Dokumentacja Techniczna Aplikacji do Zarządzania Rezerwacjami

Autor : Bartosz Jan Pękała

Data : 27.10.2024 r.

Spis treści

1. Wstęp

Niniejsza dokumentacja techniczna opisuje aplikację do zarządzania rezerwacjami, która została stworzona przy użyciu języka programowania Python oraz bazy danych PostgreSQL. Aplikacja została zaprojektowana w celu umożliwienia użytkownikom łatwego zarządzania terminami spotkań i rezerwacjami. Główne funkcje aplikacji obejmują dodawanie nowych rezerwacji, sprawdzanie dostępności terminów oraz przeglądanie istniejących spotkań.

Aplikacja opiera się na graficznym interfejsie użytkownika stworzonym z użyciem biblioteki Tkinter, co zapewnia intuicyjną obsługę oraz prostą nawigację. PostgreSQL pełni rolę głównej bazy danych, przechowującej informacje o wszystkich rezerwacjach, co gwarantuje skalowalność i bezpieczne przechowywanie danych.

Dokumentacja zawiera instrukcje dotyczące instalacji i uruchomienia projektu i opis jego struktury

2. Opis Projektu

Aplikacja do zarządzania rezerwacjami jest narzędziem służącym do obsługi i organizacji spotkań oraz rezerwacji terminów. Została zaprojektowana tak, aby umożliwić użytkownikom szybkie dodawanie nowych rezerwacji, sprawdzanie dostępności terminów oraz przeglądanie istniejących spotkań. Aplikacja posiada intuicyjny interfejs graficzny, który umożliwia łatwą nawigację.

Główne cechy aplikacji:

- Dodawanie nowych rezerwacji z losowym numerem spotkania, z jednoczesnym sprawdzeniem dostępności wybranego przedziału czasowego.
- Sprawdzanie dostępności terminów o wybranej dacie oraz przedziale czasowym.
- Przeglądanie istniejących spotkań w formie tabeli.

Aplikacja została stworzona przy użyciu języka programowania Python oraz biblioteki Tkinter do obsługi interfejsu graficznego. Za przechowywanie danych odpowiada baza danych PostgreSQL, która zapewnia skalowalność oraz bezpieczne przechowywanie informacji o rezerwacjach.

3. Wymagania techniczne

- Python (wersja 3.8 lub wyższa)
- Biblioteki Pythona:
    - Tkinter (standardowa biblioteka w Pythonie)
    - Psycopg2 (psycopg2-binary)
- PostgreSQL (wersja 12 lub wyższa)
- pgAdmin (zarządzanie bazą danych graficznie)

4. Instrukcja uruchomienia

Uruchomienie aplikacji przy użyciu konsoli

1. Upewnij się, że wszystkie wymagania techniczne są spełnione, w tym instalacja Pythona i bazy danych PostgreSQL.
2. Otwórz terminal lub wiersz poleceń.
3. Przejdź do katalogu, w którym znajduje się plik aplikacji (np. main.py).
4. Uruchom aplikację za pomocą polecenia: *python main.py*
5. Po uruchomieniu aplikacji zobaczysz okno interfejsu graficznego.

Uruchomienie aplikacji w Visual Studio Code

1. Upewnij się, że masz zainstalowany Visual Studio Code oraz Python na Twoim komputerze.
2. Otwórz Visual Studio Code.
3. Wybierz „Plik” -> „Otwórz folder” i wskaż folder, w którym znajduje się Twój plik skryptu (np. *main.py*).
4. Kliknij dwukrotnie na plik *main.py*, aby otworzyć go w edytorze.

5. Aby uruchomić skrypt, kliknij prawym przyciskiem myszy w obrębie edytora kodu i wybierz opcję „Run Python File in Terminal” lub naciśnij klawisz F5 albo kliknij przycisk „Run”.
6. Skrypt uruchomi się w zintegrowanym terminalu w Visual Studio Code. Zobaczysz okno interfejsu graficznego aplikacji.

Przeglądanie istniejących spotkań

1. Kliknij przycisk „Odczytaj dane z bazy”.
2. Aplikacja pobierze i wyświetli wszystkie istniejące rezerwacje w formie tabeli.
3. Możesz przeglądać tabelę i zobaczyć szczegóły każdego spotkania.

Dodawanie nowej rezerwacji

1. W oknie aplikacji wybierz odpowiednią datę z listy rozwijanej „Data”.
2. Wybierz dostępny przedział czasowy z listy rozwijanej „Przedział czasowy”.
3. Kliknij przycisk „Zapisz dane do bazy”.
4. Aplikacja sprawdzi, czy wybrany przedział czasowy jest wolny, i jeśli tak, zarezerwuje termin z unikalnym numerem spotkania.
5. Po dodaniu rezerwacji, tabela zostanie zaktualizowana, aby pokazać nowo dodane spotkanie.

Sprawdzanie dostępności terminu

1. Wybierz odpowiednią datę z listy rozwijanej „Data”.
2. Wybierz przedział czasowy z listy rozwijanej „Przedział czasowy”.
3. Kliknij przycisk „Sprawdź dostępność”.
4. Aplikacja sprawdzi, czy wybrany termin jest wolny. Jeśli jest zajęty, wyświetli istniejące spotkanie w tabeli.
5. Jeśli termin jest wolny, wyświetli odpowiedni komunikat.

Zamknięcie aplikacji

1. Aby zakończyć pracę z aplikacją, zamknij okno aplikacji, klikając przycisk zamknięcia okna (X) w prawym górnym rogu.
2. Jeśli uruchomiłeś aplikację w terminalu, zakończ pracę, naciskając Ctrl+C w terminalu.

5. Struktura techniczna projektu

Architektura aplikacji

Aplikacja została zaprojektowana w oparciu o model klient-serwer, gdzie interfejs użytkownika (GUI) pełni funkcję klienta, a baza danych PostgreSQL pełni funkcję serwera danych. Komunikacja między aplikacją a bazą danych odbywa się za pomocą zapytań SQL, obsługiwanych przez bibliotekę psycopg2.

Technologie użyte w projekcie

- Python: Główny język programowania użyty do stworzenia aplikacji.
- Tkinter: Biblioteka GUI Pythona używana do tworzenia interfejsu graficznego aplikacji.
- PostgreSQL: Baza danych relacyjna, w której przechowywane są informacje o rezerwacjach.
- Psycopg2: Biblioteka Pythona służąca do połączenia z bazą danych PostgreSQL.

Struktura kodu

Kod źródłowy aplikacji podzielony jest na kilka modułów odpowiadających za różne funkcje:

1. Moduł GUI (Tkinter): Odpowiada za tworzenie interfejsu graficznego, w tym przycisków, pól wprowadzania, list rozwijanych oraz tabeli do wyświetlania danych.

```
# Tworzymy główne okno aplikacji
root = tk.Tk()
root.title("Rezerwacje")

# Rozmiaru okna
root.geometry("1000x500")

# Ramka na przyciski i pola
button_frame = tk.Frame(root)
button_frame.pack(side="left", fill="y", padx=10, pady=10)

# Przycisk do odczytu danych z bazy
read_button = tk.Button(button_frame, text="Odczytaj dane z bazy", command=read_data)
read_button.pack(pady=10)

# Lista rozwijana do wyboru daty
date_label = tk.Label(button_frame, text="Data:")
date_label.pack(pady=5)

# Generowanie dat do wyboru (30 dni od dzisiaj)
dates = [(datetime.now() + timedelta(days=i)).strftime("%Y-%m-%d") for i in range(30)]
date_var = tk.StringVar(value=dates[0])
date_dropdown = tk.OptionMenu(button_frame, date_var, *dates)
date_dropdown.pack(pady=5)

# Lista rozwijana do wyboru przedziału czasowego
timeslot_label = tk.Label(button_frame, text="Przedział czasowy:")
timeslot_label.pack(pady=5)

# Generowanie przedziałów czasowych do wyboru
time_slots = ["09:00-10:00", "10:00-11:00", "11:00-12:00", "12:00-13:00"]
timeslot_var = tk.StringVar(value=time_slots[0])
timeslot_dropdown = tk.OptionMenu(button_frame, timeslot_var, *time_slots)
timeslot_dropdown.pack(pady=5)
```

Rysunek 1 Kod tworzenia interfejsu cz. 1

```
# Przycisk do zapisu danych do bazy
save_button = tk.Button(button_frame, text="Zapisz dane do bazy", command=save_data)
save_button.pack(pady=10)

# Przycisk do sprawdzania dostępności terminu
check_button = tk.Button(button_frame, text="Sprawdź dostępność", command=check_availability)
check_button.pack(pady=10)

# Tworzenie ramki na tabelkę
table_frame = tk.Frame(root)
table_frame.pack(side="left", fill="both", expand=True, padx=10, pady=10)

# Tworzenie tabelki przy użyciu Treeview
data_table = ttk.Treeview(table_frame, columns=("ID", "Date", "Przedział czasowy", "Opis", "Status"), show="headings")
data_table.heading("ID", text="ID")
data_table.heading("Date", text="Data")
data_table.heading("Przedział czasowy", text="Przedział czasowy")
data_table.heading("Opis", text="Opis")
data_table.heading("Status", text="Status")

# Ustawienie szerokości kolumn
data_table.column("ID", width=50)
data_table.column("Date", width=100)
data_table.column("Przedział czasowy", width=120)
data_table.column("Opis", width=200)
data_table.column("Status", width=100)

# Dodanie tabelki do okna
data_table.pack(fill="both", expand=True)

# Uruchomienie głównej pętli aplikacji
root.mainloop()
```

Rysunek 2 Kod tworzenia interfejsu cz. 2

2. Moduł obsługi bazy danych (Psycopg2): Zawiera funkcje odpowiedzialne za połączenie z bazą danych, pobieranie i zapisywanie danych, oraz wykonywanie operacji SQL.

```python
import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
import psycopg2
from datetime import datetime, timedelta
import random

# Funkcja do łączenia z bazą danych PostgreSQL
def connect_to_postgresql():
    try:
        connection = psycopg2.connect(
            dbname="Moja_Baza_Danych",
            user="postgres",
            password="niedowierzalny12!",
            host="localhost",
            port="5432"
        )
        print("Połączono z PostgreSQL")
        return connection
    except Exception as error:
        print("Błąd połączenia:", error)
        return None
```

Rysunek 3 Biblioteki i łączenie z bazą danych

```python
# Funkcja do odczytu danych z bazy
def read_data():
    connection = connect_to_postgresql()
    if connection:
        with connection.cursor() as cursor:
            cursor.execute("SELECT * FROM rezerwacje")
            records = cursor.fetchall()

            # Czyszczenie istniejących wierszy w tabeli
            for row in data_table.get_children():
                data_table.delete(row)

            # Dodawanie wierszy z bazy do tabeli
            for record in records:
                data_table.insert("", "end", values=record)

        connection.close()
        print("Odczytano dane z bazy danych!")
    else:
        messagebox.showerror("Błąd", "Nie udało się połączyć z bazą danych.")
```

Rysunek 4 Funkcja odczytu bazy danych

```python
# Funkcja do zapisu danych do bazy (dodanie nowej rezerwacji)
def save_data():
    date = date_var.get()
    timeslot = timeslot_var.get()

    if not date or not timeslot:
        messagebox.showerror("Błąd", "Wprowadź datę i przedział czasowy!")
        return

    connection = connect_to_postgresql()
    if connection:
        with connection.cursor() as cursor:
            # Losowanie unikalnego numeru spotkania
            unique_number = None
            while True:
                proposed_number = random.randint(1, 100)
                cursor.execute("SELECT COUNT(*) FROM rezerwacje WHERE opis = %s", (f"Spotkanie {proposed_number}",))
                if cursor.fetchone()[0] == 0:
                    unique_number = proposed_number
                    break

            # Sprawdzanie, czy wybrany przedział czasowy jest wolny
            cursor.execute(
                "SELECT COUNT(*) FROM rezerwacje WHERE date = %s AND przedzial_czasowy = %s",
                (date, timeslot)
            )
            if cursor.fetchone()[0] > 0:
                messagebox.showerror("Błąd", "Wybrany przedział czasowy jest już zajęty!")
                connection.close()
                return

            # Wstawianie danych do tabeli
            cursor.execute(
                "INSERT INTO rezerwacje (date, przedzial_czasowy, opis, status) VALUES (%s, %s, %s, %s)",
                (date, timeslot, f"Spotkanie {unique_number}", 'potwierdzony')
            )
```

```
        connection.commit()
        connection.close()
        messagebox.showinfo("Sukces", "Rezerwacja dodana pomyślnie!")
        read_data()
    else:
        messagebox.showerror("Błąd", "Nie udało się połączyć z bazą danych.")
```

Rysunek 5 Funkcja zapisywania rezerwacji do bazy danych

```
# Funkcja do sprawdzania dostępności terminu
def check_availability():
    date = date_var.get()
    timeslot = timeslot_var.get()

    if not date or not timeslot:
        messagebox.showerror("Błąd", "Wybierz datę i przedział czasowy!")
        return

    connection = connect_to_postgresql()
    if connection:
        with connection.cursor() as cursor:
            # Sprawdzanie, czy wybrany przedział czasowy jest wolny
            cursor.execute(
                "SELECT * FROM rezerwacje WHERE date = %s AND przedzial_czasowy = %s",
                (date, timeslot)
            )
            records = cursor.fetchall()

            # Czyszczenie istniejących wierszy w tabeli
            for row in data_table.get_children():
                data_table.delete(row)

            if records:
                messagebox.showinfo("Wynik", "Wybrany termin jest zajęty!")
                for record in records:
                    data_table.insert("", "end", values=record)
            else:
                messagebox.showinfo("Wynik", "Wybrany termin jest wolny.")

        connection.close()
    else:
        messagebox.showerror("Błąd", "Nie udało się połączyć z bazą danych.")
```

Rysunek 6 Funkcja sprawdzania dostępności rezerwacji w danym dniu i czasie

3. Tabela rezerwacje w bazie danych: Przechowuje informacje o wszystkich rezerwacjach, takie jak data, przedział czasowy, numer spotkania i status.

	id [PK] integer	date date	przedzial_czasowy character varying (20)	opis text	status character varying (20)
1	1	2024-11-02	10:00-11:00	Spotkanie 24	potwierdzony
2	2	2024-11-11	10:00-11:00	Spotkanie 9	potwierdzony
3	3	2024-11-11	11:00-12:00	Spotkanie 49	potwierdzony
4	4	2024-11-04	11:00-12:00	Spotkanie 33	potwierdzony
5	5	2024-11-19	11:00-12:00	Spotkanie 83	potwierdzony
6	6	2024-11-07	11:00-12:00	Spotkanie 20	potwierdzony
7	7	2024-10-26	09:00-10:00	Spotkanie 3	potwierdzony
8	8	2024-11-18	09:00-10:00	Spotkanie 71	potwierdzony
9	9	2024-11-20	10:00-11:00	Spotkanie 46	potwierdzony
10	10	2024-10-28	09:00-10:00	Spotkanie 40	potwierdzony
11	11	2024-10-29	09:00-10:00	Spotkanie 54	potwierdzony
12	12	2024-11-04	09:00-10:00	Spotkanie 73	potwierdzony
13	13	2024-11-04	10:00-11:00	Spotkanie 29	potwierdzony
14	14	2024-11-18	12:00-13:00	Spotkanie 31	potwierdzony
15	15	2024-11-15	12:00-13:00	Spotkanie 57	potwierdzony
16	16	2024-11-03	12:00-13:00	Spotkanie 4	potwierdzony
17	17	2024-11-21	11:00-12:00	Spotkanie 93	potwierdzony
18	18	2024-10-27	10:00-11:00	Spotkanie 43	potwierdzony
19	19	2024-11-03	11:00-12:00	Spotkanie 99	potwierdzony
20	20	2024-11-17	12:00-13:00	Spotkanie 34	potwierdzony
21	21	2024-11-09	11:00-12:00	Spotkanie 19	potwierdzony

Rysunek 7 Tabela rezerwacji

6. Podsumowanie

Aplikacja do zarządzania rezerwacjami została stworzona jako intuicyjne narzędzie do organizowania spotkań i terminów. Dzięki wykorzystaniu języka Python, biblioteki Tkinter do tworzenia interfejsu graficznego oraz bazy danych PostgreSQL, projekt zapewnia skalowalne i elastyczne rozwiązanie do zarządzania danymi.

Główne funkcjonalności aplikacji obejmują dodawanie nowych rezerwacji, sprawdzanie dostępności terminów oraz przeglądanie istniejących spotkań w formie tabeli. Dzięki modułowej budowie i zastosowaniu nowoczesnych technologii, aplikacja jest łatwa w rozbudowie oraz modyfikacji w celu dodania nowych funkcji lub obsługi większej liczby użytkowników.

Projekt został stworzony z myślą o łatwości obsługi, intuicyjnej nawigacji oraz bezpiecznym zarządzaniu danymi. W przyszłości aplikacja może zostać rozwinięta o dodatkowe funkcje, takie jak powiadomienia o rezerwacjach czy integracja z zewnętrznymi kalendarzami.