DDIARIO

¿Qué es?

Ddiario es un software diseñado para gestionar rutinas, dietas y entrenamientos.

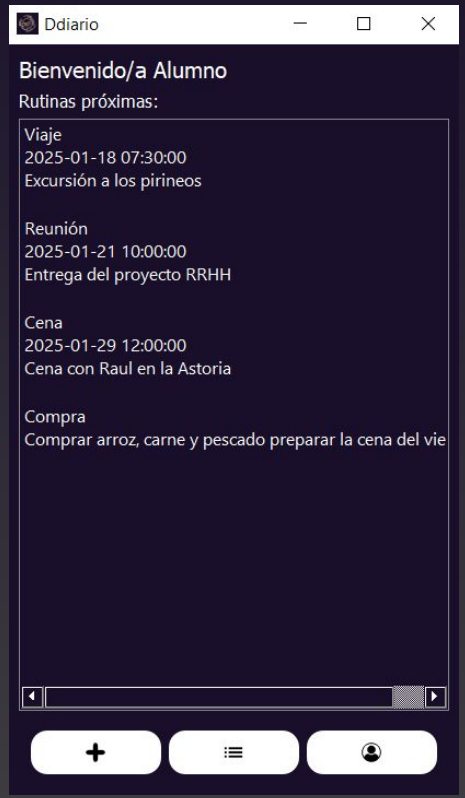
Ddiario busca satisfacer las necesidades de aquellos usuarios cuya agenda diaria sea una prioridad, permitiéndoles mantener una gestión óptima de su tiempo.

Funciones

El programa cuenta con diversas funciones que involucran a las rutinas, permitiendo crearlas, eliminarlas y editarlas, además de mostrarlas según fecha.

Público objetivo

A pesar de que la app esta desarrollada para cualquier usuario, principalmente nos centramos en usuarios de media edad que puedan necesitar una forma de mantener organizados sus días.



Estructura lógica

La estructura del programa se divide en 2 grandes partes, la estructura gráfica y la estructura lógica

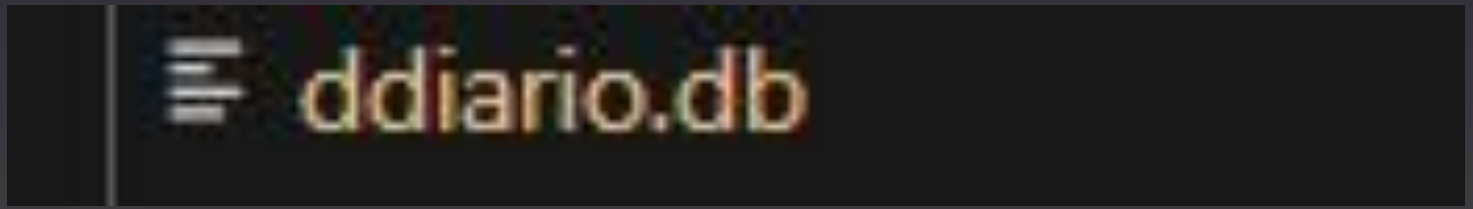
```python
class DatabaseManager:
    def __init__(self, db_name="res/ddiario.db"):
        '''Inicia la conexión con la base de datos y crea las tablas'''
        self.connection = sqlite3.connect(db_name)
        self.create_tables()

    def create_tables(self):
        '''Crea las tablas de la db si estas no existen'''

        # Inicia el cursor con la conexión a la db
        cursor = self.connection.cursor()

        # Tabla de usuarios
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS users (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                username TEXT NOT NULL,
                mail TEXT NOT NULL UNIQUE,
                password TEXT NOT NULL
            )
        ''')

        # Tabla de rutinas
        cursor.execute('''
            CREATE TABLE IF NOT EXISTS routines (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                user_id INTEGER NOT NULL,
                name TEXT NOT NULL,
                description TEXT,
                date TEXT NOT NULL,
                is_recurring BOOLEAN NOT NULL,
                FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
            )
        ''')


        # Commit
        self.connection.commit()
```

```python
# Widget principal y sus propiedades
main_window = QWidget()
main_window.setWindowTitle('Ddiario')
main_window.setGeometry(100, 100, 350, 600)
main_window.setWindowIcon(QIcon('res\\icon_main.png'))
with open('res\\css.css', "r", encoding='utf-8') as file:
    main_window.setStyleSheet(file.read())

# Layout principal (StackedLayout)
layout = QStackedLayout()
main_window.setLayout(layout)
screens = {
    0: WelcomeScreen(lambda i: layout.setCurrentIndex(i)),
    1: LoginScreen(lambda i: layout.setCurrentIndex(i), db, set_current_user),
    2: RegisterScreen(lambda i: layout.setCurrentIndex(i), db, set_current_user),
    3: MainScreen(lambda i: layout.setCurrentIndex(i), db, get_current_user),
    4: CreateRoutineScreen(lambda i: layout.setCurrentIndex(i), db, get_current_user),
    5: EditRoutineScreen(lambda i: layout.setCurrentIndex(i), db, get_current_user),
    6: PorfileScreen(lambda i: layout.setCurrentIndex(i), db, get_current_user, set_current_user),
}

def on_tab_changed(index):
    # Si el index cambia a estas ventanas, actualiza los datos de sus listas
    if index == 3:  # Pantalla principal
        screens[3].update_routines()
    elif index == 4:  # Pantalla de edición
        screens[4].update()
    elif index == 5:  # Pantalla de edición
        screens[5].update_routines()
    elif index == 6: # Pantalla perfil
        screens[6].update_info()
layout.currentChanged.connect(on_tab_changed)
```

Base de datos

El programa cuenta con una db SQLite local que almacena los registros de usuarios y las rutinas vinculadas a estos.

La base de datos se mantiene en local con el fin de evitar posibles problemas técnicos durante la presentación.



Interfaz

La interfaz ha sido diseñada con la librería de Python PYQT-5, combinada con un archivo css que interactúa con los elementos de dicha librería.

css.css > ...

```css
*{
    background-color : #190f2a;
    font-size: 16px;
    color : white;
}

.QPushButton{
    background-color: white;
    color: #190f2a;
    font-weight: bold;

    padding: 10;
    border-radius: 15px;
}

.QPushButton:hover{background-color: #c8c8c8;}
```

Prueba

Lo siguiente es una prueba práctica de las funciones que incluye la app:

Dificultades

- Cese del desarrollo móvil
- Persistencia de datos al cerrar la app
- Gestión de las condiciones en las rutinas
- Recorte en el diseño gráfico original

Futuro

Algunas de las posibles implementaciones en futuros desarrollos son:

- Compartir rutinas
- Diferentes interacciones entre rutinas
- Adaptación móvil y/o web

FIN