1. Cel: Aplikacja do Zarządzania Zasobami w Magazynie.

2. Dostępne operacje aplikacji:

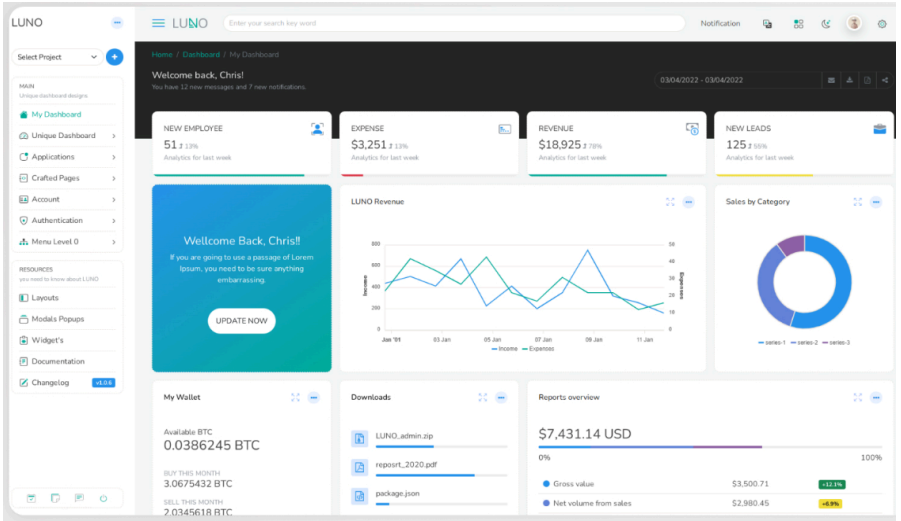
a. Rejestracja nowego towaru
   i. (rejestracja nowego towaru z nazwą, opisem, ceną i rozmiarem)
b. Edycja danych produktu
   i. (zmiana nazwy, ceny, opisu, rozmiaru)
c. Usuwanie produktu
   i. (np. jak produkt został wycofany)
d. Przyjęcie towaru na magazyn
   i. (dodanie dostawy i zwiększenie stanu magazynowego)
e. Wydanie towaru z magazynu
   i. (zmniejszenie stanu, np. wydanie do sklepu)
f. Wykonanie transakcji między dwoma magazynami
   i. (odpowiednie zmiany stanu magazynowego)
g. Wyszukiwanie produktów i podgląd np. stanu magazynowego
   i. (po kategorii, miesiące transakcji itd)
h. Tworzenie wykresów w aplikacji
i. Dodanie nowego:
   i. Magazynu
   ii. Klienta (np. sklep)
   iii. Supplier
j. Wygenerowanie raportu dla danych magazynów czy sklepów

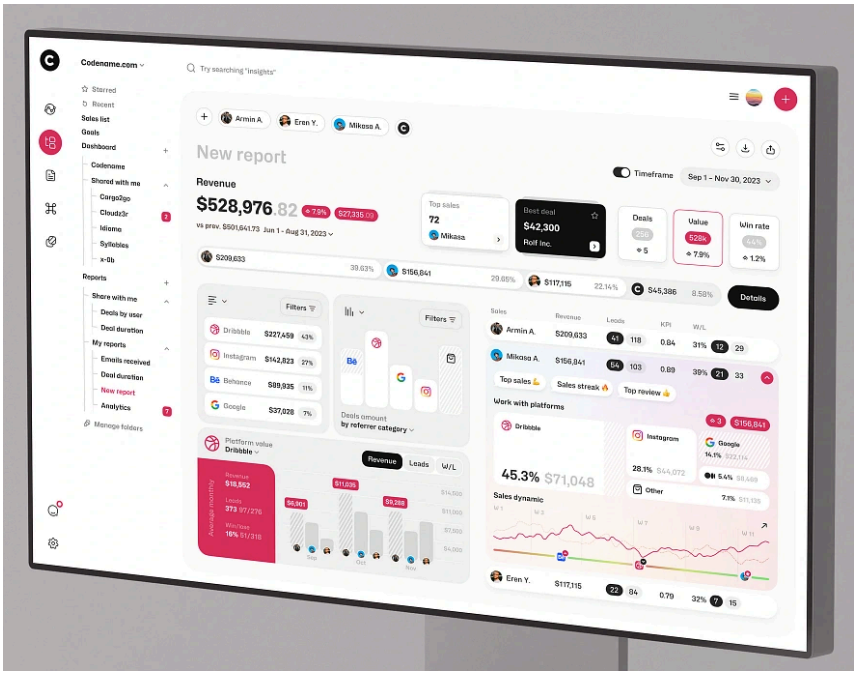
3. Architektura

- Backend: Java + Spring Boot
- Frontend: React

4. UI

Aplikacja będzie mieć nowoczesny wygląd przypominający dashboard biznesowy, podobny do tych ze wzoru:





Układ będzie przejrzysty i intuicyjny, dostosowany do jednej osoby zarządzającej magazynami.

Po lewej stronie znajdzie się menu z głównymi opcjami: zarządzanie produktami, operacje magazynowe (przyjęcia i wydania), raporty etc.
W centralnej części ekranu użytkownik zobaczy panel główny z najważniejszymi informacjami — m.in. stanami magazynowymi, alertami o niskim stanie towaru oraz podstawowymi wykresami pokazującymi ruch produktów.

Będą też dostępne szybkie akcje, jak dodanie nowego produktu czy wygenerowanie raportu. Całość będzie czytelna, estetyczna i przygotowana tak, by ułatwić codzienną pracę w magazynie.

Propozycje danych na “kafelki”:

- **Liczba wszystkich produktów** w magazynie (np. "Produkty: 528 sztuk").
- **Liczba kategorii produktów** (np. "Kategorie: 12").
- **Aktualna liczba przyjętych dostaw** w miesiącu (np. "Przyjęcia: 45").
- **Aktualna liczba wydanych towarów** w miesiącu (np. "Wydania: 38").
- **Produkty z niskim stanem magazynowym** (np. "Alerty: 7 produktów").
- **Najczęściej wydawany produkt** (np. "Top produkt: Wiertarka XYZ").
- **Całkowita wartość zapasów** (np. "Wartość magazynu: 120 000 zł").
- **Wartość przyjęć i wydań** w ostatnim tygodniu/miesiącu (np. "Obrót: +15 000 zł").
- **Ostatnie operacje magazynowe** (np. "Ostatnie przyjęcie: 22.04.2025").

5. **Logika w procedurach/triggerach:**

Procedury:

1. **Obsługa transakcji magazynowych (dodawanie transakcji):**

- Tworzenie nowej transakcji w tabeli `Transaction`.
- Dodawanie powiązanych produktów (`TransactionProduct`).
- Aktualizacja stanów magazynowych (`ProductInventory`):
  - Przy transakcji **SUPPLIER_TO_WAREHOUSE** ➔ zwiększ stan magazynowy.
  - Przy transakcji **WAREHOUSE_TO_CUSTOMER** ➔ zmniejsz stan magazynowy.
  - Przy **WAREHOUSE_TO_WAREHOUSE** ➔ zmniejsz w jednym magazynie, zwiększ w drugim.

2. **Obsługa przyjęcia nowej dostawy:**

- Procedura przyjmująca produkty od dostawcy (SUPPLIER ➔ WAREHOUSE).
- Automatyczne tworzenie transakcji i aktualizacja `ProductInventory`.

3. **Obsługa sprzedaży klientowi:**

- Procedura generująca sprzedaż (WAREHOUSE ➔ CLIENT).
- Sprawdzanie dostępnej ilości przed sprzedażą.

4. **Aktualizacja lub tworzenie raportów (`Report`):**

- Procedura generująca raport stanu magazynu na konkretny dzień.

Triggery:

1. **Pilnowanie pojemności magazynów (`Warehouse`):**

- **Przed** aktualizacją `ProductInventory`: sprawdzenie, czy suma `OccupiedCapacity` + rozmiary nowych produktów nie przekroczy `Capacity` magazynu.
- Aktualizacja `OccupiedCapacity` po każdej zmianie stanów magazynowych.

2. **Walidacja danych przed insertem/aktualizacją:**

- **Trigger przed INSERT lub UPDATE** na `Transaction` ➔ sprawdzenie, czy spełniony jest warunek `transaction_type_check`, zanim baza wyrzuci błąd constraint.

3. **Synchronizacja `OccupiedCapacity`:**

- Przy zmianie ilości produktów w `ProductInventory`, przeliczenie `OccupiedCapacity` magazynu automatycznie w triggerze.