

最終課題

山口秀和
2024/02/19

目次

1. タスク管理アプリ
2. ToDoリストアプリ
3. 貸借管理アプリ
4. まとめ

1. タスク管理アプリ

- ・使用言語
 - ・ Java: ver. 17.0.2、PostgreSQL: ver. 11.2、HTML5、CSS3、JavaScript
- ・制作の目的
 - ・ 背景
 - ・ 重要な情報を紛失
 - ・ 何から手を付ければいいのかわからない
 - ・ 目的
 - ・ タスクリストを作成 ・ 進行度をチェック

今回作成したアプリはタスク管理アプリ、ToDoリストアプリ、貸借管理アプリの3つです。

タスク管理アプリに使用している言語は、Java、PostgreSQL、HTML、CSS、JavaScriptです。今回、タスク管理アプリを作成しようと思った拝啓には就職活動をする際に、重要な情報が記載してあるメールがどんどん流れていき探すのが大変だったということと、大量にしなければならないことが一気に増え、何から手を付ければいいのかわからなくなった。という背景があります。今回このアプリを作成することでやらなければならないことを洗い出し、整理し、進行度をチェックすることでしっかり管理できればと思い作成しました。

1. タスク管理アプリ

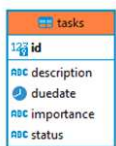
・機能

- ・タスクリストを表示
- ・新規タスクの追加
- ・既存タスクの更新
- ・既存タスクの削除

機能としては、データベースから取得したタスクリストを表示、新しいタスクの追加、既にあるタスクの更新・削除があります。

1. タスク管理アプリ

・詳細 (ER図)



tasksテーブル

id: ID
description: タスク名
due date: 期日
importance: 重要度
status: ステータス

データベースで使用しているテーブルの詳細は、tasksテーブルが1つあります。

tasksテーブルの内容は、属性として ID、タスク名、期日、重要度、ステータスがあります。

IDはserial型と主キーを設定しています。タスク名・重要度・ステータスはtext型、期日はdate型で設定しています。

1. タスク管理アプリ

・詳細 (メイン画面)

タスク名	期日	重要度	ステータス
開発者向け	2024-01-20	高	完了
公開に注意	2024-02-01	高	進行中
公開に注意	2024-02-01	低	未着手
公開に注意	2024-02-01	高	未着手
公開に注意	2024-02-15	中	未着手

Add New Task
タスク名: 期日: 重要度: ステータス:

Tasks

- ・タスク名
- ・期日
- ・重要度
- ・ステータス

Add New Task

- ・タスク名
- ・期日
- ・重要度(高・中・低)
- ・ステータス(未着手)

こちらがメイン画面になります。Tasksの表にデータベースから取得したリストを表示しています。

表の下の変更ボタンをクリックすると、変更画面に移動します。

Add New Taskの部分で新規追加を行います。

タスク名と期日に値が入っていないとエラーを返します。

タスク追加ボタンをクリックすると、ステータスの値を未着手とした状態でデータベースに保存されます。

1. タスク管理アプリ

- ・詳細（変更画面）

Tasks

タスク名	期日	締め期	ステータス	
得意先訪問	2024/01/10	済	完了	変更 削除
会議の記録	2024/02/101	済	進行中	変更 削除
会議の記録	2024/02/101	済	予定済	変更 削除
会議の記録	2024/02/11	済	予定済	変更 削除
会議の記録	2024/02/11	済	予定済	変更 削除
会議の記録	2024/02/11	済	予定済	変更 削除

タスクの変更

・全項目変更可能
 ただし行ずつ

タスクの削除

削除確認画面なし

1. タスク管理アプリ

- ・詳細（変更画面）

タスクの変更
・全項目変更可能
ただし行ずつ

タスクの削除
削除確認画面なし

Tasks

タスク名	期日	始期日	ステータス	
勉強会参加	2024/11/10	済	完了	変更 削除
会議に出席	2024/10/10	済	進行中	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除

戻る

1. タスク管理アプリ

- ・詳細（変更画面）

タスクの変更
・全項目変更可能
ただし行ずつ

タスクの削除
削除確認画面なし

Tasks

タスク名	期日	始期日	ステータス	
勉強会参加	2024/11/10	済	完了	変更 削除
会議に出席	2024/10/10	済	進行中	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除

戻る

1. タスク管理アプリ

- ・詳細（変更画面）

タスクの変更
・全項目変更可能
ただし行ずつ

タスクの削除
削除確認画面なし

Tasks

タスク名	期日	始期日	ステータス	
勉強会参加	2024/11/10	済	完了	変更 削除
会議に出席	2024/10/10	済	進行中	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除

戻る

1. タスク管理アプリ

- ・詳細（変更画面）

タスクの変更
・全項目変更可能
ただし行ずつ

タスクの削除
削除確認画面なし

Tasks

タスク名	期日	始期日	ステータス	
勉強会参加	2024/11/10	済	完了	変更 削除
会議に出席	2024/10/10	済	進行中	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除
会議に出席	2024/10/10	済	予定済	変更 削除

戻る

こちらが変更画面となります。

全項目の変更を可能にしています。

タスクの変更・削除は1行ずつ実行します。

注意：削除する時に確認画面が表示されません。

戻るボタンをクリックするとメイン画面に移動します。

こちらが変更画面となります。

全項目の変更を可能にしています。

タスクの変更・削除は1行ずつ実行します。

注意：削除する時に確認画面が表示されません。

戻るボタンをクリックするとメイン画面に移動します。

こちらが変更画面となります。

全項目の変更を可能にしています。

タスクの変更・削除は1行ずつ実行します。

注意：削除する時に確認画面が表示されません。

戻るボタンをクリックするとメイン画面に移動します。

こちらが変更画面となります。

全項目の変更を可能にしています。

タスクの変更・削除は1行ずつ実行します。

注意：削除する時に確認画面が表示されません。

戻るボタンをクリックするとメイン画面に移動します。

こちらが変更画面となります。

全項目の変更を可能にしています。

タスクの変更・削除は1行ずつ実行します。

注意：削除する時に確認画面が表示されません。

戻るボタンをクリックするとメイン画面に移動します。

2. ToDoリストアプリ

- ・ 使用言語
 - ・ Python: ver. 3.11.4, SQLite: ver. 3.41.2
- ・ 制作の目的
 - ・ 背景
 - ・ しなければならないことを忘れる
 - ・ 目的
 - ・ やるべきことを整理するリストを作成

2. ToDoリストアプリ

- ・ 使用言語
 - ・ Python: ver. 3.11.4, SQLite: ver. 3.41.2
- ・ 制作の目的
 - ・ 背景
 - ・ しなければならないことを忘れる
 - ・ 目的
 - ・ やるべきことを整理するリストを作成

2. ToDoリストアプリ

- ・ 使用言語
 - ・ Python: ver. 3.11.4, SQLite: ver. 3.41.2
- ・ 制作の目的
 - ・ 背景
 - ・ しなければならないことを忘れる
 - ・ 目的
 - ・ やるべきことを整理するリストを作成

2. ToDoリストアプリ

- ・ 使用言語
 - ・ Python: ver. 3.11.4, SQLite: ver. 3.41.2
- ・ 制作の目的
 - ・ 背景
 - ・ しなければならないことを忘れる
 - ・ 目的
 - ・ やるべきことを整理するリストを作成

2. ToDoリストアプリ

- ・ 使用言語
 - ・ Python: ver. 3.11.4, SQLite: ver. 3.41.2
- ・ 制作の目的
 - ・ 背景
 - ・ しなければならないことを忘れる
 - ・ 目的
 - ・ やるべきことを整理するリストを作成

2. ToDoリストアプリ

- ・ 使用言語
 - ・ Python: ver. 3.11.4, SQLite: ver. 3.41.2
- ・ 制作の目的
 - ・ 背景
 - ・ しなければならないことを忘れる
 - ・ 目的
 - ・ やるべきことを整理するリストを作成

2. ToDoリストアプリ

- ・機能
 - ・新規追加
 - ・リストの表示
 - ・更新
 - ・削除

2. ToDoリストアプリ

- ・機能
 - ・新規追加
 - ・リストの表示
 - ・更新
 - ・削除

2. ToDoリストアプリ

- ・機能
 - ・新規追加
 - ・リストの表示
 - ・更新
 - ・削除

2. ToDoリストアプリ

- ・機能
 - ・新規追加
 - ・リストの表示
 - ・更新
 - ・削除

2. ToDoリストアプリ

- ・機能
 - ・新規追加
 - ・リストの表示
 - ・更新
 - ・削除

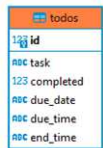
2. ToDoリストアプリ

- ・機能
 - ・新規追加
 - ・リストの表示
 - ・更新
 - ・削除

機能としては、新規タスクの追加、データベースから取得したリストの表示、既存タスクの更新と削除が可能です。

2. ToDoリストアプリ

・詳細 (ER図)



todos	
id	PK
task	
completed	
due_date	
due_time	
end_time	

todosテーブル

id:	ID
task:	タスク名
due_date:	日付
due_time:	開始時間
end_time:	終了時間

データベースで使用しているデータはtodosテーブルのみとなっています。

テーブルの属性は、ID、タスク名、日付、開始時間、終了時間の5つです。

主キーはIDでserial型で設定しています。他の4つはtext型で設定しています。

2. ToDoリストアプリ

・詳細 (入力部分)



タスク

年月日(yyyy-mm-dd)

開始時間(hh:mm)

終了時間(hh:mm)

送信

入力内容

- ・タスク名
- ・年月日
- ・開始時間 省略可
- ・終了時間 省略可

こちらが新規追加部分となります。

開始時間と終了時間は省略可となっており、省略した場合、未設定としてデータベースに保存されます。

2. ToDoリストアプリ

・詳細 (出力部分)



タスク	年月日	開始時間	終了時間
タスク1	2024-01-01	00:00	23:59
タスク2	2024-01-02	00:00	23:59
タスク3	2024-01-03	00:00	23:59
タスク4	2024-01-04	00:00	23:59
タスク5	2024-01-05	00:00	23:59
タスク6	2024-01-06	00:00	23:59
タスク7	2024-01-07	00:00	23:59
タスク8	2024-01-08	00:00	23:59
タスク9	2024-01-09	00:00	23:59
タスク10	2024-01-10	00:00	23:59

出力内容

- ・タスク名
- ・年月日
- ・開始時間
- ・終了時間

こちらがデータベースから取得したデータを表示する部分になります。

2. ToDoリストアプリ

- ・詳細（変更・削除）
 - ・変更
 - 入力部分に変更後の値を入力
 - 出力部分の変更したい項目を選択
 - 変更ボタンをクリック
 - ・削除
 - 出力部分の削除したい項目を選択
 - 削除ボタンをクリック

変更する際は、入力部分に変更後の値を入力していただき、
変更したい項目を出力部分で選択し、
変更ボタンをクリックする。
削除は、出力部分の削除したい項目を選択し、
削除ボタンをクリックする。

3. 貸借管理アプリ

- ・使用言語
 - ・VBA、SQLite: ver. 3.41.2
- ・制作の目的
 - ・背景
 - ・利息の計算が面倒くさい
 - ・目的
 - ・管理を楽にする

3つ目は、貸借管理アプリ（別名:天上不知唯我独損
(ハコワレ)、命名:喜久山)です。
このアプリに使用している言語は VBA、SQLite です。
このアプリは自分で使用することを目的で作成しており、利息の計算をするのが面倒なので
前々から欲しいと思っていたながら、ずっと作らずに
いたアプリを制作しました。

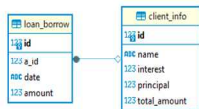
3. 貸借管理アプリ

- ・機能
 - ・ログイン
 - ・取得
 - ・追加
 - ・更新

機能としては、ログイン、データの取得、データの追加、データの更新があります。

3. 貸借管理アプリ

・詳細 (ER図)



loan_borrowテーブル

a_id: client_infoテーブルのid
date: 日付
amount: 金額

client_infoテーブル

name: 氏名
interest: 利息
principal: 元本
total_amount: 総金額

データベースに使用しているテーブルには
loan_borrowテーブルとclient_infoテーブルがあります。

loan_borrowテーブルには属性として、ID、
client_infoテーブルのID、日付、金額があります。
主キーはidをserial型で設定しています。a_idには
client_infoテーブルのidを外部キーとして設定してい
ます。

a_idとamountは数値として、dateには文字列型を設
定してます。

client_infoテーブルには属性として、ID、氏名、利
息、元本、総金額があります。

主キーにはidをserial型で設定しています。nameに
は文字列型、interest・principal・total_amountには
数値型で設定しています。

3. 貸借管理アプリ

・詳細 (ログイン)

他人に見られないよう
にログイン機能を搭載

OKボタンクリックで
データ取得

ファイルを起動すると最初にpasswordを求められる
ので入力します。

passwordが一致した場合データベースからデータを
取得して画面に表示します。

3. 貸借管理アプリ

・詳細 (表示データ)

氏名	id	元本	利息	総金額	表示内容
test1	1	18000	65	18065	・氏名
test2	2	13000	46	13046	・id
test3	3	5000	60	5060	・元本
test4	4	0	0	0	・利息
test5	5	0	0	0	・総金額
test6	6	0	0	0	

表示内容は、氏名、id、元本、利息、総金額です。

3. 貸借管理アプリ

・詳細 (氏名追加)

- ・新規追加ボタンをクリック
- ・氏名を入力
- ・追加をクリック

新規追加ボタンをクリックすると、氏名を入力する画面が出るので入力。追加ボタンをクリックすると、データベースに氏名を追加
その際に、元本、利息、総金額に0を入力

3. 貸借管理アプリ

・詳細 (AR追加)

- ・氏名をリストから選択
- ・IDは触らない
- ・年月日を入力
形式はyyyy-MM-dd
- ・金額を入力
数値のみ

ARボタン(accounts receivable)をクリックするとユーザーフォームが表示されます。氏名をリストから選択するとIDが自動で入力されます。
年月日にyyyy-MM-ddの形式で入力
金額に数値を入力してOKボタンをクリックする。
利息(年利5%)の計算をしたうえでデータベースに反映する。

3. 貸借管理アプリ

・詳細 (AP追加)

- ・氏名をリストから選択
- ・IDは触らない
- ・年月日を入力
形式はyyyy-MM-dd
- ・利息と元本に数値を入力
空欄がないよう入力

APボタン(Accounts payable)をクリックすると、ユーザーフォームが表示されるので氏名をリストから選択すると自動的にIDに数値が入力され、利息と元本の値が表示されます。
年月日をyyyy-MM-ddの形式で入力
利息と元本に0～表示されている値までの数値を入力する。
OKボタンをクリックするとデータベースに反映する。

3. 貸借管理アプリ

・詳細 (AP追加)

test1	日付	金額
詳細	2024/2/1	10000
	2024/2/1	2000
	2024/2/2	1000
	2024/2/3	1000
	2024/2/4	1000
	2024/2/5	2000
	2024/2/6	-2000
	2024/2/10	5000
	2024/2/13	-2000

- ・ J2のセルでリストを選択
- ・ 詳細ボタンをクリック
- ・ 詳細を表示

J2のセルで氏名のリストを選択し、詳細ボタンをクリックすると
データベースからリストを取得し、表示する。

4. まとめ

- ・ 作成したアプリ
 - ・ タスク管理アプリ
 - ・ ToDoリストアプリ
 - ・ 貸借管理アプリ

今回作成したアプリはタスク管理アプリ、ToDo リストアプリ、貸借管理アプリの3つです。
タスク管理アプリと ToDo リストアプリの違いを把握せずに作成してしまったので、ほとんど同じような動作になってしまいました。

ToDoリストアプリ

```
1 import tkinter as tk
2 from tkinter import messagebox
3 import sqlite3
4 from datetime import datetime
5
6 conn = None
7 cursor = None
8
9 def connect_database():
10     global conn, cursor
11     conn = sqlite3.connect('todo.db')
12     cursor = conn.cursor()
13
14 def close_database():
15     global conn
16     conn.close()
17
18 def create_todo_table(cursor):
19     # ToDoテーブルを作成
20     cursor.execute('''
21         CREATE TABLE IF NOT EXISTS todos (
22             id INTEGER PRIMARY KEY AUTOINCREMENT,
23             task TEXT NOT NULL,
24             completed BOOLEAN NOT NULL,
25             due_date DATE,
26             due_time TIME,
27             end_time TIME
28         )
29     ''')
30
31 def submit_form():
32     try:
33         task = entry_task.get()
34         duedate = entry_due_date.get()
35         duetime = entry_due_time.get()
36         endtime = entry_end_time.get()
37
38         # フォーマットが正しいか確認
39         datetime.strptime(duedate, '%Y-%m-%d')
40         if duetime.strip():
41             datetime.strptime(duetime, '%H:%M')
42         if endtime.strip():
43             datetime.strptime(endtime, '%H:%M')
44
45         # ToDoをデータベースに追加
46         connect_database()
47         add_todo(task, duedate, duetime, endtime)
48         close_database()
49
50         messagebox.showinfo("フォームの内容", f"タスク: {task},
51             日付: {duedate} が送信されました。")
52
53         # フォーム送信後にデータを再取得して表示
54         connect_database()
55         display_todos()
56         close_database()
57
```

```

58     except ValueError as e:
59         messagebox.showerror("エラー",
60             f"日付または時間の形式が正しくありません。詳細: {str(e)}")
61
62 def add_todo(task, due_date=None, due_time=None, end_time=None):
63     # 日付と時間を文字列からdatetimeオブジェクトに変換
64     # due_date = datetime.strptime(f'{due_date} {due_time}',
65         '%Y-%m-%d %H:%M:%S') if due_date else None
66     due_date = datetime.strptime(due_date, '%Y-%m-%d') if due_date else None
67     due_time = datetime.strptime(due_time, '%H:%M') if due_time else None
68     end_time = datetime.strptime(end_time, '%H:%M') if end_time else None
69
70     cursor.execute('INSERT INTO todos (task, completed, due_date, due_time, end_time)
71         VALUES (?, ?, ?, ?, ?)',
72         (task, False, due_date, due_time, end_time))
73     conn.commit()
74
75 def delete_selected_item():
76     global conn, cursor
77
78     selected_index = listbox.curselection()
79
80     if selected_index:
81         # 選択されたアイテムのインデックスを取得
82         selected_index = selected_index[0]
83
84         # 選択されたアイテムを取得
85         selected_item = listbox.get(selected_index)
86
87         # |で分割して、各要素を取得
88         parts = [part.strip() for part in selected_item.split('|')]
89         print(parts)
90
91         # "タスク: "の部分を取得して削除
92         task_info = parts[0].split(': ')[-1]
93         due_date_info = parts[1].split(': ')[-1]
94         due_time_info = parts[2].split(': ')[-1]
95         end_time_info = parts[3].split(': ')[-1]
96
97         # データベースから対応する要素を削除
98         connect_database()
99         cursor.execute("DELETE FROM todos WHERE task = ? AND due_date LIKE ?
100             AND due_time LIKE ? AND end_time LIKE ?",
101             (task_info, f'%{due_date_info}%', f'%{due_time_info}%', f'%{end_time_info}%'))
102         # データベース変更をコミット
103         conn.commit()
104         close_database()
105
106         # 選択されたアイテムを削除
107         listbox.delete(selected_index)
108
109         # 新しい値でアイテムを挿入
110         new_value = "削除"
111         listbox.insert(selected_index, new_value)
112
113         # フォーム送信後にデータを再取得して表示
114         # connect_database()
115         # display_todos()

```

```

116         # close_database()
117
118     def update_selected_item():
119         global conn, cursor
120
121         try:
122             task = entry_task.get()
123             duedate = entry_due_date.get()
124             duetime = entry_due_time.get()
125             endtime = entry_end_time.get()
126
127             # フォーマットが正しいか確認
128             datetime.strptime(duedate, '%Y-%m-%d')
129             if duetime.strip():
130                 datetime.strptime(duetime, '%H:%M')
131             if endtime.strip():
132                 datetime.strptime(endtime, '%H:%M')
133
134
135             selected_index = listbox.curselection()
136
137             if selected_index:
138                 # 選択されたアイテムのインデックスを取得
139                 selected_index = selected_index[0]
140
141                 # 選択されたアイテムを取得
142                 selected_item = listbox.get(selected_index)
143
144                 # | で分割して、各要素を取得
145                 parts = [part.strip() for part in selected_item.split('|')]
146
147                 # "タスク: "の部分を取得して削除
148                 task_info = parts[0].split(': ')[-1]
149                 due_date_info = parts[1].split(': ')[-1]
150                 due_time_info = parts[2].split(': ')[-1]
151                 end_time_info = parts[3].split(': ')[-1]
152
153                 # データベースから対応する要素を削除
154                 connect_database()
155                 cursor.execute("UPDATE todos SET task=?, due_date=?, due_time=?,
156                                end_time=? WHERE task=? AND due_date LIKE ? AND due_time LIKE ? AND end_time LIKE ?",
157                                (task, duedate, duetime, endtime, task_info, f'%{due_date_info}%',
158                                f'%{due_time_info}%', f'%{end_time_info}%'))
159                 # データベース変更をコミット
160                 conn.commit()
161                 close_database()
162
163                 # 選択されたアイテムを削除
164                 listbox.delete(selected_index)
165
166                 # 新しい値でアイテムを挿入
167                 new_value = f"タスク: {task:<20} | 日付: {duedate:<15} | 開始時間: {duetime:<10} |
168                             終了時間: {endtime:<10}"
169                 listbox.insert(selected_index, new_value)
170
171         except ValueError as e:
172             messagebox.showerror("エラー", f"日付または時間の形式が正しくありません。詳細: {str(e)}")
173

```

```

174
175 def display_todos():
176     # 日付と時間でソートしてToDoリストを取得
177     cursor.execute("SELECT task, strftime('%Y-%m-%d', due_date), strftime('%H:%M', due_time),
178         strftime('%H:%M', end_time) FROM todos ORDER BY due_date, due_time")
179     todos = cursor.fetchall()
180
181     # エントリーをクリア
182     entry_task.delete(0, tk.END)
183     entry_due_date.delete(0, tk.END)
184     entry_due_time.delete(0, tk.END)
185     entry_end_time.delete(0, tk.END)
186
187     # リストボックスをクリア
188     listbox.delete(0, tk.END)
189
190     # ToDoをリストボックスに表示
191     for todo in todos:
192         task = todo[0] if todo[0] is not None else "未設定"
193         due_date = todo[1] if todo[1] is not None else "未設定"
194         due_time = todo[2] if todo[2] is not None else "未設定"
195         end_time = todo[3] if todo[3] is not None else "未設定"
196
197         listbox.insert(tk.END, f"タスク: {task:<20} | 日付: {due_date:<15} |
198             開始時間: {due_time:<10} | 終了時間: {end_time:<10}")
199
200     # Tkinterウィンドウの作成
201     root = tk.Tk()
202     root.title("タスク入力画面")
203
204     # ラベルとエントリー(タスク)
205     label_task = tk.Label(root, text="タスク")
206     label_task.pack(padx=10)
207     entry_task = tk.Entry(root, width=30) # 幅を20に設定
208     entry_task.pack(padx=10)
209
210     # ラベルとエントリ(日付)
211     label_due_date = tk.Label(root, text="年月日(yyyy-mm-dd)")
212     label_due_date.pack(padx=10)
213     entry_due_date = tk.Entry(root, width=30) # 幅を20に設定
214     entry_due_date.pack(padx=10)
215
216     # ラベルとエントリー(開始時間)
217     label_due_time = tk.Label(root, text="開始時間(hh:mm)")
218     label_due_time.pack(padx=10)
219     entry_due_time = tk.Entry(root, width=30) # 幅を20に設定
220     entry_due_time.pack(padx=10)
221
222     # ラベルとエントリー(終了時間)
223     label_end_time = tk.Label(root, text="終了時間(hh:mm)")
224     label_end_time.pack(padx=10)
225     entry_end_time = tk.Entry(root, width=30) # 幅を20に設定
226     entry_end_time.pack(padx=10)
227
228
229     # 送信ボタン
230     submit_button = tk.Button(root, text="送信", command=submit_form)
231     submit_button.pack(pady=10, padx=10)

```

```
232 |
233 | # リストボックスの作成
234 | listbox = tk.Listbox(root, width=80)    # 幅を50に設定
235 | listbox.pack(pady=10, padx=10)
236 |
237 | # 削除ボタンの作成
238 | delete_button = tk.Button(root, text=" 選択したアイテムを削除", command=delete_selected_item)
239 | delete_button.pack(pady=10)
240 |
241 | # 変更ボタンの作成
242 | update_button = tk.Button(root, text=" 選択したアイテムを変更", command=update_selected_item)
243 | update_button.pack(pady=10)
244 |
245 | # フォーム起動時にデータを表示
246 | connect_database()
247 | display_todos()
248 | close_database()
249 |
250 | # Tkinterメインループ
251 | root.mainloop()
```

貸借管理アプリ

ThisWorkbook(ワークブックの開閉時の処理)

```
1 Option Explicit
2
3 Private Sub Workbook_BeforeClose(Cancel As Boolean)
4     ' Excelが閉じられる前に実行したい処理をここに追加
5     ClearData
6 End Sub
7
8 Sub ClearData()
9     ' B3からF列までの範囲をクリア
10    Range("B3:F" & Cells(rows.Count, "B").End(xlUp).row).ClearContents
11    Range("L3:M" & Cells(rows.Count, "L").End(xlUp).row).ClearContents
12    Range("J2").ClearContents
13
14 End Sub
15
16 Private Sub Workbook_Open()
17     UserForm1.Show
18 End Sub
```

Module1

```
1 Option Explicit
2
3 Sub ShowLoginForm()
4     UserForm1.Show
5 End Sub
6
7 Sub ShowAddForm()
8     ' 新規追加ボタンがクリックされた時の処理
9     ' フォームを表示
10    newName.Show
11 End Sub
12
13 Sub ShowARForm()
14     ' ARボタンがクリックされた時の処理
15     ' ARフォームを表示
16     ARFormName.Show
17 End Sub
18
19 Sub ShowAPForm()
20     ' APボタンがクリックされた時の処理
21     ' APフォームを表示
22     APFormName.Show
23 End Sub
24
25 Sub RefreshDataList()
26     ' データをリストとして表示している範囲をクリア
27     Range("B3:F1000").ClearContents
28
29     ' データベースからデータを取得してリストを再表示
30     Dim conn As Object
31     Set conn = CreateObject("ADODB.Connection")
32
33     ' SQLite3への接続文字列
34     Dim connStr As String
35     ' connStr = "DRIVER={SQLite3 ODBC Driver};Database=C:\Users¥7d10¥Desktop¥8_最終課題¥vbafinal.db;"
36     Dim dbPath As String
```

```

37 dbPath = ThisWorkbook.Path & "\vbafinal.db"
38 connStr = "DRIVER={SQLite3 ODBC Driver};Database=" & dbPath & ";";
39
40 ' 接続を開始
41 conn.Open connStr
42
43 ' SQLクエリを構築してデータを取得
44 Dim sql As String
45 sql = "SELECT id, name, principal, interest, total_amount FROM client_info;"
46
47 ' クエリを実行して結果を取得
48 Dim rs As Object
49 Set rs = CreateObject("ADODB.Recordset")
50 rs.Open sql, conn
51
52 ' データをExcelに表示
53 Dim row As Long
54 row = 3 ' 表示を始める行
55 Do Until rs.EOF
56     ' データをExcelに表示
57     Range("B" & row).value = rs("name").value
58     Range("C" & row).value = rs("id").value
59     Range("D" & row).value = rs("principal").value
60     Range("E" & row).value = rs("interest").value
61     Range("F" & row).value = rs("total_amount").value
62
63     row = row + 1
64     rs.MoveNext
65 Loop
66
67 ' 接続を閉じる
68 rs.Close
69 conn.Close
70 End Sub
71
72 Sub target_info()
73     Dim targetValue As Variant
74     Dim ws As Worksheet
75     Dim targetRow As Range
76     Dim a_id As String
77     Dim conn As Object
78     Dim rs As Object
79     Dim sql As String
80     Dim dataList As New Collection
81     Dim dataItem As Variant
82     Dim connStr As String
83
84     ' データをリストとして表示している範囲をクリア
85     Range("L3:M1000").ClearContents
86
87     ' ExcelのJ2セルの値を取得
88     targetValue = Sheets("Sheet1").Range("J2").value
89
90     ' 対応する行をB列から検索
91     Set ws = Sheets("Sheet1")
92     Set targetRow = ws.Columns("B:B").Find(What:=targetValue, LookIn:=xlValues, LookAt:=xlWhole)
93
94     If Not targetRow Is Nothing Then

```

```

95 ' B列で一致する行が見つかった場合
96     a_id = targetRow.Offset(0, 1).value ' C 列の値を取得
97
98 ' データベースに接続
99     Set conn = CreateObject("ADODB.Connection")
100     ' connStr = "DRIVER={SQLite3 ODBC Driver};Database=C:\Users\7d10\Desktop\8_ 最終課題\vbafinal.db;"
101     Dim dbPath As String
102     dbPath = ThisWorkbook.Path & "\vbafinal.db"
103     connStr = "DRIVER={SQLite3 ODBC Driver};Database=" & dbPath & ";"
104     conn.Open connStr
105
106     ' loan_borrowテーブルからデータを取得
107     sql = "SELECT date, amount FROM loan_borrow WHERE a_id = '" & a_id & "';"
108     Set rs = CreateObject("ADODB.Recordset")
109     rs.Open sql, conn
110
111 ' 結果をExcelのL列とM列に表示
112 Dim rows As Long
113 rows = 3 ' 表示を始める行
114 Do Until rs.EOF
115     ' データをExcelに表示
116     Range("L" & rows).value = rs("date").value
117     Range("M" & rows).value = rs("amount").value
118
119     rows = rows + 1
120     rs.MoveNext
121 Loop
122
123 ' 接続を閉じる
124 conn.Close
125 Else
126     MsgBox "指定された値が見つかりませんでした。"
127 End If
128 End Sub
129
130 Sub CommandButton_Select_Click()
131     ' 選択ボタンがクリックされた時の処理
132     ' 選択されたnameに付属するデータを取得してフォームに表示するなどの処理をここに追加
133
134     ' 例: 選択されたnameを取得
135     Dim selectedName As String
136     selectedName = ListBox_Names.value
137
138     ' 例: 選択されたnameに付属するa_idを取得
139     Dim a_id As Integer
140     ' ここでSQLite3からのデータ取得などの処理を行う
141
142     ' 例: フォームにa_idを表示
143     TextBox_a_id.value = a_id
144 End Sub
145
146 Sub CommandButton_Register_Click()
147     ' 登録ボタンがクリックされた時の処理
148     ' 入力されたデータをデータベースに登録する処理をここに追加
149
150     ' 例: 入力されたデータを取得
151     Dim a_id As Integer
152     a_id = TextBox_a_id.value

```



```

153
154     Dim dateValue As Date
155     dateValue = DatePicker_Date.value
156
157     Dim amount As Double
158     amount = CDbI(TextBox_Amount.value)
159
160     ' 例：データベースに登録する処理(SQLite3へのデータの挿入など)
161     ' ここでSQLite3へのデータベース接続とクエリ実行を行う
162
163     ' 登録後、リストを再取得
164     RefreshDataList
165
166     ' 登録が完了した旨のメッセージを表示
167     MsgBox "データを登録しました。"
168
169     ' フォームを閉じる
170     Unload Me
171 End Sub

UserForm1( ログイン画面 )
1 Option Explicit
2
3 Private Sub CommandButton1_Click()
4     btnLogin_Click
5 End Sub
6
7 Private Sub btnLogin_Click()
8     Dim password As String
9     Dim conn As Object
10    Set conn = CreateObject("ADODB.Connection")
11
12    ' パスワードを取得
13    password = Me.txtPassword.value
14
15    ' パスワードの検証
16    If password = "1234" Then
17        ' ログイン成功時の処理
18        Me.lblMessage.Caption = " ログイン成功"
19        ' ここにログイン後の処理を追加
20
21        On Error Resume Next
22
23        Dim dbPath As String
24        dbPath = ThisWorkbook.Path & "\vbafinal.db"
25        conn.Open "DRIVER={SQLite3 ODBC Driver};Database=" & dbPath & ";"
26        On Error GoTo 0
27
28        ' 接続が成功したかどうかを確認
29        If conn.State = 1 Then
30            MsgBox "SQLite データベースへの接続が成功しました。"
31            ' ここで必要な処理を実行
32
33            ' SQLクエリを構築
34            Dim sql As String
35            sql = "SELECT id, name, principal, interest, total_amount FROM client_info;"
36
37            ' クエリを実行して結果を取得

```

```
38         Dim rs As Object
39         Set rs = CreateObject("ADODB.Recordset")
40         rs.Open sql, conn
41
42         ' 結果をExcelに表示
43         Dim row As Long
44         row = 3 ' 表示を始める行
45     Do Until rs.EOF
46         ' データをExcelに表示
47         Range("B" & row).value = rs("name").value
48         Range("C" & row).value = rs("id").value
49         Range("D" & row).value = rs("principal").value
50         Range("E" & row).value = rs("interest").value
51         Range("F" & row).value = rs("total_amount").value
52
53         row = row + 1
54         rs.MoveNext
55     Loop
56
57     Unload Me ' フォームを閉じる
58 Else
```