WOJSKOWA AKADEMIA TECHNICZNA

WYDZIAŁ INŻYNIERII LĄDOWEJ I GEODEZJI

PRZEDMIOT**: Programowanie w Systemach Informacji Przestrzennej**

SEMESTR: **VII**

ROK AKADEMICKI: **2023/2024**

WYKŁADOWCA: **ppłk dr inż. Marek Wyszyński**

# Projekt

TEMAT: Projekt systemu do zarządzania oddziałami pracowników w firmie informatycznej.

**WYKONAŁ:**

**sierż. pchor. Karol LASKA**

GRUPA: WIG20GG1S0

Nr albumu: 78706

**Spis treści:**

1. **Cel projektu (3)**
2. **Wykorzystane oprogramowanie (3)**
3. **Opis kodu realizującego poszczególne zadania**
   1. **Import bibliotek (3-4)**
   2. **Klasy obiektów w bazie danych (4)**
   3. **Połączenie z bazą danych (5)**
   4. **Dodawanie pracownika/oddziału (5-6)**
   5. **Usuwanie pracownika/oddziału (6-7)**
   6. **Wyświetlanie pracownika/oddziału (7)**
   7. **Aktualizacja pracowników/oddziałów (7-8)**
   8. **Mapa lokalizacji pracowników/oddziałów (8-9)**
   9. **Interfejs (9-11)**
   10. **Logowanie (12)**
4. **Link do platformy GitHub z dostępem do zrealizowanego kodu aplikacji (12)**
5. **Podsumowanie (12-13)**
6. **Cel laboratorium**

Do celu projektu należało opracowanie skryptu w języku python, który da nam możliwość obsługi systemu do zarządzania pracownikami i oddziałami w firmie informatycznej. Zakres działania to dodawanie, edytowanie, usuwanie, wyświetlanie danych zebranych w bazie danych oraz wyświetlanie ich na mapie.

1. **Wykorzystane oprogramowanie**

- Python (wersja 3.12)

- Pycharm (wersja 2023.2.3)

- Platforma GitHub

- PostgreSQL (16)

- pgAdmin (4)

1. **Opis kodu realizującego poszczególne zadania**
   1. **Import bibliotek**

Na początku pracy wykonano import niezbędnych bibliotek do realizacji kodu. Do tych bibliotek należały: psycopg2, folium, sqlalchemy, wkb oraz Nominatim.

Psycopg2 odpowiada za ustanawianie połączenia z bazą danych, wykonywanie zapytań SQL, przetwarzanie wyników zapytań oraz zarządzanie transakcjami. Dzięki psycopg2 można efektywnie komunikować się z bazą danych PostgreSQL, co jest istotne w projektach wymagających przechowywania i pobierania danych z tej konkretnej bazy danych.

Folium jest używana do tworzenia interaktywnych map internetowych. Odpowiada za łatwe i efektywne generowanie map w formie interaktywnych obiektów HTML. Umożliwia dodawanie markerów, kształtów, linii, obszarów i innych elementów na mapie, a także integruje się z danymi geograficznymi, takimi jak GeoJSON.

Biblioteka SQLAlchemy jest narzędziem do pracy z bazami danych w języku Python. Oferuje wygodny sposób interakcji z bazami danych, umożliwiając programistom manipulowanie danymi, tworzenie zapytań SQL, mapowanie obiektowo-relacyjne i wiele innych.

WKB jest powszechnie używane w kontekście systemów informacji geograficznej (GIS) i baz danych przestrzennych. Biblioteki, które obsługują WKB, pozwalają na konwersję danych geometrii pomiędzy formą binarną a formą zrozumiałą dla człowieka oraz umożliwiają manipulację tymi danymi w programach komputerowych.

Biblioteka Nominatim jest często wykorzystywana w programowaniu, zwłaszcza w kontekście aplikacji, które wymagają przekształcania adresów na współrzędne geograficzne (geokodowanie) lub współrzędnych na adresy (odwrotne geokodowanie).

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

*Rys.1 Zaimportowane biblioteki.*

* 1. **Klasy obiektów w bazie danych**

Klasa pracownik posłużyła do stworzenia tabeli, która będzie zawierać najważniejsze informacje o pracownikach firmy. Do tych informacji należą imię, nazwisko, miejscowość, specjalność, numer oddziału i lokalizacja. Dodatkowo zastosowana metoda inicjalizacyjna \_\_init\_\_(), która przyjmuję wprowadzane dane i przypisuje je do odpowiednich kolumn tabeli w bazie danych.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

*Rys.2 Klasa pracownik.*

W przypadku klasy oddziałów sytuacja jest analogiczna do tej powyższej, z małą zmianą dotyczącą zawartości tabeli. Do tabeli oddziałów przypisałem takie wartości jak nazwa, miejscowość, kraj, numer oddziału oraz lokalizację.

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

*Rys.3 Klasa oddział.*

* 1. **Połączenie z bazą danych**

W celu uzyskania możliwości tworzenia funkcji w pierwszej kolejności należało połączyć się z bazą danych PostgreSQL przy pomocy biblioteki SQLAlchemy. W pierwszej kolejności określiłem niezbędne parametry potrzebne do połączenia z bazą (db\_params). Następnie stworzyłem silnik odpowiedzialny za zarządzanie połączeniem do bazy danych. Silnik pozyskuje parametry wcześniej określone. Następnie stworzyłem obiekt Base służący do definiowania klas reprezentujących tabele w bazie danych. Na koniec określiłem instancję sesji wykorzystując klasę Session, by później móc ją wykorzystywać w dalszych etapach.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

*Rys.4 Połączenie z bazą danych.*

* 1. **Dodawanie pracownika/oddziału**

Pierwszą funkcją dotyczącą pracy na bazie danych została funkcja odpowiadająca za dodawanie pracownika do bazy danych. Funkcja polega na pobraniu i zapisaniu zmiennych w oparciu o input(). Następnie tworzę nowy obiekt dla klasy Worker, który posiada wprowadzone wcześniej dane. Następnie dodaje nowego pracownika do sesji poprzez add(). Na koniec pozostaje zapisać zmiany w bazie danych poprzez funkcję commit(). W przypadku klasy Workplace dodawanie działa tak samo, tylko przyjmuję inne dane, wcześniej zdeklarowane.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

*Rys.5 Dodawanie użytkownika.*

* 1. **Usuwanie pracownika/oddziału**

Kolejną zdefiniowana została funkcja odpowiedzialna za usuwanie pracownika lub pracowników z bazy danych. W skrócie funkcja pobiera nazwisko pracownika do usunięcia po przez funkcję input(). Następnie tworze zapytanie do bazy danych, by wyszukać pracowników o podanym nazwisku do czego wykorzystuję opcję filter(). Następnie tworzę pustą tymczasową listę, która przechowa wyszukanych pracowników. Następnie wyświetlam listę pracowników z przypisanymi im numerami. Na koniec okodowywuję warunek działania funkcji tak, by dla wprowadzonej wartości 0 usuwała wszystkich wyszukanych pracowników albo po prostu tak by usuwała tylko wybranego pracownika poprzez wprowadzenie przyporządkowanej mu liczbie. Usuwam przy użyciu delete(), a następnie zapisuję zmiany commit().

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

*Rys.6 Usuwanie pracownika.*

Usuwanie oddziału polega na takiej samej zasadzie działania. Jedynie są funkcję usuwania po innych wartościach pracowników np. funkcja lub oddział. Wtedy zmianą jest tylko zmienna pozyskiwana w celu wyszukania pracowników spełniających dany warunek.

* 1. **Wyświetlanie pracowników/oddziałów**

Jest to stosunkowo prosta funkcja, w której wykorzystano znów funkcję query() w celu wyszukania wszystkich danych o pracownikach. Następnie dodałem warunek, jeżeli zostanie zwrócona pusta lista to wyświetli napis brak danych. Jeżeli lista będzie zawierać dane to zostaną one po kolei wyświetlone.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

*Rys.7 Wyświetlanie pracowników.*

Wyświetlanie oddziałów działa tak samo jak i w przypadku poprzednich funkcji. Jedyną zmianą są sposoby wyświetlania pracowników z jednego oddziału lub o jednej specjalności. Wtedy do zapytania dodaję warunek filter(), który wyszukuję dane, które zawierają wcześniej określoną wartość np. numer oddziału.

* 1. **Aktualizacja pracowników/oddziałów**

Ta funkcja polega na pobraniu nazwiska pracownika poprzez input(). Następnie pobiera wszystkich pracowników z bazy danych. Dalej jest pętla, która przeszukuję listę pracowników i jeżeli znajdzie pracownika o danym nazwisku to zaczyna prosić o podanie nowych danych pracownika. Po wprowadzeniu wszystkich danych funkcja aktualizuje współrzędne lokalizacji pracownika. Na samym końcu wszystko jest zapisywane commit().

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

*Rys.8 Aktualizacja pracownika.*

Analogicznie działa edycja oddziałów. Największą zmianę posiadają funkcję do edycji pracowników z konkretnego oddziału lub o konkretnej specjalności. Tam dochodzi kilka innych elementów takich jak pobranie zmiennej edit czyli numeru oddziału. Następnie jest wyszukiwanie wszystkich pracowników z tego danego oddziału i umieszczanie ich w tymczasowej liście. W dalszej części kolejna pętla wyświetla wszystkich wyszukanych pracowników i przypisuję im numery. Następnie jest pętla, która dodaje jednego wybranego użytkownika do nowej listy. Na koniec ostatnia pętla pobiera nowe dane o pracowniku oraz aktualizuje jego współrzędne lokalizacji. Ostatnim krokiem jest zapisanie wszystkich zmian.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Opis wygenerowany automatycznie

*Rys.9 Aktualizacja pracownika z danego oddziału.*

* 1. **Mapa lokalizacji pracowników/oddziałów**

Funkcja rozpoczyna się od stworzenia obiektu mapy o określonych parametrach – folium.map(). Następie są pobierani wszyscy pracownicy z bazy danych. Dalej pętla pobiera współrzędne dla każdego pracownika odnosząc się do miejscowości przypisanej do niego. Dodatkowo w miejscu określonym przez współrzędne dodaje znacznik informujący jaki pracownik w danym miejscu mieszka. Na koniec wyświetla napis „Wygenerowano mapę” i zapisuje mapę poprzez map.save() pod wskazaną nazwą do pliku HTML.

Dodatkową funkcją jest funkcją get\_cordinates, która odpowiada za pobranie współrzędnych. Wykorzystuje ona bibliotekę geopy i moduł Nominatim, który jest geokoderem.Następnie location = geolocator.geocode(address): Wywołuje metodę geocode obiektu geolocator w celu uzyskania obiektu Location, który zawiera informacje o geolokalizacji podanego adresu. Następnie są pobierane współrzędne długość i szerokość. Na końcu w wyniku są zwracane jako krotka. Warto zauważyć, że funkcja ta może zwrócić wartość None, jeśli nie uda się zlokalizować danego adresu.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

*Rys.10 Tworzenie mapy pracowników.*

*Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie*

*Rys.11 Pobieranie współrzędnych.*

Jeżeli chodzi o mapę wszystkich oddziałów to działa tak samo tylko ma zmieniony wycentrowanie mapy i jej skalę bo obejmuję obszar całej Europy. Jedynie większe różnice posiadają mapy pracowników jednego oddziału, ponieważ tam dochodzi element wprowadzenia danych np. numeru oddziału, a następnie są pobierani wszyscy pracownicy, którzy spełniają podany warunek. Na koniec są pobierane współrzędne ich lokalizacji i analogicznie jest tworzona mapa tak jak w poprzedniej funkcji dla wszystkich pracowników.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

*Rys.12 Mapa pracowników jednego oddziału.*

* 1. **Interfejs**

Najważniejszą funkcją jest interfejs użytkownika. To on łączy wszystkie wcześniej stworzone funkcję i umożliwia korzystanie z nich dla użytkownika. Do stworzenia menu została wykorzystana pętla while, która działa do momentu aż natrafi na break. W pierwszej kolejności w pętli wyświetla się nasze menu z opisanymi możliwościami do wyboru i przyporządkowanymi im numerami. Następnie pobieram od użytkownika numer funkcji którą wybrał poprzez int(input()), ponieważ jest to liczba. Dalej wyświetla się komunikat jaką funkcję się wybrało. W dalszej części jest stworzona instrukcja match, która inicjuję blok dopasowania wzorców. W niej znajdują się bloki kodu, czyli case, które zostaną wykonanę gdy wybrana liczba przez użytkownika zostanie dopasowana do wartości przy danym case. Kolejno umieszczam wywołania danych funkcji pod odpowiednimi parametrami co owocuje w poprawnie działający interfejs.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie

*Rys.13 Interfejs użytkownika.*

Obraz zawierający tekst, zrzut ekranu

Opis wygenerowany automatycznie

*Rys.14 Interfejs użytkownika.*

*Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie*

*Rys.15 Interfejs użytkownika.*

Obraz zawierający tekst, zrzut ekranu, Czcionka, menu

Opis wygenerowany automatycznie

*Rys.16 Wizualizacja interfejsu.*

* 1. **Logowanie**

Ostatnią funkcją jest logowanie się do systemu. W tym przypadku zostały wykorzystane bardzo prymitywne metody polegające na wykorzystaniu nieskończonej pętli, która nie zatrzyma się, dopóki nie zostaną wprowadzone odpowiednie wartości. Dalej funkcja żąda wprowadzenia loginu i hasła po przez zwykłą opcję input(). Dalej funkcja sprawdza czy login i hasło zgadzają się z tymi określonymi wcześniej. Jeżeli tak to funkcją uruchamia interfejs użytkownika czyli GUI(). Na końcu znajduję się jeszcze break po to, żeby cały program wyłączył się po zatrzymaniu interfejsu. Drugą opcją jest wprowadzenie błędnych danych, wtedy wyświetla się komunikat o niepowodzeniu logowania i ponawia się próba logowania.

*Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie*

*Rys.17 Logowanie do systemu.*

1. **Link do platformy GitHub z dostępem do zrealizowanego kodu aplikacji**

https://github.com/Karol2532/sandbox\_psip

1. **Podsumowanie**

Praca nad projektem wymagała wykorzystania wiedzy i umiejętności pozyskanych na laboratorium. Temat projektu i wymagania z nim związane były bardzo podobne do wykonywanych zadań na ćwiczeniach laboratoryjnych. Jedynie zakres pracy i obszerność wymaganych elementów w skrypcie zmuszała nas do większego rozbudowania kodu. Mając możliwość ponownego opracowywania takiego zadania wykorzystałem inną opcję, ponieważ na laboratoriach korzystałem z cursora i tworzyłem zapytania SQL. W tej pracy postanowiłem skorzystać z opcji orm i tworzenia zapytań w oparciu o sesje. W początkowej fazie projektu stwarzało mi to problemy, jednakże z dokonywanymi postępami dochodziłem do wprawy w wykonywaniu zapytań i interesujących mnie funkcji. Dzięki temu zadaniowi poszerzyłem jeszcze bardziej swoją wiedzę i umiejętności z zakresu kodowania. Niestety nie udało mi się wykonać systemu w formie interfejsu graficznego. Mimo to z czasem na pewno wrócę do zagadnienia i postaram się go wykonać.

Wracając do zadania projektowego, napotkałem jeden problem z programem Pycharm. W sytuacji tworzenia funkcji logowania się i rozbudowywania jej o ciekawsze możliwości, czyli np. ukrywanie hasła, ograniczony czas logowania lub liczba prób, blokady czasowe itp. napotkałem problem, że aplikacja nie umożliwiała poprawnego działania funkcji. Jedyną opcją było korzystanie z terminala do poleceń i w nim częściowo to działało, lecz nadal nie całkowicie poprawnie. W innych programach moje okodowane funkcję działały prawidłowo więc można wywnioskować, że wina jest jednoznaczna po stronie Pycharma.

Podsumowując zadanie projektowe uważam za wykonane prawidłowo, zostały spełnione wszystkie wytyczne przekazane przez prowadzącego.