|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Wydział  WIMiIP | Imię i nazwisko  1. Maciej Adamus  2. Kacper Bielak | Projekt nr 1  SQL | Kieurnek  Informatyka techniczna | |
| **Temat projektu:**  System bazodanowy do obsługi sklepu internetowego - NoSQL | | | | |
| Data wykonania  04.06.2024r. | Przedmiot:  Zawansowane architektury baz danych | | | Grupa  GL01 |

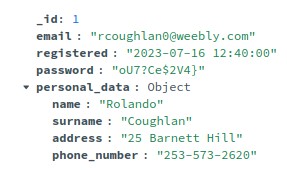
1. **Wstęp i cel projektu**

Celem pierwszego projektu na przedmiot Zawansowane architektury baz danych było zaimplementowanie systemu bazodanowego do obsługi sklepu internetowego z różnym asortymentem. Baza danych stworzona została w języku programowania NoSQL w systemie do zarządzania nierelacyjnymi bazami danych MongoDB. System ma na celu umożliwienie zarządzanie użytkownikami, produktami oraz zamówieniami w sklepie internetowym.

1. **Struktura systemu bazodanowego**

Zaimplementowany system bazodanowy składa się z trzech kolekcji:

* *User* – zawiera użytkowników zarejestrowanych w systemie – klientów, poniżej przedstawiono przykładowego użytkownika:



* *Product* – zawiera produkty w sklepie, w celach testowych zostały tam dodane produkty z trzech kategorii, które mają różne pola: telefony, samochody i maszyny budownicze, poniżej przedstawiono po jednym z każdych kategorii przykładowych produktów:

Obraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznieObraz zawierający tekst, zrzut ekranu, Czcionka

Opis wygenerowany automatycznie  
Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Opis wygenerowany automatycznie

* *Order* – zawiera informację o zamówieniach, posiada m.in. przypisanego użytkownika, listę produktów czy cenę całkowitą, poniżej przedstawiono przykładowy dokument z tej kolekcji:

Obraz zawierający tekst, Czcionka, zrzut ekranu, algebra

Opis wygenerowany automatycznie

W ramach systemu stworzony został również szereg funkcji agregujących, które w większości zostały wykorzystane w systemie przeznaczonym do prezentacji. Zostały one przedstawione poniżej:

1. Funkcja agregująca nr 1: wyświetlanie wszystkich użytkowników.
2. Funkcja agregująca nr 2: aktualizacja ceny produktu.
3. Funkcja agregująca nr 3: wyświetlanie wszystkich zamówień danego klienta.
4. Funkcja agregująca nr 4: wyświetlanie wszystkich zamówień, które mają obejmują dany produkt.
5. Funkcja agregująca nr 5: wyświetlanie wszystkich niedostępnych produktów.
6. Funkcja agregująca nr 6: wyświetlanie najdroższego produktu.
7. Funkcja agregująca nr 7: wyświetlanie najtańszego produktu.
8. Funkcja agregująca nr 8: wyświetlanie zamówień z odrzuconymi płatnościami.
9. Funkcja agregująca nr 9: zmiana hasła użytkownika.
10. Funkcja agregująca nr 10: wyświetlanie danych personalnych wszystkich użytkowników.
11. Funkcja agregująca nr 11: zmiana numeru telefonowi użytkownika.
12. Funkcja agregująca nr 12: wyświetlanie wszystkich produktów - telefonów.
13. Funkcja agregująca nr 13: wyświetlanie wszystkich produktów - samochodów.
14. Funkcja agregująca nr 14: wyświetlanie najtańszego produktów - maszyn.
15. Funkcja agregująca nr 15: wyświetlanie nazwy produktów, które są telefonami i liczby jego zamówień posortowanych w dół względem liczby zamówień.
16. Funkcja agregująca nr 16: stworzona na podstawie funkcji nr 15 zmodyfikowanej o zwracanie id telefonów i ich wszystkich informacji z kolekcji product
17. Funkcja agregująca nr 17: wyświetlanie użytkownika z największą liczbą zamówień.
18. Funkcja agregująca nr 18: wyświetlanie użytkownika z najwyższą łączną wartością wszystkich zamówień.
19. Funkcja agregująca nr 19: wyświetlanie wszystkich użytkowników, którzy mają w jakimkolwiek zamówieniu status płatności odrzucony.
20. Funkcja agregująca nr 20: wyświetlanie zamówień gdzie adres dostawy jest inny niż adres użytkownika.
21. Funkcja agregująca nr 21: zmiana produktu na niedostępny.
22. **Implementacja systemu – kod źródłowy, wykorzystane narzędzia**

System bazodanowy zaimplementowany został w nierelacyjnej bazie danych MongoDB w języku NoSQL. Przy implementacji wykorzystano system kontroli wersji Git w którym znajduje się całościowy kod źródłowy systemu wraz z jego opisem. Ponadto przy implementacji wykorzystano IDE Microsoft Visual Studio Code oraz gotowe narzędzie do zarządzania bazą danych MongoDB Compass. Link do repozytorium z implementacją znajduje się poniżej:

[Repozytorium GitHub](https://github.com/Bielak2000/zsbd-projekt-2)

Proces uruchomieniowy systemu został zautomatyzowano dzięki wykorzystaniu oprogramowania Docker, które jest open-source’owe i służy do konteneryzacji. Do tego celu stworzono plik *docker-compose.yml* zawierający konfigurację serwera bazodanowego i służący do jego uruchomienia na podstawie najnowszej wersji obrazu mongo. Jego implementacja pozwala na automatyczną inicjalizację wszystkich kolekcji oraz zasilenie ich danymi znajdującymi się w katalogu *database* przy uruchamianiu kontenera.

1. **Część zaimplementowana do prezentacji systemu**

W ramach prezentacji stworzonego systemu bazodanowego zaimplementowane aplikacje webową w język TypeScript z wykorzystaniem frameworka React. Powstałe oprogramowanie pozwala przeprowadzić przykładowe operację na bazie danych, a także umożliwia odpowiednią prezentację danych.

Poniżej przedstawiono zrzuty ekranu z aplikacji wraz z opisem funkcjonalności:

* Główny widok z listą top 10 uczniów, top 10 klas, uczniów zagrożonych i uczniów starających się o stypendium
* Widok z listą klas:
* Widok z listą studentów:
* Widok z listą nauczycieli:
* Widok pojedynczego studenta oraz możliwość dodania nowej oceny, usunięcia lub edycji istniejącej:
* Widok pojedynczego nauczyciela:
* Widok pojedynczej klasy:

1. **Wnioski i możliwości dalszego rozwoju**

Zaimplementowany system bazodanowy do obsługi sklepu internetowego gwarantuje możliwość jego zarządzaniem. Stworzone kolekcję pozwalają przechowywać dane o użytkownikach, zamówieniach oraz produktach. Zaimplementowane funkcję agregujące mogą zdecydowanie pomóc przy implementacji całościowego systemu pełniącego rolę sklepu internetowego (np. aplikacja webowa czy mobilna). Ich wykorzystanie wydaje się być zdecydowanie bardziej wydajną opcją niż pobieranie wszystkich danych i ich odpowiednie filtrowanie / sortowanie w implementacji samej aplikacji.

Wybór nierelacyjnej bazy danych zamiast relacyjnej był dobrym krokiem przy jej projektowaniu. Z racji dużej ilości kategorii produktów, które zazwyczaj znajdują się w takim systemie i różnymi parametrami, którymi się charakteryzują nierelacyjna baza danych wydaje się być zdecydowanie lepszym podejściem. Ponadto jej zastosowanie pozwala na bezproblemowe dodawanie nowych kategorii produktów i rozwijanie systemu. Sprzyjają również temu funkcje agregujące, które są łatwe w implementacji i pozwalają na obsługę różnorodnych danych.

Zaimplementowana część do prezentacji napisana w języku TypeScript z wykorzystaniem frameworka React pomaga w zobrazowaniu działania stworzonego systemu bazodanowego. Operację, która ona oferuje (wyświetlania, dodawanie, modyfikacje czy usuwanie danych) pomagają w lepszy sposób przedstawić jego strukturę i przykładowe użycie.

Jak już wspomniano wcześniej, stworzony system bazodanowy może zostać wykorzystany w końcowym systemie pełniącym rolę sklepu internetowego. Ponadto zaimplementowana część do prezentacji może być jego punktem wejściowym. Jeśli chodzi o samą bazę danych to może okazać się przydatne jego rozszerzenie o funkcjonalność integracji z zewnętrznymi systemami płatności, które służyły by do przeprowadzania opłat za zamówienia. Ponadto warto byłoby również zaszyfrować niektóre dane użytkowników, aby zapobiec ich łatwemu wyciekowi.