Dokumentacja

Spis treści:

1. Spis użytych technologii

1.1 Python w wersji 3.11.4.
1.2 Biblioteki w pythonie, między innymi sqlalchemy. Bardziej szczegółowy spis użytych bibliotek w pliku requirements.txt
1.3 Baza danych MariaDB, na serwerze zajęciowym
1.4 ERD editor dla wizualizacji zależności między tabelami
1.5 R w wersji 4.4.1
1.6 R Studio
1.7 tynitex

2. Lista plików i opis ich zawartości

```
Pimp my wheels
|
| -   data (folder z danymi dla projektu)
|               \
|               | - parameters (folder ze podstawowymi stałymi)
|               |                       \
|               |                       | - dates.json
|               |                       |
|               |                       | - employees.json
|               |                       |
|               |                       | - services_parts.json
|               |
|               | - raw (folder z imionami i nazwiskami osób, przed obróbką)
|               |                       \
|               |                       | - female_first_names.csv (imiona żeńskie, przed obróbką)
|               |                       |
|               |                       | - female_surnames.csv (nazwiska żeńskie, przed obróbką)
|               |                       |
|               |                       | - male_first_names.csv (imiona męskie, przed obróbką)
|               |                       |
|               |                       | - male_surnames.csv (nazwiska męskie, przed obróbką)
|               |
|               | - brands.csv (marki, modele i ceny samochodów)
|               |
|               | - equipment.csv(nazwy, typy i ceny rynkowe części samochodów)
```

```
|               |
|               | - female_surnames.csv(nazwiska żeńskie, po obróbce)
|               |
|               | - male_surnames.csv(nazwiska męskie, po obróbce)
|               |
|               | - names.csv(imiona, po obróbcę)
|
| -   src(folder dla podstawowej generacji bazy)
|               \
|               | - emulation (folder dla emulacji akcji podejmowanych przez osób)
|               |                   \
|               |                   | - __init__.py
|               |                   |
|               |                   | - customer_decision_maker.py (akcji podejmowane przez klientów)
|               |                   |
|               |                   | - emulation.py (emulacja jednego dnia pracy)
|               |                   |
|               |                   | - equipment_generator.py (generator narzędzi)
|               |                   |
|               |                   | - workshop_decision_maker.py (akcji dotyczące napraw w warsztatu)
|               |                   |
|               |                   | - workshop_emulator.py (akcji dotyczące sprzedaży/kupna warsztatu)
|               |
|               | - generators
|               |                   \
|               |                   | - __init__.py
|               |                   |
|               |                   | - equipment_generator.py (generator narzędzi dla tabel)
|               |                   |
|               |                   | - - personal_data_generator.py (generator danych personalnych)
|               |
|               | - models (folder architektur i zależności w bazie)
|               |                   \
|               |                   | - __init__.py
|               |                   |
|               |                   | - base.py (classa bazowa dla pozostałych tabel w sqlalchemhy)
|               |                   |
|               |                   | - customer.py (tabela klientów)
|               |                   |
|               |                   | - employee.py (tabela pracowników)
|               |                   |
|               |                   | - equipment.py (tabela katalogu narzędzi)
|               |                   |
|               |                   | - inventory.py (tabela urzytych narzędzi)
|               |                   |
|               |                   | - service.py (tabela napraw)
|               |                   |
|               |                   | - transaction.py (tabela transakcji bankowych)
|               |                   |
|               |                   | - vehicle.py (tabela kupionych, sprzedanych i naprawionych pojazdach)
|               |                   |
```

| | | - workshop.py (tabela warsztatów)
| | |
| | - person
| | |
| | - utils
| | | \
| | | - __init__.py
| | | |
| | | - names.py (wczytywanie danych z data/raw)
| | |
| | - __init__.py
|
| - venv (wirtualne środowisko pythona)
|
| - Dokumentacja.docx (dokumentacja)
|
| - main.py (plik generujący całą bazę)
|
| - PimpMyWheelsSchema.vuerd.json (wizualizacja zależności i tabel w bazie)
|
| - report_PimpMyWhells.qmd (generator raportu w quarto)
|
| - report_PimpMyWhells.pdf (gotowy raport pdf)
|
| - PimpMyWheelsSchema.vuerd.json (schemat bazy w erd editorze)
|
| - requirements.txt (spis użytych bibliotek pythona)

3. Kolejność i sposób uruchamiania plików

3.1 Na początek należy zainstalować python w wersji 3.11.4. Można to zrobić na oficjalnej stronie:

https://www.python.org/downloads/release/python-3114/

Wraz z pythonem instaluje się pip (administrator bibliotek dla pythona). Można sprawdzić czy się zainstalował za pomocą wpisania polecenia:

pip list

w cmd. Jeżeli pojawia się lista zaistalowanych bibliotek, to wszystko działa, jeżeli nie, można zainstalować administrator anaconda:

https://www.anaconda.com/download

3.2 Dalej należy zainstalować pakiet statystyczny R. Można zainstalować R w wersji 4.4.1 pod tym linkiem:
https://cran.r-project.org/bin/windows/base/

Również należy zainstalować RStudio, można to zrobić pod tym linkiem:

https://posit.co/download/rstudio-desktop/

3.3 Należy zainstalować wszystkie biblioteki dla projektu. Można to łatwo zrobić za pomocą komendy:

```
pip install -r requirements.txt
```

Znajdując się w głównym folderze PimpMyWheels

3.4 Instalujemy Visual Studio Code. Można to zrobić z oficjalnej strony: https://code.visualstudio.com/download

3.5 W Visual Studio Code instalujemy ERD Editor.

3.6 Dalej można zapoznać się z ogólną architekturą bazy w pliku PimpMyWheelsSchema.vuerd.json.

3.7 Ostatnim krokiem wchodzimy do pliku report_PimpMyWhells.qmd i generujemy pdf raporta za pomocą quarto i tynitex. Można zainstalować tynitex za pomocą komendy:

```
quarto install tinytex
```

Wpisanej w konsoli rstudio.

3.8 Teraz można otworzyć Raport.pdf

4. Schemat projektu bazy danych i zależności funkcyjne

Wizualizacja schematu projektu bazy danych jest w pliku:
PimpMyWheelsSchema.vuerd.json

Poniżej przedstawimy opis każdej tabeli:

4.1 Customers
id(Int, unsigned, not null, autoincrement , primary key)
name(String(50) , not null)
surname(String(50) , not null)
email(String(60) , not null)
phone_number(String(12) , not null)
birth_date(Date, not null)
address(String(200) , not null)
account_creation_date(Date, not null)
account_deletion_date(Date)
last_active(Date)

4.2 Employees
id (Int, unsigned, not null, autoincrement, primary key)
name(String(50), not null)
surname(String(50), not null)
email(String(60), not null)
phone_number(String(12), not null)
birth_date(Date, not null)
address(String(200), not null)
workshop_id(foreign key("workshops.id"))
position(String(100), not null)
hire_date(Date, not null)
resignation_date(Date)
salary(DECIMAL(8,2), unsigned, not null)

4.3 Equipment
id(Int, unsigned, not null, autoincrement, primary key)
name(String(255), not null)
type(String(50), not null)

cost(DECIMAL(8, 2), not null)

4.4 Inventory
id(Int, unsigned, not null, autoincrement, primary key)
equipment_id(foreign key(equipment.id))
service_id(foreign key(services.id))
workshop_id(foreign key(workshops.id))
delivery_date(Date, not null)

4.5 Services
id(Int, unsigned, not null, autoincrement, primary key)
employee_id(foreign key(employee.id))
start_date(Date, not null)
end_date(Date)
work_cost(DECIMAL(8, 2), not null)
transaction_id(foreign key(transactions.id))
description(String(100), not null)
vehicle_id(foreign key(vehicle.id))

4.6 Transactions
id(Int, unsigned, not null, autoincrement, primary key)
transaction_method(Enum(“cash”, “card”), not null)
other_party(foreign key(customers.id))
date(Date, not null)
transaction_type(Enum(“income”, “cost”), not null)
value(DECIMAL(8, 2), not null)

4.7 Vehicles
id(Int, unsigned, not null, autoincrement, primary key)
purchase_id(foreign key(transactions.id), nullable), jeżeli purchase i sale null to samochód był naprawiony, a nie sprzedany
sale_id(foreign key(transactions.id), nullable)
workshop_id(Int, unsigned, not null , foreign key(workshops.id))
brand(String(15))
model(String(15))

4.8 Workshops
id(Int, unsigned, not null, autoincrement, primary key)
address(String(100), not null)
phone_number(String(12), not null)
stations_number(Int, unsigned, not null), ilość stanowisk do naprawy
opening_date(Date, not null)

5. EKNF

Zależności funkcyjne:

1. Customers

   id → name, surname, email, phone_number, birth_date, address, account_creation_date, account_deletion_date, last_active

2. Employees

   id → name, surname, email, phone_number, birth_date, address, workshop_id, hire_date, resignation_date, salary

3. Equipment

id → name, type, cost

name → cost

4. Inventory

   id → equipment_id, service_id, workshop_id, delivery_date

5. Services

   id → employee_id, start_date, end_date, work_cost, transaction_id, description

6. Transactions

   id → transaction_method, other_party, date, transaction_type, value

7. Vehicles

   id → purchase_id, sale_id, workshop_id, brand, model

8. Workshops

   id → address, phone_number, stations_number, opening_date

W tabeli Equipment:

name → cost.

Normalizacja mówi, że należało by rozdzielić tabeli na dwie, aczkolwiek optymalizacja wskazuje, że można zostać przy jednej tabeli.

6. Najtrudniejsze podczas realizacji projektu

Najtrudniejsze podczas realizacji projektu było stworzyć architekturę bazy, oraz napisać kod, wypełniający ją.