Wyższa Szkoła Ekonomii i Informatyki w Krakowie

Dokumentacja Użytkowa Systemu Magazynowego StockMaster

Autor: Damian Adamski

Wstęp i charakterystyka systemu

StockMaster to nowoczesny, mobilny system klasy WMS, zaprojektowany z myślą o cyfryzacji procesów logistycznych w małych i średnich przedsiębiorstwach. Rozwiązanie to przekształca standardowy smartfon w profesjonalny terminal magazynowy, wykorzystując wbudowaną kamerę jako precyzyjny skaner kodów. Oczywiście również można użyć do obsługi skanerów/terminali mobilnych z systemem android, coraz to nowsze terminale na rynku zaczynają pojawiać się właśnie z systemem android.

Aplikacja umożliwia błyskawiczną realizację kluczowych operacji, takich jak przyjęcia, przesunięcia i wydania towaru, zintegrowane z automatycznym generowaniem faktur i paragonów. Dzięki architekturze opartej na chmurze, system zapewnia podgląd stanów magazynowych w czasie rzeczywistym, skutecznie eliminując papierową dokumentację oraz błędy ludzkie. Bezpieczeństwo danych gwarantuje wbudowany system autoryzacji z podziałem na role (Admin, Magazynier, wizytor dla rewidentów i kontroli), zapewniający pełną kontrolę dostępu.

Pracę nad aplikacją trwają ciągle, funkcje które są aktualnie mogą w przyszłości zostać przebudowane w zupełnie inne, aby upłynnić pracę na aplikacji oraz zapewnić pełną obsługę nad systemem bez warunku zmiany kodu aplikacji.

Link do repozytorium z aplikacją: https://github.com/Dadamsky/StockMaster

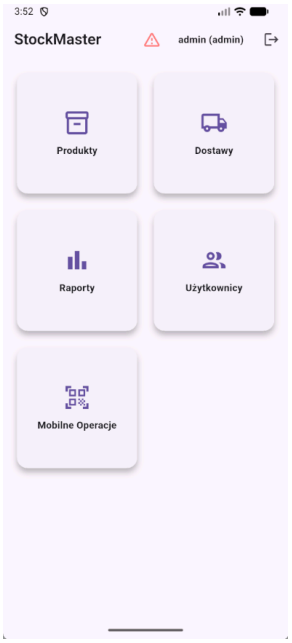
1. Logowanie i Dostęp do Systemu

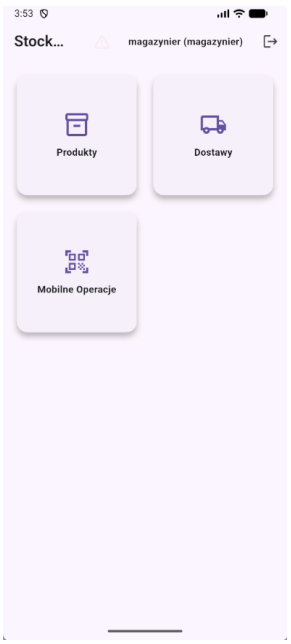
System zabezpieczony jest mechanizmem autentykacji. Dostęp do poszczególnych modułów zależy od przypisanej roli użytkownika.

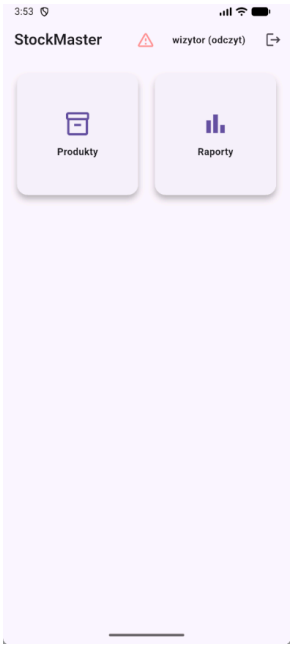
- **Ekran Logowania:** Wymaga podania unikalnego **Loginu** oraz **Hasła**.

1. Konto admin : Login admin hasło adminek
2. Konto magazynier : Login magazynier hasło magazynier
3. Konto wizytor : Login wizytor hasło wizytor

- **Wylogowanie:** Dostępne z poziomu Dashboardu (ikona w prawym górnym rogu).







1.1 **Autentykacja i definicja ról.**

Poniższy kod inicjalizuje kluczowe komponenty ekranu głównego. Mapa **`_rolePermissions`** definiuje statyczną politykę dostępu, przypisując konkretne moduły systemu do ról użytkowników. Metoda `_canView` realizuje logikę weryfikacji, przyznając pełne uprawnienia dla roli administratora oraz selektywne dla pozostałych grup.

```
final Map<String, List<String>> _rolePermissions = {
  'admin': ['📦 Produkty', '🚚 Dostawy', '📊 Raporty', '⚙️ Użytkownicy', '📱 Mobilne Operacje'],
  'magazynier': ['📦 Produkty', '🚚 Dostawy', '📱 Mobilne Operacje'],
  'odczyt': ['📊 Raporty', '📦 Produkty'],
};
```

```
bool _canView(String role, String title) {
  if (role == 'admin') return true;
  final allowedTitles = _rolePermissions[role] ?? [];
  return allowedTitles.contains(title);
}
```

Poniższy fragment metody `build` odpowiada za warstwę prezentacji Dashboardu, reagującą w czasie rzeczywistym na status użytkownika. Wykorzystano widget `StreamBuilder`, który utrzymuje aktywne połączenie z dokumentem w bazie Firestore. Kod dynamicznie ekstrahuje bieżącą rolę (`userRole`), definiuje pełną pulę dostępnych modułów (`menuItems`), a następnie tworzy listę `filteredItems`. Proces ten polega na przefiltrowaniu dostępnych opcji przy użyciu logiki weryfikacyjnej (`_canView`) i zmapowaniu ich na odpowiednie ekrany docelowe, co gwarantuje wyświetlenie wyłącznie autoryzowanych elementów interfejsu.

```
@override
Widget build(BuildContext context) {
  return StreamBuilder<DocumentSnapshot>(
    stream: _currentUserId != null
        ? _db.collection('users').doc(_currentUserId).snapshots()
        : null,
    builder: (context, snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
        return const Scaffold(
          body: Center(child: CircularProgressIndicator()),
        ); // Scaffold
      }

      final userRole = (snapshot.data?.data() as Map<String, dynamic>?)?['role'] ?? 'odczyt';

      // Definicja wszystkich możliwych przycisków
      final Map<String, IconData> menuItems = {
        '📦 Produkty': Icons.inventory_2_outlined,
        '🚚 Dostawy': Icons.local_shipping_outlined,
        '📊 Raporty': Icons.bar_chart_outlined,
        '⚙️ Użytkownicy': Icons.group_outlined,
        '📱 Mobilne Operacje': Icons.qr_code_scanner,
      };

      // Filtrowanie elementów na podstawie roli
      final filteredItems = menuItems.keys.where((title) => _canView(userRole, title)).map((title) {
        final icon = menuItems[title]!;
        final screenMap = {
          '📦 Produkty': const ProductListScreen(),
          '🚚 Dostawy': const DeliveriesScreen(),
          '📊 Raporty': const ReportsScreen(),
          '⚙️ Użytkownicy': const UserManagementScreen(),
          '📱 Mobilne Operacje': const mob_ops.MobileOperationsScreen(),
        };
```

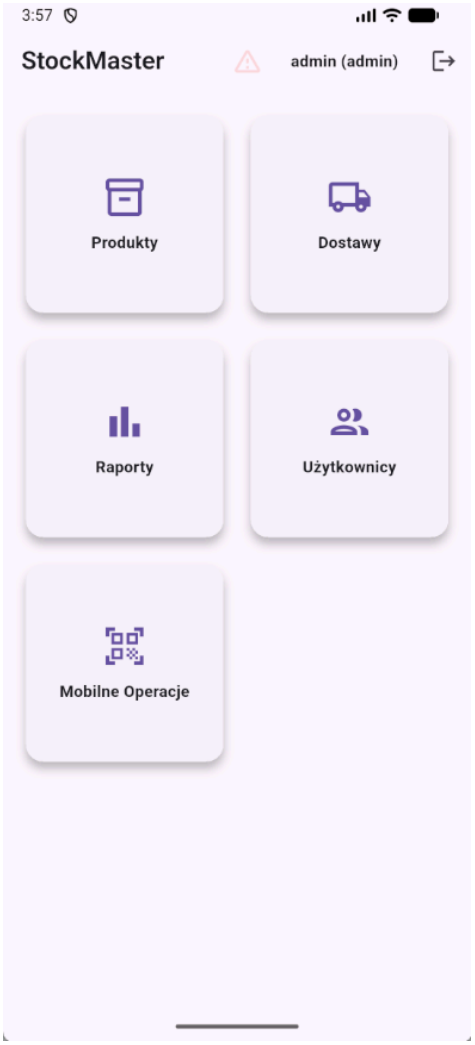
```
        return _buildCard(context, title, icon, () {
          if (screenMap[title] is! Container) {
            _navigateTo(context, screenMap[title]!);
          }
        });
      }).toList();
```

2. Panel Główny (Dashboard)

Centrum sterowania aplikacją. Widok dostosowuje się dynamicznie do uprawnień zalogowanego użytkownika.

Elementy interfejsu:

1. **Nazwa użytkownika:** Wyświetlana w nagłówku.
2. **Przycisk wylogowania w prawym górnym:** Bezpieczne zamknięcie sesji.
3. **Ikona Alertu :** Interaktywny przycisk ostrzegawczy, który pojawia się, gdy stan magazynowy któregokolwiek produktu spadnie poniżej ustalonego minimum.
4. **Kafelki Funkcyjne:** Szybki dostęp do modułów: Produkty, Dostawy, Raporty, Użytkownicy, Mobilne Operacje.

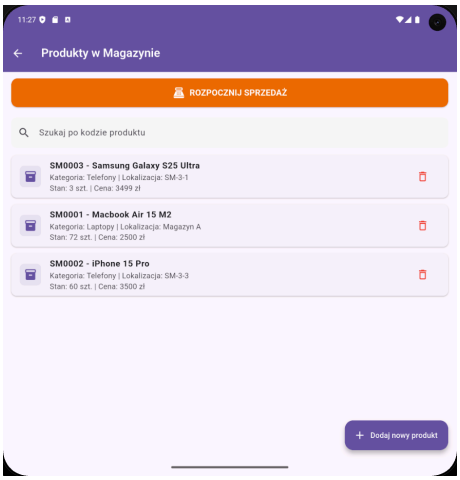


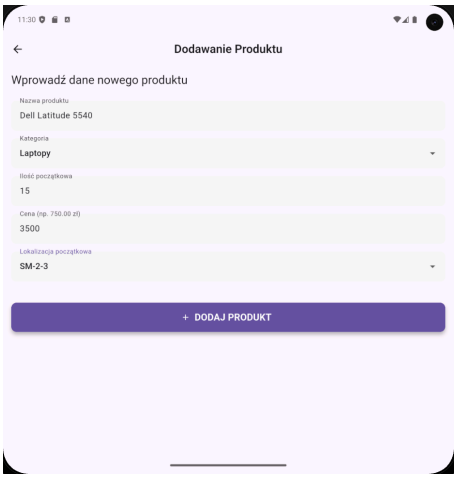
3. Moduł: Produkty (Zarządzanie Asortymentem)

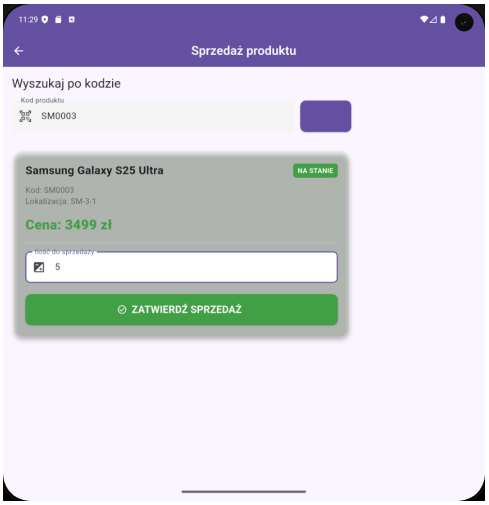
Moduł służący do przeglądania bazy towarowej i szybkiej sprzedaży detalicznej.

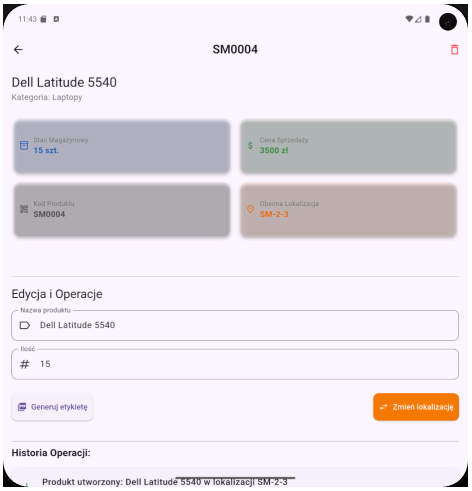
Funkcje:

- **Lista Produktów:** Wyświetla nazwę, cenę, kategorię oraz unikalny numer seryjny (SM). Mamy możliwość zmiany ilości, nazwy lub wygenerowania kodu kreskowego czyli wydrukowanie etykiety do wybranej drukarki.
- **Wyszukiwanie:** Pasek wyszukiwania umożliwiający filtrowanie po kodzie produktu, terminale danych posiadające skaner mogą skanować etykietę aby szybciej wyszukiwać produkt.
- **Dodawanie Produktu:** Formularz rejestracji i przyjęcia nowego produktu, generującym tym samym kod seryjny unikalny i etykietę do druku.
- **Rozpocznij Sprzedaż:**
    1. Wpisanie numeru seryjnego lub zeskanowanie etykiety.
    2. Deklaracja ilości sprzedawanej.
    3. Automatyczne generowanie **Paragonu**.
    4. Komunikat końcowy: "Sprzedaż zakończona pomyślnie".









3.1 Definiowanie modułu sprzedaży, wyszukiwania i listy produktowej.

Implementacja głównego przycisku akcji "Rozpocznij Sprzedaż"

Fragment kodu definiujący kluczowy element interfejsu (Call to Action) na liście produktów. Zastosowano widget `ElevatedButton.icon` z niestandardowym stylem (kolorystyka, zaokrąglenie), który służy do szybkiej inicjacji procesu sprzedażowego. Obsługa zdarzenia `onPressed` realizuje nawigację do nowego widoku (`SalesScreen`) przy użyciu `Navigator.push`

```
Container(
  padding: const EdgeInsets.symmetric(horizontal: 16, vertical: 12),
  child: Row(
    children: [
      Expanded(
        child: ElevatedButton.icon(
          onPressed: () {
            // Docelowo tutaj nawigujemy do ekranu sprzedaży
            Navigator.push(
              context,
              MaterialPageRoute(builder: (_) => const SalesScreen()),
            );
          },
          icon: const Icon(Icons.point_of_sale_outlined, size: 24),
          label: const Text(
            'ROZPOCZNIJ SPRZEDAŻ',
            style: TextStyle(fontSize: 16, fontWeight: FontWeight.bold),
          ), // Text
          style: ElevatedButton.styleFrom(
            backgroundColor: Colors.orange.shade800,
            foregroundColor: Colors.white,
            padding: const EdgeInsets.symmetric(vertical: 15),
            shape: RoundedRectangleBorder(
              borderRadius: BorderRadius.circular(12),
            ), // RoundedRectangleBorder
          ),
        ), // ElevatedButton.icon
      ), // Expanded
    ],
  ), // Row
), // Container
```

Implementacja wyszukiwania w czasie rzeczywistym (Live Search).

Funkcja `onChanged` przy każdej zmianie tekstu normalizuje zapytanie (metody `trim` i `toLowerCase`) oraz wywołuje `setState`, wymuszając natychmiastowe przefiltrowanie i odświeżenie widoku listy produktów na bazie unikalnego ID produktu.

```
Padding(
  padding: const EdgeInsets.symmetric(horizontal: 16.0, vertical: 8.0),
  child: TextField(
    controller: _searchController,
    decoration: InputDecoration(
      labelText: 'Szukaj po kodzie produktu',
      prefixIcon: const Icon(Icons.search),
      filled: true,
      fillColor: Colors.grey[100],
      border: OutlineInputBorder(
        borderRadius: BorderRadius.circular(10),
        borderSide: BorderSide.none,
      ), // OutlineInputBorder
    ), // InputDecoration
    onChanged: (value) {
      setState(() {
        _searchTerm = value.trim().toLowerCase();
      });
    },
  ), // TextField
), // Padding
```

Poniższy kod realizuje proces wyświetlania asortymentu w czasie rzeczywistym

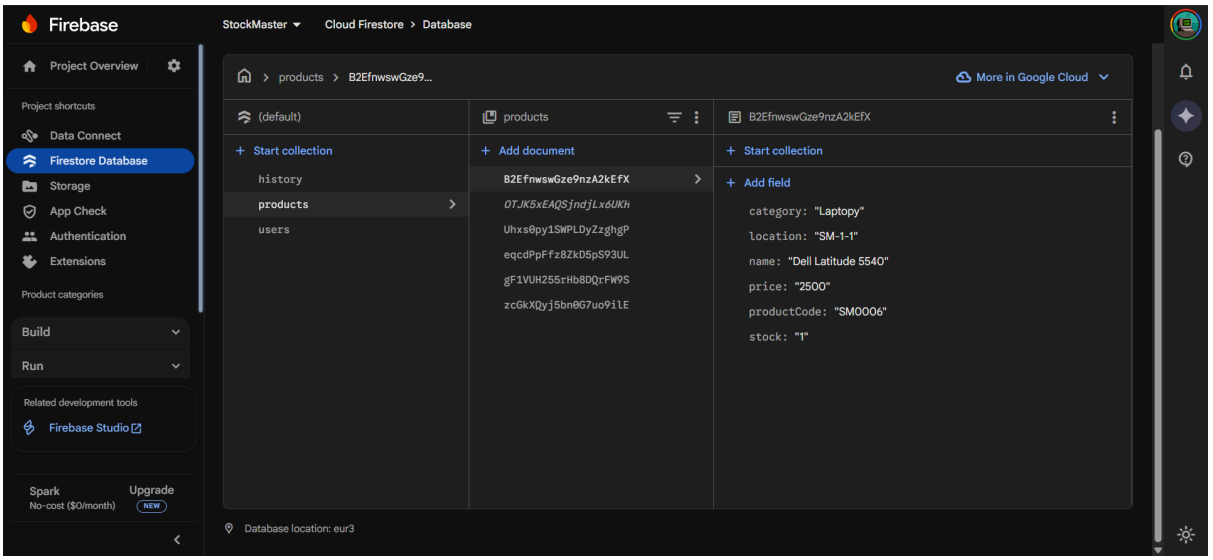
```
Expanded(
  child: StreamBuilder<QuerySnapshot>(
    stream: _firebaseService.getProducts(),
    builder: (context, AsyncSnapshot<QuerySnapshot> snapshot) {
      if (snapshot.connectionState == ConnectionState.waiting) {
        return const Center(child: CircularProgressIndicator());
      }
      if (!snapshot.hasData || snapshot.data!.docs.isEmpty) {
        return const Center(child: Text('Brak produktów w magazynie'));
      }

      final products = snapshot.data!.docs;
      final filteredProducts = products.where((doc) {
        final data = doc.data() as Map<String, dynamic>;
        final code = (data['productCode'] ?? '').toString().toLowerCase();
        return code.contains(_searchTerm);
      }).toList();

      return ListView.builder(
        itemCount: filteredProducts.length,
        itemBuilder: (context, index) {
          final product = filteredProducts[index];
          final productData = product.data() as Map<String, dynamic>;

          final price = productData['price'] ?? '0.00';
          final stock = productData['stock'] ?? '0';
          final category = productData['category'] ?? 'Brak';
          final location = productData['location'] ?? 'Brak';
          final name = productData['name'] ?? 'Brak nazwy';
          final productCode = productData['productCode'] ?? 'Brak kodu';
```

Poniższy ekran obrazuje strukturę bazy danych aplikacji na FireBase. Baza danych zawiera tabele history, która zbiera historie operacji produktów, tabele products w której zawarte są produkty dodane do systemu i tabele users zawierającą listę użytkowników razem z zdefiniowanymi rolami.



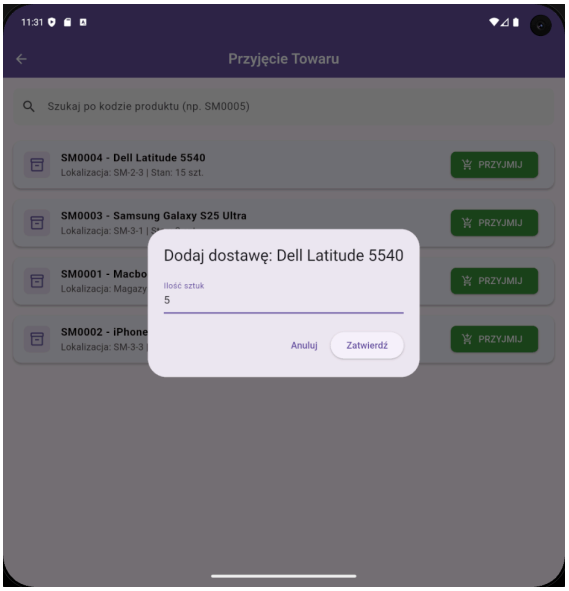
---

4. Moduł: Dostawy (Przyjęcia Magazynowe)

Uproszczony moduł do szybkiego aktualizowania stanów magazynowych z poziomu listy.

Funkcje:

- **Wyszukiwanie:** Szybkie znajdowanie produktu po kodzie, można użyć skanera skanującego etykiety generowane przy tworzeniu produktu.
    - **Przycisk "Przyjmij":** Dostępny przy każdym produkcie. Po kliknięciu wyświetla monit o podanie ilości sztuk do dodania do stanu bieżącego.



---

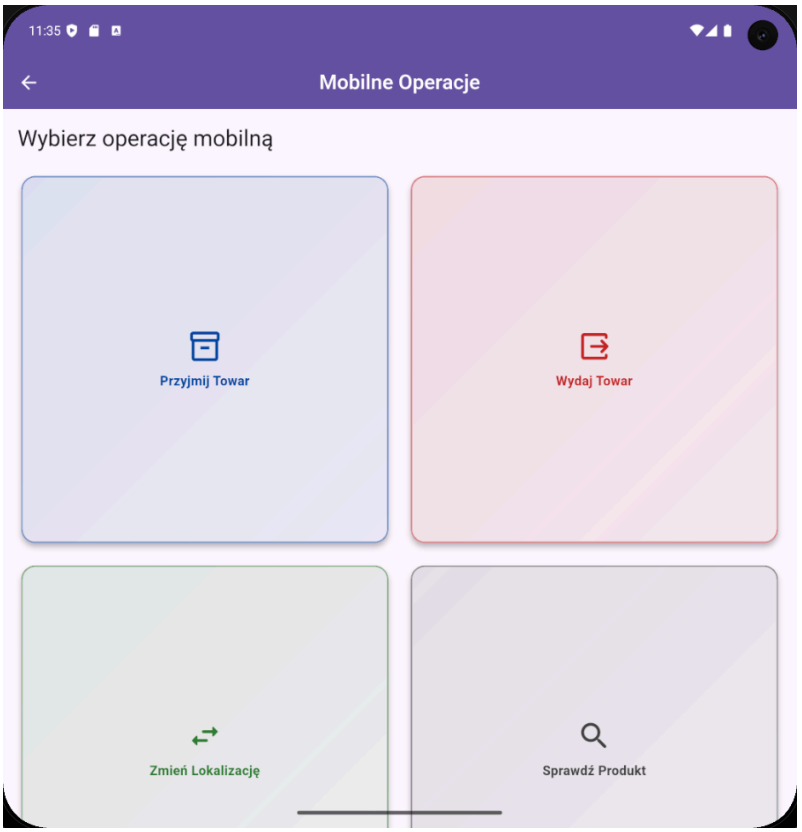
5. Moduł: Mobilne Operacje (Obsługa Skanera)

Moduł dedykowany głównie dla roli **Magazynier**, zoptymalizowany pod użycie wbudowanego skanera kodów kreskowych/QR.

Dostępne Operacje:

1. **Przyjmij Towar:**
    - Skanowanie etykiety towaru.
    - Wprowadzenie ilości przyjmowanej na stan.
2. **Wydanie Towaru (Sprzedaż z Dokumentem):**
    - Skanowanie produktu.
    - Wybór typu dokumentu: **Paragon** lub **Faktura**.
    - *Dla Faktury:* Możliwość wprowadzenia danych kontrahenta (np. NIP).
3. **Zmiana Lokalizacji:**
    - Skanowanie produktu.
    - Wybór nowej lokalizacji (przesunięcie międzymagazynowe).
4. **Sprawdź Produkt (Info-Skan):**

- Skanowanie etykiety w celu natychmiastowego wyświetlenia szczegółów (Stan, Cena, Lokalizacja, generowanie kodu kreskowego).

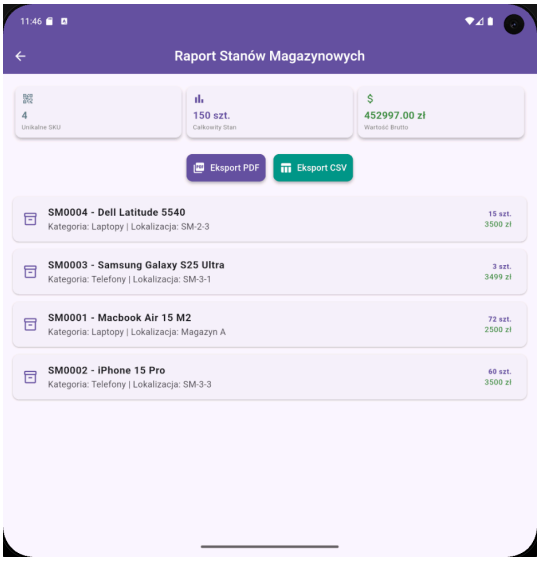


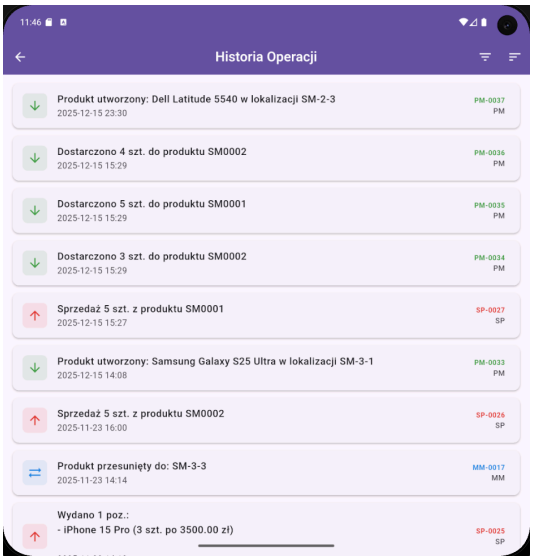
---

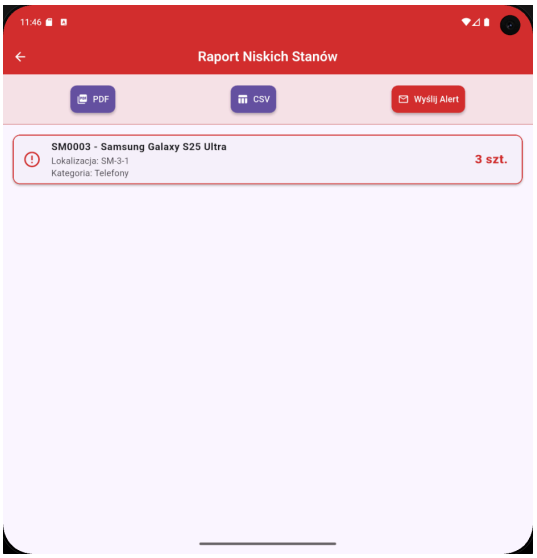
6. Moduł: Raporty i Analizy

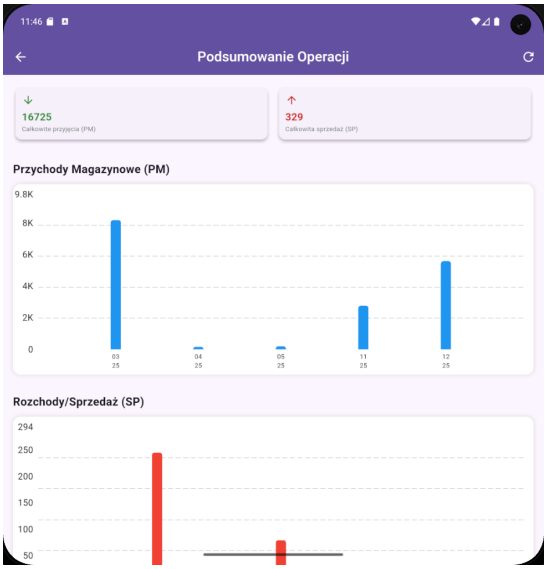
Centrum analityczne aplikacji, oferujące cztery kluczowe zestawienia.

1. **Raport Stanów Magazynowych:** Tabela z aktualnymi ilościami i wartością magazynu (dostępna opcja eksportu).
2. **Raport Operacji (Historia):** Pełny log zdarzeń (Dostarczenie, Sprzedaż, Przesunięcie).
    - *Filtrowanie:* Możliwość sortowania po dacie lub typie operacji (np. tylko PM).
3. **Raport Niskich Stanów:** Lista produktów krytycznych (stan bliski zeru).
4. **Podsumowanie Sprzedaży (Wykresy):** Graficzna wizualizacja przychodów (PM) i sprzedaży (SP).









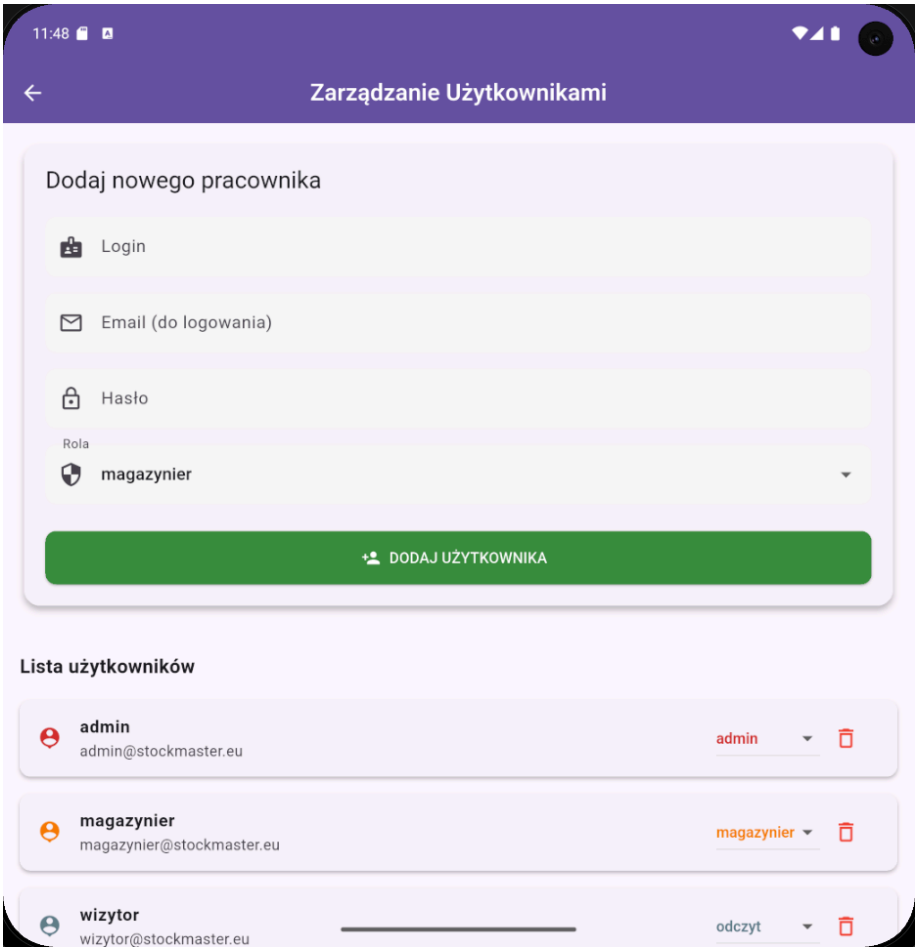
---

7. Moduł: Użytkownicy (Administracja)

Moduł dostępny dla Administratora służący do zarządzania personelem.

Funkcje:

- **Tworzenie Użytkownika:** Formularz (Login, E-mail, Hasło).
- **Zarządzanie Rolami:** Przypisywanie uprawnień:
    - **Admin:** Pełny dostęp.
    - **Magazynier:** Dostęp do Operacji Mobilnych i Produktów.
    - **Odczyt:** Podgląd Raportów (bez edycji).
- **Edycja/Usuwanie:** Możliwość zmiany roli istniejącego użytkownika lub usunięcie konta.


11:48
Zarządzanie Użytkownikami
Dodaj nowego pracownika
Login
Email (do logowania)
Hasło
Rola
magazynier
DODAJ UŻYTKOWNIKA
Lista użytkowników
admin
admin@stockmaster.eu
admin
magazynier
magazynier@stockmaster.eu
magazynier
wizytor
wizytor@stockmaster.eu
odczyt