**Zaproponowany temat:  
*Analiza wydajności zapytań i struktur danych w systemie sprzedażowym***

**Zakres tematyczny pracy:**

1. **Projekt modelu danych i generowanie danych testowych:**
   * **Zaprojektowanie relacyjnego modelu danych dla systemu sprzedażowego (tabele: sprzedaż, klienci, produkty, czas, lokalizacja),**
   * **Generowanie danych testowych w różnych zakresach liczności (np. 1 000, 10 000, 100 000, 1 000 000 rekordów),**
   * **Wstępna transformacja danych i ich zapis do bazy danych PostgreSQL (symulacja procesu ETL),**
   * **Przygotowanie zapytań operujących na danych zagregowanych (SUM, AVG, GROUP BY, JOIN) oraz filtrowaniu danych według różnych kryteriów.**
2. **Analiza wydajności zapytań i czynników wpływających na czas odpowiedzi systemu:**
   * **Porównanie trzech strategii realizacji zapytań:**
     + **Java + ORM (findAll() i filtrowanie przy użyciu streamów),**
     + **Spring Data JPA z HQL lub metodami zapytań (findByXyz... lub @Query),**
     + **Native SQL zapisywany ręcznie (@Query(..., nativeQuery = true)),**
   * **Pomiar i analiza czasu wykonania zapytań oraz zużycia pamięci dla każdej strategii,**
   * **Analiza liczby wygenerowanych zapytań oraz problemów takich jak N+1,**
   * **Badanie wpływu struktury danych: poziomu normalizacji (tabele połączone vs denormalizowane),**
   * **Testowanie wpływu obecności i rodzaju indeksów na kolumnach filtrujących i sortujących,**
   * **Analiza działania i efektywności Hibernate Second-Level Cache przy wielokrotnym dostępie do tych samych danych.**

**Co analizujemy:**

* **Wydajność zapytań przy różnych rozmiarach zbioru,**
* **Różnice w wydajności i zużyciu zasobów pomiędzy:**
  + **filtrowaniem danych w Javie (po stronie aplikacji),**
  + **zapytaniami w HQL (ORM),**
  + **zapytaniami w czystym SQL (nativeQuery),**
* **Wpływ indeksów na wydajność – zapytania z i bez indeksów,**
* **Różnice wydajności w zależności od poziomu normalizacji danych,**
* **Zachowanie ORM w zależności od strategii ładowania danych (eager vs lazy),**
* **Liczbę zapytań generowanych automatycznie przez Hibernate i ich wpływ na czas odpowiedzi,**
* **Skuteczność cache przy ponownym odczycie danych (Hibernate Second-Level Cache).**

**Metoda badawcza:**

* ***Eksperymentalna* – porównanie wydajności trzech strategii realizacji zapytań w zróżnicowanych warunkach (wielkość danych, obecność indeksów, poziom normalizacji),**
* ***Analityczna* – opis przyjętych modeli danych, strategii ORM oraz czynników wpływających na wydajność operacji bazodanowych.**

**Frontend (część wizualna):**

* **Prosty panel webowy do wykonywania testów i prezentacji wyników:**
  + **generowanie danych testowych,**
  + **wykonywanie zapytań przy pomocy wybranej strategii,**
  + **wizualizacja wyników (czasy odpowiedzi, ilość zapytań, zużycie pamięci),**
* **Technologia frontendowa: Vue 3 + Chart.js lub Angular + Angular Material.**