.Ali: tumbona → TUMBONES(codi)

VNN: usuari

VNN: tumbona

3.3. DDL (Data Definition Language)

L'estructura bàsica de la base de dades està definida amb codi SQL mitjançant un mètode o funció de Python que s'encarrega de crear la base de dades en un fitxer anomenat xiringuito.db.

A continuació es detalla el codi del mètode que crea la base de dades amb les seues taules i atributs.

```python
def crear_bd():
    try:
        # Conexió a la base de dades (crea l'arxiu si no existeix)
        conn = sqlite3.connect('xiringuito.db')
        cursor = conn.cursor()

        # Crea la taula Usuaris (si no existeix)
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS Usuaris (
                Dni VARCHAR(9) PRIMARY KEY,
                Nom VARCHAR(30),
                Telefon INT
            )
        ''')
```

Fig.1 Definició del mètode crear_bd que crea l'arxiu xiringuito.db si no ha sigut creat abans. També es crea la taula Usuaris amb un dni que identifica l'usuari, un nom i telèfon.

```python
        # Crea la taula Tumbones (si no existeix)
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS Tumbones (
                Codi INT PRIMARY KEY,
                Estat VARCHAR(10) NOT NULL,
                Preu DECIMAL(6,2)
            )
        ''')
```

Fig.2 Es crea la taula Tumbones amb un codi identificatiu, un estat que ha de tindre obligatòriament i un preu.

```python
        # Crea la taula Lloguers (si no existeix)
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS Lloguers (
                id INTEGER PRIMARY KEY,
                Usuari VARCHAR(9) NOT NULL,
                Tumbona INT NOT NULL,
                Data_Inici TIMESTAMP DEFAULT NULL,
                Data_Fi TIMESTAMP DEFAULT NULL,
                FOREIGN KEY(Usuari) REFERENCES Usuaris(dni),
                FOREIGN KEY(Tumbona) REFERENCES Tumbones(codi)
            )
        ''')
```

Fig.3 Creació de la taula Lloguers amb un id, un usuari, tumbona, data_inici del lloguer i data_fi del lloguer.

En la taula “LLOGUERS” tant l’usuari com la tumbona son claus foràanees de les taules “USUARI” i “TUMBONA” respectivament. A més, com que la taula “LLOGUERS” té una relació d’existència respecte de les altres dos taules, l’usuari i la tumbona ha de tindre valor NO NULL, ja que no pot existir ningún lloguer sense usuari ni tumbona associats.

```
# Crea la taula Users (si no existeix)
cursor.execute('''
    CREATE TABLE IF NOT EXISTS Users (
        Id INTEGER PRIMARY KEY,
        NomUsuari VARCHAR(30) UNIQUE,
        Contrasenya VARCHAR(30),
        Permisos VARCHAR(30)
    )
''')
```

Fig.4 Creació de la taula “USERS”

La taula “USERS” no té relació amb cap altra, ja que realment està creada per al registre i control dels diferents usuaris i permisos.

Conté un identificador, un nom d’usuari (admin, user), una contrassenya i uns permisos.

```
# Inserta els usuaris inicials (admin i usuari) amb les seves contrasenyes i permisos
cursor.execute('INSERT OR IGNORE INTO Users (NomUsuari, Contrasenya, Permisos) VALUES (?, ?, ?)', ('admin', '1234', 'Administrador'))
cursor.execute('INSERT OR IGNORE INTO Users (NomUsuari, Contrasenya, Permisos) VALUES (?, ?, ?)', ('user', '0000', 'Usuari'))
```

Fig.5 Insert dels valors DEFAULT de la taula extra “USERS”.

3.4. DML (Data Manipulation Language)

Llogar una tumbona:

Quan un usuari pretén llogar una tumbona, se li otorga una tumbona determinada amb un codi i se li demanen les dades personals (dni, nom i telèfon) i es registrat a la taula Usuaris de la base de dades.

```
# Funció per a gestionar el lloguer de la tumbona
def llogar_tumbona():
    codi = codi_entry.get()
    dni = dni_entry.get()
    nom = nom_entry.get()
    telefon = telefon_entry.get()
```

```
# Inserció del nou lloguer amb clàusula INSERT OR IGNORE
cursor.execute('INSERT OR IGNORE INTO Usuaris (dni, nom, telefon) VALUES (?, ?, ?)', (dni, nom, telefon,))
```

Automàticament, la tumbona canvia el seu estat de “Lliure” a “Ocupada”.

```
cursor.execute('UPDATE Tumbones SET estat = "Ocupada" WHERE codi = ?', (codi,))
```

I es crea un nou registre de l’usuari amb la seua tumbona i la data en la qual s’ha efectuat el lloguer.

```
# Actualització de la taula Lloguers
data_actual = datetime.now().strftime('%d-%m-%Y %H:%M:%S')
cursor.execute('INSERT INTO Lloguers (Usuari, Tumbona, Data_Inici) VALUES (?, ?, ?)' , (dni, codi , data_actual,))
```

Cobrar lloguer:

Posteriorment, quan es vol fer efectiu el cobrament d’una tumbona, s’introdueix el codi de la tumbona i el dni del client.

```
def calcular_cobrament():
    codi = codi_entry.get()
    dni = dni_entry.get()
```

Així mateix l’estat de la tubona, que queda disponible de nou, torna a ser actualitzat a “Lliure” en la taula Tumbones i s’afegeix la data en la que finalitza el servici en la taula Lloguers.

```
# Actualitzar l'estat de la tumbona a "Lliure" i la data_Fi
cursor.execute('UPDATE Tumbones SET estat = "Lliure" WHERE codi = ?', (codi,))
data_actual = datetime.now().strftime('%d-%m-%Y %H:%M:%S')
cursor.execute('UPDATE Lloguers SET Data_Fi = ? WHERE Tumbona = ? AND Data_Fi IS NULL', (data_actual, codi,))
```

3.5. DQL (Data Query Language)

Consultar beneficis:

L’aplicació té l’opció (només per l’usuari administrador) de consultar els beneficis obtinguts. Per a obtindre aquesta informació es necessiten dos consultes diferents:

1. La quantitat de lloguers cobrats de la taula Lloguers

```
# Consulta per obtenir el recompte de beneficis
cursor.execute('SELECT COUNT(*) FROM Lloguers WHERE Data_Fi IS NOT NULL')
tumbones_cobrades = cursor.fetchone()
```

2. El preu de la tumbona de la taula Tumbones

```
cursor.execute('SELECT preu FROM Tumbones')
preu_tumbona = cursor.fetchone()
benefici = tumbones_cobrades[0] * preu_tumbona[0]
```

Consultar el codi i la quantitat de tumbones lliures:

També està l'opció de poder consultar si queden tumbones lliures i quines en específic són les que queden disponibles. Per a poder fer aquestes consultes es necessari fer ús de dos clàusules Select.

```
# Consulta per obtenir el recompte de tumbones lliures
cursor.execute('SELECT COUNT(*) FROM Tumbones WHERE estat = "Lliure"')
total_lliures = cursor.fetchone()[0]

# Consulta per obtenir el codi de les tumbones lliures
cursor.execute('SELECT codi FROM Tumbones WHERE estat = "Lliure"')
codi_lliures = cursor.fetchall()
codis = ", ".join(str(codi[0]) for codi in codi_lliures)
```

Consultar les dades de l'usuari:

Si en algún moment es vol saber la informació personal del client que està fent ús d'una o varies tumbones per donar-li un servici més personal o per algún altre motiu, es pot fer mitjançant l'opció mostrar_usuari. Per a fer aquesta consulta de la taula Usuaris, s'ha de fer el SELECT de la taula Lloguers que es la que relaciona els usuaris amb el codi de les tumbones.

```
# Obtenir el DNI de l'usuari que ocupa la butaca a partir de la taula Lloguers
cursor.execute('''
    SELECT Usuaris.dni, Usuaris.nom, Usuaris.telefon
    FROM Lloguers
    JOIN Usuaris ON Lloguers.Usuari = Usuaris.dni
    WHERE Lloguers.Tumbona = ? AND Lloguers.Data_Fi IS NULL
''', (codi,))

dades_usuari = cursor.fetchone()
```

3.6. DCL (Data Control Language)

En aquest apartat no s'ha fet ús de codi SQL per a crear usuaris, concedir permisos amb la clàusula GRANT, ni tampoc revocar-los amb la clàusula REVOKE, ja que el gestor SQLite no permet el seu ús. De totes formes si que s'ha implementat aquesta funcionalitat mitjançant codi Python i la consulta d'una taula 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.

```python
def validar_login():
    username = username_entry.get()
    password = password_entry.get()

    # Validar les credencials amb la base de dades
    if validar_credencials(username, password):
        # Accés permès; tanca la finestra de login i mostra la finestra principal
        login_window.destroy()
        mostrar_finestra_principal(username)  # Passa el nom d'usuari a la finestra principal
    else:
        messagebox.showerror("Error", "Credencials incorrectes")
```

Fig.6 Login

```python
# Crea la taula Users (si no existeix)
cursor.execute('''
    CREATE TABLE IF NOT EXISTS Users (
        Id INTEGER PRIMARY KEY,
        NomUsuari VARCHAR(30) UNIQUE,
        Contrasenya VARCHAR(30),
        Permisos VARCHAR(30)
    )
''')
```

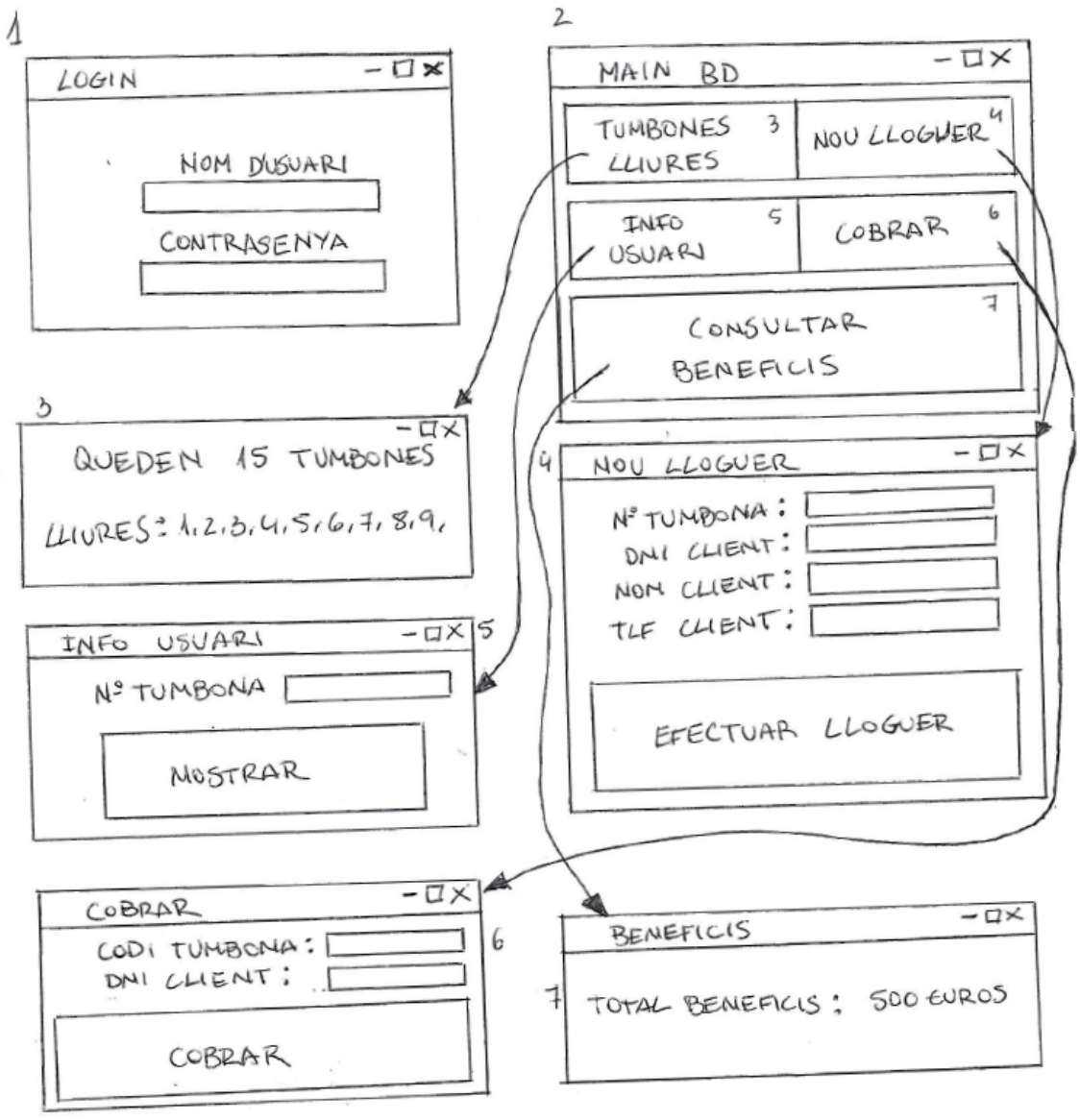
Fig.7 Taula extra "Users"

```python
cursor.execute('SELECT Permisos FROM Users WHERE NomUsuari = ?', (username,))
resultat = cursor.fetchone()
```

Fig.8 Consulta dels permisos de l'usuari

4. Perspectiva dinámica

Sketch:



Enumeració dels diferents sketchs i casos d'ús.

Especificació de casos d'ús

1. [Login]

Cas d'ús	Login	**Identificador:** CU01
Actors	Usuari (user i admin)	
Tipus	Primari	
Precondició	Executar la 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.	

5. 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 desenvolupat un sistema de gestió de tumbones amb una base de dades relacional.
   - Es va començar amb un diagrama Entitat-Relació (ER) i es va transformar a taules mitjançant codi SQL.
   - La base de dades s'ha creat amb SQLite 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.

-

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.

6. Bibliografia i webgrafia

- Apunts del Tema 4 sobre base de dades en SQL.

- 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