

**WYDZIAŁ
ELEKTROTECHNIKI
I INFORMATYKI
POLITECHNIKI RZESZOWSKIEJ**

Patryk Walat, Adrian Socha

Aplikacja Oracle APEX zakładu produkcyjnego

Sprawozdanie z projektu

Rzeszów, 2024

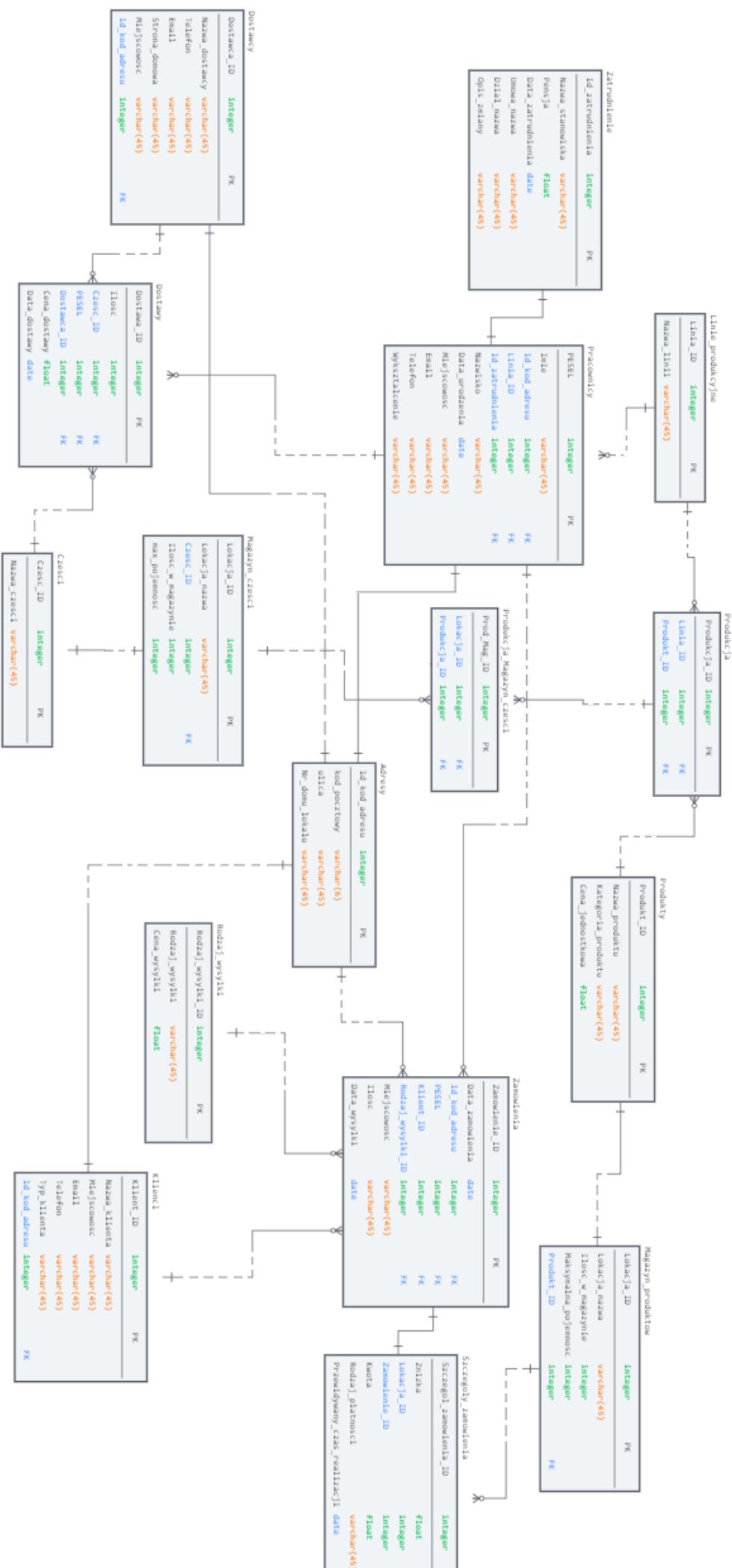
Spis treści

Spis treści	3
1. Wstęp	4
2. Schemat bazy danych.....	4
3. Pakiet Zarządzania produkcją i pracownikami	6
3.1 Cel pakietu.....	6
3.2 Funkcje wchodzące w skład pakietu	6
3.3 Procedury wchodzące w skład pakietu	8
4. Pakiet Zarządzania magazynami, dostawami i zamówieniami	10
4.1 Cel pakietu.....	10
4.2 Funkcje wchodzące w skład pakietu	10
4.3 Procedury wchodzące w skład pakietu.....	13
5. Wykorzystane Obiekty	15
6. Panel aplikacji.....	17
6.1 Strona Sprawdź i Zrealizuj Zamówienie	18
6.2 Strona Uzupełnij Stan Magazynu Części	22
6.3 Strona Wprowadzanie zamówień	24
6.4 Strona Procentowe zapełnienie Magazynu Produktów.....	27
6.5 Strona Wprowadzanie Dostaw Zniżka	28
6.6 Strona Wprowadzanie Adresów	32
6.7 Strona Wprowadzanie Klientów.	35
6.8 Strona Wprowadzanie Pracowników	41
6.9 Strona Procentowe Zapełnienie Magazynu Części	48
6.10 Strona Aktualizuj Magazyny Po Produkcji.....	49
6.11 Strona Wyświetl Klientów.....	52
6.12 Strona Możliwości Produkcyjne	53
6.13 Strona Potrzebne Części Do Produkcji	54
6.14 Strona Braki Potrzebne Części.....	56
6.15 Strona Dyspozycja Asortymentu	58
6.16 Strona Wyświetl Zamówienia.....	61
7 Eksport aplikacji i import.....	62
8. Podsumowanie.....	66
9. Dane do logowania.....	66
10. Kod PL/SQL.....	66

1. Wstęp

Projekt bazy danych zakładu produkcyjnego zakłada możliwość prowadzenia produkcji oraz nadzorowania wytwarzania towarów w fabryce. W celu sprawnego zarządzania działalnością firmy potrzebna jest baza danych odzwierciedlająca procesy zachodzące w przedsiębiorstwie oraz zależności jakie łączą je ze sobą. Podstawę firmy stanowią pracownicy, którzy to odpowiadają za produkcję różnych towarów, realizację zamówień na produkty składanych przez klientów przedsiębiorstwa, odbieranie dostaw towarów od dostawców zewnętrznych oraz obsługujący magazyn zaopatrzony w części potrzebne do procesu produkcyjnego oraz magazynu wytworzonych produktów przechowywanych do czasu zakupu przez klienta. Utrzymanie płynności funkcjonowania fabryki nie było by możliwe bez klientów, którzy chcąc kupić wytwarzane towary składają zamówienia.

2. Schemat bazy danych



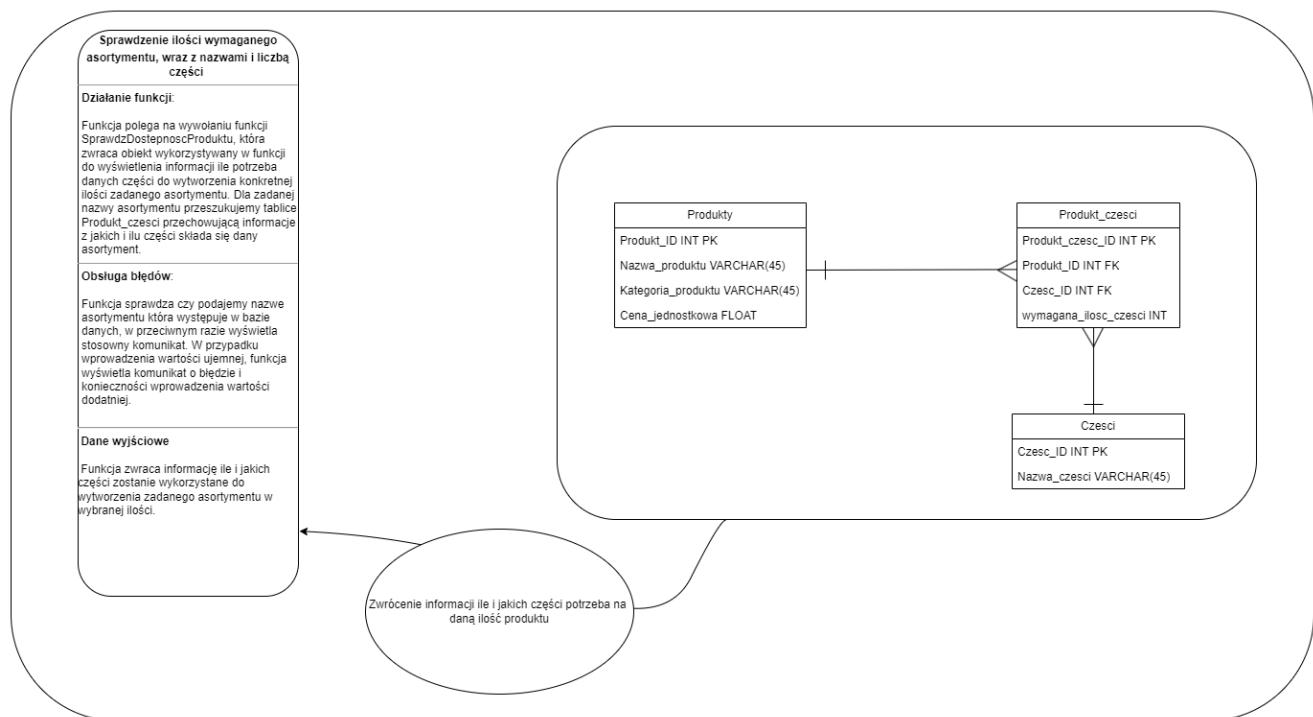
3. Pakiet Zarządzania produkcją i pracownikami

3.1 Cel pakietu

Pakiet zawiera zestaw funkcji i procedur, które umożliwiają skuteczne zarządzanie procesem produkcji oraz pracownikami w zakładzie produkcyjnym. Pakiet ten został stworzony w celu usprawnienia i zautomatyzowania kluczowych aspektów działalności produkcyjnej przedsiębiorstwa

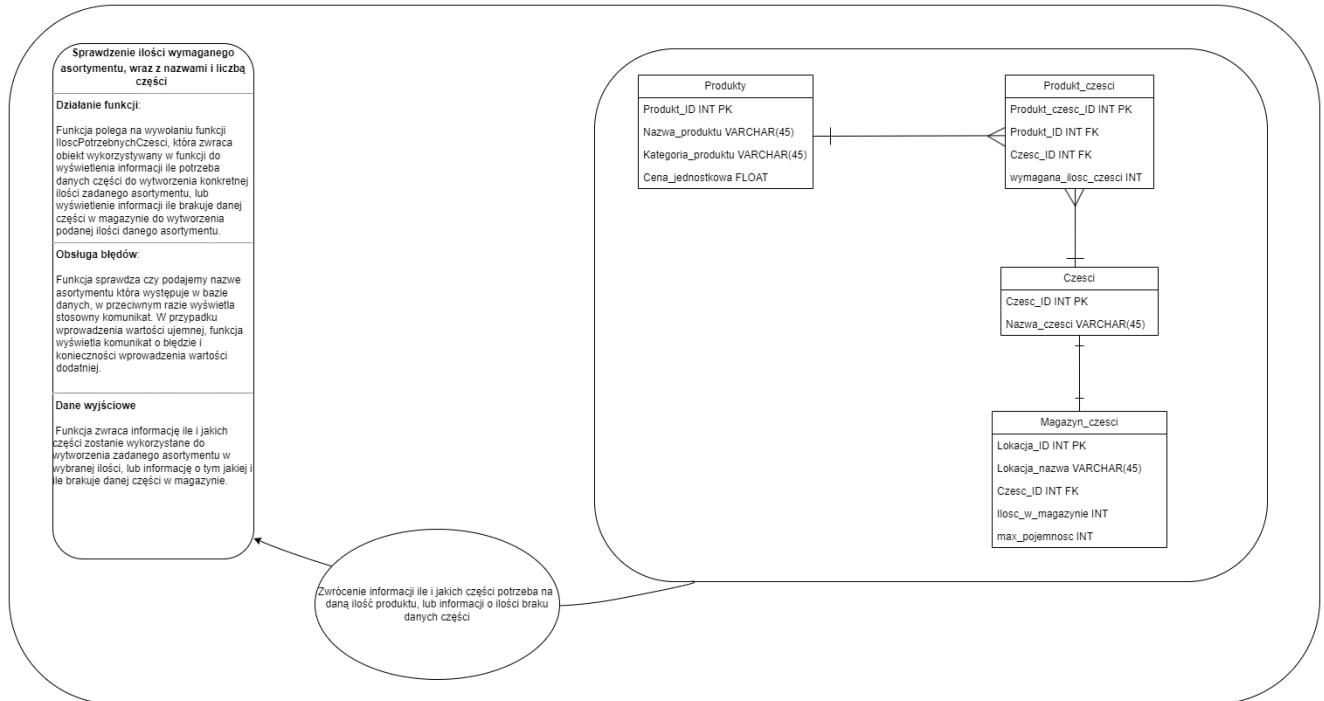
3.2 Funkcje wchodzące w skład pakietu

- Potrzebne części do Produkcji – funkcja przyjmująca dwa parametry (Produkt_ID, ilość danego asortymentu) wyświetla informację o tym jakich i ile części będziemy potrzebować do wyprodukowania zadanego asortymentu w konkretnej ilości.

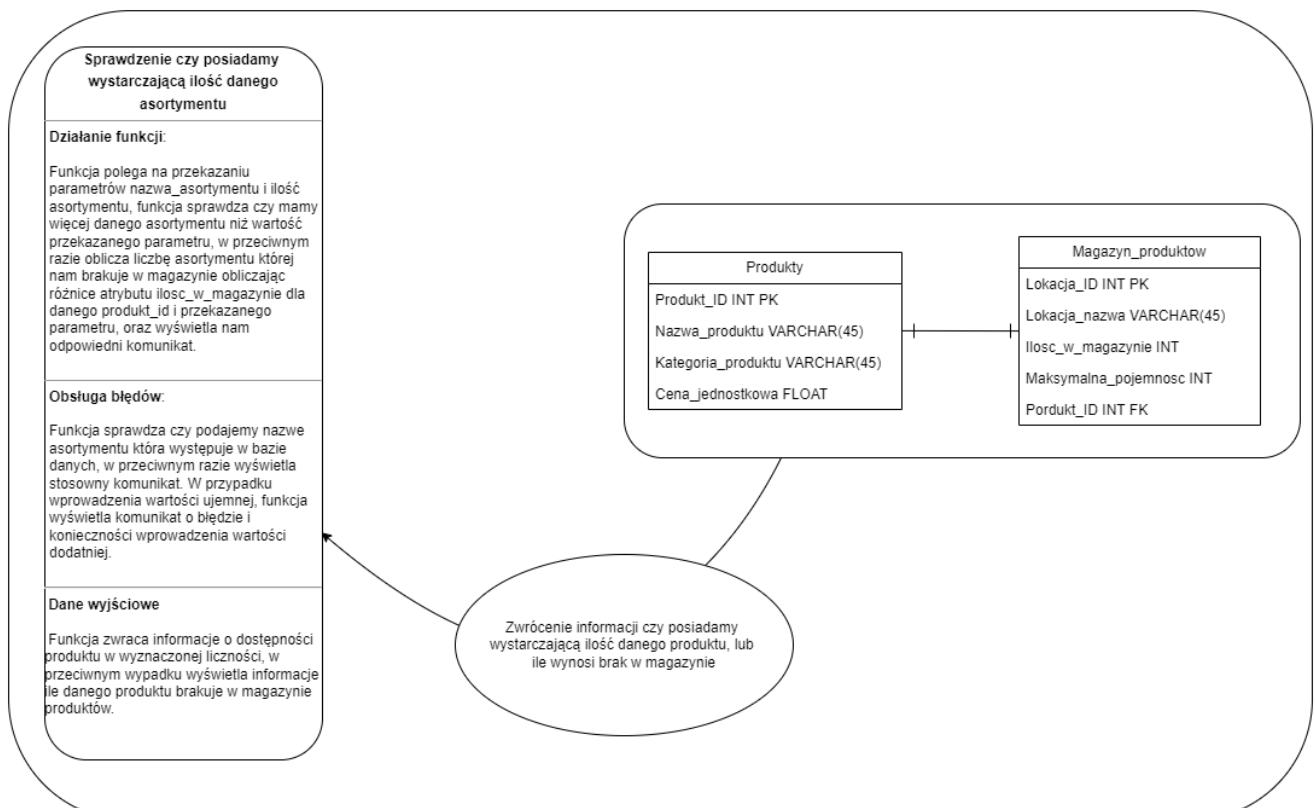


- Braki Potrzebne Części – funkcja przyjmująca jako parametr kurSOR działania funkcji Potrzebne części do Produkcji, oprócz udostępniania informacji o potrzebnych częściach, przeszukuje tabele Magazyn części i wyświetla informacje o ewentualnych brakach danej części, jeśli do

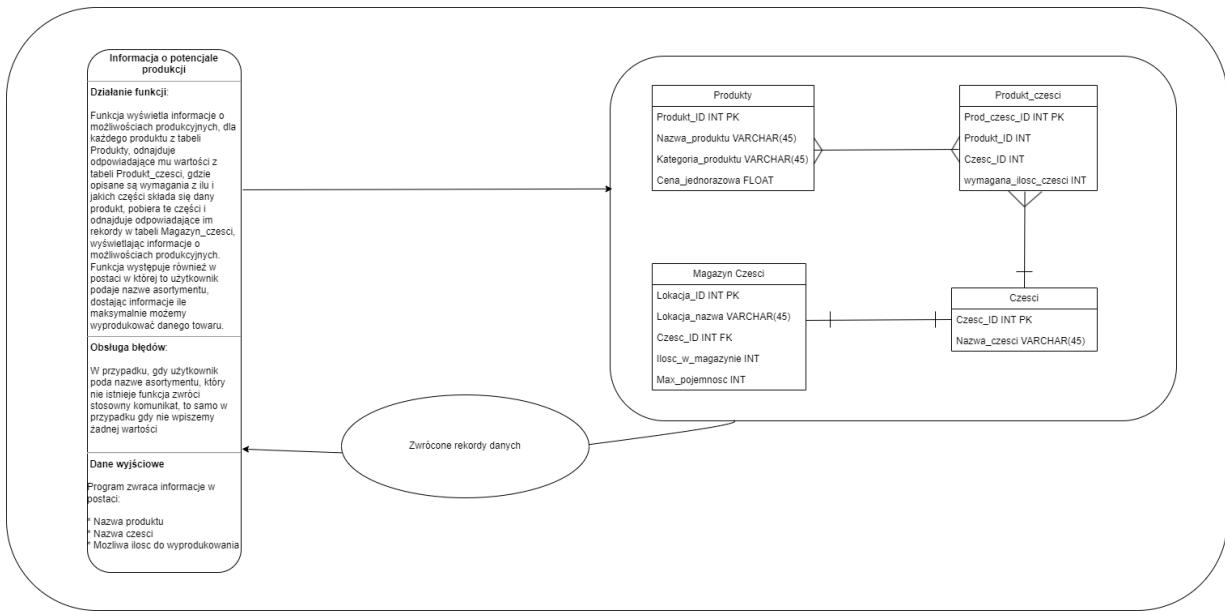
wyprodukowania danego asortymentu potrzeba więcej konkretnej części niż jest w magazynie.



- Dyspozycja asortymentu – funkcja zwracająca informację czy dysponujemy wystarczającą ilością danego asortymentu, w przypadku gdy nie posiadamy wystarczającej ilości podanej przez klienta wyświetlana jest informacji ile wynosi brak danego asortymentu.

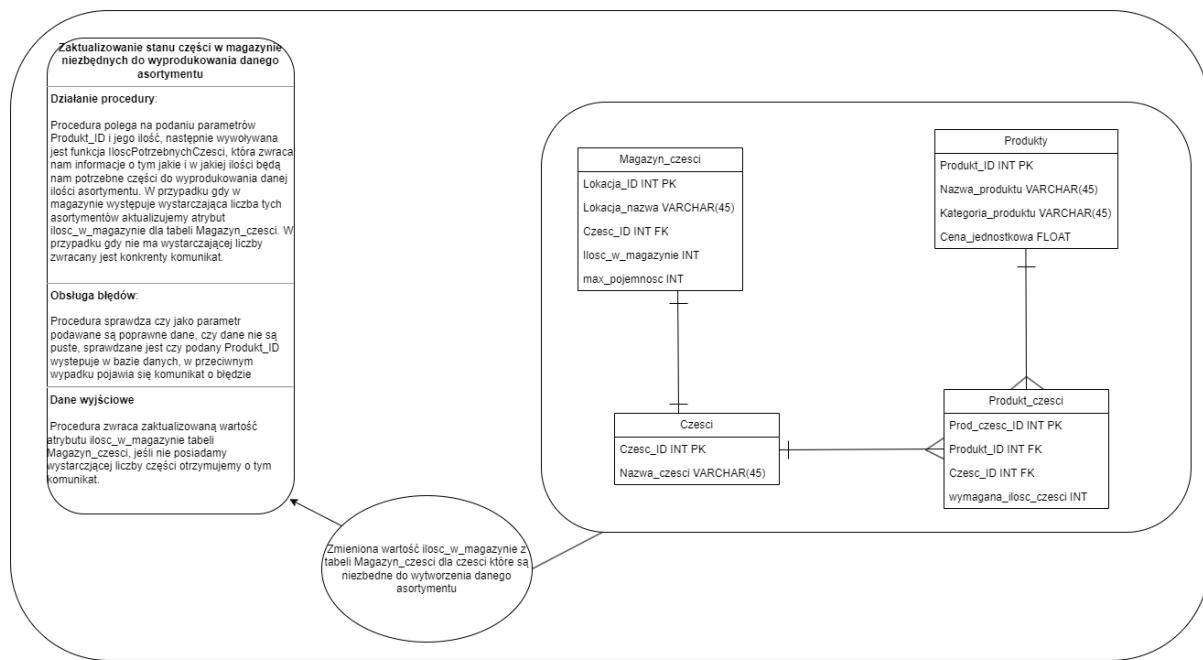


- Możliwości produkcyjne – funkcja pozwalająca na uzyskanie informacji ile jesteśmy w stanie wyprodukować asortymentów bazując na stanie magazynu części, czyli ile produktów zostanie jeszcze wyprodukowanych zanim wyczyścimy swoje zapasy.

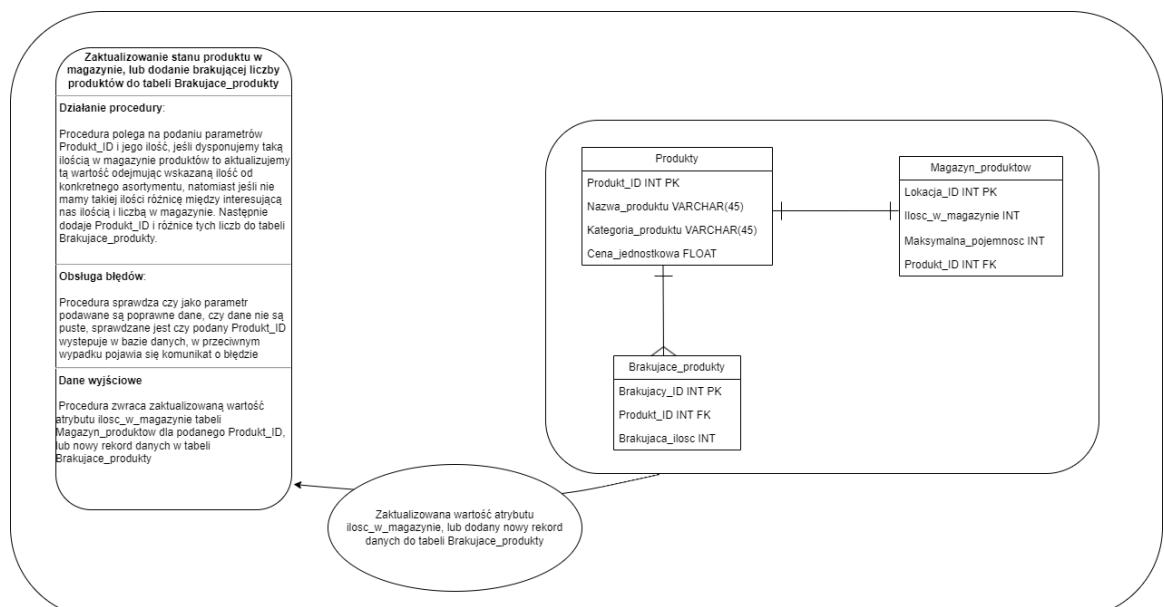


3.3 Procedury wchodzące w skład pakietu

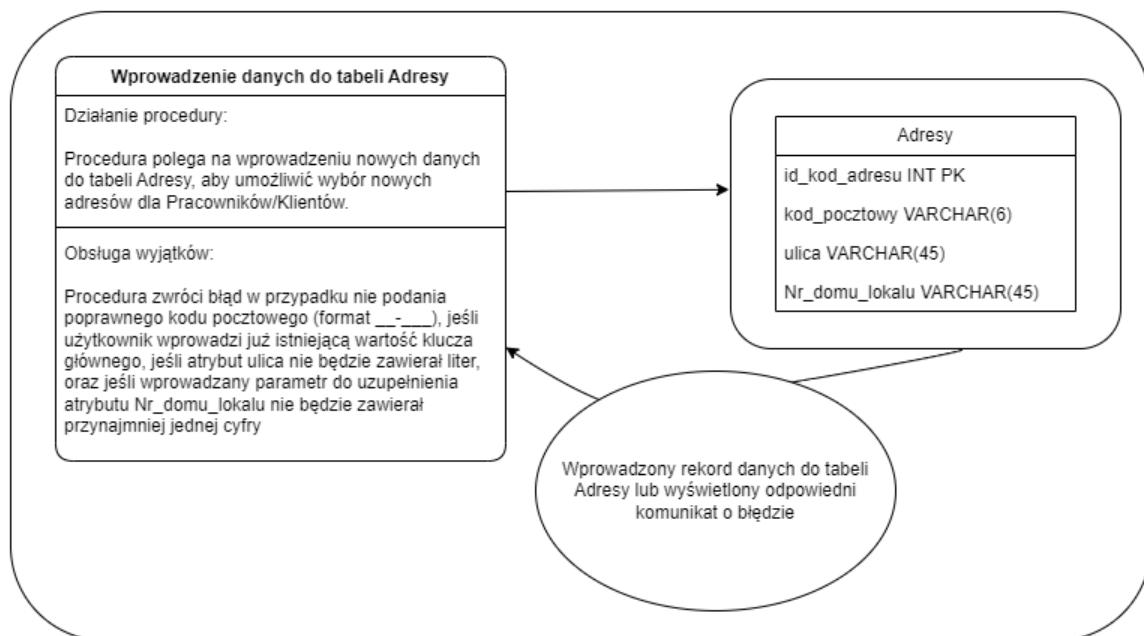
- Aktualizuj magazyny po produkcji – procedura, która jest wykorzystywana podczas tworzenia nowego produktu, sprawdza jakie części i w jakiej ilości są wymagane do wyprodukowania danego produktu w zadanej ilości, a następnie w tabeli Magazyn_części aktualizuje wartości atrybutu ilość_w_magazynie dla danych części, oraz tabele Magazyn_produktow inkrementując stan magazynowy danego produktu (zwiększenie pola o ilość którą wytworzyliśmy).



- Wprowadzanie pracowników – procedura, polegająca na równoczesnym wprowadzeniu nowego rekordu danych do tabeli Pracownicy i Zatrudnienie, podczas zatrudniania nowego pracownika, istnieje możliwość wprowadzenia nowego rekordu danych do tabeli Adresy, jeśli pracownik wprowadzi dane dotyczące zamieszkania, które jeszcze nie występują w bazie danych.
- Sprawdź i zrealizuj zamówienie – procedura która sprawdza czy zakład jest w stanie zrealizować zamówienie z produktów które są w zapasie, w przeciwnym razie, różnicę danego asortymentu i jego ilości dodajemy do kolejnych rekordów tabeli Brakujące Produkty.



- Wprowadzanie adresów – procedura wprowadzająca nowe rekordy danych do tabeli Adresy, walidująca przekazywane parametry np. czy kod pocztowy zawiera poprawny format.



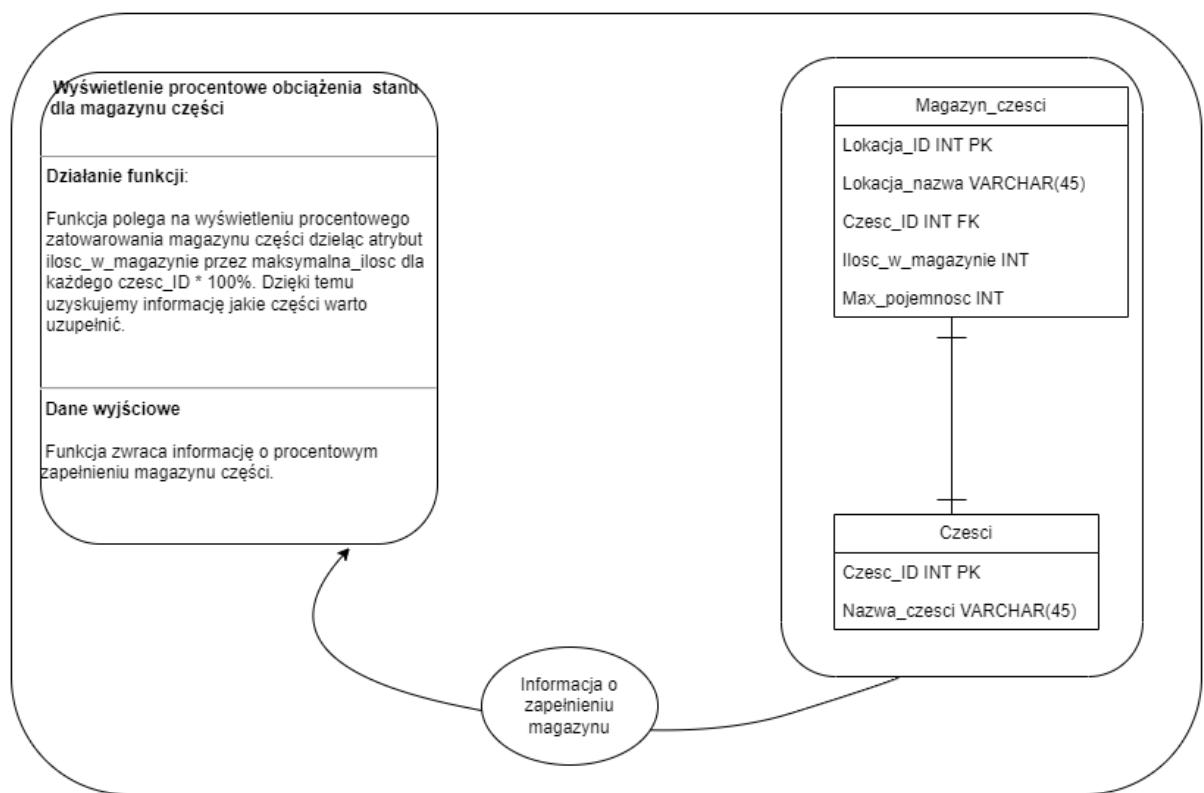
4. Pakiet Zarządzania magazynami, dostawami i zamówieniami

4.1 Cel pakietu

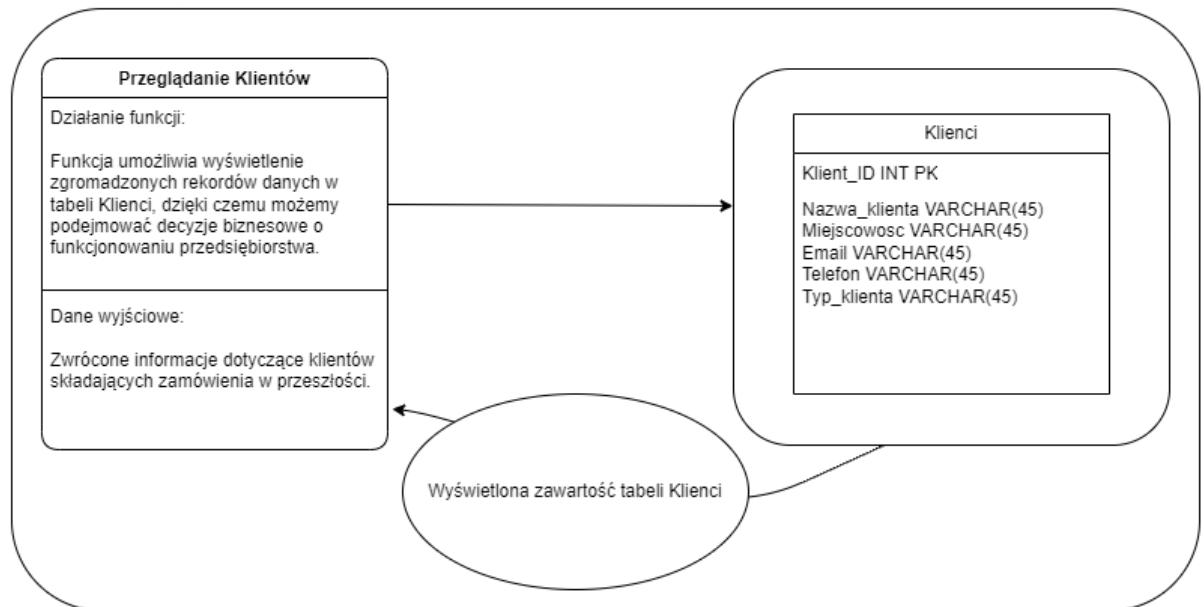
Pakiet został stworzony w celu usprawnienia i zautomatyzowania zarządzania magazynami, dostawami oraz zamówieniami w zakładzie produkcyjnym. Zapewnia kompleksowe narzędzia do monitorowania stanów magazynowych, obsługi dostaw od dostawców oraz zarządzania zamówieniami klientów.

4.2 Funkcje wchodzące w skład pakietu

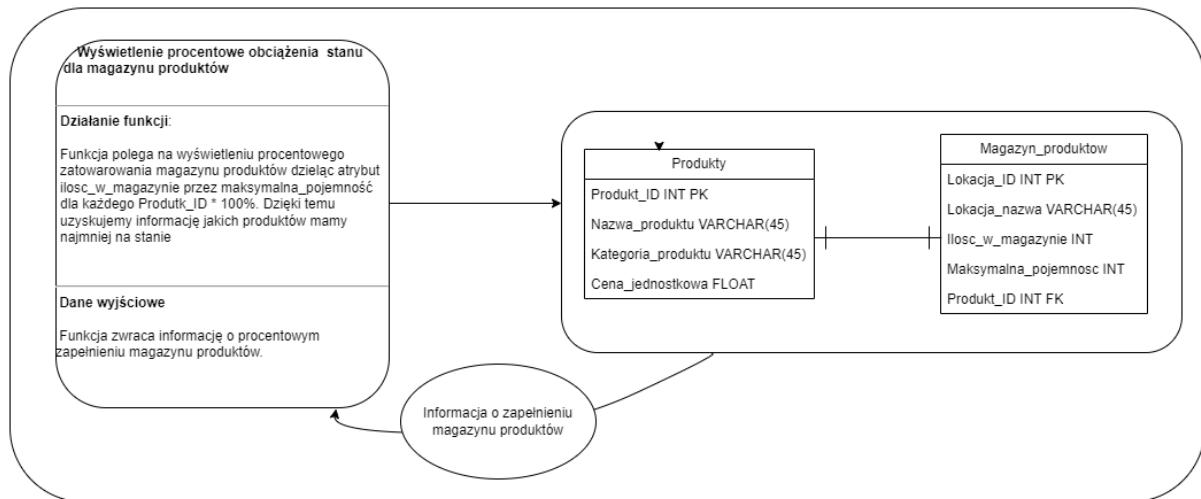
- Procentowe zapełnienie magazynu części – funkcja polegająca na wyświetleniu procentowego obciążenia magazynu części, dzięki temu pozyskujemy informacje jakich części nie należy zamawiać w najbliższym czasie



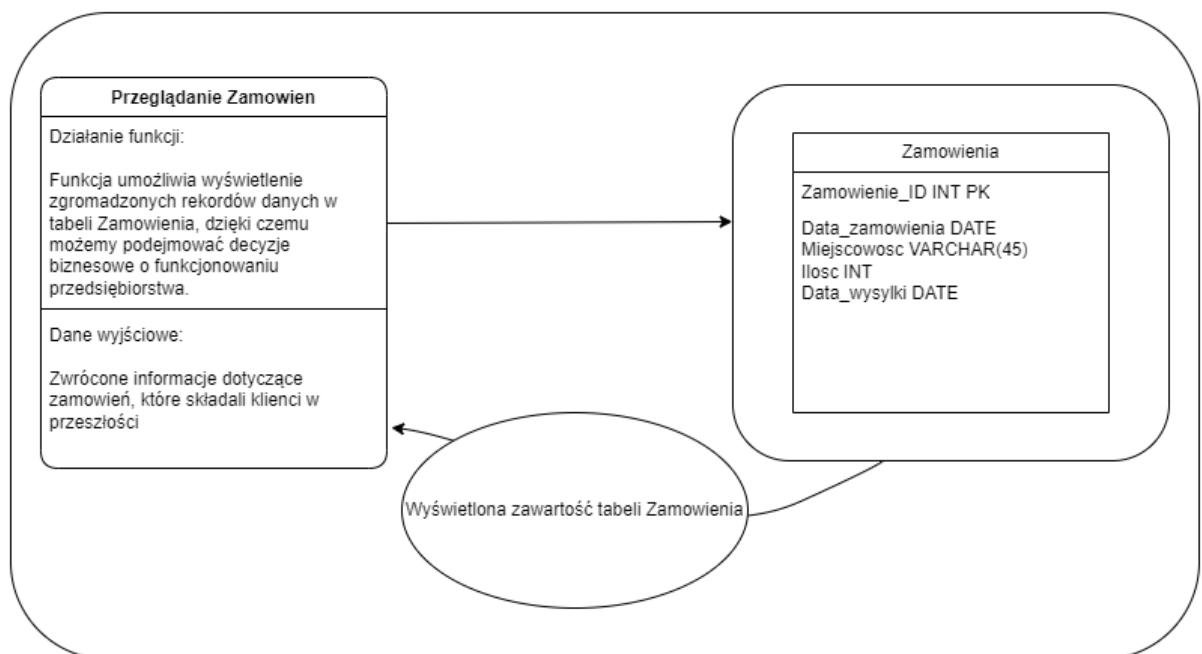
- Wyświetl Klientów – funkcja pozwalająca na wyświetleniu całej tabeli Klientów, dzięki czemu możemy sprawdzić najczęstszych klientów, oraz jak dużą grupą odbiorców dysponujemy.



- Procentowe zapełnienie magazynu produktów – funkcja zwracająca informacje o procentowym zapełnieniu magazynu produktów, dzięki czemu uzyskujemy informacje dla jakich asortymentów należy wstrzymać produkcję, a dla jakich ją zwiększyć.

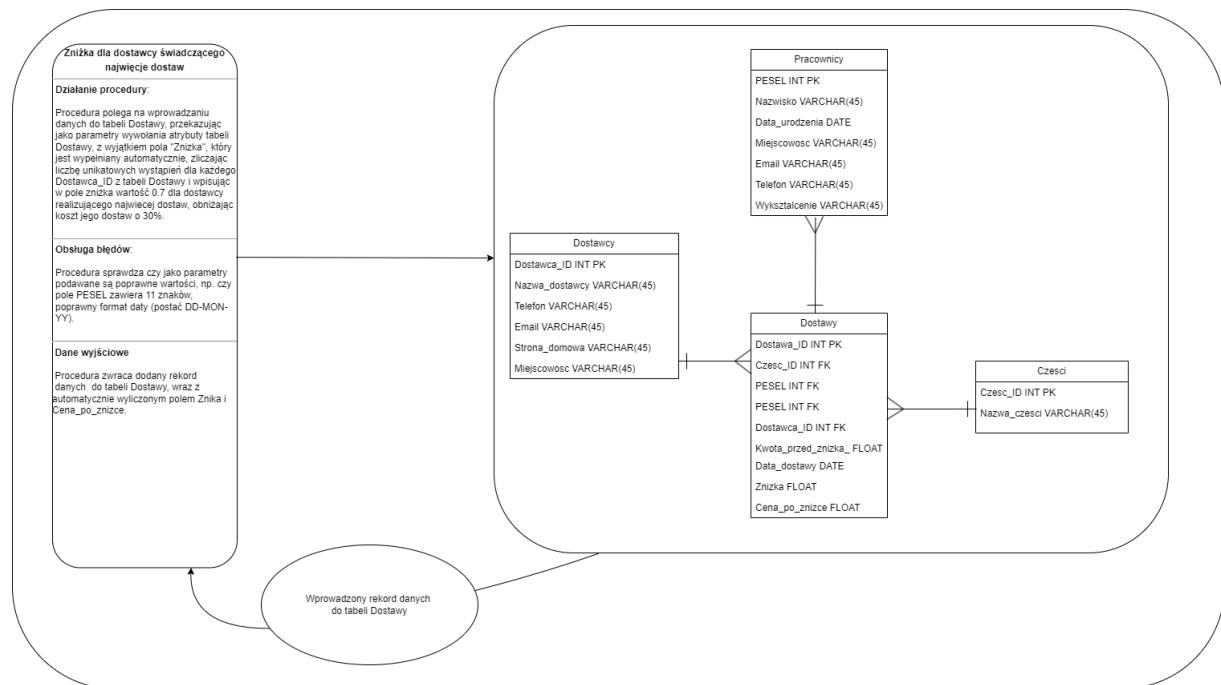


- Wyświetl Zamowienia – funkcja pozwalająca na wyświetlenie informacji dotyczących aktualnych i byłych zamówień, może posłużyć jako źródło informacji dotyczących zmian i trendów branżowych, oraz zachowań rynku.

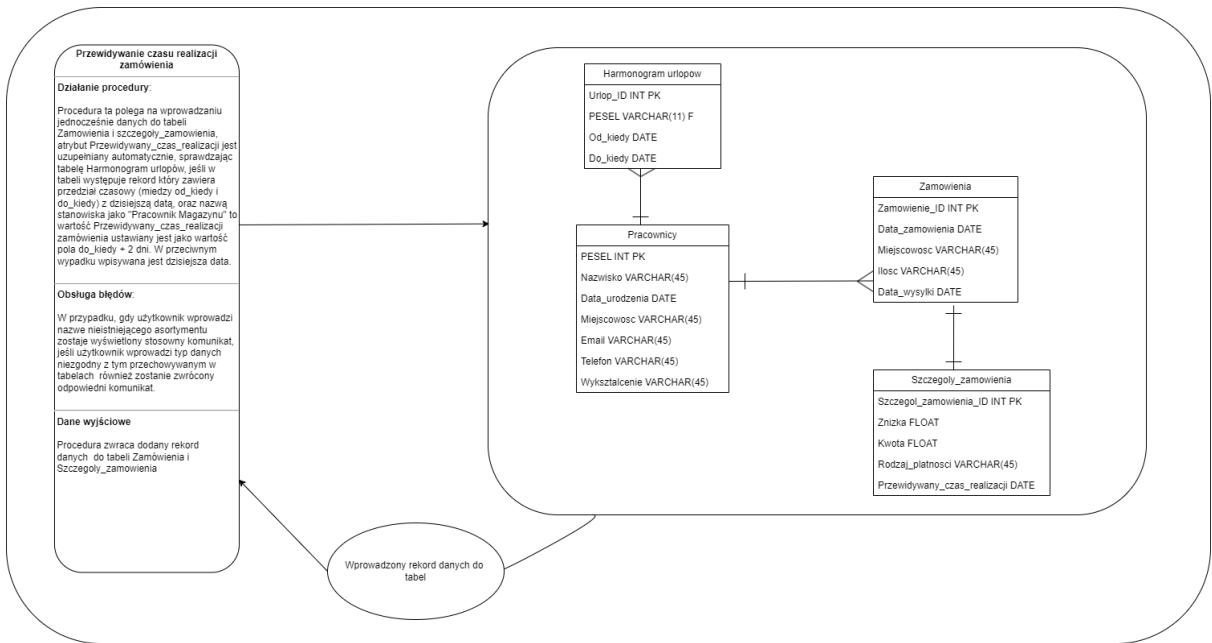


4.3 Procedury wchodzące w skład pakietu.

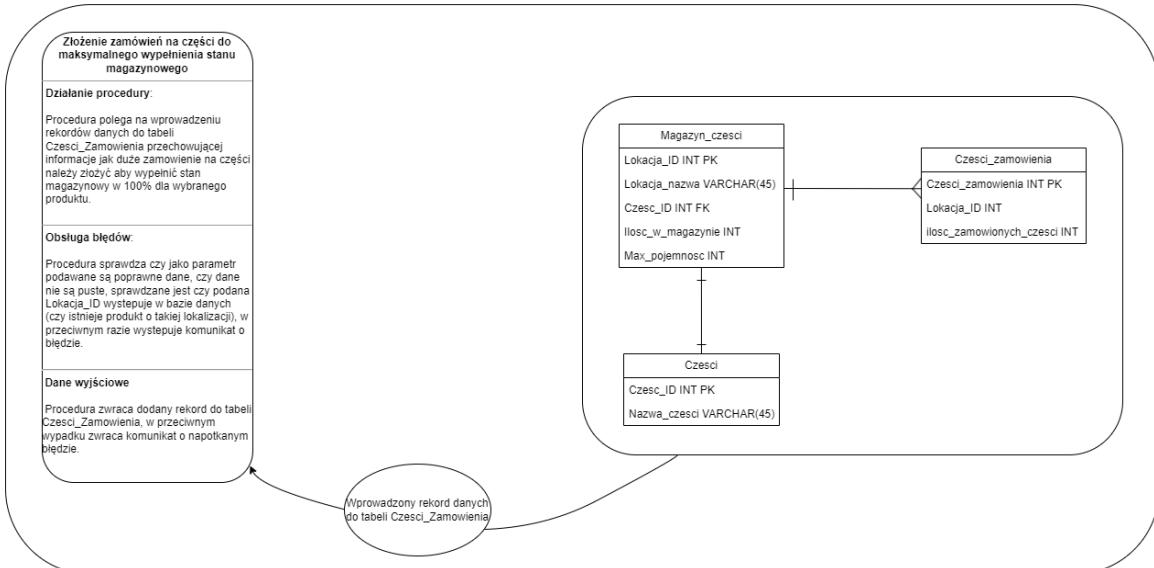
- Wprowadzenie dostaw zniżka – procedura polegająca na automatycznym uzupełnieniu atrybutu cena dostawy, podczas wprowadzania rekordów danych do tabeli Dostawy, jeśli podamy id dostawcy który do tej pory zrealizował dla przedsiębiorstwa najwięcej zleceń, dostaje on zniżkę dla aktualnego zamówienia w postaci 0.7 ceny pierwotnej.



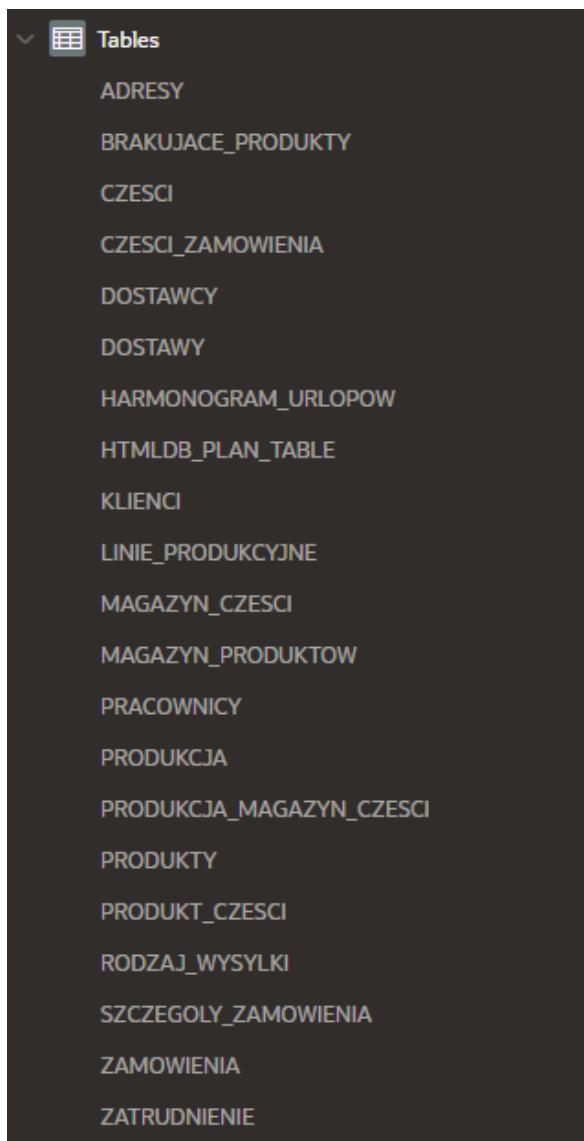
- Wprowadzanie Zamowien Przewidywany Czas Realizacji – procedura polegająca na automatycznym uzupełnieniu pola Przewidywany czas realizacji zamówienia podczas wprowadzania rekordu do tabeli Zamowienia, sprawdzana jest tabela Harmonogram urlopów, jeśli dzisiejsza data nie zawiera się w którymś rekordzie danych w tabeli Harmonogram urlopów dla pracownika o stanowisku 'Pracownik Magazynu' to atrybut Przewidywany_czas_realizacji_zamowienia będzie ustalony na dzisiejszą date, w przeciwnym wypadku będzie to dzisiejsza data + 3 dni.



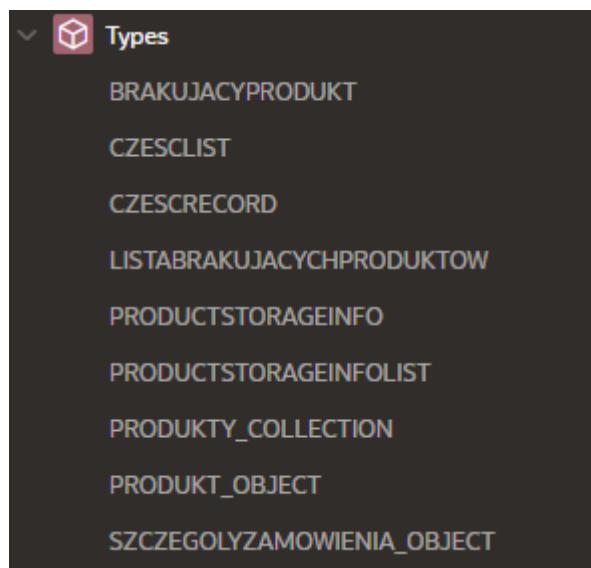
- Uzupełnij Stan Magazynu Częci – funkcja polegająca na uzupełnieniu stanu magazynowego danej części do jej maksymalnego obciążenia, poprzez dodanie do tabeli Czesci_Zamowienia wartości różnicy pól max_pojemnosc – ilość_w_magazyni dla danej Części.



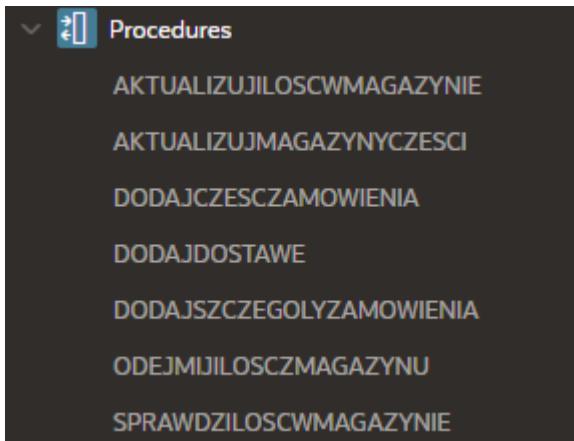
5. Wykorzystane Obiekty



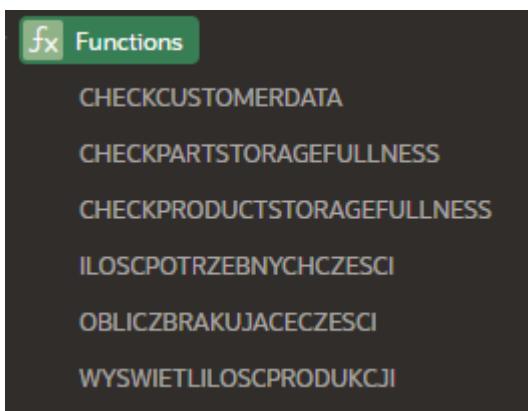
Rysunek 1 Utworzne Tabele w aplikacji



Rysunek 2 Utworzone obiekty

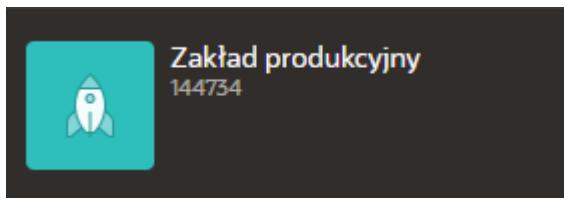


Rysunek 3 Przykłady wykorzystanych procedury



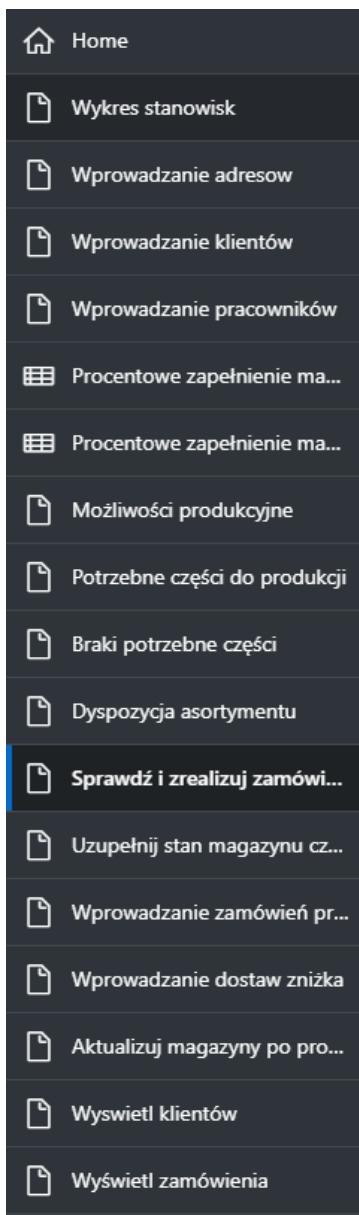
Rysunek 4 Przykłady wykorzystanych funkcje

6. Panel aplikacji



Rysunek 5 Aplikacja Zakładu Produkcyjnego

Aplikacja przechowuje wszystkie strony i funkcjonalności, oraz ich możliwe zastosowania

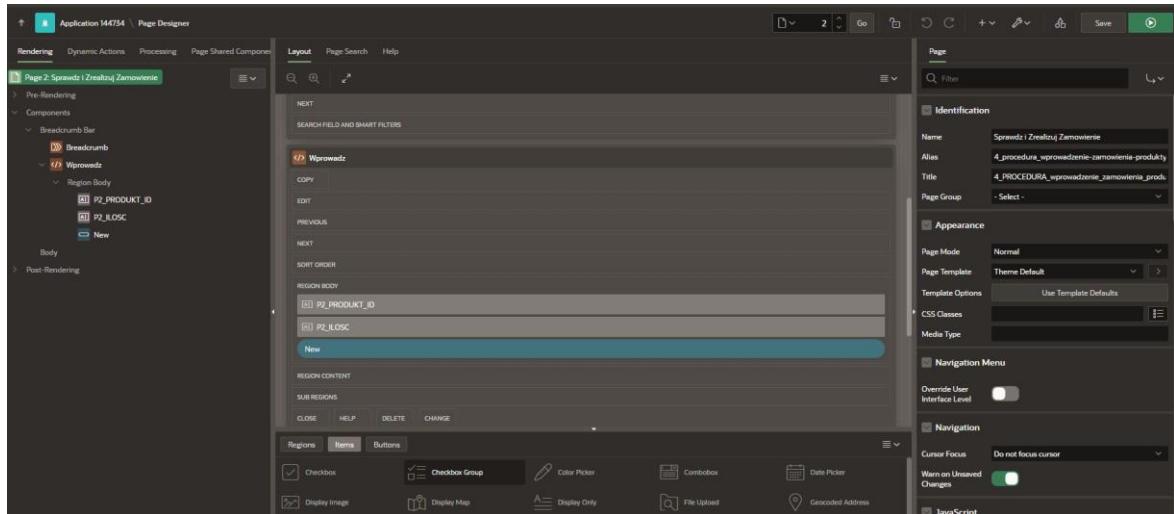


Rysunek 6 Menu boczne strony

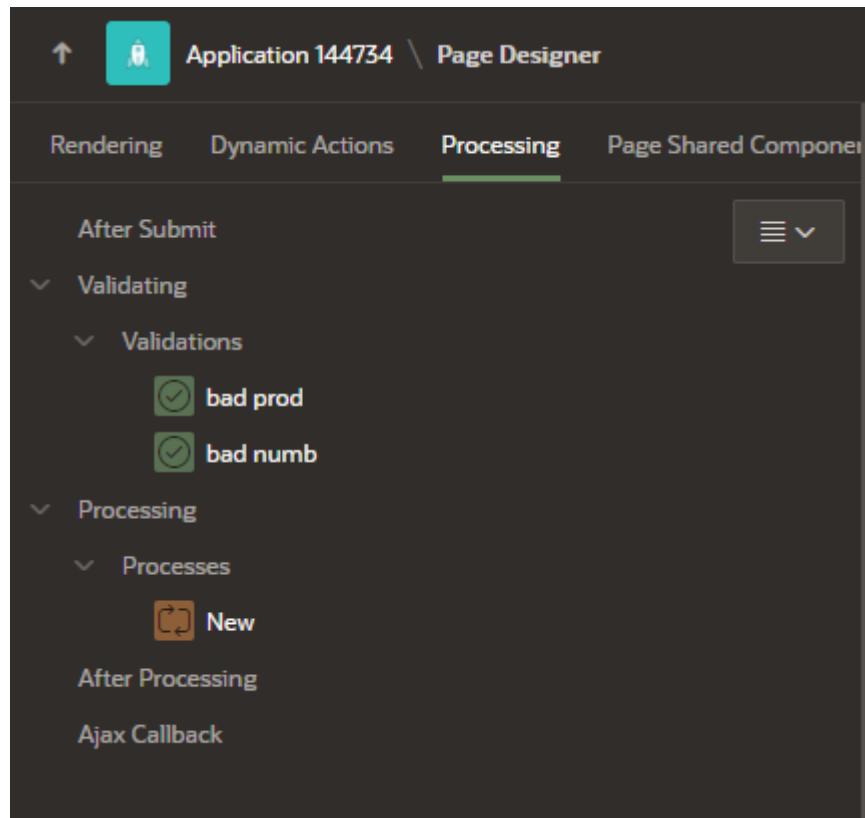
Menu boczne stanowi idealny skrót do szybkiego wybierania interesującej nas strony z dostępnymi funkcjonalnościami

6.1 Strona Sprawdź i Zrealizuj Zamówienie

Strona, dzięki której zakład może realizować zamówienia które są możliwe do realizacji ze składu magazynu produktów, w przypadku gdy nie jest możliwa natychmiastowa realizacja zamówienia dodajemy liczbę brakującego asortymentu do tabeli Brakujące Produkty

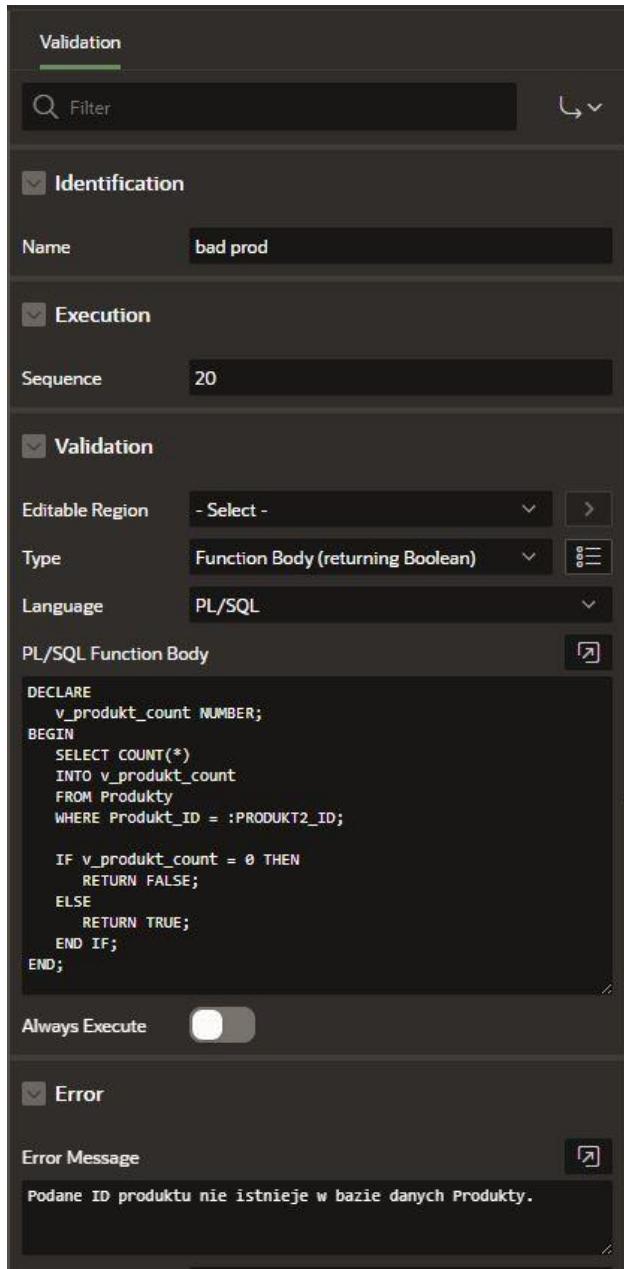


Rysunek 7 Struktura strony Sprawdź i Zrealizuj Zamówienie



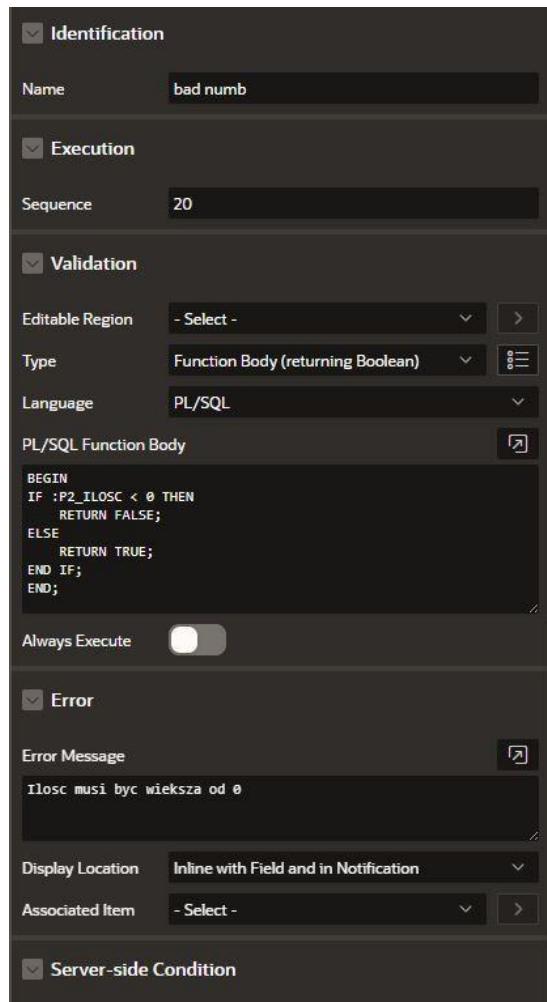
Rysunek 8 Zalożenia walidacyjne

Powyższy zrzut ekranu prezentuje zastosowaną validację parametru Produkt_ID, kod PL/SQL zwróci błąd w przypadku, gdy wartość przekazanego parametru nie będzie występować w tabeli Produkty w atrybucie Produkt_ID, w takim przypadku wyświetli komunikat „Podane ID produktu nie istnieje w bazie danych”.

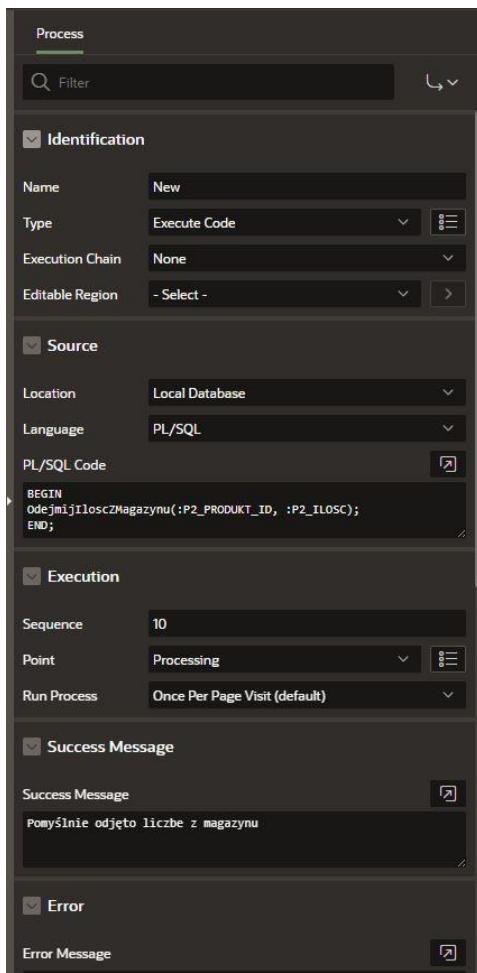


Rysunek 9 Walidacja dla zlego Produkt_ID

Zrzut ekranu przedstawiający walidację bad numb, która pilnuje aby przekazywany parametr ilość_asortymentu zamówienia nie była wartością mniejszą od 0.



Rysunek 10 Walidacja dla niepoprawnej liczby asortymentu zamówienia



Rysunek 11 Przedstawienie wywołania procedury w stronie dla przekazanych parametrów

Poniższy zrzut ekranu pokazuje jak wygląda walidacja danych na „front-endzie”, jeśli przekazujemy źle zdefiniowane wartości parametrów otrzymujemy komunikat o konieczności przekazania poprawnych danych, wraz z konkretnymi informacjami o błędach (tutaj źle Produkt_ID i ilość asortymentu).

A screenshot of a web application window titled 'test'. The main content area says 'Sprawdź i zrealizuj zamówienie'. Below it is a form with two input fields: 'Product id' containing '-1' and 'Podaj ilosc asortymentu' containing '-1'. At the bottom left is a 'New' button. In the top right corner, there is a red error message box with the title '2 errors have occurred':

- Podane ID produktu nie istnieje w bazie danych Produkty.
- Ilosc musi byc wieksza od 0

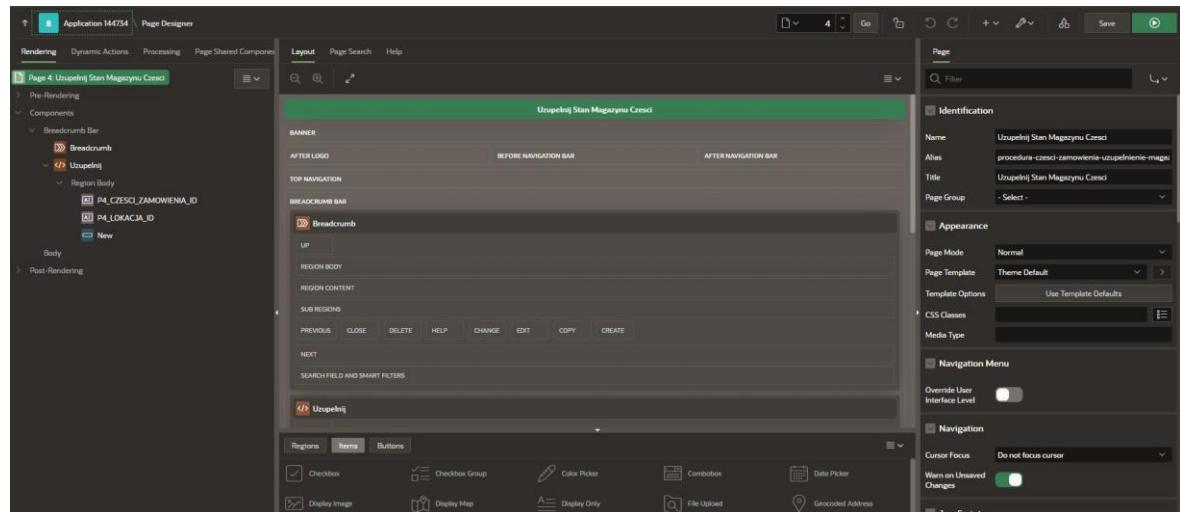
Rysunek 12 Graficzna interpretacja walidacji danych

W momencie przekazania poprawnych wartości parametrów ukazuje nam się informacja o poprawnym wywołaniu procedury, wraz z informacją, że wszystko przebiegło pomyślnie.

The screenshot shows a web application interface with a blue header bar containing the text 'test'. Below the header, there is a green success message box with the text 'Pomyślnie odjęto liczbę z magazynu'. The main content area has a title 'Sprawdź i zrealizuj zamówienie' and a sub-section 'Wprowadz'. It contains two input fields: 'Product Id' with value '15' and 'Podaj ilosc asortymentu' with value '100'. A 'New' button is also present.

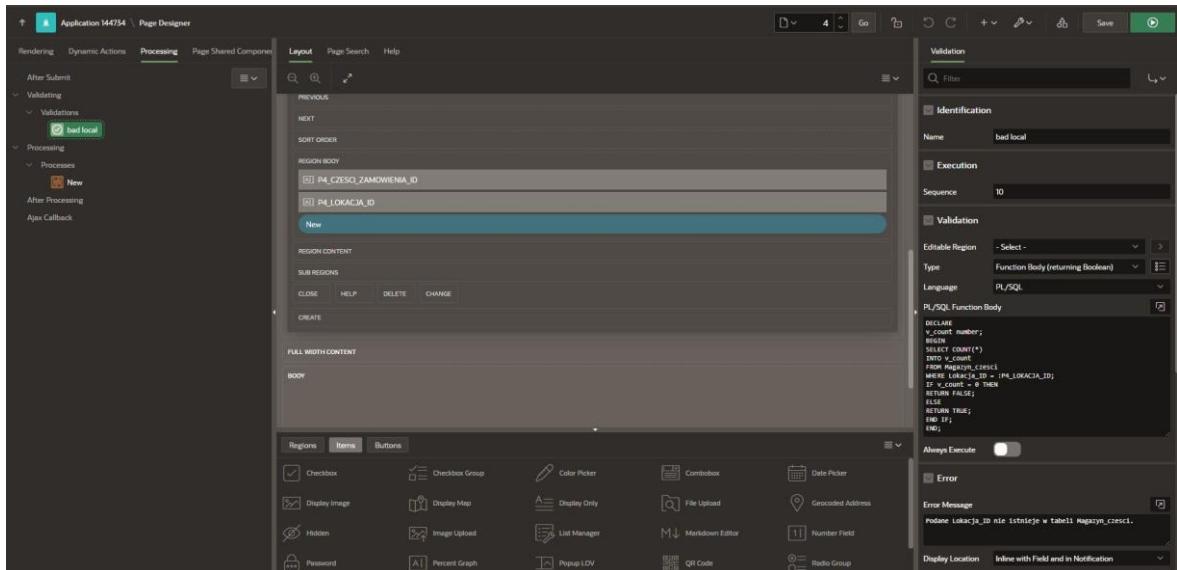
6.2 Strona Uzupełnij Stan Magazynu Części.

Strona, dzięki której użytkownik (w domyśle pracownik magazynu) może złożyć zamówienie na wybrane przez siebie części do wypełnienia ich pełnego zatowarowania. Strona posiada walidacje błędu, w przypadku gdy podajemy Lokalizację części która nie istnieje w bazie.

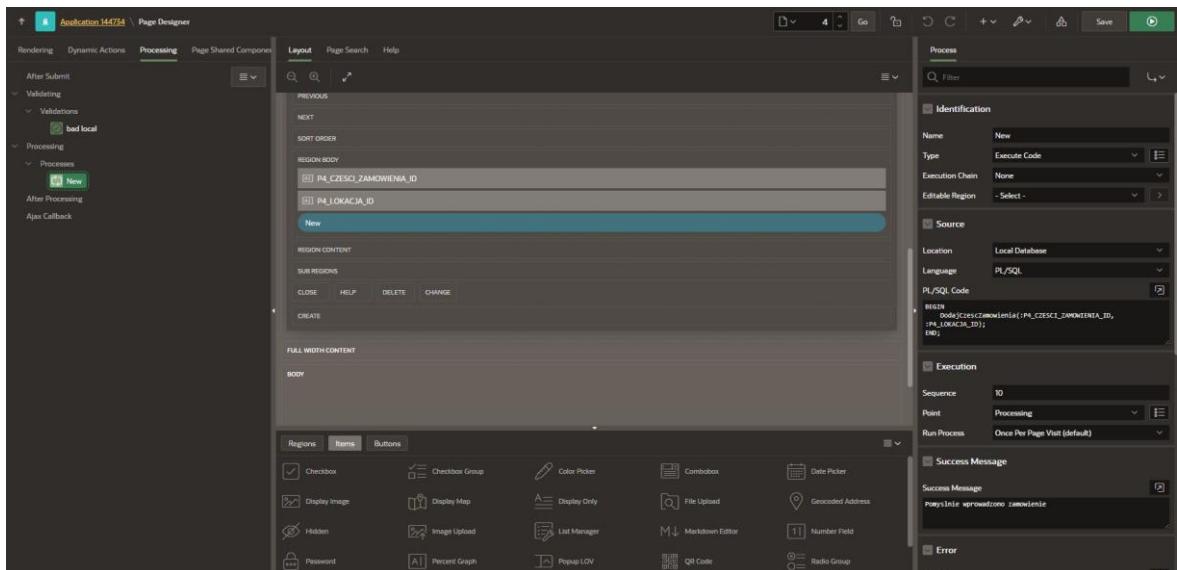


Rysunek 13 Struktura strony Uzupełnij Stan Magazynu Części

Zrzut przedstawiający zaimplementowaną walidację w przypadku podania błędnej lokalizacji części.



Rysunek 14 Walidacja w przypadku podania nie istniejącej lokalizacji części



Rysunek 15 Przedstawienie wywołania procedury dla zadanych parametrów

Wizualizacja w przypadku podania do parametru Lokacja_ID wartości która nie występuje w tabeli Magazyn_czesci w kluczu głównym (Lokacja_ID).



Rysunek 16 Wyświetlenie informacji o błędzie

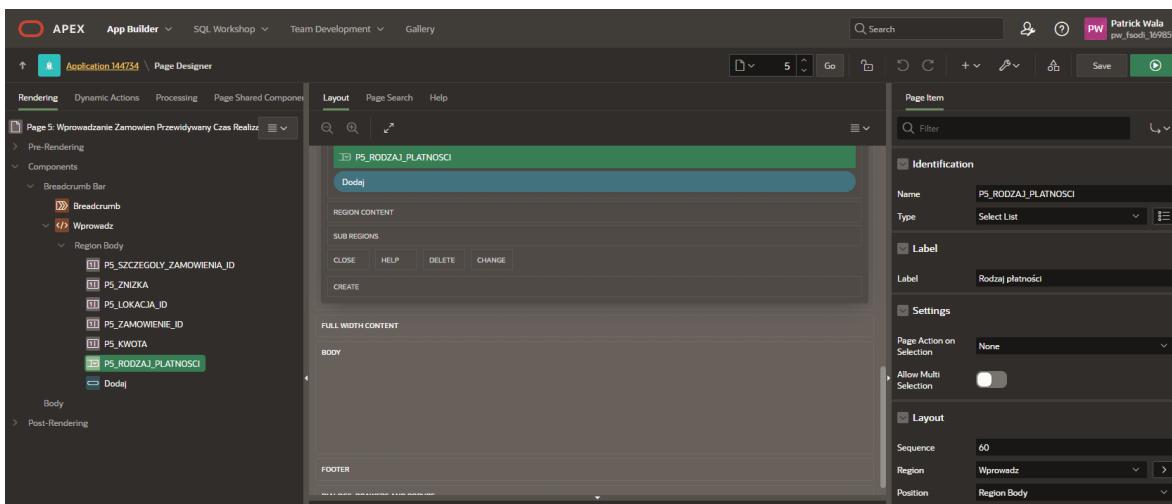
W przypadku podania poprawnej wartości ukazuje nam się komunikat o poprawnym wywołaniu procedury na stronie.

The screenshot shows a web browser window titled "test". The main content area has a header "Uzupełnij stan magazynu części". Below it, there is a form with two input fields: "Części Zamówienia Id" containing the value "3" and "Lokacja Id" containing the value "2". A button labeled "Dodać" is visible. In the top right corner, there is a green notification bar with the text "Pomyślnie wprowadzono zamówienie" (Successfully entered order).

Rysunek 17 Poprawne wykonanie kodu

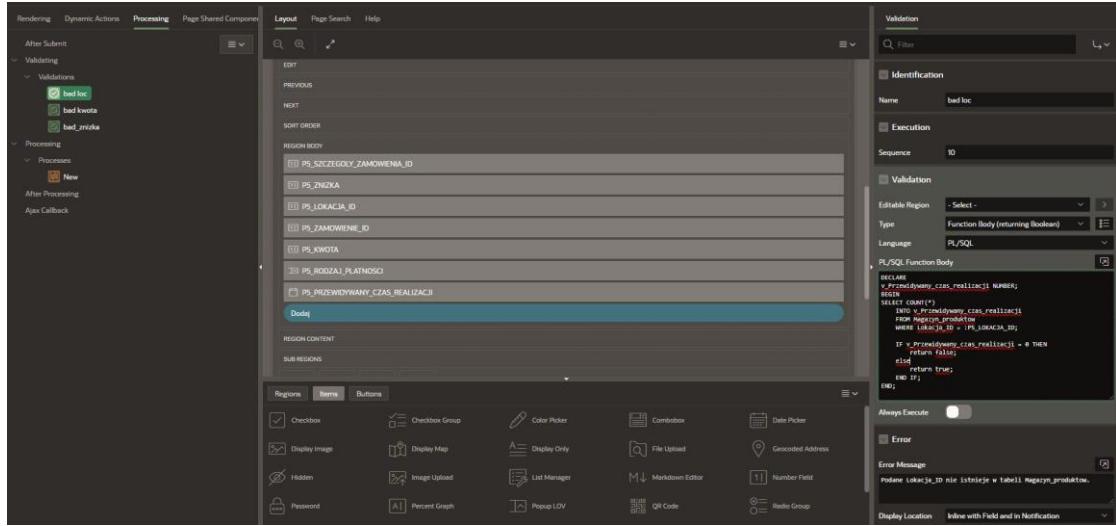
6.3 Strona Wprowadzanie zamówień

Strona, będąca wykorzystywana przez pracownika zakładu, obsługującego wprowadzanie zamówień do systemu, procedura obsługuje automatycznie uzupełnienie atrybutu Przewidywany_czas_realizacji z tabeli Zamowienie, przeszukując czy w tabeli Harmonogram_urlopów dla pracownika o stanowisku „Pracownik Magazynu” miedzy wartościami atrybutów od_kiedy do_kiedy nie znajduje się dzisiejsza data, jeśli nie to przewidywany czas realizacji zamówienia jest dzisiejszą datą.



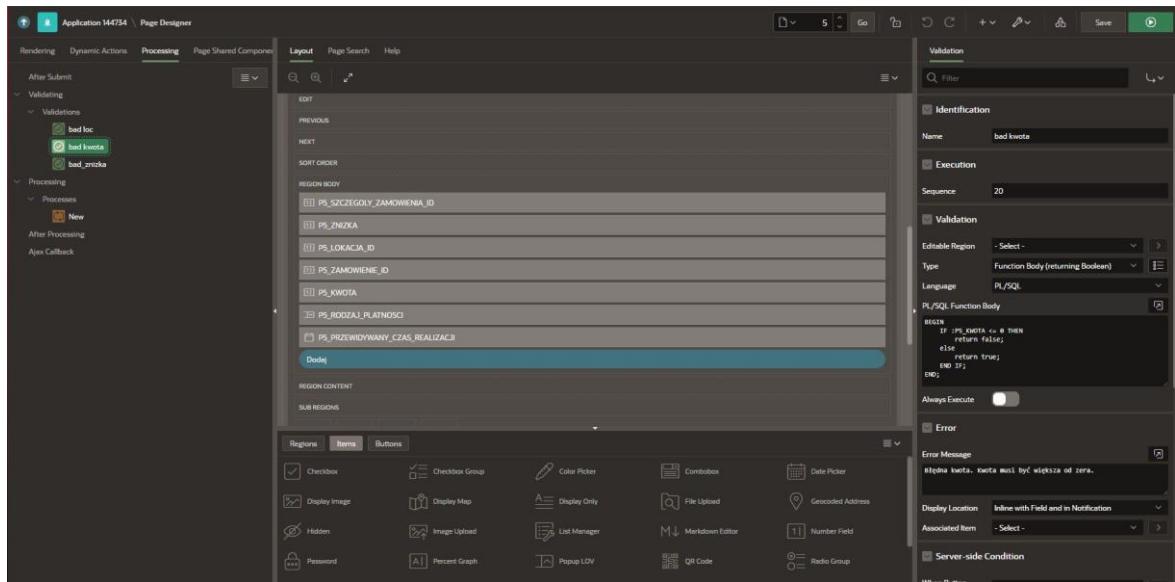
Rysunek 18 Struktura strony, wraz z widocznymi parametrami

Zrzut ekranu przedstawia walidacje parametru Lokacja_ID, jeśli przekazana wartość tego parametru nie będzie występować w tabeli Magazyn_produkty jako klucz główny, otrzymamy stosowny komunikat o błędzie.



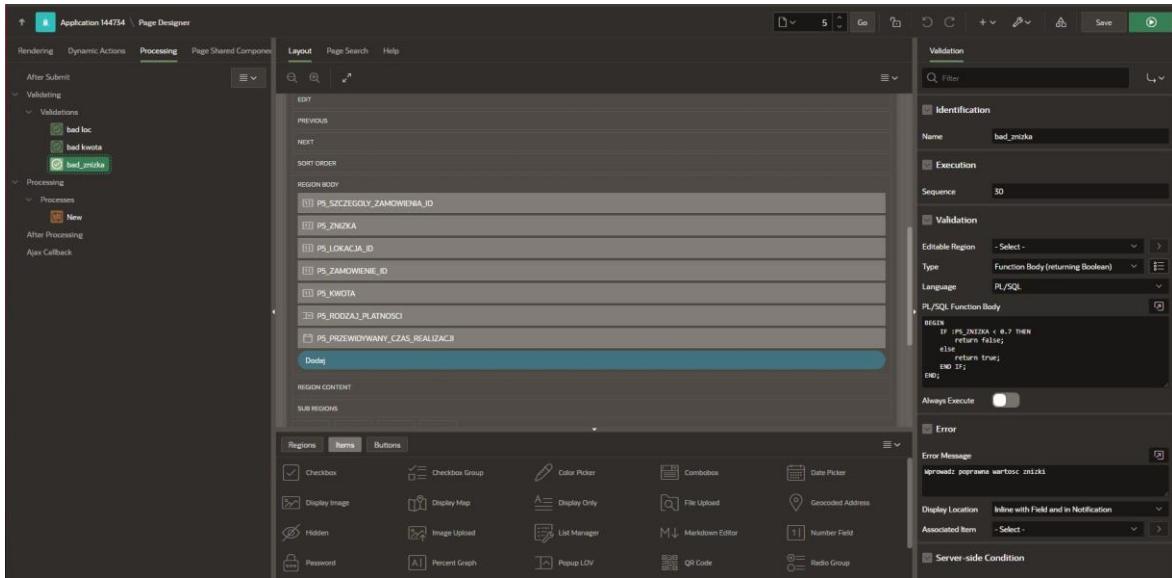
Rysunek 19 Walidacja bad_loc

Zrzut ekranu przedstawia walidacje parametru Kwota, jeśli przekazana wartość tego parametru będzie mniejsza lub równa 0, wyświetli nam się komunikat o konieczności wprowadzenia poprawnej wartości zamówienia.

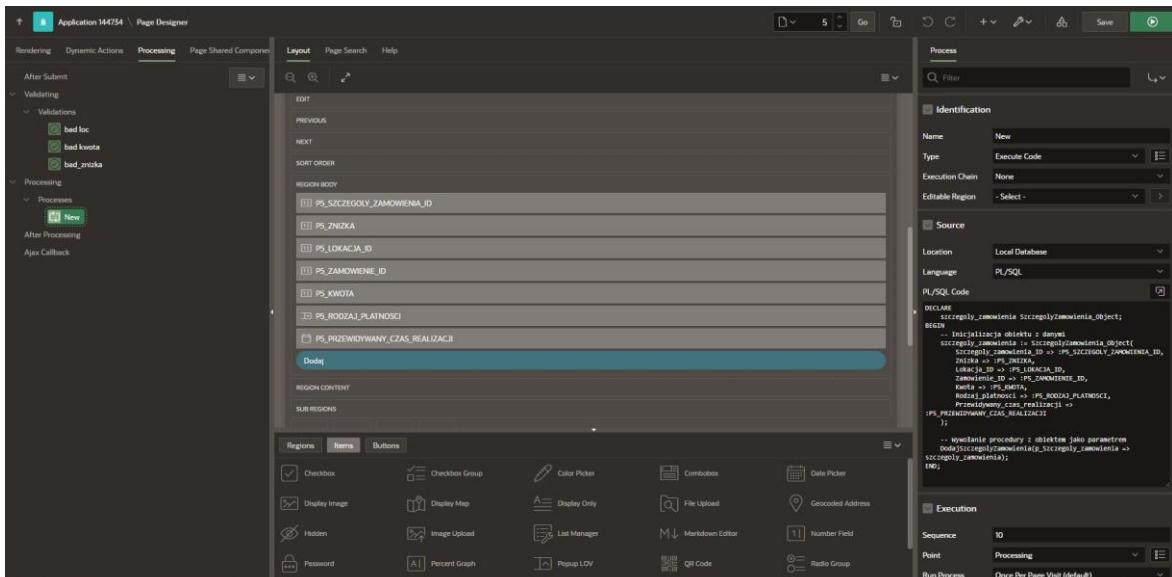


Rysunek 20 Walidacja bad_kwota

Zrzut ekranu przedstawia walidację parametru Zniżka, jeśli przekazana wartość tego parametru będzie mniejsza lub równa 0.7, wyświetli nam się komunikat o konieczności korekcji wartości tego parametru.



Rysunek 21 Walidacja bad_znizka



Rysunek 22 Kod przedstawiajacy wywolanie procedury

Poniższy zrzut ekranu przedstawia komunikat jaki wyświetli się użytkownikowi strony w przypadku nie wprowadzenia poprawnych wartości dla zadeklarowanych parametrów.

The screenshot shows a form titled "Wprowadzanie zamówień przewidywany czas realizacji". The form fields include:

- Szczegóły Zamówienia Id: -10
- Znizza: -1
- Lokacja ID: -1
- Zamówienie ID: -1
- Kwota: -10
- Rodzaj płatności: Karta płatnicza

A red error message box at the top right states: "3 errors have occurred":

- Podane Lokacja ID nie istnieje w tabeli Magazyn_produkty.
- Błędna kwota. Kwota musi być większa od zera.
- Wprowadź poprawną wartość znizki

Rysunek 23 Wizualizacja validacji

W przypadku wprowadzenia poprawnych danych, użytkownikowi wyświetli się komunikat o poprawnym działaniu.

The screenshot shows the same form as in Rysunek 23, but with valid input:

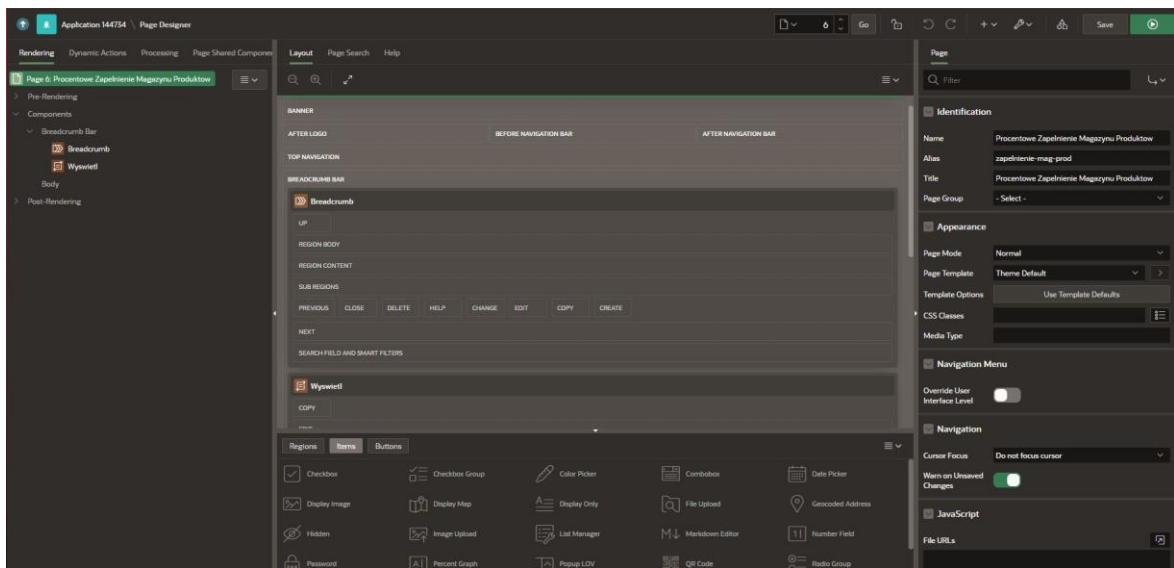
- Szczegóły Zamówienia Id: 57
- Znizza: 1
- Lokacja ID: 12
- Zamówienie ID: 33
- Kwota: 1234
- Rodzaj płatności: Karta płatnicza

A green success message box at the top right says: "Sukces".

Rysunek 24 Poprawne wykonanie kodu

6.4 Strona Procentowe zapelnienie Magazynu Produktów

Strona ma za zadanie informowanie użytkownika o procentowym zapelnieniu magazynu Produktów, co niesie za sobą informacje o wysokim znaczeniu biznesowym, wiemy na jakie produkty jest największy popyt na rynku i czego produkcje warto zwiększyć, oraz jakich produktów warto ograniczyć produkcję.



Rysunek 25 Struktura strony Procentowe Zapelnienie Magazynu Produktów

test

Procentowe zapelnienie magazynu produktów

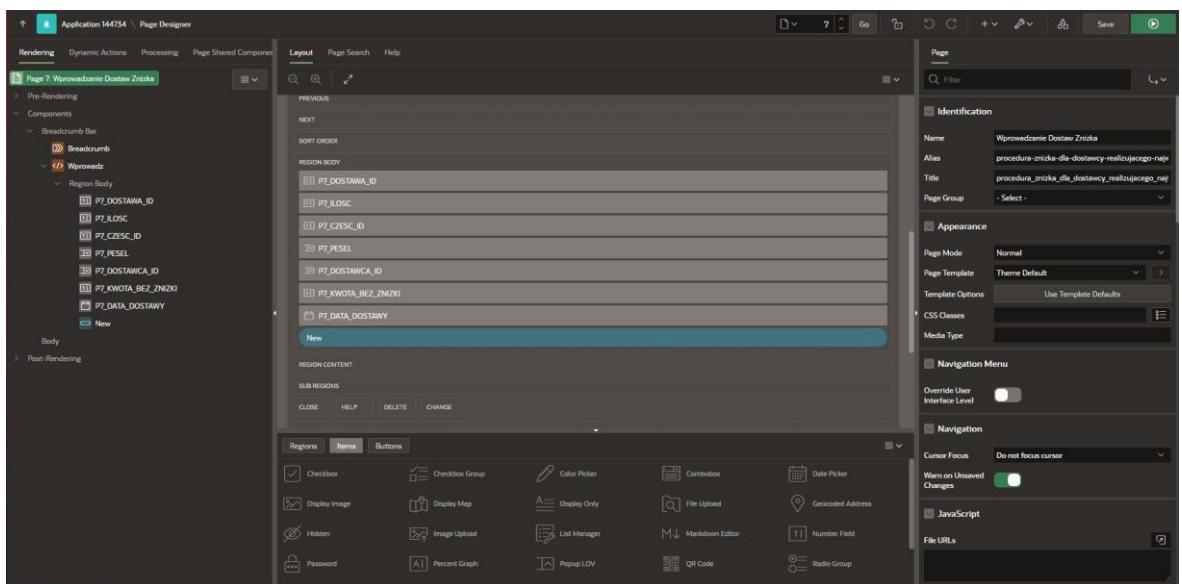
Wyswietl

Lokacja: 1, produkt: 2, zapelnienie: 49.67%
 Lokacja: 2, produkt: 4, zapelnienie: 34%
 Lokacja: 3, produkt: 6, zapelnienie: 86.96%
 Lokacja: 4, produkt: 8, zapelnienie: 49%
 Lokacja: 5, produkt: 10, zapelnienie: 75.33%
 Lokacja: 6, produkt: 12, zapelnienie: 63.6%
 Lokacja: 7, produkt: 14, zapelnienie: 74%
 Lokacja: 8, produkt: 1, zapelnienie: 78.8%
 Lokacja: 9, produkt: 3, zapelnienie: 97%
 Lokacja: 10, produkt: 5, zapelnienie: 90%
 Lokacja: 11, produkt: 7, zapelnienie: 21.2%
 Lokacja: 12, produkt: 9, zapelnienie: 35.71%
 Lokacja: 13, produkt: 11, zapelnienie: 39.5%
 Lokacja: 14, produkt: 13, zapelnienie: 62%
 Lokacja: 15, produkt: 15, zapelnienie: 14.67%

Rysunek 26 Dzialanie strony

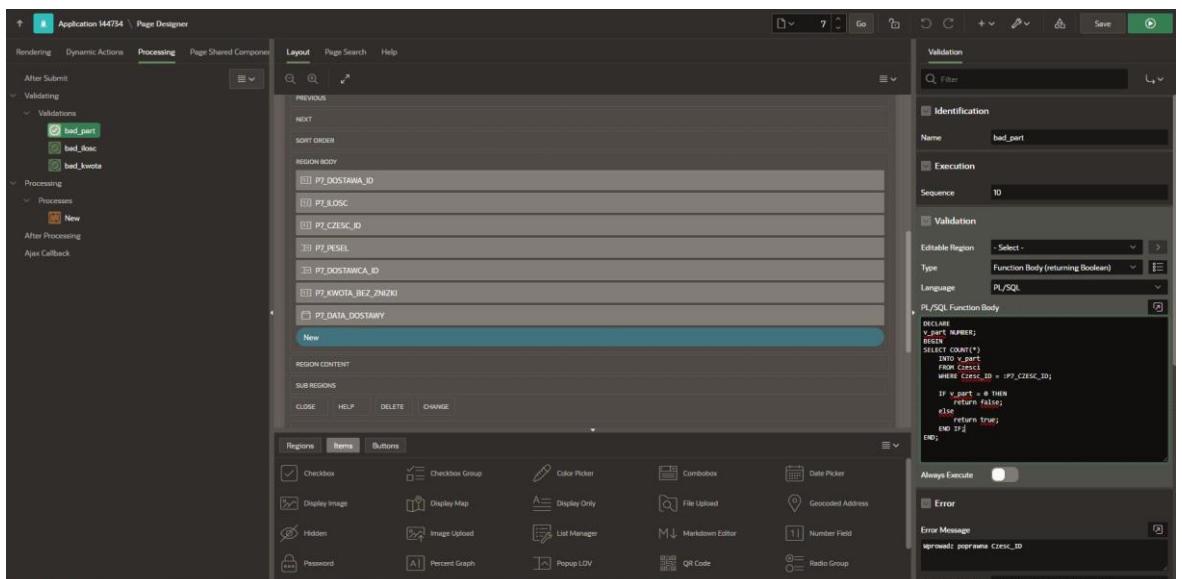
6.5 Strona Wprowadzanie Dostaw Zniżka

Strona której celem jest uproszczenie wprowadzania danych dotyczących dostaw części do zakładu produkcyjnego, dodatkowo kod procedury automatycznie oblicza należytą zniżkę dla dostawcy i końcową cenę dostawy.



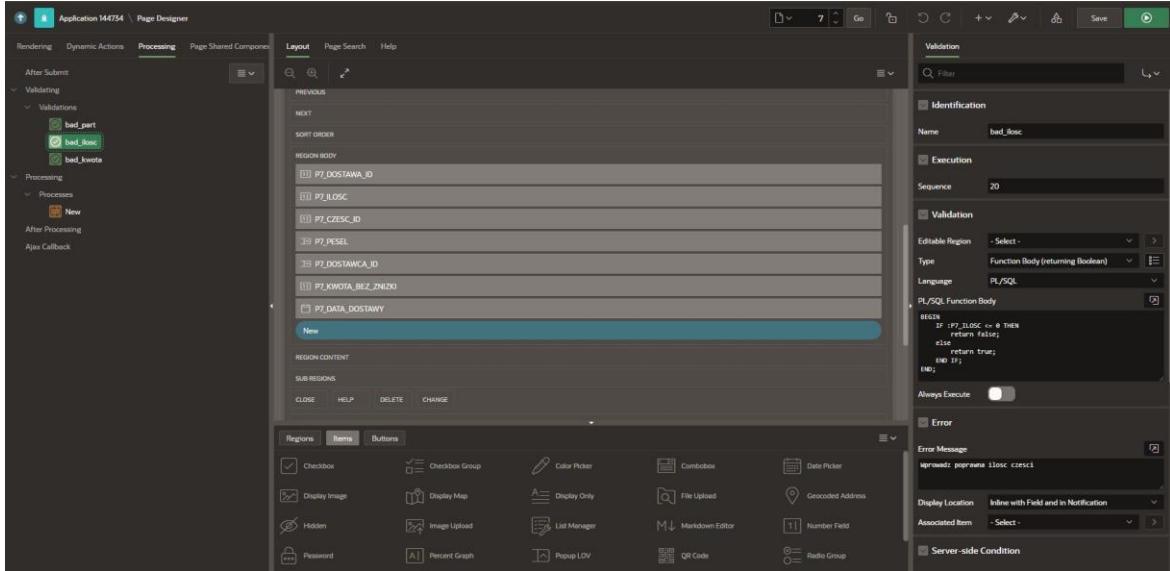
Rysunek 27 Struktura strony i wymagane parametry do uzupełnienia

Poniższy zrzut ekranu przedstawia walidacje polegającą na wyświetleniu komunikatu błędu w przypadku, gdy użytkownik przekaże jako parametr Czesc_ID, która nie występuje w tabeli Części.



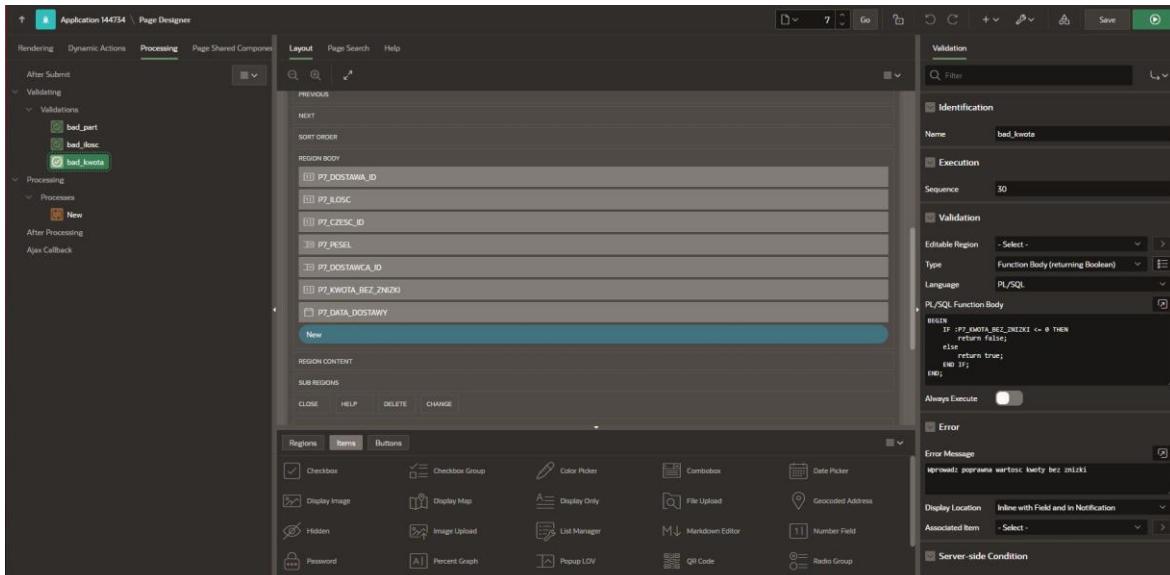
Rysunek 28 Walidacja bad_part

Poniższy zrzut ekranu przedstawia validację polegającą na wyświetleniu komunikatu błędu w przypadku, gdy użytkownik przekaże do parametru ilosc wartość mniejszą, lub równą 0.

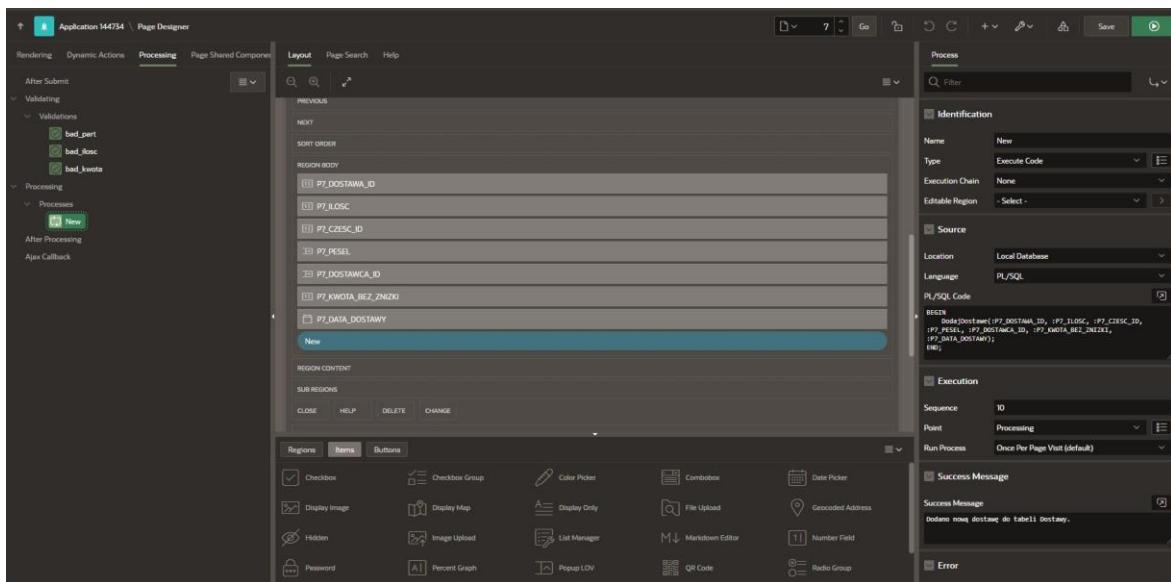


Rysunek 29 Walidacja bad_ilosc

Poniższy zrzut ekranu przedstawia validację polegającą na wyświetleniu komunikatu błędu w przypadku, gdy użytkownik przekaže do parametru kwota_bez_znizki wartość mniejszą, lub równą 0.



Rysunek 30 Walidacja bad_kwota



Rysunek 31 Wywołanie procedury na stronie

Poniższy zrzut ekranu przedstawia wizualizacje walidacji w momencie podania błędnych wartości dla wymaganych parametrów.

Dostawa_id	31
Ilosc	-10
Czesc_ID	-10
PESEL	23258741369
Dostawca_ID	3
Kwota bez zników	-100
Data dostawy	1/31/2024
Dodaj	

Rysunek 32 Wizualizacja walidacji

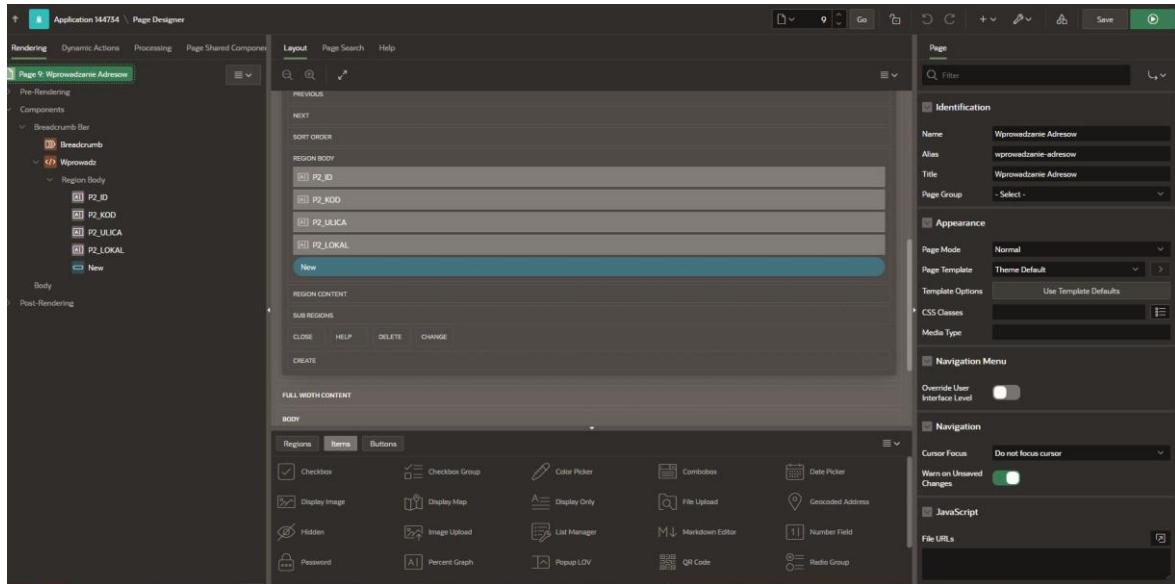
Zrzut ekranu przedstawiający komunikat w momencie poprawnego przekazania parametrów.

Dostawa_id	5
Ilosc	100
Czesc_ID	3
PESEL	23258741369
Dostawca_ID	10
Kwota bez zników	100
Data dostawy	1/25/2024
Dodaj	

Rysunek 33 Poprawne wykonanie procedury

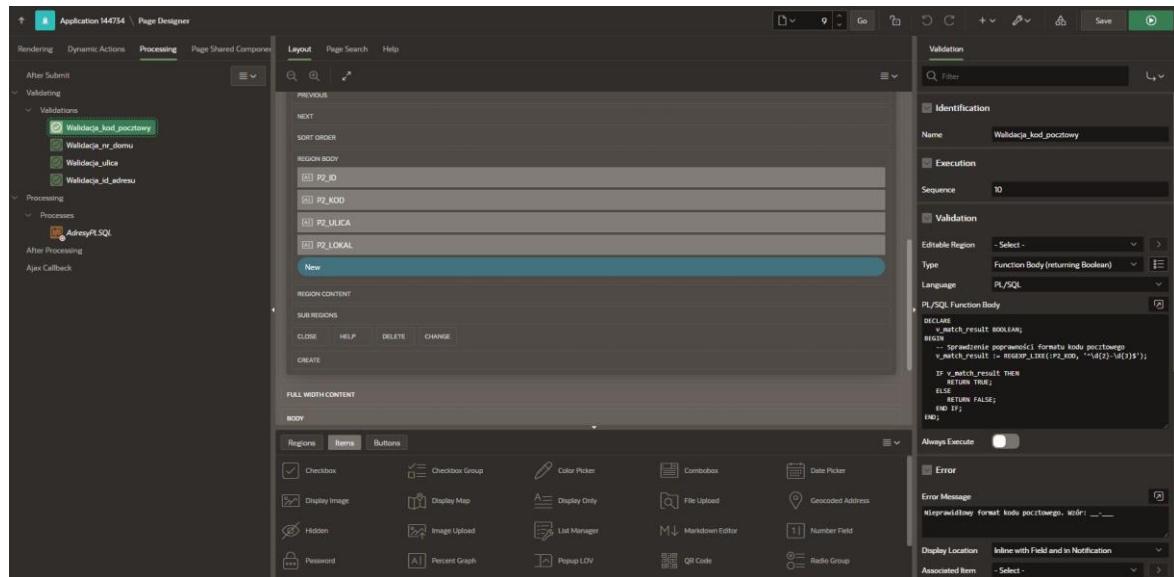
6.6 Strona Wprowadzanie Adresów

Strona, która ułatwia użytkownikowi aplikacji wprowadzanie danych do tabeli Adresy, zapewniając jednocześnie walidacje danych.



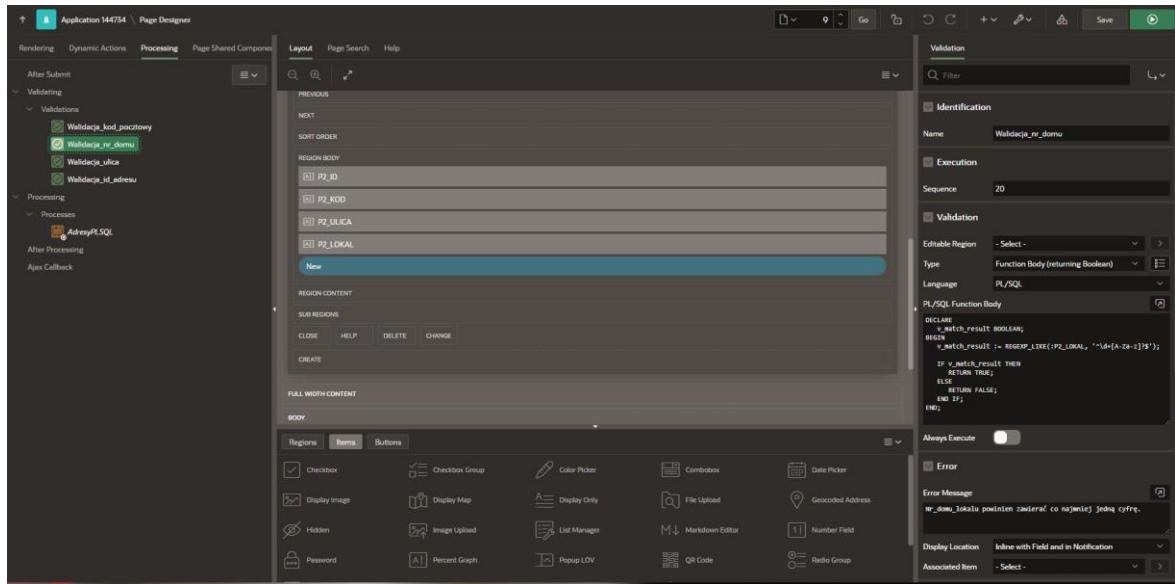
Rysunek 34 Struktura strony, wraz z widocznymi parametrami

Zrzut ekranu przedstawiający walidację sprawdzającą czy parametr kod_pocztowy jest w postaci (_-_), tak aby wartość była zgodna z rzeczywistymi danymi.



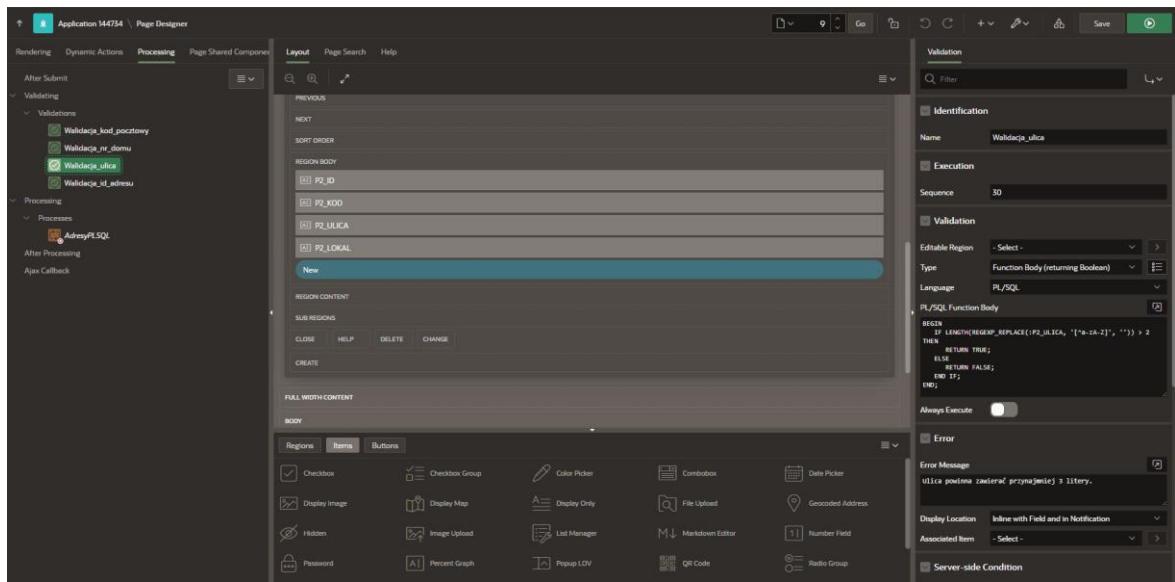
Rysunek 35 Walidacja kod_pocztowy

Walidacja parametru nr_domu sprawdzająca czy w przekazywanej wartości znajduje się przynajmniej jedna cyfra.



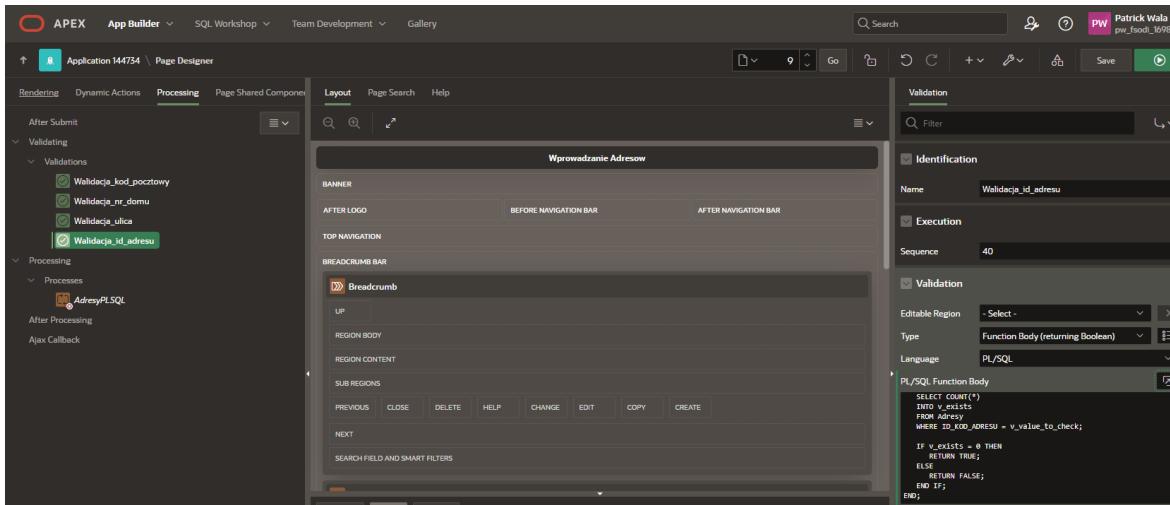
Rysunek 36 Walidacja nr_domu

Walidacja sprawdzająca czy w przekazywanym parametrze ulica, znajdują się litery.

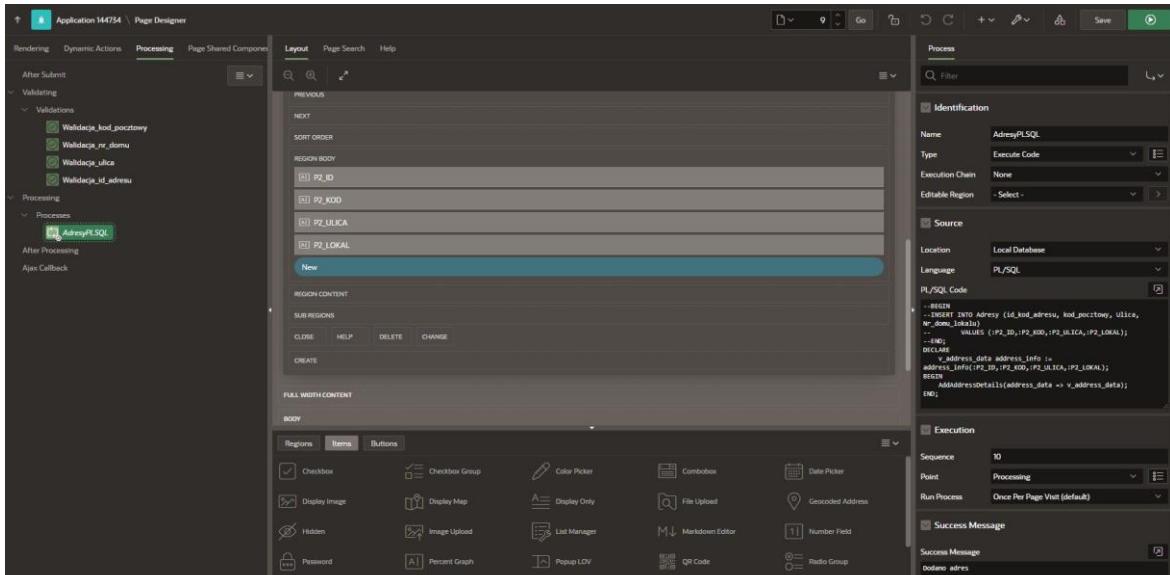


Rysunek 37 Walidacja Ulica

Walidacja sprawdzająca czy wartość przekazywanego parametru id_adresu występuje w kluczu głównym tabeli Adresy.

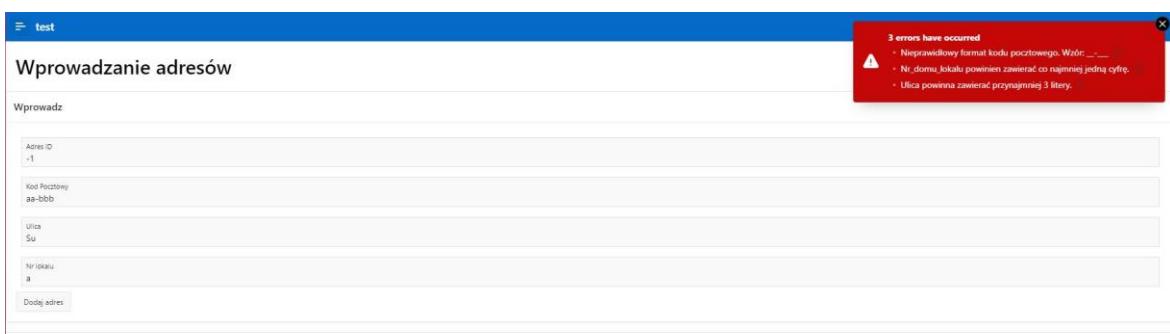


Rysunek 38 Walidacja id_adresu



Rysunek 39 Wywolanie procedury

Wyświetlenie błędów na stronie w przypadku podania błędnych wartości parametrów.



Rysunek 40 Wizualizacja błędów

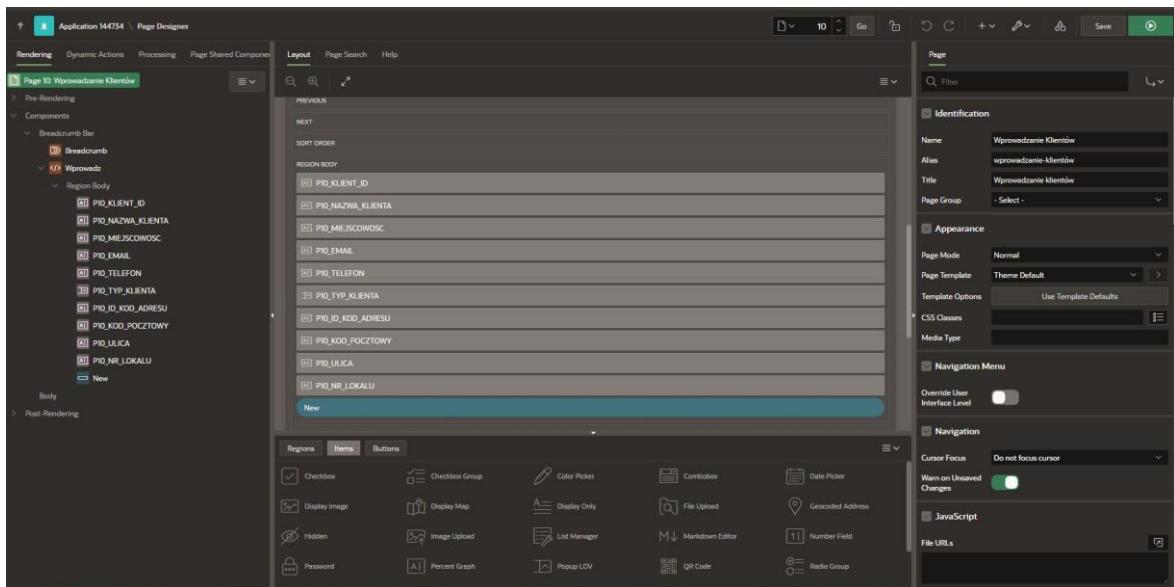
Zrzut ekranu przedstawiający komunikat w momencie poprawnego wykonania się procedury.

The screenshot shows a web application window titled "test". The main content area is titled "Wprowadzanie adresów" (Address Entry). It contains several input fields: "Adres ID" with value "176", "Kod Pocztowy" with value "11-111", "Ulica" with value "Super", and "Nr lokalu" with value "3a". Below these fields is a button labeled "Daj dodaj adres". In the top right corner, there is a green notification bar with a checkmark icon and the text "Sukces. Dodano adres".

Rysunek 41 Poprawne wykonanie kodu

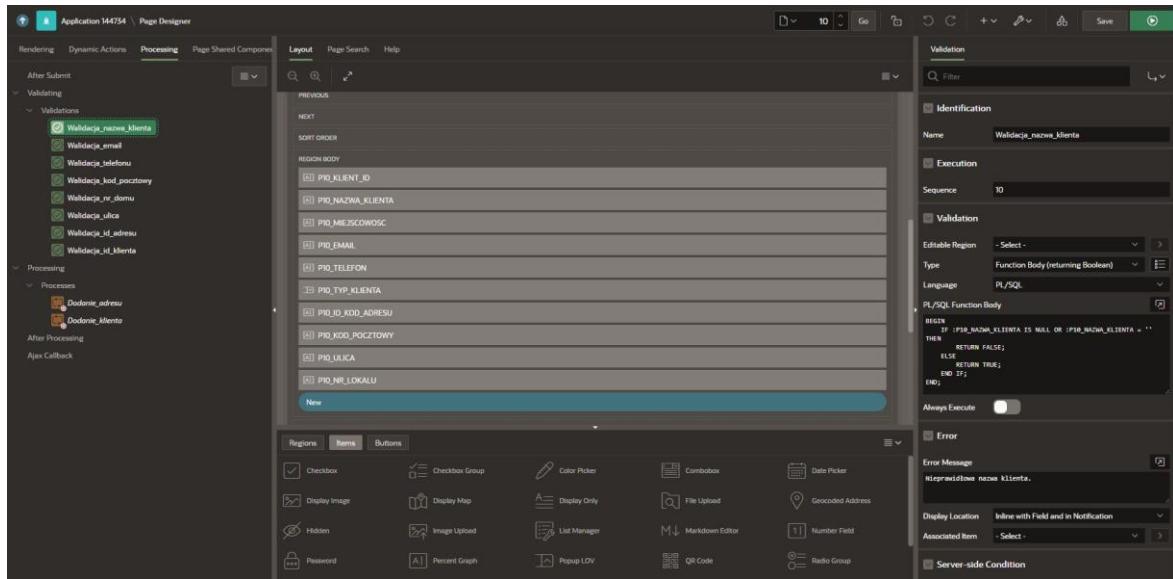
6.7 Strona Wprowadzanie Klientów.

Strona, która ułatwia użytkownikowi aplikacji wprowadzanie danych do tabeli Klienci, oraz jeśli przekazane parametry nie występują w rekordach tabeli Adresy, to dodajemy dane również tam, zapewniając jednocześnie walidacje danych.



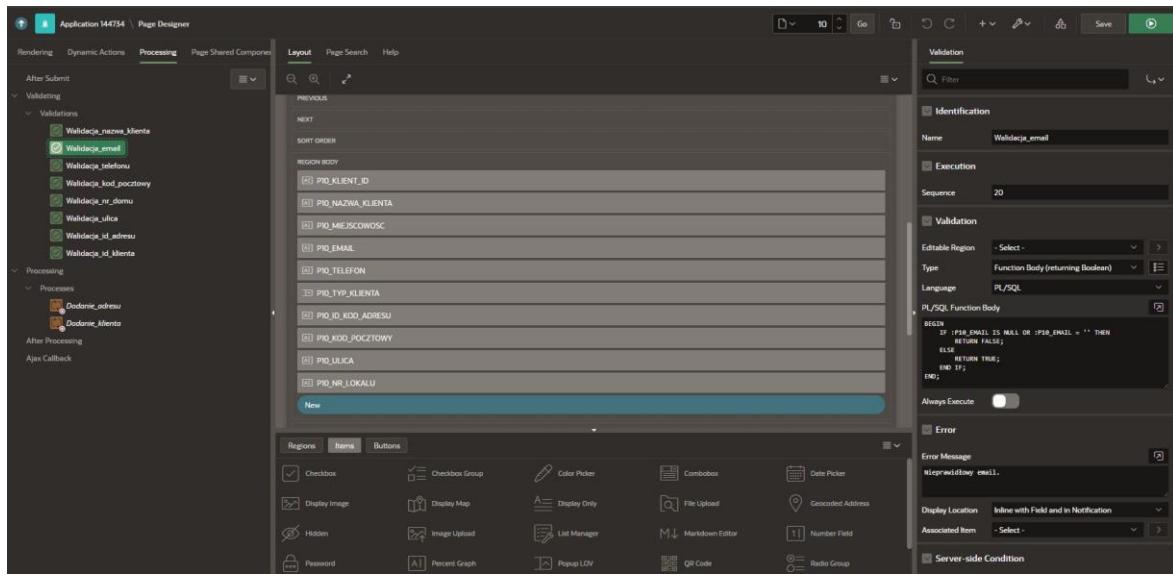
Rysunek 42 Struktura strony i przekazane parametry

Walidacja parametru nazwa_klienta, jeśli wartość tego parametru jest pusta wyświetla nam się stosowny komunikat o błędzie.



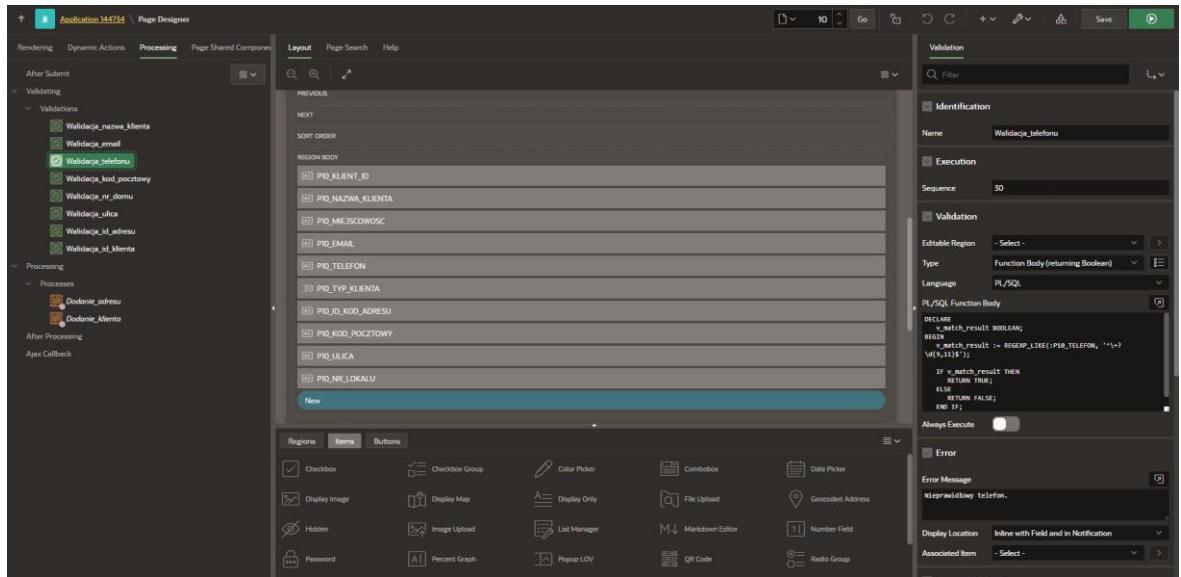
Rysunek 43 Walidacja nazwa_klienta

Walidacja sprawdzająca czy przekazana wartość parametru email zawiera znak „@”.



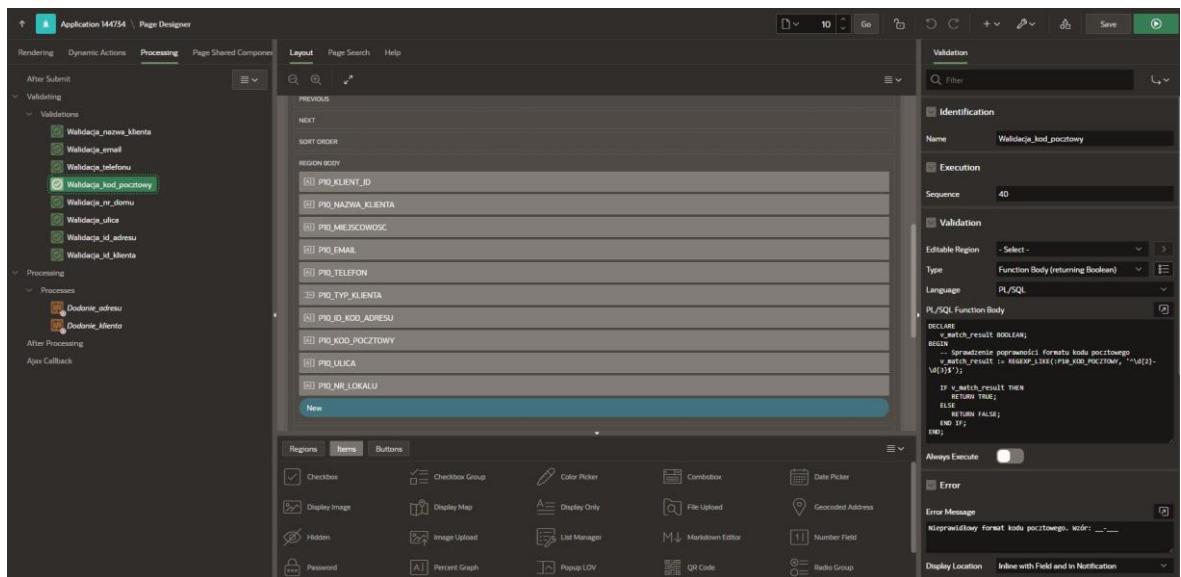
Rysunek 44 Walidacja email

Walidacja parametru Telefon, aby nie wyświetlała się informacja o błędzie wartość tego parametru powinna mieć między 9 i 11 znaków.



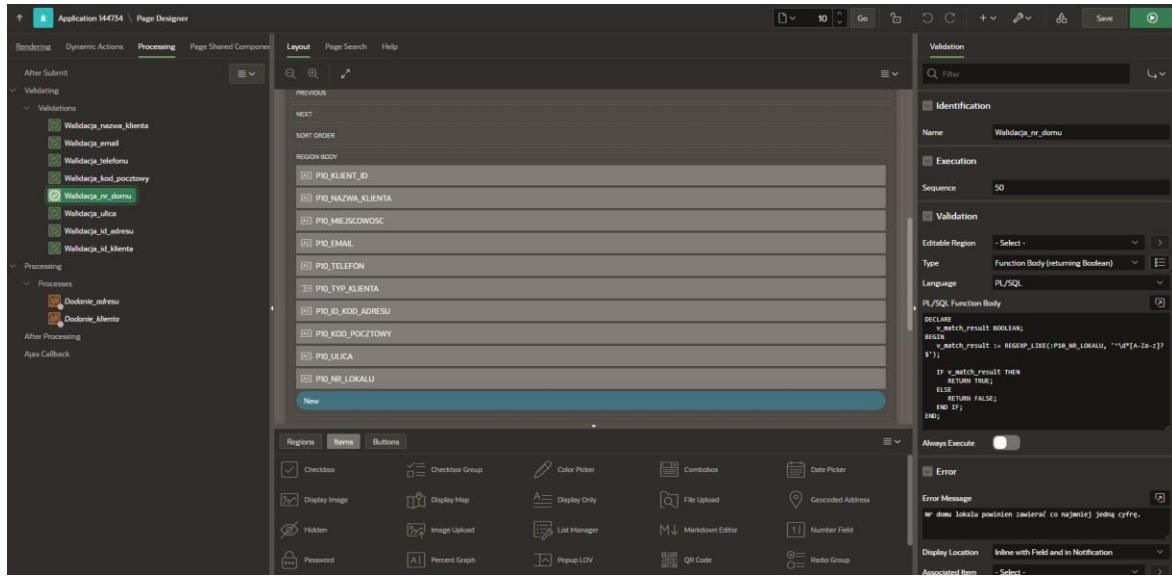
Rysunek 45 Walidacja telefonu

Walidacja parametru kod_pocztowy, parametr powinien mieć postać zgodną z rzeczywistymi wartościami kodu pocztowego (format __-__-__).



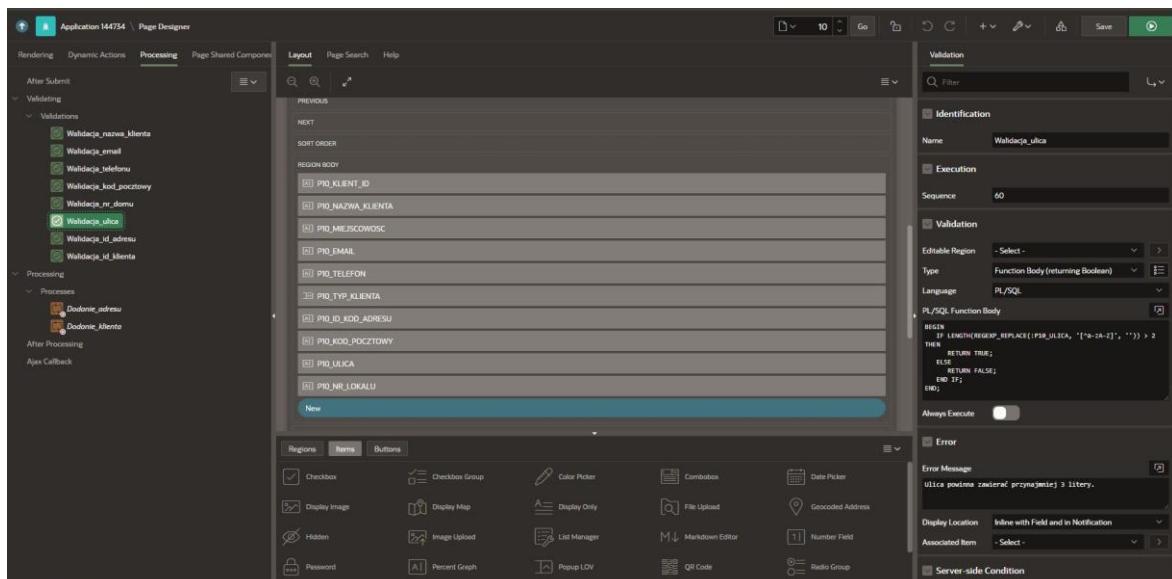
Rysunek 46 Walidacja kod_pocztowy

Walidacja parametru nr_domu sprawdzająca czy w przekazywanej wartości znajduje się przynajmniej jedna cyfra.



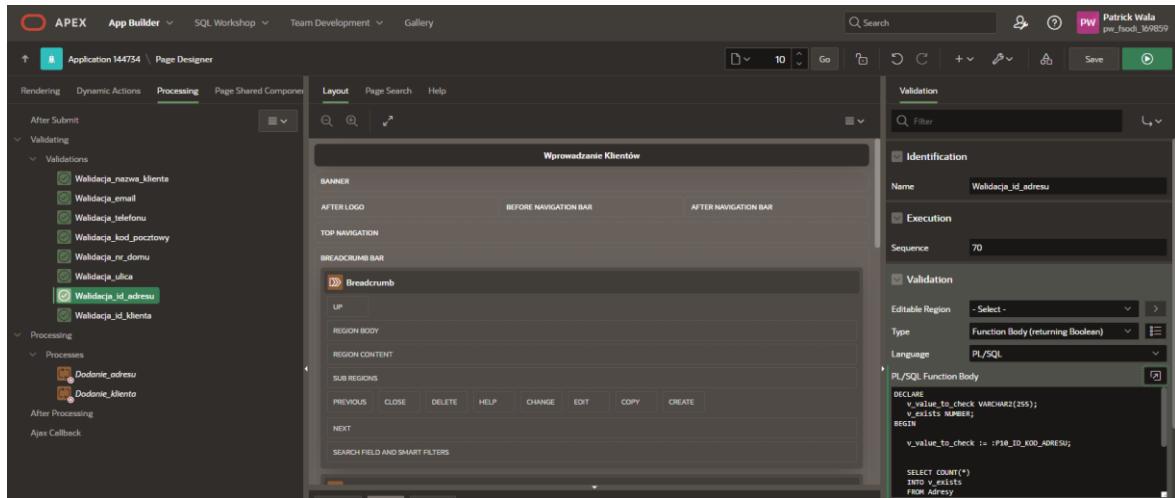
Rysunek 47 Walidacja nr_domu

Walidacja sprawdzająca czy w przekazywanym parametrze ulica, znajdują się litery.



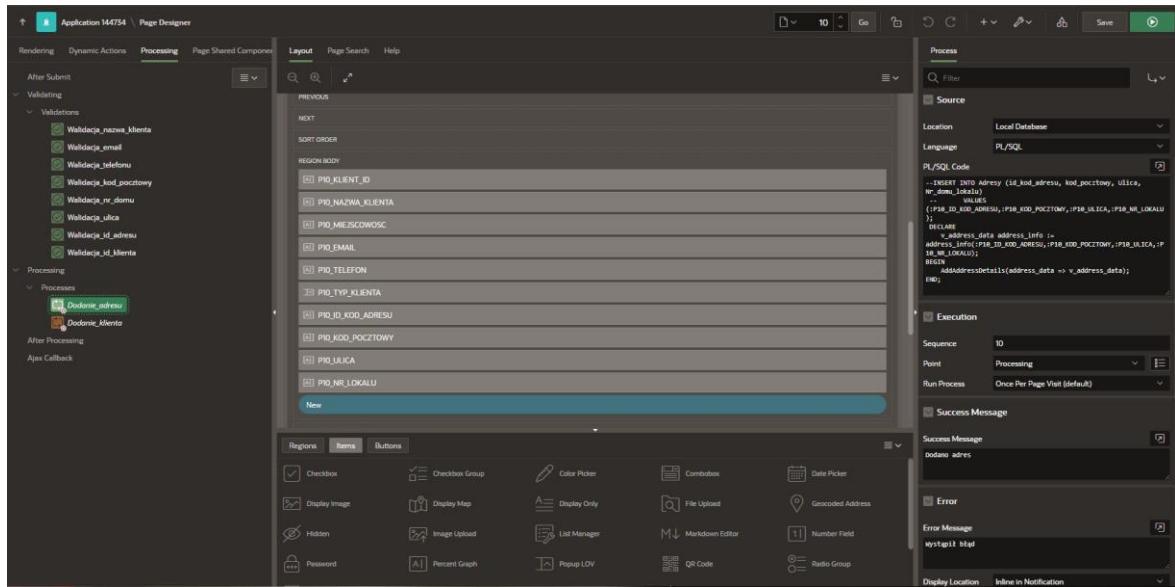
Rysunek 48 Walidacja ulica

Walidacja sprawdzająca czy wartość przekazywanego parametru id_adresu występuje w kluczu głównym tabeli Adresy.



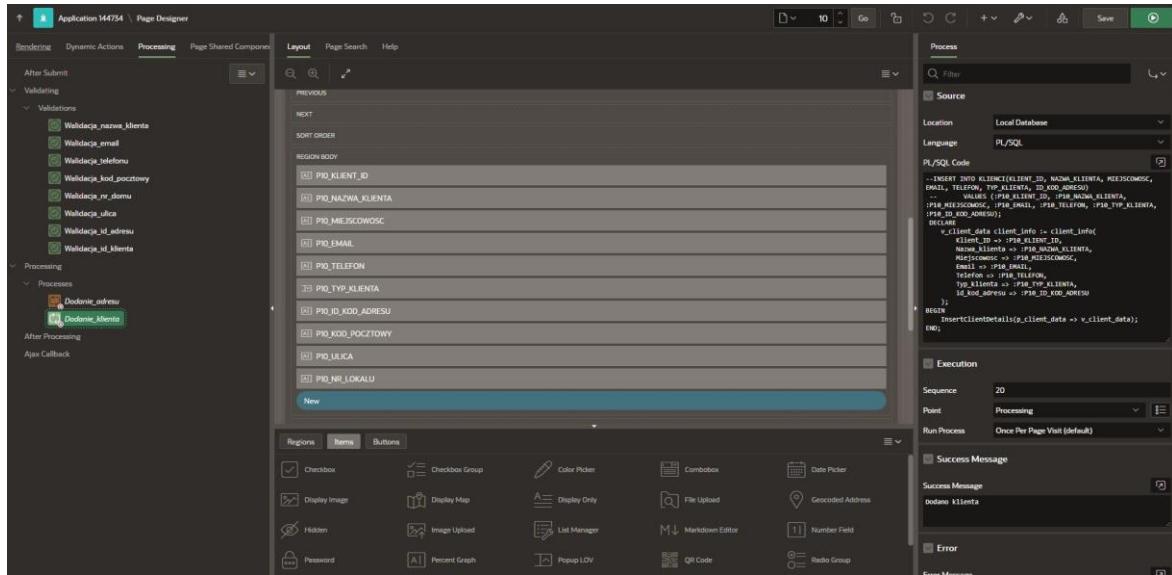
Rysunek 49 Walidacja id_adresu

Zrzut ekranu przedstawiający wywołanie procedury służącej do wprowadzenia danych do tabeli Adresy, jeśli dane przekazane do parametrów nie występuje w rekordach tej tabeli.



Rysunek 50 Wywołanie procedury

Zrzut ekranu przedstawiający wywołanie procedury służącej do wprowadzenia danych do tabeli Klienci.



Rysunek 51 Wywołanie procedury

Podgląd graficzny w momencie gdy przekażemy błędne wartości.

The screenshot shows a client input form titled 'Wprowadzanie klientów'. The fields include: Id klienta (-1), Nazwa klienta (a), Miejscowość (u), Email (a), Telefon (1), Typ klienta (Stary), Id adresu (-1), Kod pocztowy (ab-ds), Ulica (a), and Nr lokalu (-3). A 'Dodać klienta' button is at the bottom. A red error box on the right lists four validation errors:

- Nieprawidły telefon.
- Nieprawidły format kodu pocztowego. Wzór: ...-...-...
- Nr domu lokalu powinien zawańać co najmniej jedną cyfrę.
- Ulica powinna zawańać przynajmniej 3 litery.

Rysunek 52 Wizualizacja błędu

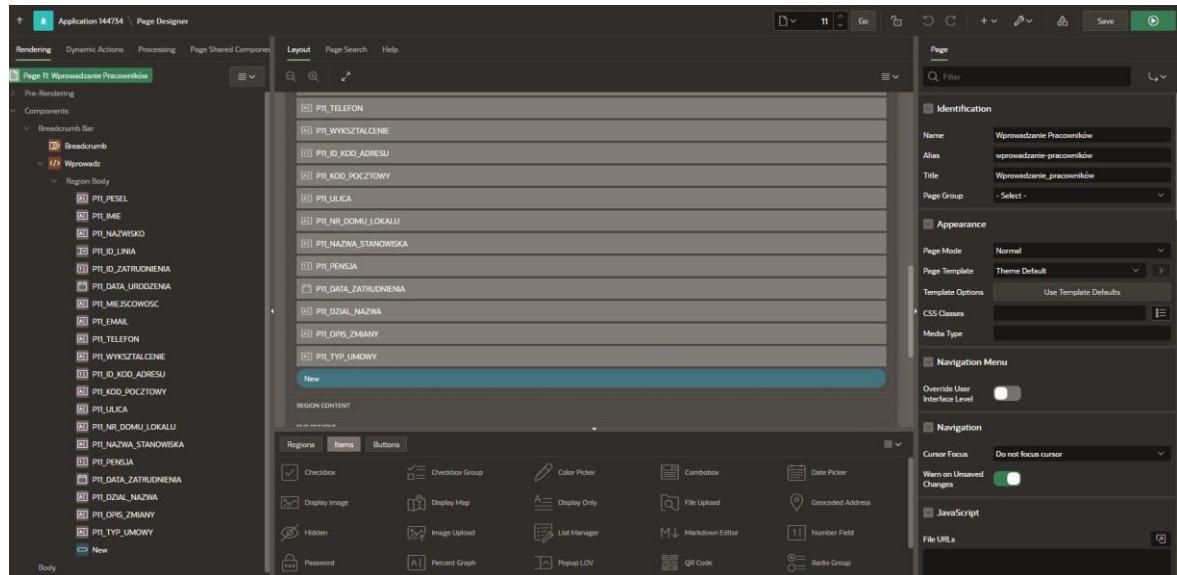
W momencie przekazania poprawnych danych wyświetla nam się komunikat.

The screenshot shows a web-based application window titled "test". The main title is "Wprowadzanie klientów". Below it, there's a form labeled "Wprowadz". The form fields include: "Id klienta" (744), "Nazwa klienta" (Super klient), "Miejscowość" (Rzeszów), "Email" (abcc@gmail.com), "Telefon" (12345678), "Typ klienta" (Stal), "Id adresu" (744), "Kod pocztowy" (12-234), "Ulica" (Super), "Nr lokalu" (6), and a "Dodaj klienta" button. A green success message "Sukces. Dodano klienta" is displayed at the top right.

Rysunek 53 Okno w momencie wykonania procedury

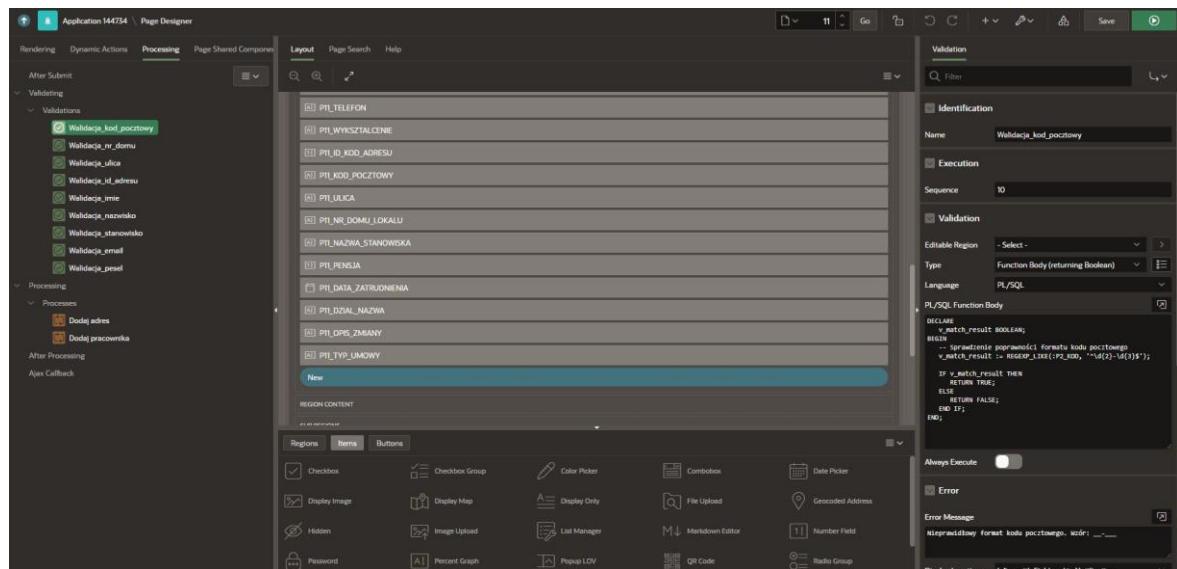
6.8 Strona Wprowadzanie Pracowników

Strona służąca do jednoczesnego wprowadzania danych do tabeli Pracownicy i Zatrudnienie, które odpowiadają za osoby zatrudnione w przedsiębiorstwie, zapewniając walidacje danych.



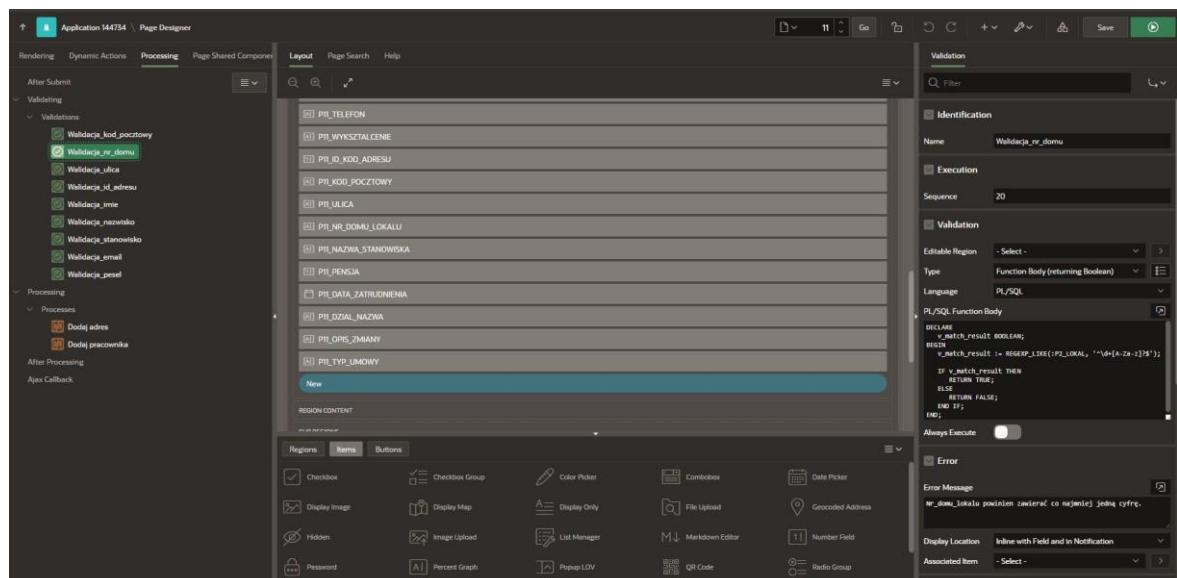
Rysunek 54 Struktura strony i wymagane parametry

Walidacja parametru kod_pocztowy, parametr powinien mieć postać zgodną z rzeczywistymi wartościami kodu pocztowego (format __-__-__).



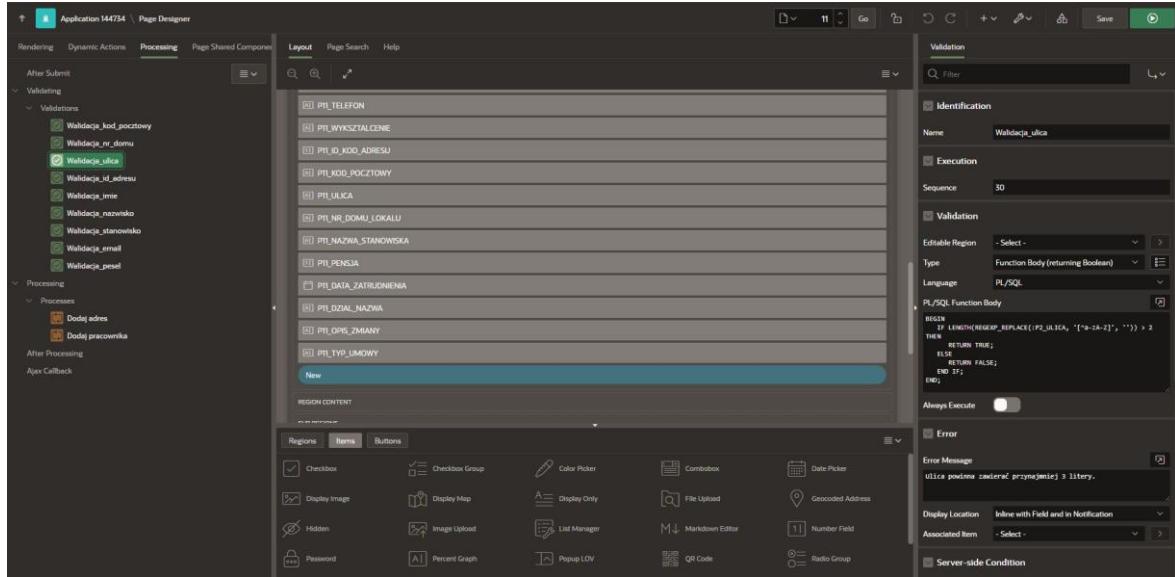
Rysunek 55 Walidacja kod_pocztowy

Walidacja parametru nr_domu sprawdzająca czy w przekazywanej wartości znajduje się przynajmniej jedna cyfra.



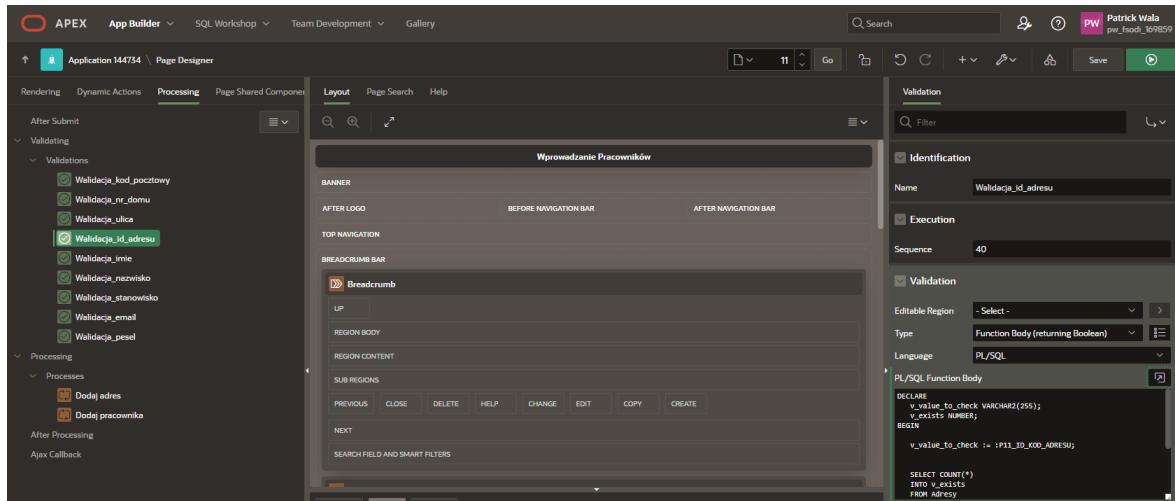
Rysunek 56 Walidacja nr_domu

Walidacja sprawdzająca czy w przekazywanym parametrze ulica, znajdują się litery.



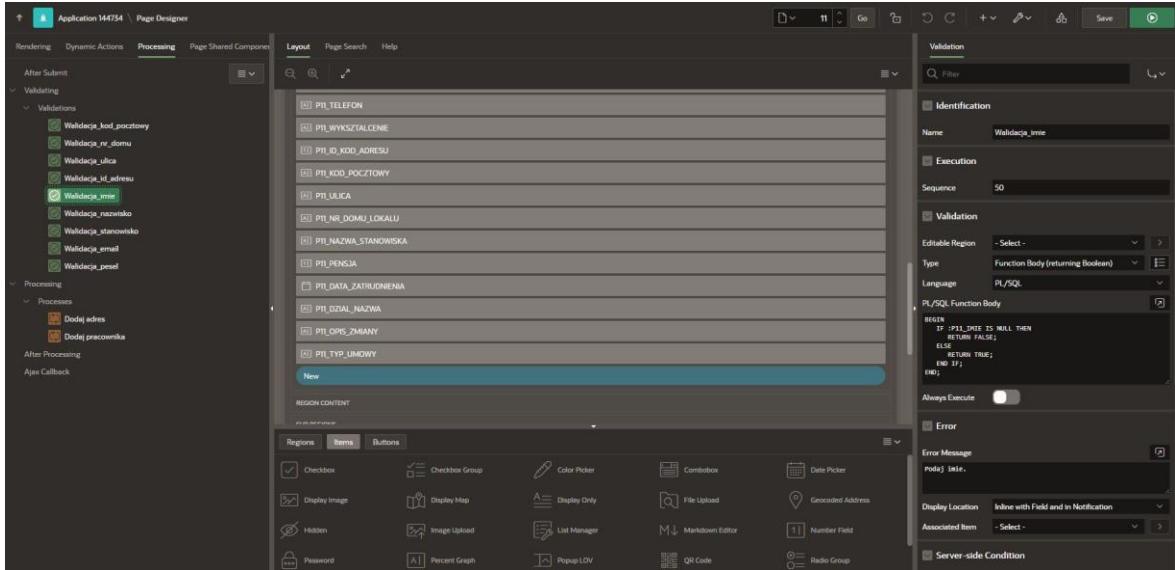
Rysunek 57 Walidacja ulica

Walidacja sprawdzająca czy wartość przekazywanego parametru id_adresu występuje w kluczu głównym tabeli Adresy.



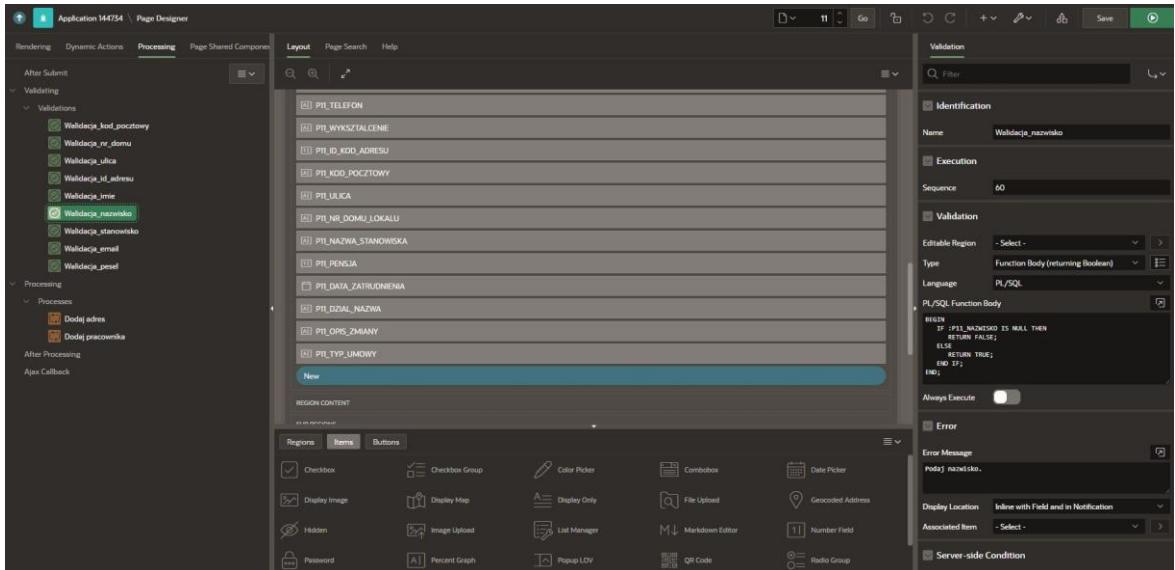
Rysunek 58 Walidacja id_adresu

Walidacja sprawdzająca czy parametr imie nie jest pusty.



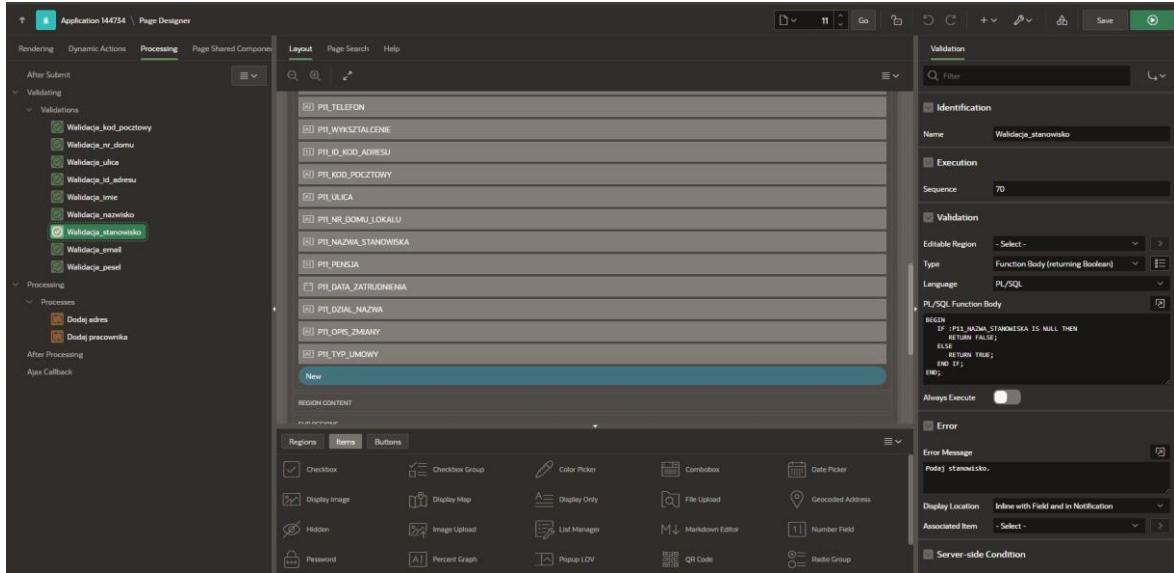
Rysunek 59 Walidacja imię

Walidacja sprawdzająca czy parametr nazwisko nie jest pusty



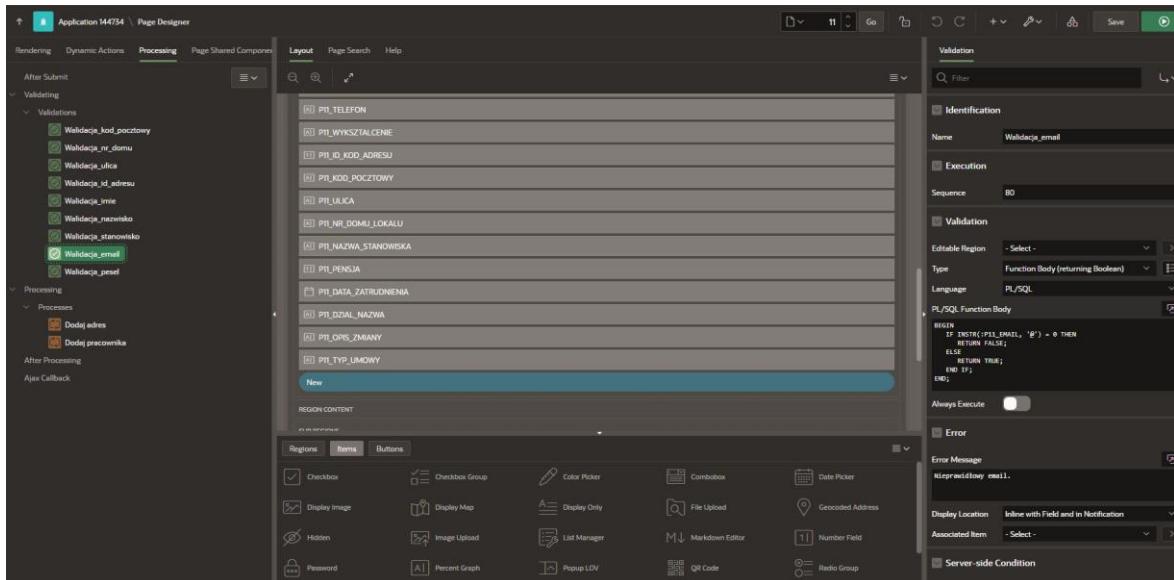
Rysunek 60 Walidacja nazwisko

Walidacja sprawdzająca czy parametr stanowisko nie jest puste.



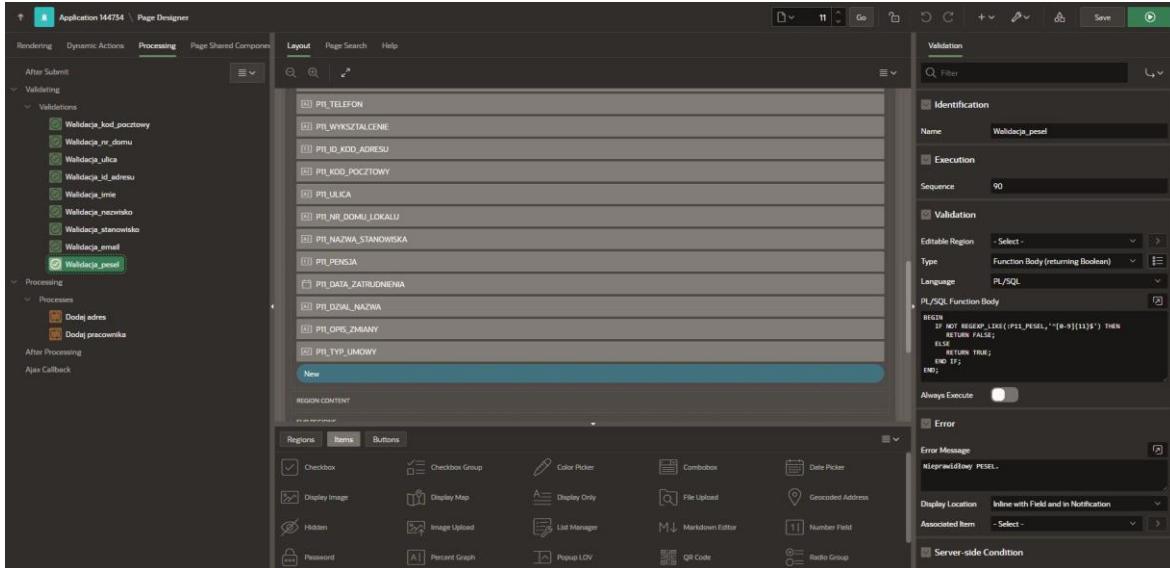
Rysunek 61 Walidacja stanowisko

Walidacja sprawdzająca czy przekazana wartość parametru email zawiera znak „@”



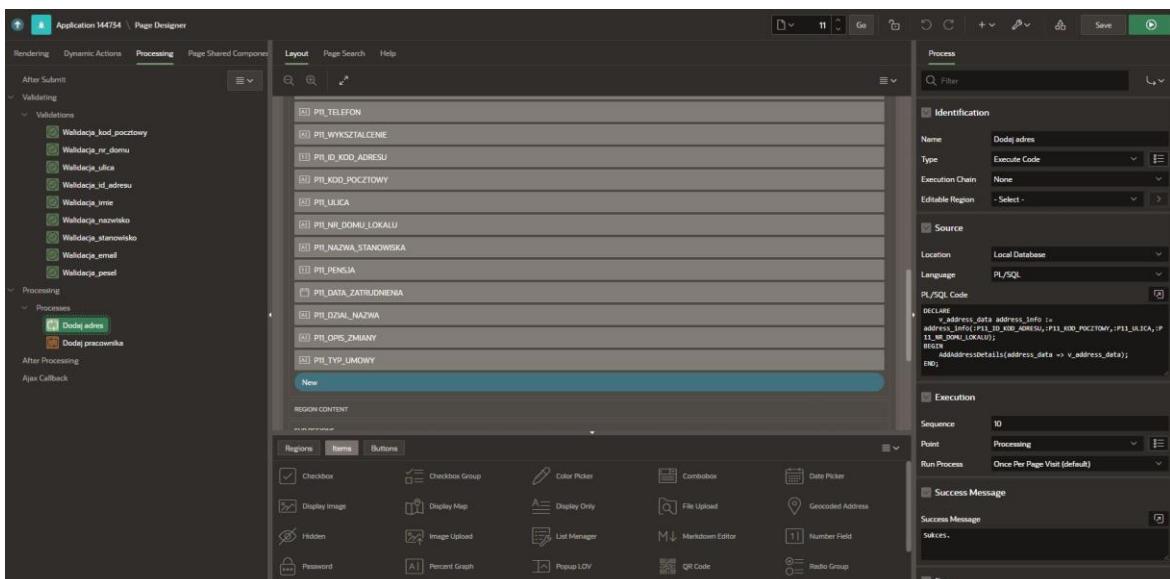
Rysunek 62 Walidacja email

Walidacja sprawdzająca czy parametr PESEL zawiera 11 znaków.



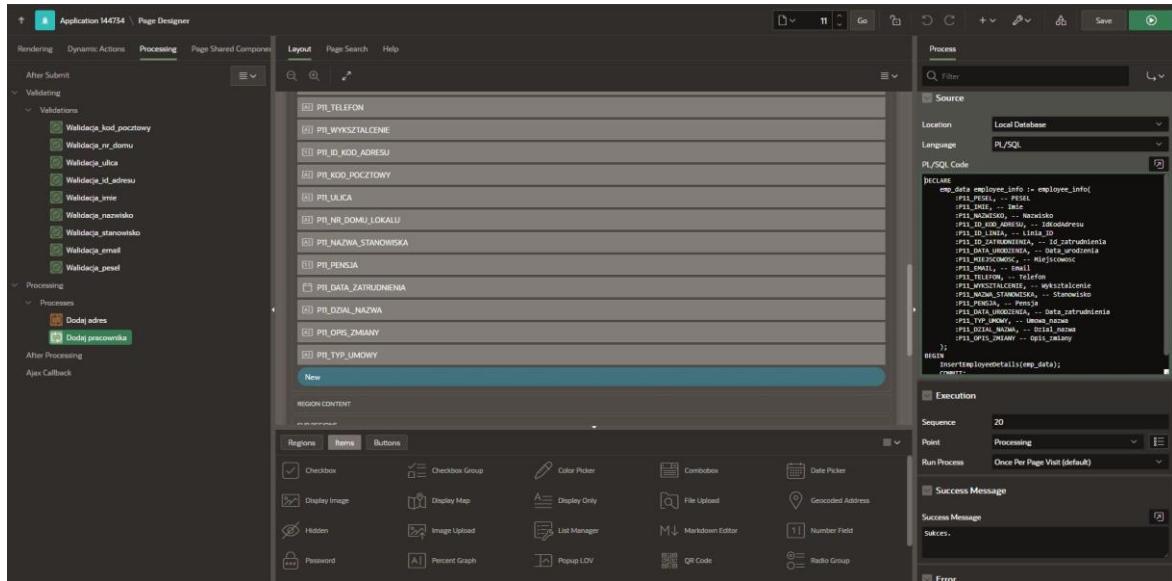
Rysunek 63 Walidacja Pesel

Zrzut ekranu przedstawiający wywołanie procedury służącej do wprowadzenia danych do tabeli Adresy, jeśli dane przekazane do parametrów nie występuje w rekordach tej tabeli.



Rysunek 64 Wywolanie procedury

Wywołanie procedury służącej do wprowadzania danych pracownika.



Rysunek 65 Wywolanie procedury DodajPracownika

Zrzut ekranu obrazujący przekazanie błędnych danych do parametru email i PESEL, oraz wyświetlenie informującego o tym komunikatu.

Rysunek 66 Graficzne przedstawienie validacji

Zrzut ekranu przedstawiający poprawnie wykonany kod procedury.

The screenshot shows a web-based application interface for managing employees. At the top, a blue header bar displays the title "test". Below it, a green success message "Dodano pracownika" (Employee added) is shown. The main content area is titled "Wprowadzanie pracowników" (Adding employees). The form contains the following fields:

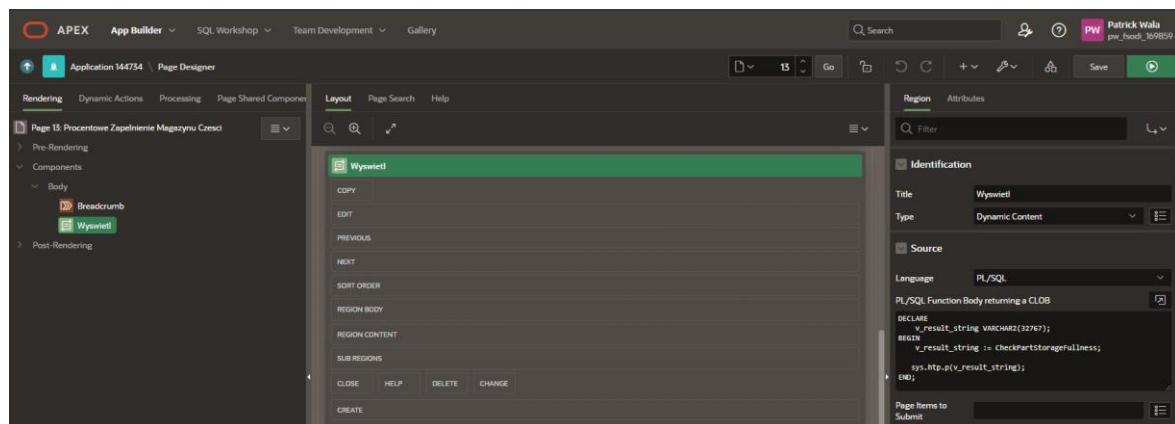
- Pesel: 12309678575
- Imię: Adam
- Nazwisko: Adam
- Unit ID: D3
- Id zatrudnienia: 534
- Data urodzenia: 1/10/1995
- Miejscowość: Rzeszów
- Email: asdf@gmail.com
- Telfon: 123456456
- Wykształcenie: Średnie
- ID Kod adresu: 645
- Kod pocztowy: 11-111

At the bottom of the form, there are several small icons for navigating the application, such as "App 144734", "Page 11", "Session", "Debug", "Quick Edit", and "Customize".

Rysunek 67 Poprawnie wykonany kod

6.9 Strona Procentowe Zapełnienie Magazynu Części

Strona ma za zadanie informowanie użytkownika o procentowym zapełnieniu magazynu Części, dzięki czemu uzyskujemy informacje o tym jakie części ulegają największemu wykorzystaniu i zużyciu w procesie tworzenia asortymentów. Otrzymujemy również informacje o tym na jakie części warto złożyć zamówienie, ponieważ w niedalekiej przyszłości mogą się wyczerpać.



Rysunek 68 Struktura strony

The screenshot shows a web page titled "Procentowe zapelnienie magazynu części". The content area is labeled "Wyswietl" and contains a list of locations with their respective fill percentages:

- Lokacja: 1, część: 2, zapelnienie: 35.71%
- Lokacja: 2, część: 4, zapelnienie: 31.62%
- Lokaja: 3, część: 6, zapelnienie: 60%
- Lokaja: 4, część: 8, zapelnienie: 59%
- Lokaja: 5, część: 10, zapelnienie: 61.54%
- Lokaja: 6, część: 12, zapelnienie: 68.89%
- Lokaja: 7, część: 14, zapelnienie: 89.59%
- Lokaja: 8, część: 1, zapelnienie: 68%
- Lokaja: 9, część: 3, zapelnienie: 61.38%
- Lokaja: 10, część: 5, zapelnienie: 27.33%
- Lokaja: 11, część: 7, zapelnienie: 55%
- Lokaja: 12, część: 9, zapelnienie: 15%
- Lokaja: 13, część: 11, zapelnienie: 59.7%
- Lokaja: 14, część: 13, zapelnienie: 27.64%
- Lokaja: 15, część: 15, zapelnienie: 50%

Rysunek 69 Działanie strony

6.10 Strona Aktualizuj Magazyny Po Produkcji

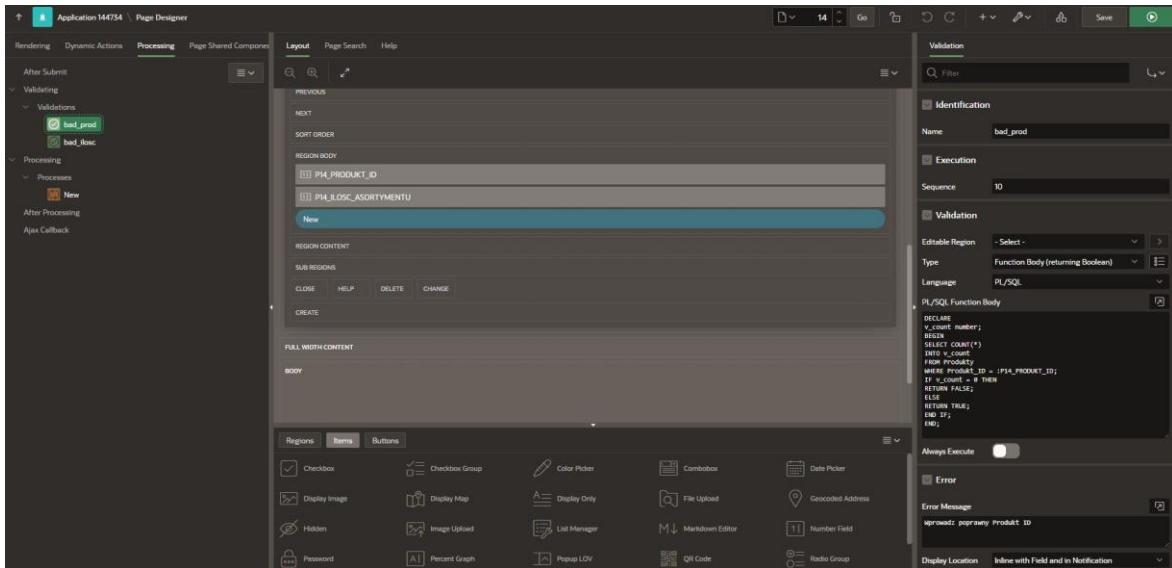
Celem tej strony jest fizyczne stworzenie danej ilości asortymentu, wykorzystując informacje zawarte w tabeli ProduktCzesci, mówiącej o tym z jakich i ilu części składa się dany produkt, oraz zaktualizowaniu rekordów tabel Magazyn_czesci (usuwamy potrzebna ilość części), oraz tabeli Magazyn_produkow (aktualizujemy ilość danego asortymentu w magazynie, o wartość którą wprowadziliśmy)

The screenshot shows the Oracle Page Designer interface with the following details:

- Page Title:** Page 14: Aktualizuj Magazyny Po Produkcji
- Layout:** The layout is set to "Normal".
- Regions:**
 - Breadcrumb Bar:** Contains "PREVIOUS" and "NEXT" buttons.
 - Sort Order:** Contains "REGION BODY" and "PIM_PRODUCT_ID" regions.
 - Region Content:** Contains "PIM_ILOSC_ASORTYMENTU" and "New" regions.
 - Body:** Contains "CLOSE", "HELP", "DELETE", and "CHANGE" buttons.
- Toolbars:** Includes "Regions", "Items", and "Buttons" tabs.
- Properties Panel:** Shows the following properties for the page:
 - Identification:** Name: Aktualizuj Magazyny Po Produkcji, Alias: tworzenie-produktu, Title: tworzenie produktu, Page Group: -Select-
 - Appearance:** Page Mode: Normal, Page Template: Theme Default, Template Options: Use Template Defaults.
 - Navigation:** Cursor Focus: Do not focus cursor, Warn on Unsaved Changes: On.
 - JavaScript:** File URLs: [empty].

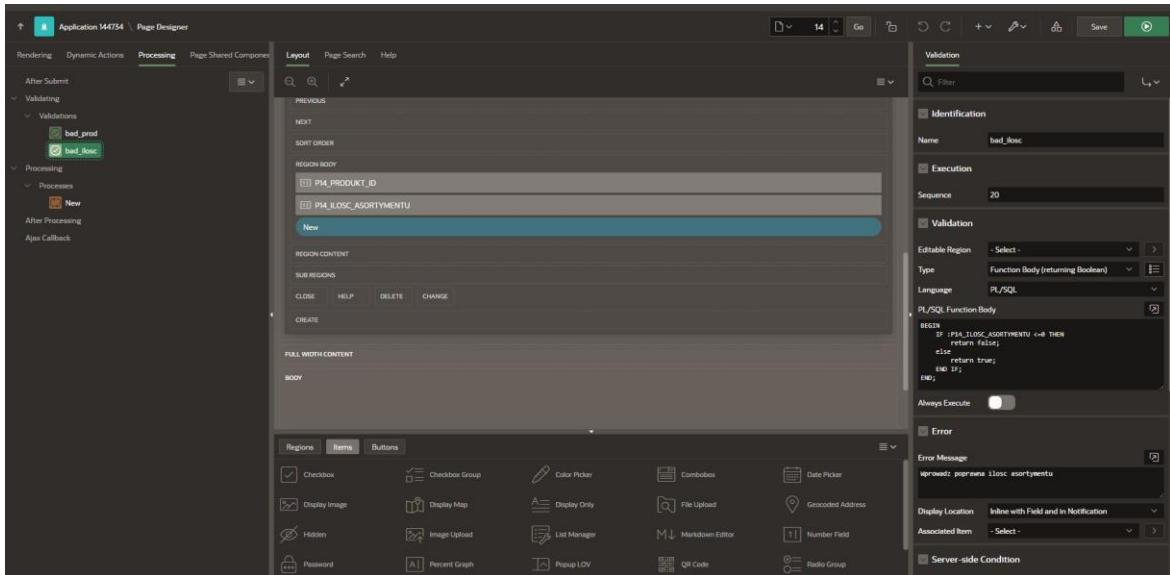
Rysunek 70 Struktura strony

Walidacja produktu, użytkownikowi strony wyświetla się błąd w momencie gdy jako parametr Produkt_ID przekazujemy wartość która nie występuje w tabeli Produkty

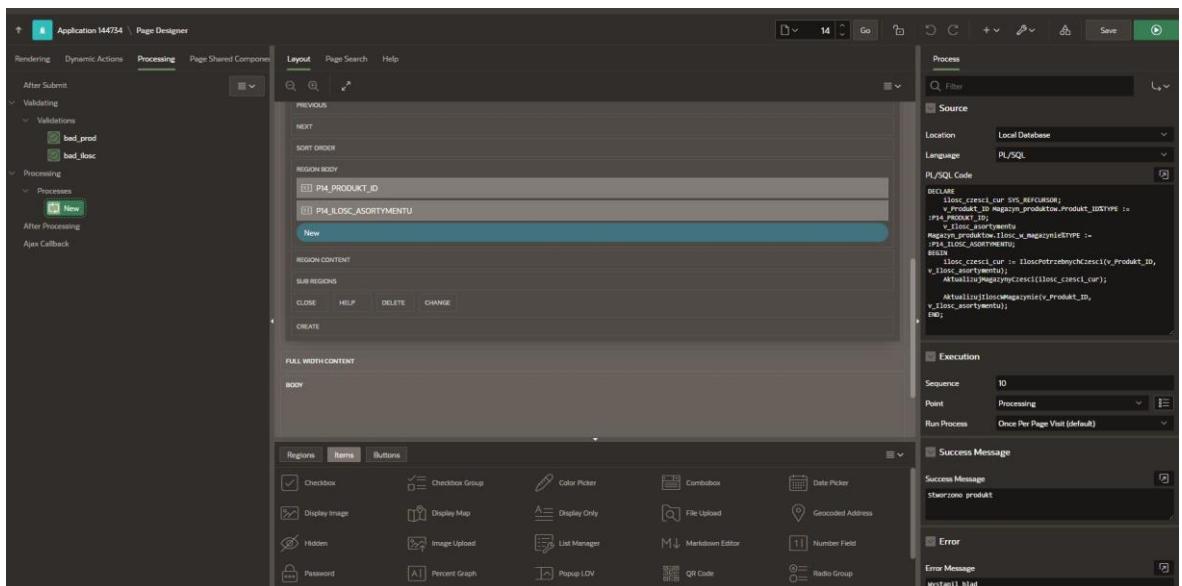


Rysunek 71 Walidacja bad_prod

Walidacja parametru ilość_asortymentu, użytkownikowi strony wyświetla się błąd w momencie, gdy jako wartość przekazujemy liczbę mniejszą lub równą 0.



Rysunek 72 Walidacja bad_ilosc



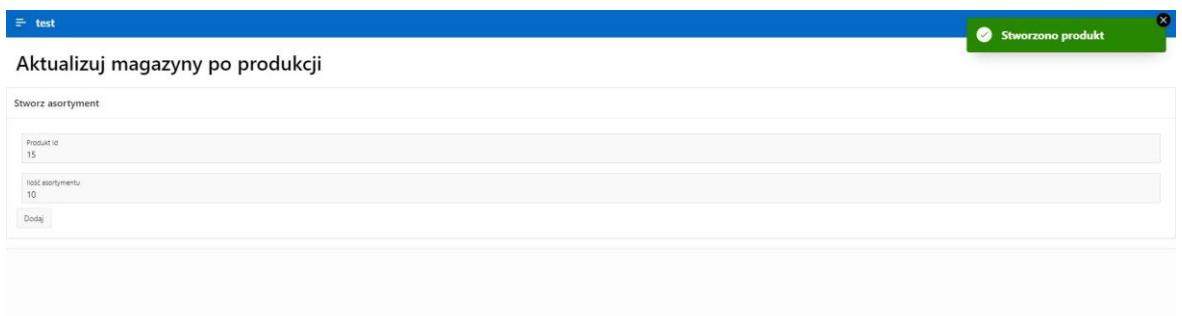
Rysunek 73 Wywolanie procedury

Widok wyświetlający się użytkownikowi, gdy wprowadzi błędne dane.



Rysunek 74 Wizualizacja walidacji

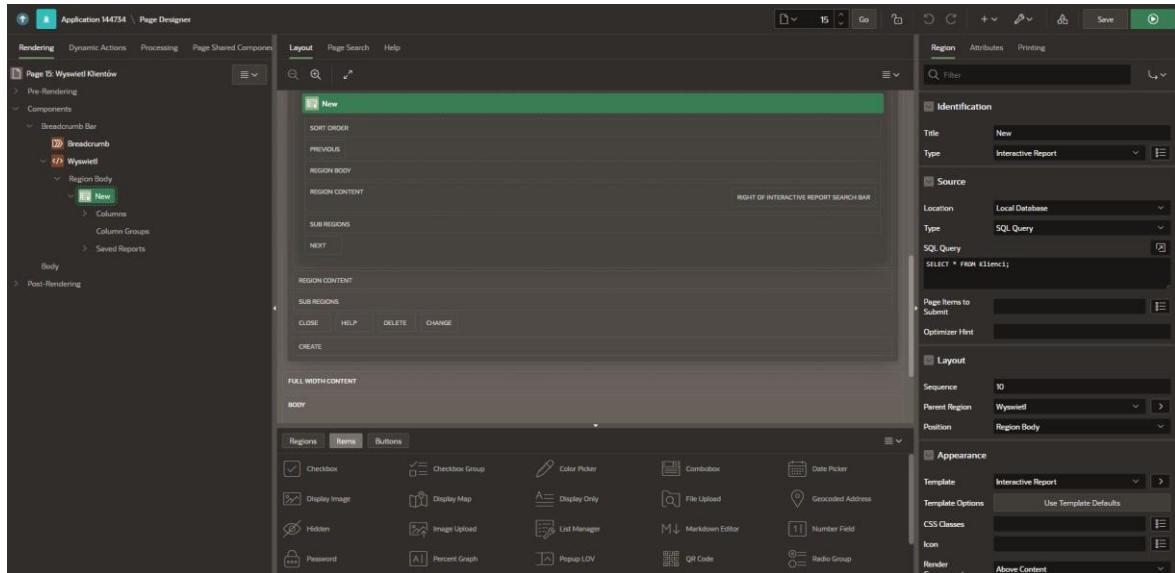
Komunikat wyświetlający się użytkownikowi w momencie poprawnego działania kodu strony.



Rysunek 75 Działanie strony

6.11 Strona Wyświetl Klientów

Działanie strony polega na wyświetlaniu zawartości tabeli Klienci, ich filtracjach i sortowaniu, oraz wyszukiwanie spersonalizowanych informacji.



Rysunek 76 Struktura strony

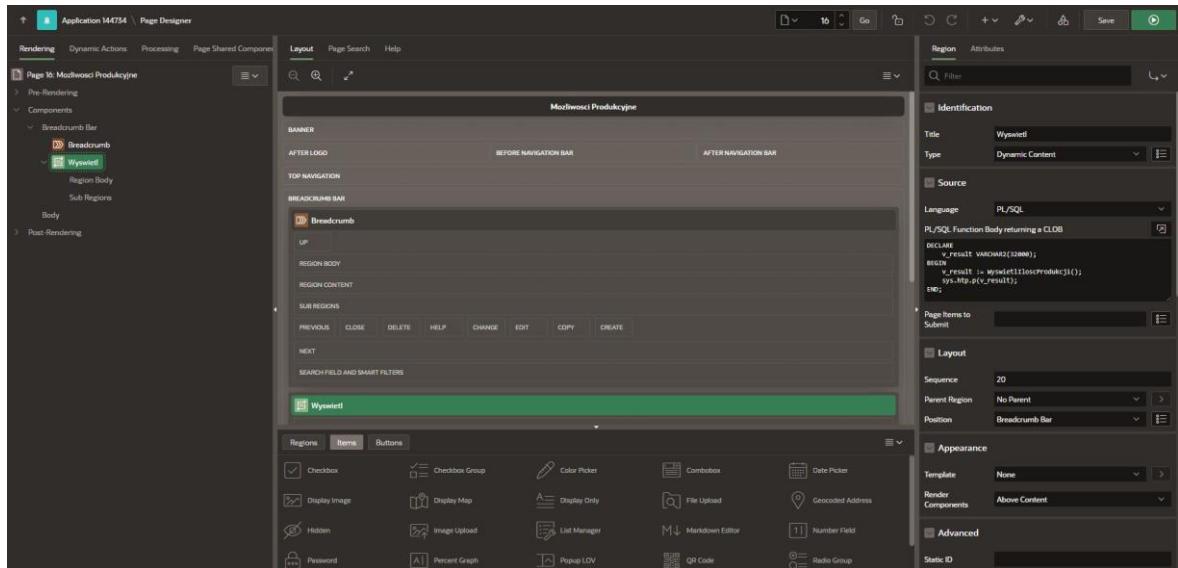
The screenshot shows the 'Wyświetl klientów' page. At the top, there's a header with a search bar and a user email 'patrykdatadev7@gmail.com'. Below the header is a table with columns: Klient Id, Nazwa Klienta, Miejscowosc, Email, Telefon, Typ Klienta, and Id Kod Adresu. The table contains 43 rows of client data. The last row is highlighted with a light blue background. At the bottom of the table, there's a page number '1 - 18'.

Klient Id	Nazwa Klienta	Miejscowosc	Email	Telefon	Typ Klienta	Id Kod Adresu
744	Super klient	Rzeszow	abcc@gmail.com	123345678	staly	744
1	AGD Master	Ciechanow	agdmast@gmail.com	721984368	normalny	1
2	Ekspert AGD	Bytom	ekspertagd@interia.com	234561297	normalny	3
3	AGD Partner	Bialystok	agdpartner@o2.pl	943712648	normalny	5
4	RTVAGD	Zary	rteagi@gmail.com	123987852	staly	7
5	AGD Solutions	Kielce	agdsolution@wp.pl	497123649	normalny	9
6	AGD Master	Ciechanow	agdmast@gmail.com	721984368	staly	11
7	Media Expert	Radom	mediaexp@interia.pl	863147456	normalny	13
8	AGD Tech	Warszawa	agdtch@gmail.com	963741258	normalny	15
9	Elektroda	Sosnowiec	elektroda@wp.pl	249731852	normalny	2
10	AGD Pro	Ciechanow	agdmast@gmail.com	721984368	normalny	4
11	Supco	Krakow	supco@wp.pl	421968376	normalny	6
12	InterComp	Rzeszow	intercomp@interia.pl	654321789	normalny	8
13	Apollo	Krakow	apollo@o2.pl	369741456	normalny	10
14	AGD Master	Ciechanow	agdmast@gmail.com	721984368	normalny	12
15	G2A	Warszawa	g2a@o2.pl	147654789	staly	14
20	asdasd	sdfsdfsf	sdfsdf@com	+123123121	normalny	27
432	Nazw sklep	Rzeszow	nazw@klep.pl	948234923	staly	156

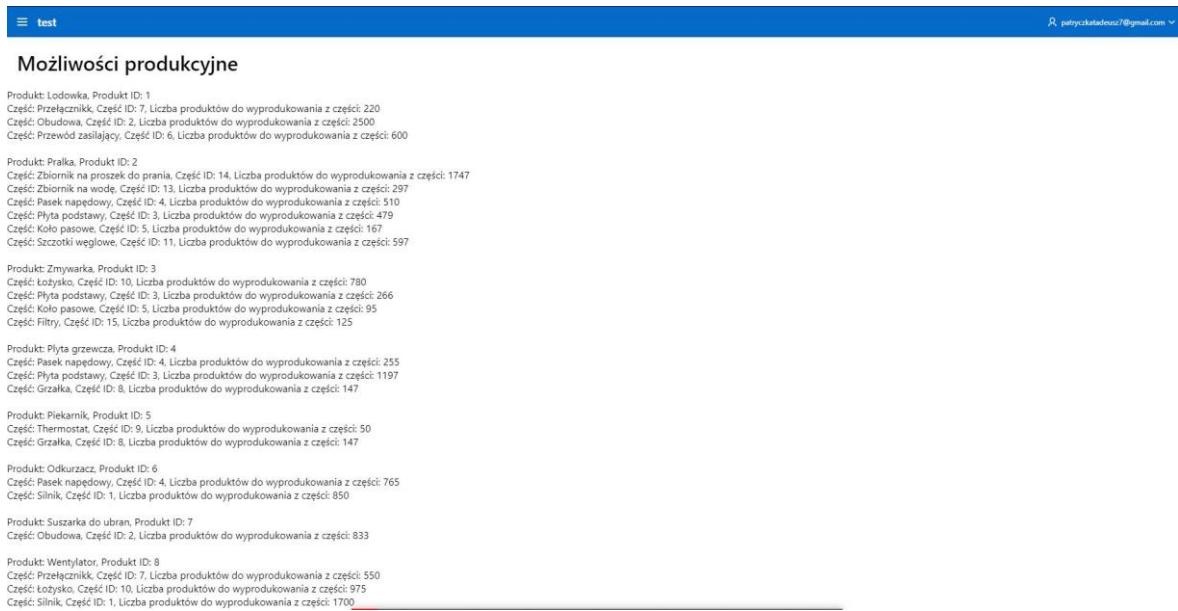
Rysunek 77 Działanie strony

6.12 Strona Możliwości Produkcyjne

Działanie strony umożliwia przegląd ile danego asortymentu jesteśmy w stanie wyprodukować z dostępnych części zgromadzonych w magazynie.



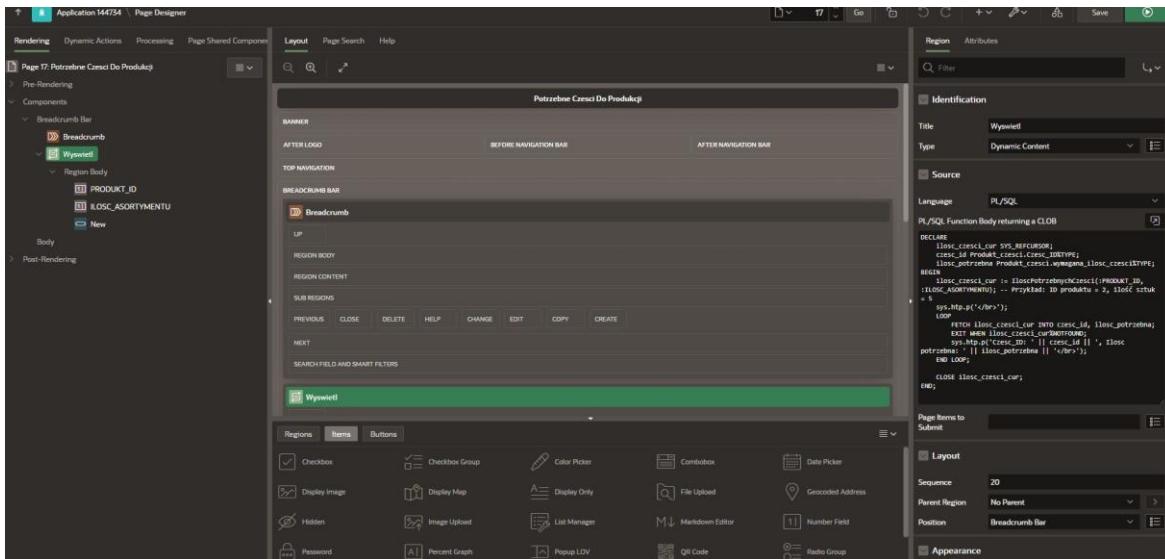
Rysunek 78 Struktura strony



Rysunek 79 Działanie strony

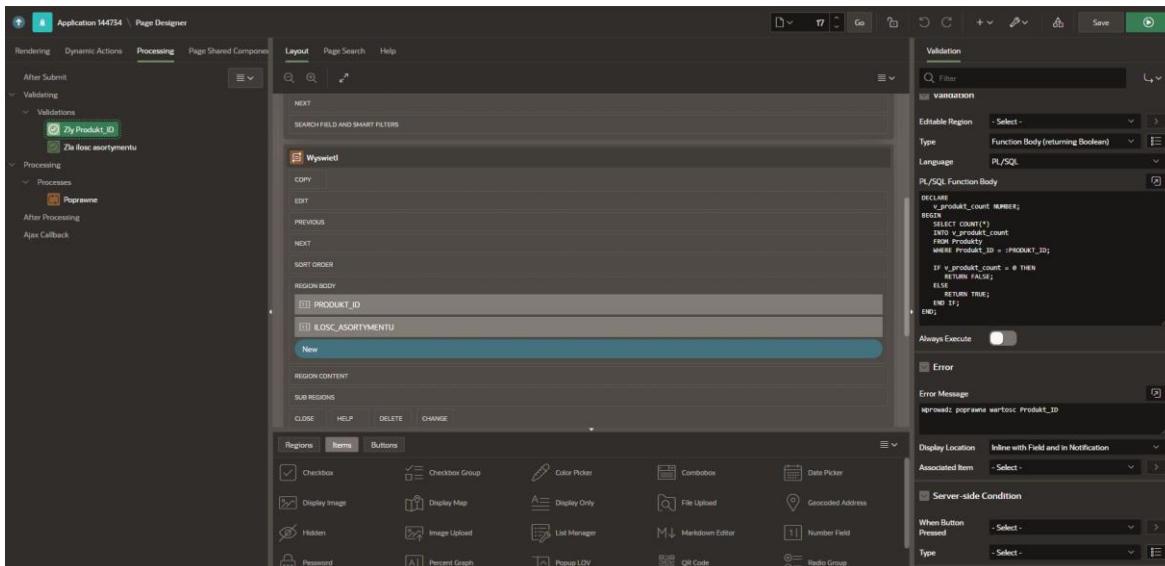
6.13 Strona Potrzebne Części Do Produkcji

Działanie strony polega na dostarczeniu użytkownikowi strony wiedzy dotyczącej ile i jakich części musi zostać zużyte aby móc wyprodukować zadaną ilość konkretnego asortymentu.



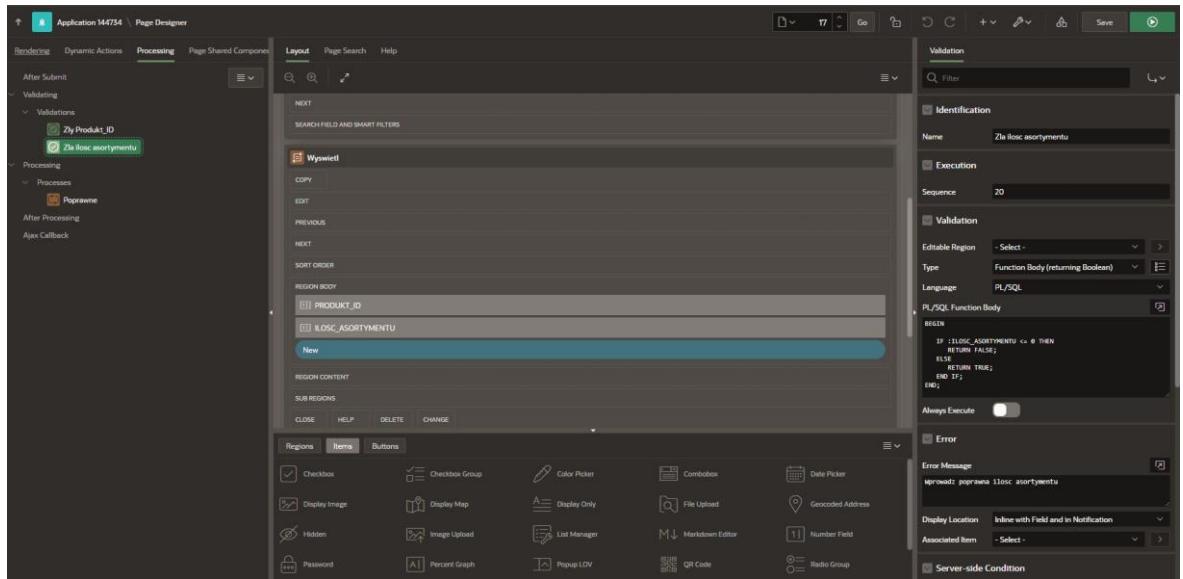
Rysunek 80 Struktura strony

Zrzut ekranu przedstawiający walidację, która wyświetla błąd w momencie gdy wartość przekazana do parametru Produkt_ID nie występuje w tabeli Produkty.

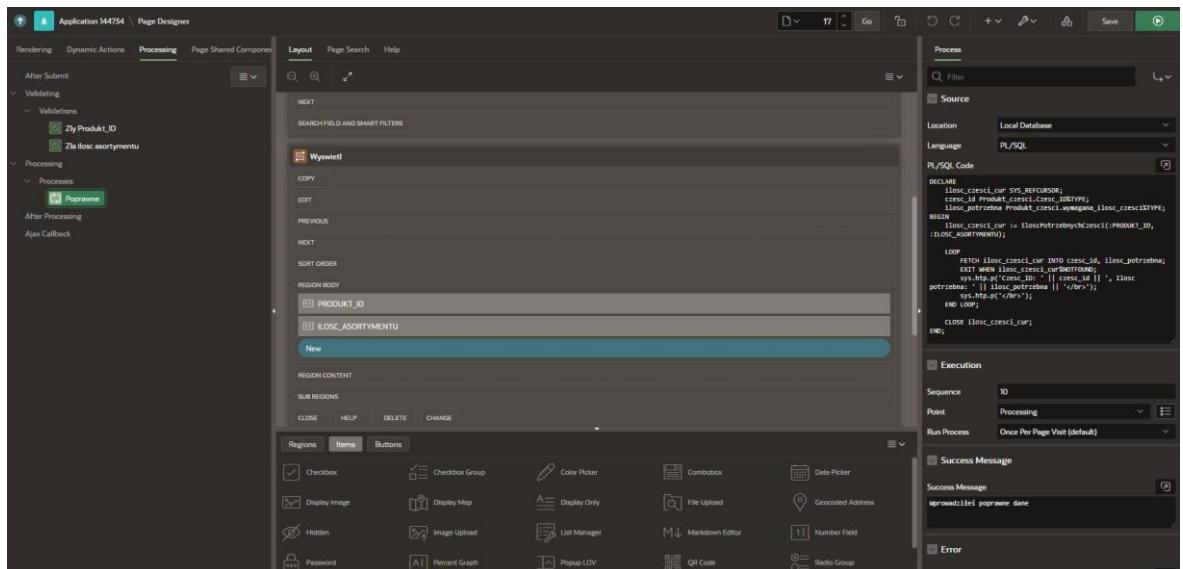


Rysunek 81 Walidacja Zły Produkt_ID

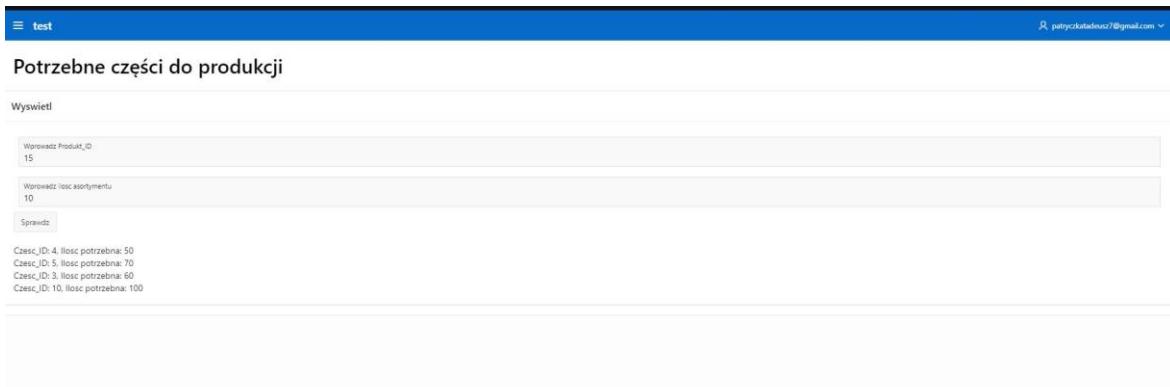
Zrzut ekranu przedstawiający validację, która wyświetla błąd w momencie, gdy przekazana wartość do parametru ilość_asortymentu jest mniejsza lub równa 0.



Rysunek 82 Walidacja Zla ilosc_asortymentu



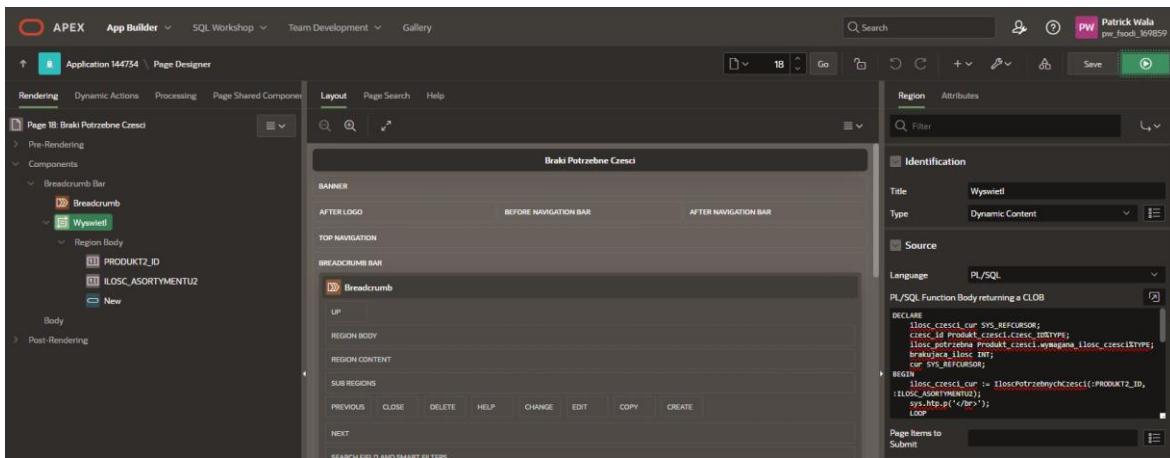
Rysunek 83 Wywolanie procedury



Rysunek 84 Działanie strony

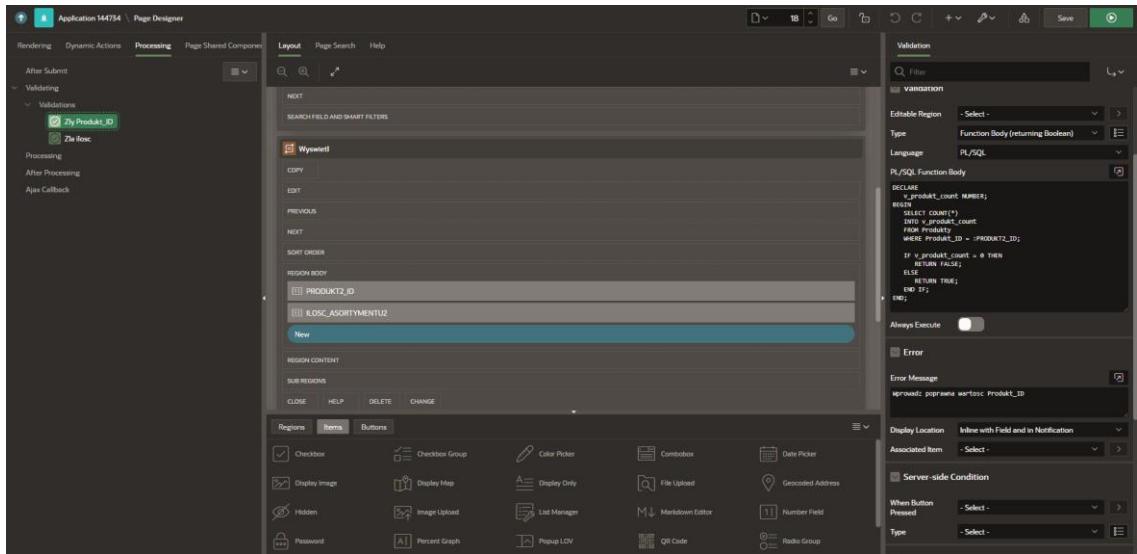
6.14 Strona Braki Potrzebne Części

Działanie strony jest modyfikacją strony Potrzebne Części do Produkcji dodając funkcjonalność wyświetlania brakującej liczby części w momencie, gdy przekazujemy bardzo dużą wartość asortymentu (taką, która przekracza zapas części w magazynie).



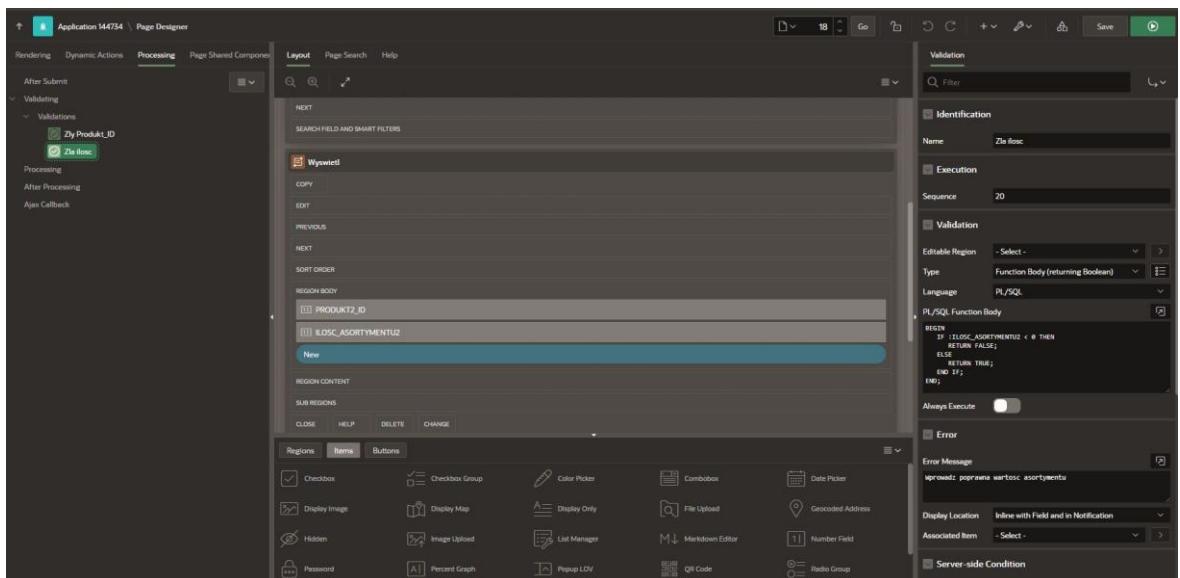
Rysunek 85 Struktura strony

Zrzut ekranu przedstawiający walidacje, która wyświetla błąd w momencie gdy wartość przekazana do parametru Produkt_ID nie występuje w tabeli Produkty.



Rysunek 86 Walidacja Zły Produkt_ID

Zrzut ekranu przedstawiający walidacje, która wyświetla błąd w momencie, gdy przekazana wartość do parametru ilość_asortymentu jest mniejsza lub równa 0.



Rysunek 87 Walidacja Zła ilość

Zrzut ekranu przedstawiający okno w momencie wprowadzenia niepoprawnych danych do parametrów.



Rysunek 88 Wizualizacja validacji

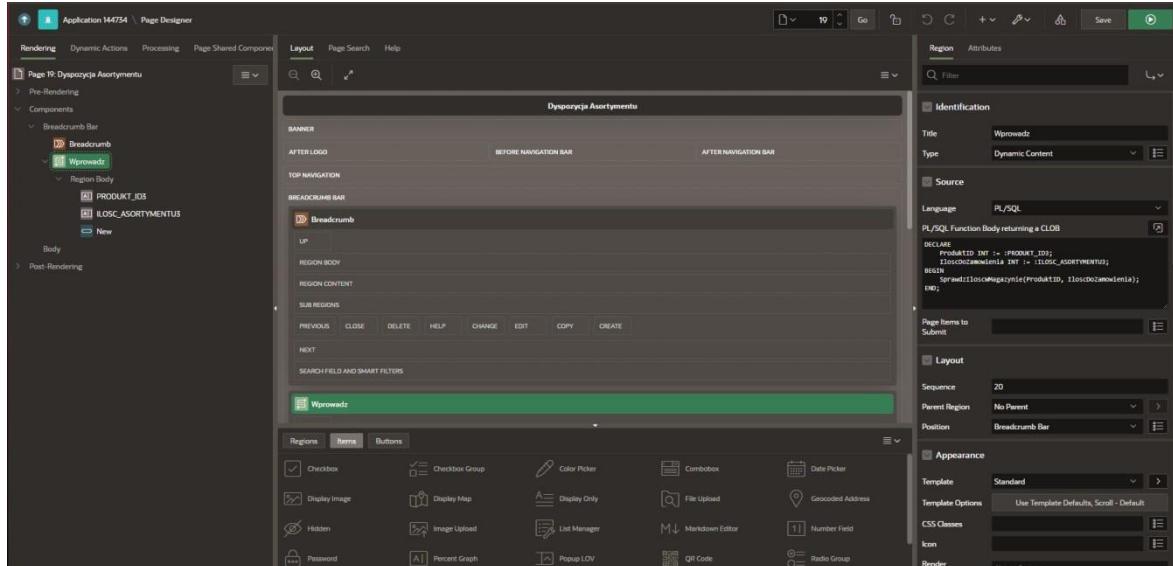
Widok przedstawiający poprawne przekazanie danych do parametrów



Rysunek 89 Działanie kodu strony

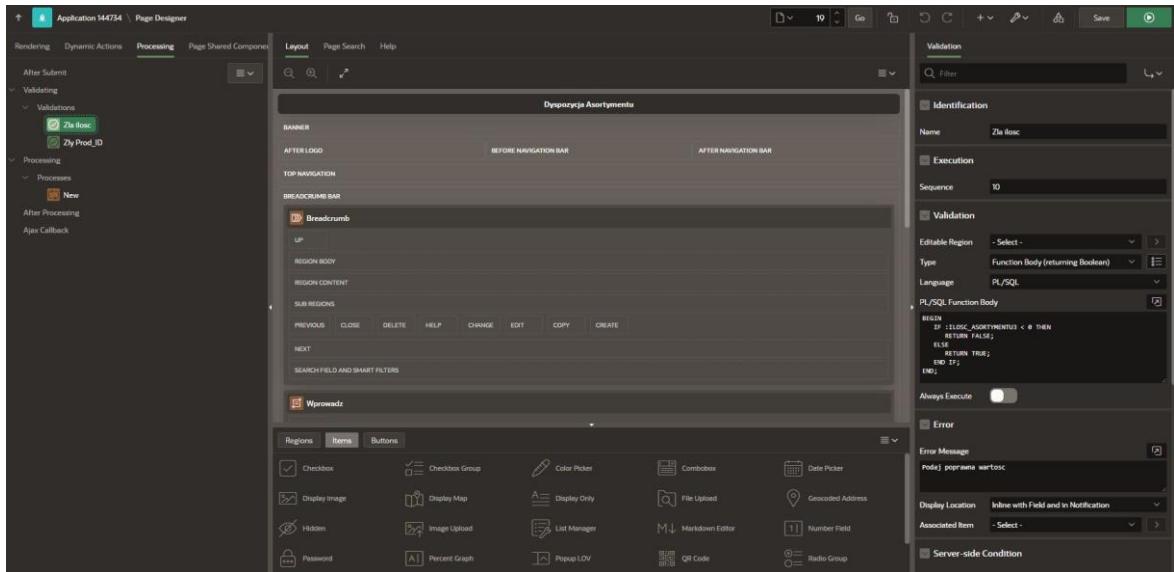
6.15 Strona Dyspozycja Asortymentu

Działanie strony ma na celu udostępnienie informacji dotyczącej zmagazynowanego asortymentu, użytkownik wprowadza dane dotyczące produktu i w odpowiedzi dostaje informacje, czy posiadamy wystarczającą ilość asortymentu w magazynie do zrealizowania zamówienia.



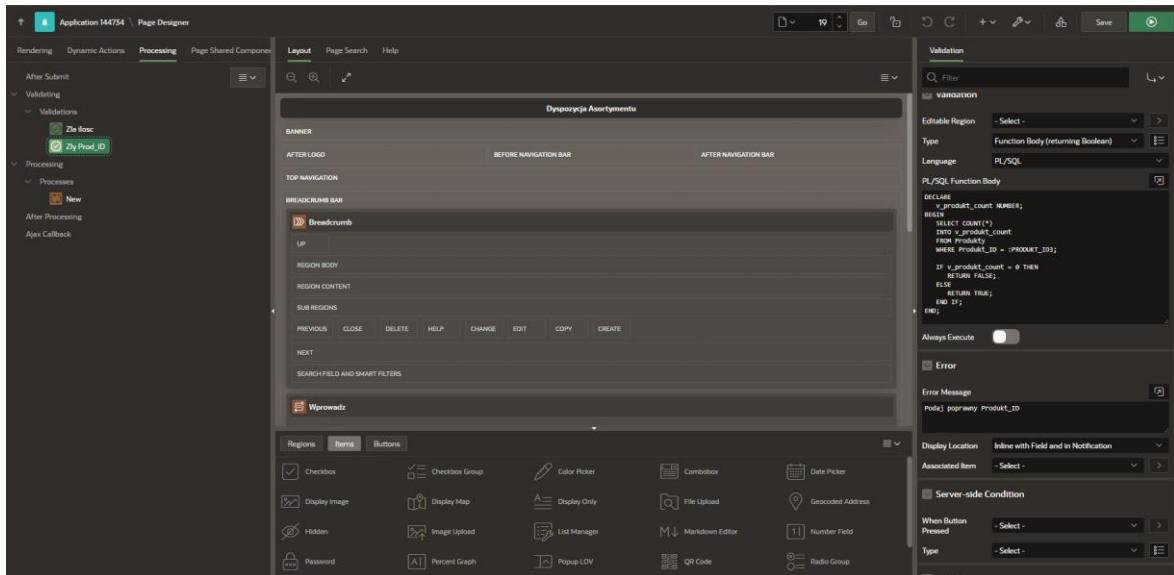
Rysunek 90 Struktura strony

Walidacja parametru ilość, zwracająca błąd w przypadku wprowadzenia wartości mniejszych od 0.

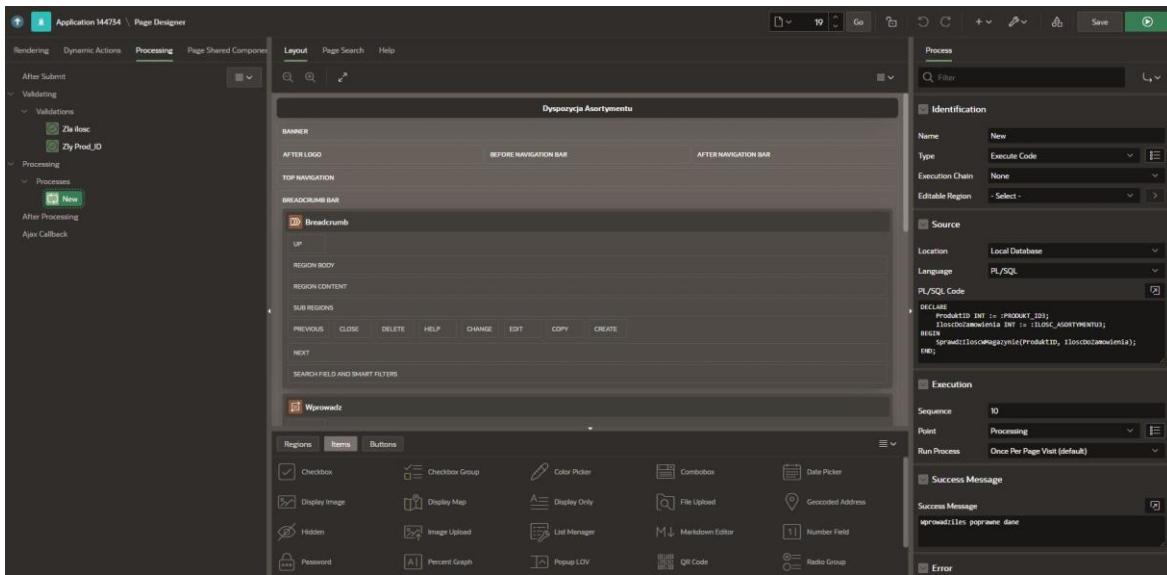


Rysunek 91 Walidacja Zla ilość

Walidacja parametru Prod_ID, użytkownikowi strony wyświetlony zostaje błąd w momencie, gdy przekazana wartość tego parametru nie będzie występować w tabeli Produkty.

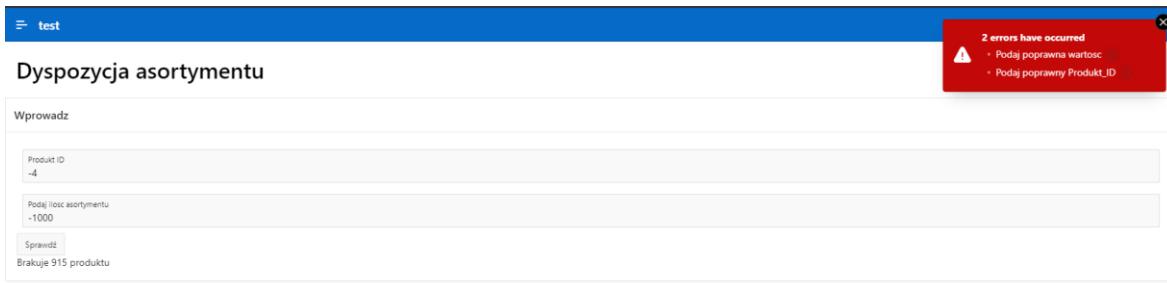


Rysunek 92 Walidacja Zly Prod_ID



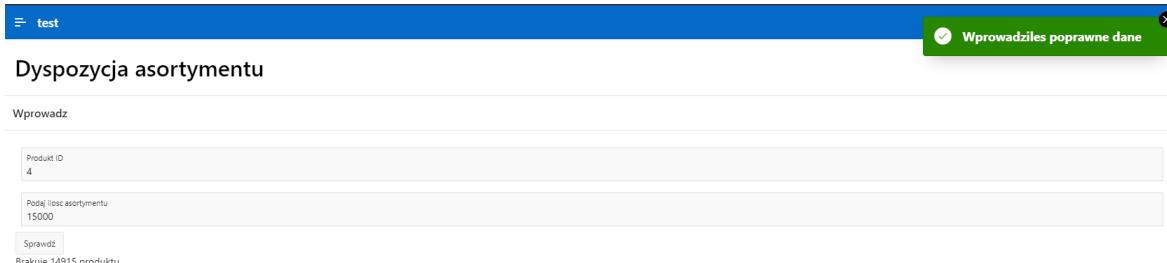
Rysunek 93 Wywołanie kodu strony

Wyświetlenie stosownego komunikatu w momencie przekazanie niewłaściwych danych.



Rysunek 94 Wizualizacja validacji

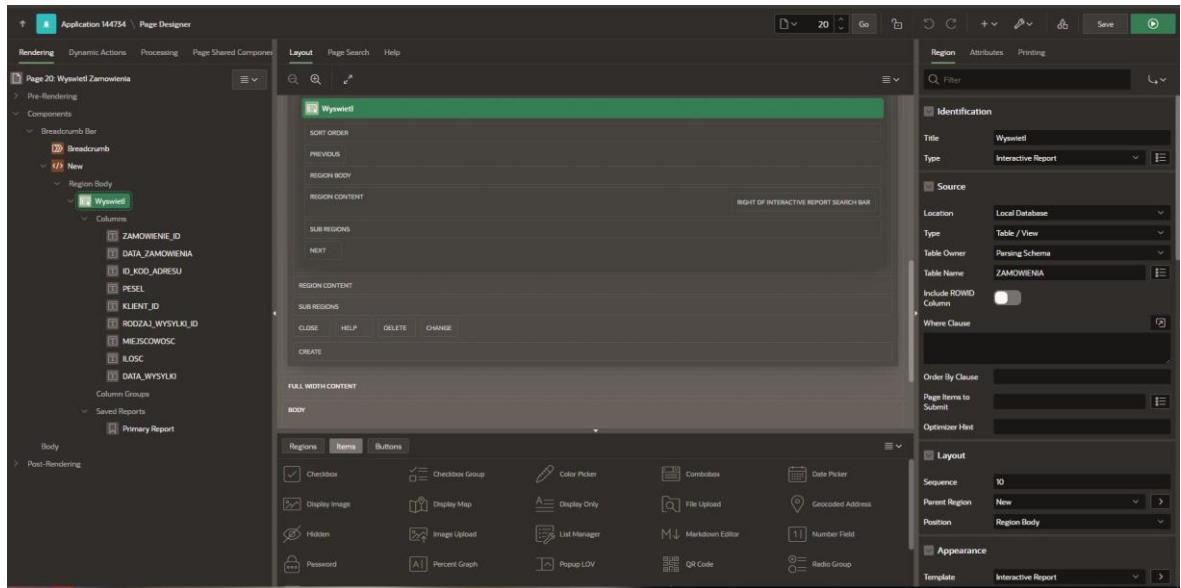
W momencie przekazania poprawnych wartości użytkownikowi wyświetla się informacja zgodna ze stanem tabeli magazyn produktów.



Rysunek 95 Działanie kodu strony

6.16 Strona Wyświetl Zamówienia

Działanie strony polega na wyświetlaniu zawartości tabeli Zamówienia, ich filtracjach i sortowaniu, oraz wyszukiwanie spersonalizowanych informacji, w celu realizacji celów biznesowych przedsiębiorstwa.



Rysunek 96 Struktura strony

The screenshot shows the 'Wyświetl zamówienia' report page. The top navigation bar includes 'test', a user icon, and an email address 'patrycja.dabrowska7@gmail.com'. The main content area has a header 'Wyświetl zamówienia' and a sub-header 'New'. Below this is a table with columns: 'Zamówienie Id', 'Data Zamówienia', 'Id Kod Adresu', 'Pesel', 'Klient Id', 'Rodzaj Wysyłki Id', 'Miejscowość', 'Ilosc', and 'Data Wysyłki'. The table contains 20 rows of data, each representing an order record with details such as date, address ID, client ID, and delivery method ID.

Zamówienie Id	Data Zamówienia	Id Kod Adresu	Pesel	Klient Id	Rodzaj Wysyłki Id	Miejscowość	Ilosc	Data Wysyłki
1	11/1/2022	1	56123698741	1	5	Kaczory	15	11/4/2022
2	11/3/2022	3	8973951456	3	4	Klecko	25	11/5/2022
3	11/6/2022	5	2375951123	5	4	Malbork	35	11/8/2022
4	11/7/2022	7	56123698741	7	3	Luban	50	11/10/2022
5	11/9/2022	9	2375951123	9	10	Łowicz	55	11/10/2022
6	11/9/2022	11	8973951456	11	1	Łeba	100	11/11/2022
7	11/11/2022	13	2375951123	13	1	Olkusz	70	11/12/2022
8	11/13/2022	15	56123698741	15	5	Olsztyn	450	11/15/2022
9	11/14/2022	2	2375951123	2	4	Sierpc	100	11/15/2022
10	11/14/2022	4	56123698741	4	3	Piawa	200	11/15/2022
11	11/17/2022	6	2375951123	6	1	Tuczno	50	11/19/2022
12	11/19/2022	8	8973951456	8	1	Rawicz	50	11/21/2022
13	11/21/2022	10	8973951456	10	1	Mielec	50	11/22/2022
14	11/23/2022	12	56123698741	12	2	Tuchola	5	11/24/2022
15	11/25/2022	14	2375951123	15	5	Strobin	80	11/26/2022
16	11/25/2022	1	2375951123	1	5	Kaczory	80	11/27/2022
17	11/26/2022	2	56123698741	2	5	Sierpc	110	11/28/2022
18	11/27/2022	13	8973951456	13	1	Olkusz	40	11/28/2022
19	11/28/2022	7	2375951123	7	3	Luban	25	11/29/2022
20	11/29/2022	4	56123698741	4	3	Piawa	80	11/20/2022

Rysunek 97 Działanie strony

7 Eksport aplikacji i import

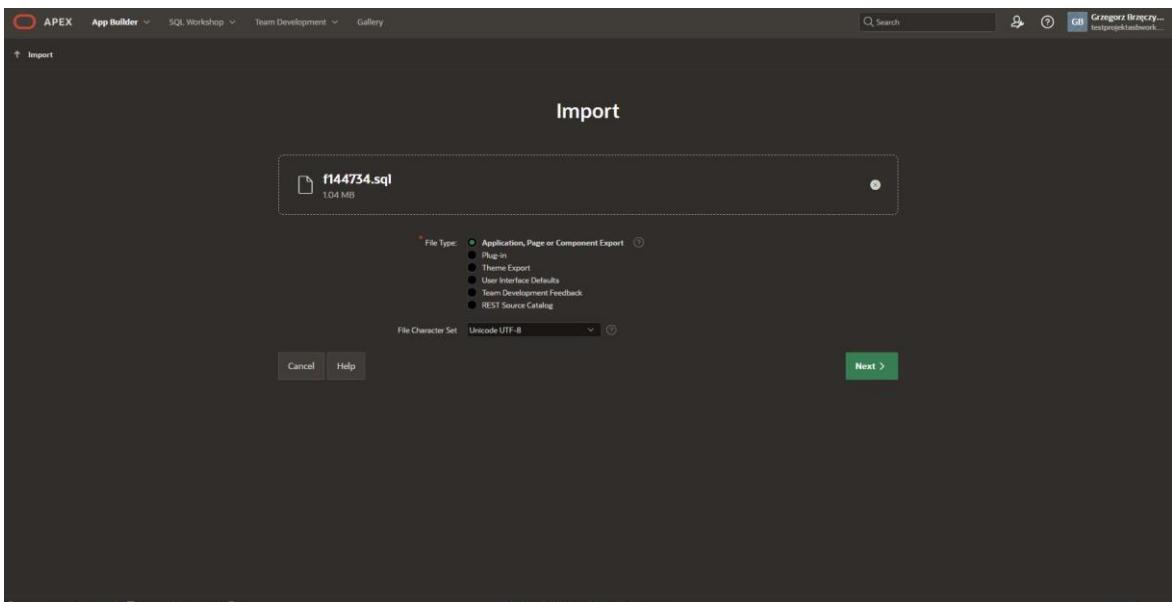
Aby dokonać eksportu aplikacji najpierw utworzyliśmy skrypty instalacyjne dla typów, tabel, procedur i funkcji w zakładce Supporting Objects. Następnie na nowo utworzonym koncie zainstalowaliśmy aplikacje.

Lock	Name	Sequence	Script	Status	Updated	Updated By	Source
✓	tabela	10	CREATE TABLE "ADRESY" ("ID_KOD_ADRESU" NUMBER(*,0) NOT NULL ENABLE, "KOD_POCZTOWY" VARCHAR2(6) NOT NULL ENABLE, "ULICA" VARCHAR2(45), "NR_DOMU_LOKALU" VARCHAR2(45), PRIMARY KEY ...);	45 minutes ago		patrikzakatausz7@gmail.com	Database Object
✓	funkcja	20	create or replace TYPE address_info AS OBJECT (id_kod_adresu INTEGER, kod_pocztyowy VARCHAR(6), ulica VARCHAR(45), Nr_domu_lokalu VARCHAR(45)) / create or replace TYPE Brakuje...	45 minutes ago		patrikzakatausz7@gmail.com	Database Object
✓	funkcje	30	create or replace FUNCTION CheckCustomerData RETURN SYS_REFCURSOR AS v_cursor SYS_REFCURSOR; BEGIN OPEN v_cursor FOR SELECT Klient_ID, Nazwa_Milenta, Miejscowosc, Email, Tel...	45 minutes ago		patrikzakatausz7@gmail.com	Database Object
✓	procedury	40	create or replace PROCEDURE AddAddressDetails (address_data IN address_info) AS BEGIN IF address_data.Nr_domu_lokalu IS NULL OR address_data.ulica IS NULL, THEN DBMS_OUTPUT.PUT_L...	45 minutes ago		patrikzakatausz7@gmail.com	Database Object

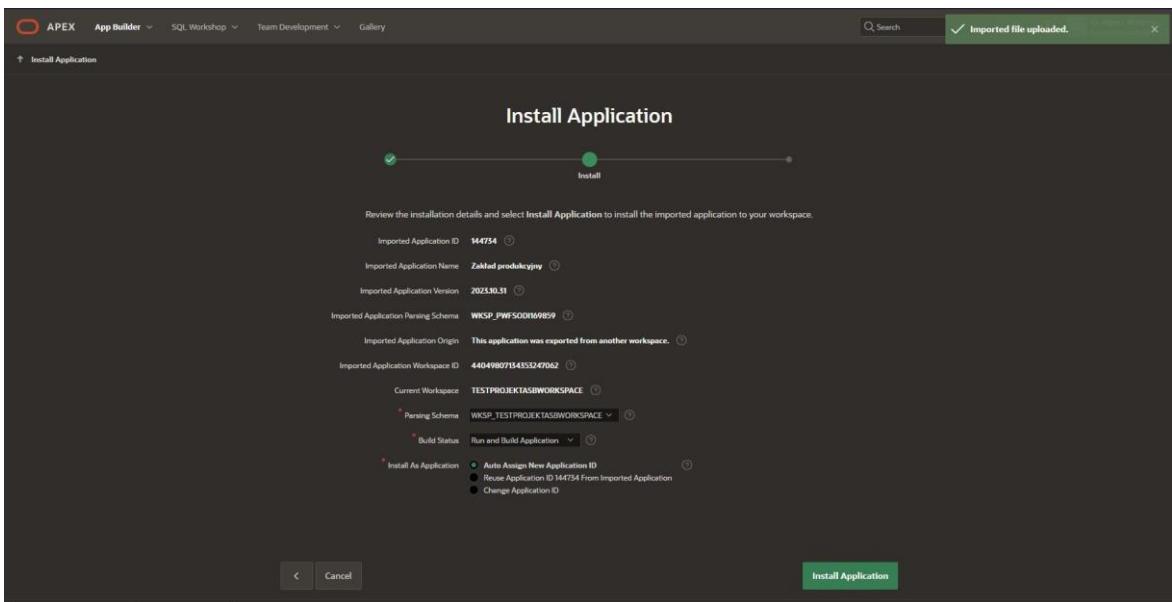
Rysunek 98 Utworzone skrypty instalacyjne

Application	Name	Pages	Updated By	Updated	Owner
144754	Zakład produkcyjny	19	PATRYCZKATADEUSZ7@GMAIL.COM	45 minutes ago	WKSP_PWFSD0169859

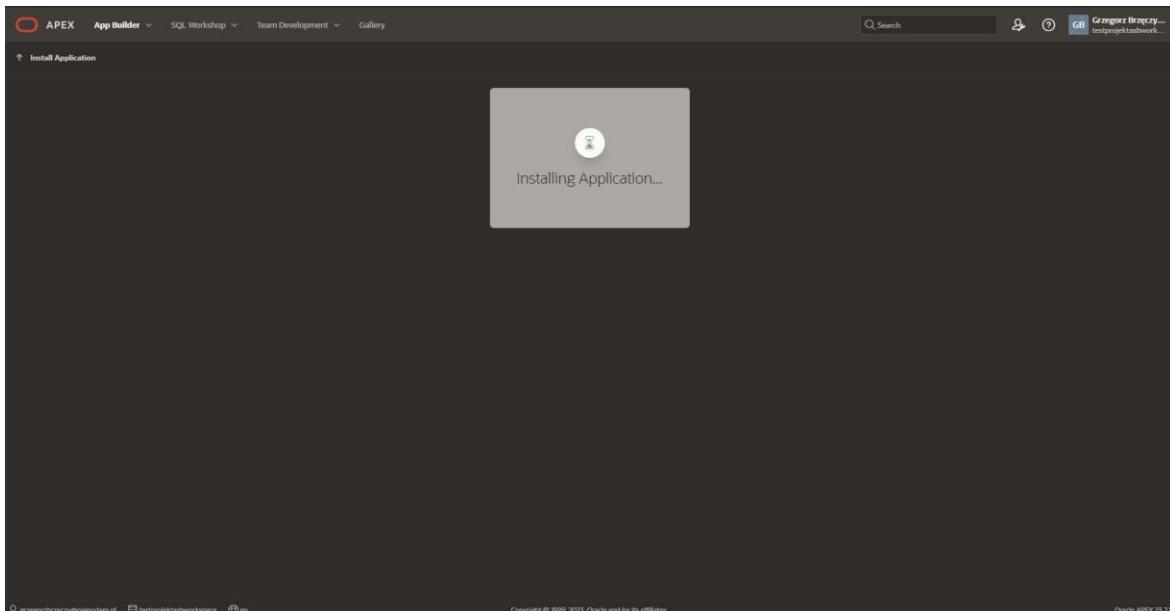
Rysunek 99 Ekran eksportu aplikacji



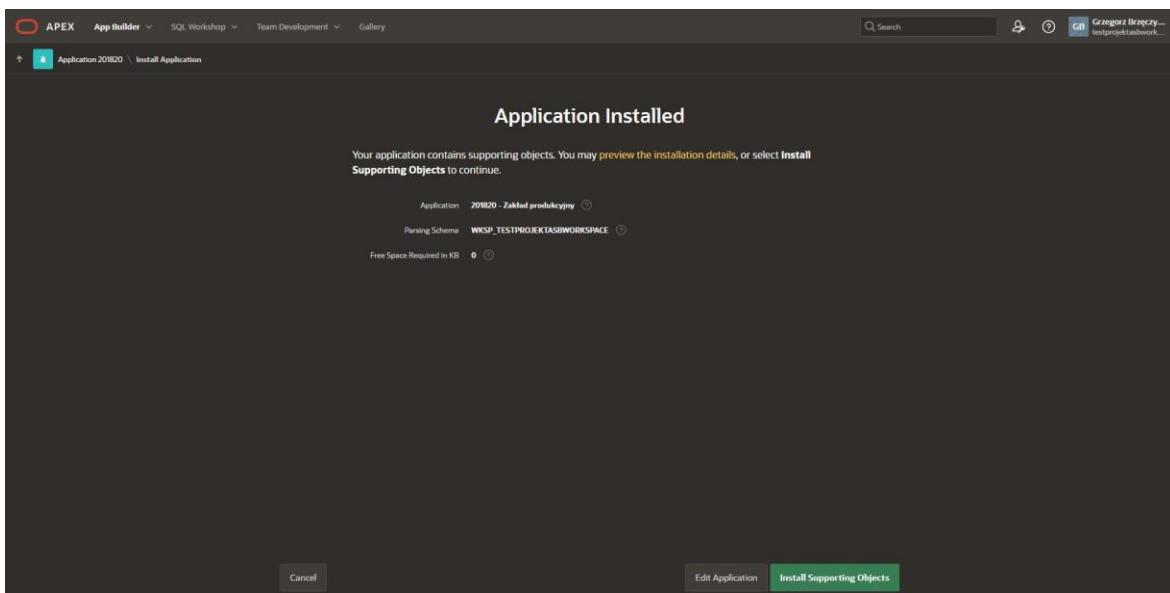
Rysunek 100 Ekran importu aplikacji – wybranie pliku z eksportem



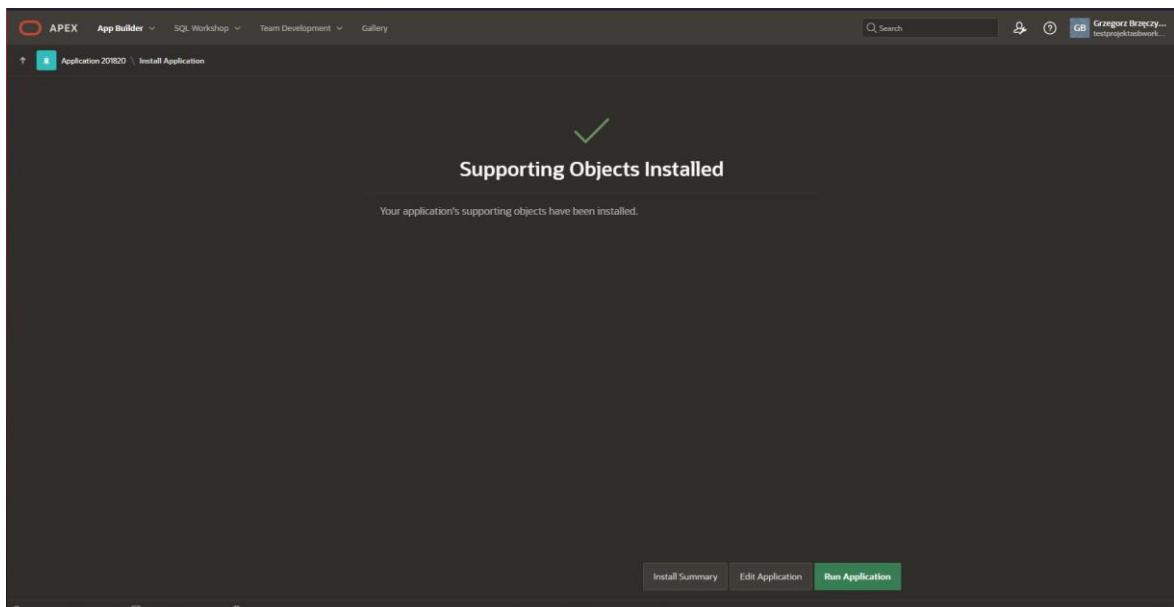
Rysunek 101 Ustawienia importu



Rysunek 102 Proces instalowania importowanej aplikacji

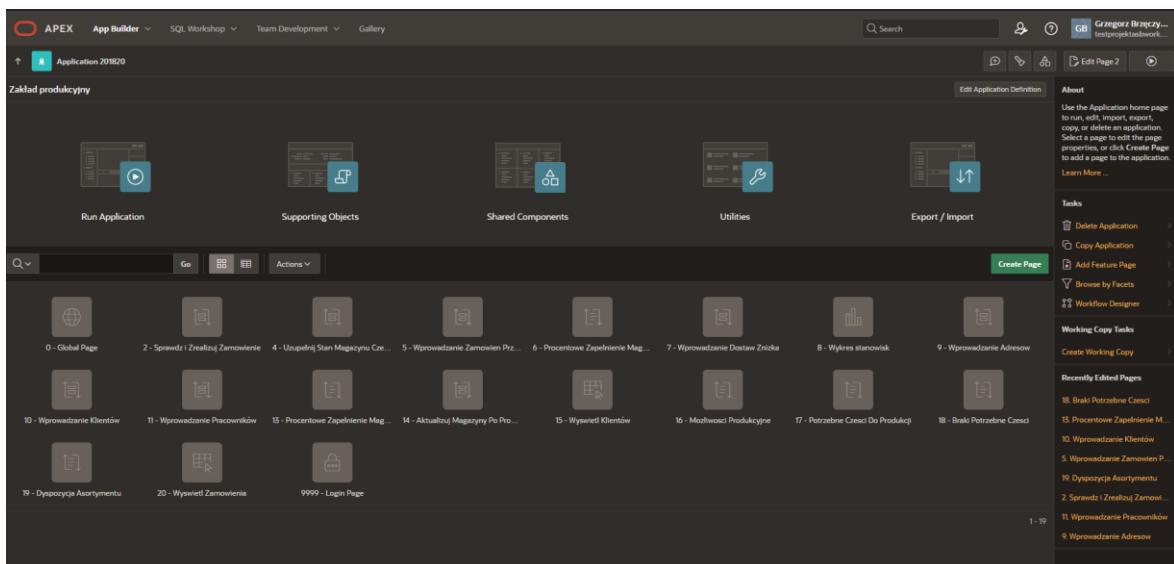


Rysunek 103 Informacja o potrzebie zainstalowania Supported Objects



Rysunek 104 Informacja o poprawnie zainstalowanych Supported Objects

Zrzut ekranu przedstawiający poprawne zainstalowanie całej aplikacji dla nowego konta użytkownika, nie powiązanego z wcześniej z aplikacją.



Rysunek 105 Strony w zainstalowanej aplikacji

Otwarcie strony po zimportowaniu na innym koncie.

Sprawdź i zrealizuj zamówienie

Pomyślnie odjęto liczbę z magazynu

Produkt Id
14

Podaj ilosc assortymentu
176

New

Wprowadź

Rysunek 106 Przykładowa strona z poprawnym działaniem

8. Podsumowanie

Realizując projekt mieliśmy za zadanie stworzenie aplikacji ułatwiającej zarządzanie bazą danych dla przedsiębiorstwa zakładu produkcyjnego. Skupiliśmy się na uwzględnieniu kluczowych aspektów pracy przedsiębiorstwa, zapewniając kompleksowe rozwiązania. Podczas tworzenia aplikacji skorzystaliśmy z zaawansowanych możliwości języka PL/SQL, opierając się m.in. na procedurach, funkcjach, kurSORach kolekcjach, dzięki walidacji błędów wpłynęliśmy na efektywność i funkcjonalność systemu. Warto podkreślić, że obsługa błędów obejmuje błędy predefiniowane, niepredefiniowane, oraz obsługę błędów użytkownika, znacząco ograniczając awaryjność systemu.

9. Dane do logowania

Aplikacja	
Workspace:	pw_fsodi_169859
Username:	patryczkatadeusz7@gmail.com
Hasło:	1qazXSW@

10. Kod PL/SQL

```
-- DECLARE TYPE
create or replace TYPE address_info AS OBJECT (
    id_kod_adresu INTEGER,
    kod_pocztyowy VARCHAR(6),
    ulica VARCHAR(45),
```

```

        Nr_domu_lokalu VARCHAR(45)
    )
    /
create or replace TYPE BrakujacyProdukt AS OBJECT (
    Produkt_ID INT,
    Brakujaca_ilosc INT
)
/
create or replace TYPE client_info AS OBJECT (
    Klient_ID Integer,
    Nazwa_klienta VARCHAR(45),
    Miejscowosc VARCHAR(45),
    Email VARCHAR(45),
    Telefon VARCHAR(45),
    Typ_klienta VARCHAR(45),
    id_kod_adresu VARCHAR(45)
)
/
create or replace TYPE Czesclist AS TABLE OF Czescrecord
/
create or replace TYPE Czescrecord AS OBJECT (
    Czesc_ID INT,
    Ilosc_potrzebna INT
)
/
create or replace TYPE employee_info AS OBJECT (
    PESEL VARCHAR(11),
    Imie VARCHAR(45),
    Nazwisko VARCHAR(45),
    id_kod_adresu INT,
    Linia_ID INT,
    id_zatrudnienia INT,
    Data_urodzenia DATE,
    Miejscowosc VARCHAR(45),
    Email VARCHAR(45),
    Telefon VARCHAR(20),
    Wyksztalcenie VARCHAR(45),
    Stanowisko VARCHAR(45),
    Pensja FLOAT,
    Data_zatrudnienia DATE,
    Umowa_nazwa VARCHAR(45),
    Dzial_nazwa VARCHAR(45),
    Opis_zmiany VARCHAR(45)
)
/

```

```

create or replace TYPE KlienciList AS TABLE OF KlientType
/
create or replace TYPE KlientType AS OBJECT (
    Klient_ID INT,
    Nazwa_klienta VARCHAR(45),
    Miejscowosc VARCHAR(45),
    Email VARCHAR(45),
    Telefon VARCHAR(20),
    Typ_klienta VARCHAR(45),
    id_kod_adresu INT
)
/
create or replace TYPE ListaBrakujacychProduktow AS TABLE OF
BrakujacyProdukt
/
create or replace TYPE ProductStorageInfo AS OBJECT (
    Lokacja_ID INT,
    Lokacja_nazwa VARCHAR(45),
    Ilosc_w_magazynie INT,
    Maksymalna_pojemnosc INT,
    Produkt_ID INT,
    Produkt_zapelnienie NUMBER
)
/
create or replace TYPE ProductStorageInfoList AS TABLE OF ProductStorageInfo
/
create or replace TYPE Produkty_Collection AS TABLE OF Produkt_Object
/
create or replace TYPE Produkt_Object AS OBJECT (
    Produkt_ID INT,
    Brakujaca_ilosc INT
)
/
create or replace TYPE SzczegolyZamowienia_Object AS OBJECT (
    Szczegoly_zamowienia_ID INT,
    Znizka FLOAT,
    Lokacja_ID INT,
    Zamowienie_ID INT,
    Kwota FLOAT,
    Rodzaj_platnosci VARCHAR(45),
    Przewidywany_czas_realizacji DATE
)
/

```

```

)
/

-- PROCEDURY

create or replace PROCEDURE AddAddressDetails (
    address_data IN address_info
) AS
BEGIN
    IF address_data.Nr_domu_lokalu IS NULL OR address_data.ulica IS NULL
THEN
        DBMS_OUTPUT.PUT_LINE('Któryś z atrybutów jest pusty: Nr_domu_lokalu
or ulica.');
        RAISE_APPLICATION_ERROR(-20202, 'Puste atrybuty');
ELSIF NOT REGEXP_LIKE(address_data.kod_pocztowy, '^\\d{2}-\\d{3}$') THEN
        DBMS_OUTPUT.PUT_LINE('Błąd. Zły format kodu pocztowego: ' ||
address_data.kod_pocztowy);
        RAISE_APPLICATION_ERROR(-20202, 'Błąd. Zły format kodu
poczgowego.');
ELSE
        INSERT INTO Adresy (id_kod_adresu, kod_pocztowy, ulica,
Nr_domu_lokalu)
        VALUES (address_data.id_kod_adresu, address_data.kod_pocztowy,
address_data.ulica, address_data.Nr_domu_lokalu);
        apex_application.g_print_success_message:='Sukces. ';
END IF;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20202, 'Wystąpił błąd: ' || SQLERRM);
END;
/

```

```

create or replace PROCEDURE AktualizujIloscWMagazynie (
    p_Produkt_ID Magazyn_produktow.Produkt_ID%TYPE,
    p_Ilosc_asortymantu Magazyn_produktow.Ilosc_w_magazynie%TYPE
)
IS
BEGIN
    UPDATE Magazyn_produktow
    SET Ilosc_w_magazynie = Ilosc_w_magazynie + p_Ilosc_asortymantu
    WHERE Produkt_ID = p_Produkt_ID;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
END AktualizujIloscWMagazynie;
/

```

```

create or replace PROCEDURE AktualizujMagazynyCzesci(p_czesci_cur
SYS_REFCURSOR) AS
    v_czesc_id Magazyn_czesci.Czesc_ID%TYPE;
    v_ilosc_potrzebna INT;
    v_brakuje BOOLEAN := false;
BEGIN
    LOOP
        FETCH p_czesci_cur INTO v_czesc_id, v_ilosc_potrzebna;
        EXIT WHEN p_czesci_cur%NOTFOUND;

        FOR magazyn IN (SELECT Ilosc_w_magazynie FROM Magazyn_czesci WHERE
Czesc_ID = v_czesc_id) LOOP
            IF v_ilosc_potrzebna > magazyn.Ilosc_w_magazynie THEN
                v_brakuje := true;
                EXIT;
            END IF;
        END LOOP;

        IF v_brakuje THEN
            sys.hpt.p('Brak dla Czesc_ID ' || v_czesc_id || ' - ' ||
v_ilosc_potrzebna);
            v_brakuje := false;
        ELSE

            UPDATE Magazyn_czesci
            SET Ilosc_w_magazynie = Ilosc_w_magazynie - v_ilosc_potrzebna
            WHERE Czesc_ID = v_czesc_id;

            sys.hpt.p('Zaktualizowano Magazyn_czesci dla Czesc_ID ' ||
v_czesc_id);
        END IF;
    END LOOP;

    CLOSE p_czesci_cur;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        sys.hpt.p('Nie znaleziono informacji o danej części w magazynie.');
    WHEN OTHERS THEN
        sys.hpt.p('Wystąpił inny błąd.');
END;
/

```

```

create or replace PROCEDURE DodajCzescZamowienia (
    p_czesci_zamowienia_ID IN Czesci_Zamowienia.czesci_zamowienia_ID%TYPE,

```

```

    p_Lokacja_ID IN Magazyn_czesci.Lokacja_ID%TYPE
) AS
    v_ilosc_zamowionych_czesci INT;
    v_count NUMBER;
    CURSOR c_magazyn_czesci (p_Lokacja_ID Magazyn_czesci.Lokacja_ID%TYPE) IS
        SELECT Max_pojemnosc - Ilosc_w_magazynie AS roznica
        FROM Magazyn_czesci
        WHERE Lokacja_ID = p_Lokacja_ID;
    BAD_LOCAL EXCEPTION;
BEGIN

    SELECT COUNT(*)
    INTO v_count
    FROM Magazyn_czesci
    WHERE Lokacja_ID = p_Lokacja_ID;

    IF v_count = 0 THEN
        RAISE BAD_LOCAL;
    END IF;

    OPEN c_magazyn_czesci(p_Lokacja_ID);
    FETCH c_magazyn_czesci INTO v_ilosc_zamowionych_czesci;
    CLOSE c_magazyn_czesci;

    INSERT INTO Czesci_Zamowienia (czesci_zamowienia_ID, Lokacja_ID,
    ilosc_zamowionych_czesci)
    VALUES (p_czesci_zamowienia_ID, p_Lokacja_ID,
    v_ilosc_zamowionych_czesci);

    COMMIT;
    sys.htp.p('Dodano nowy rekord do tabeli Czesci_Zamowienia.');
EXCEPTION
    WHEN BAD_LOCAL THEN
        sys.htp.p('Podane Lokacja_ID nie istnieje w tabeli
Magazyn_czesci.');
    WHEN NO_DATA_FOUND THEN
        sys.htp.p('Brak danych dla podanej Lokacja_ID.');
    WHEN OTHERS THEN
        sys.htp.p('Wystąpił błąd: ' || SQLERRM);
        ROLLBACK;
END;
/
create or replace PROCEDURE DodajDostawe (
    p_Dostawa_ID IN Dostawy.Dostawa_ID%TYPE,
    p_Ilosc IN Dostawy.Ilosc%TYPE,
    p_Czesc_ID IN Dostawy.Czesc_ID%TYPE,
    p_PESEL IN Dostawy.PESEL%TYPE,
    p_Dostawca_ID IN Dostawy.Dostawca_ID%TYPE,
    p_Kwota_bez_zniki IN Dostawy.Kwota_bez_znizki%TYPE,
    p_Data_dostawy IN Dostawy.Data_dostawy%TYPE

```

```

) AS
    v_Znizka FLOAT;
    v_Cena_dostawy FLOAT;
BEGIN
    SELECT CASE WHEN d.Dostawca_ID = most_common.Dostawca_ID THEN 0.7 ELSE 1
END
    INTO v_Znizka
    FROM (
        SELECT Dostawca_ID, COUNT(*) AS count_dostawca
        FROM Dostawy
        GROUP BY Dostawca_ID
        ORDER BY count_dostawca DESC
    ) most_common
    INNER JOIN Dostawcy d ON most_common.Dostawca_ID = d.Dostawca_ID
    WHERE ROWNUM = 1;

    v_Cena_dostawy := p_Kwota_bez_zniki * v_Znizka;

    INSERT INTO Dostawy (Dostawa_ID, Ilosc, Czesc_ID, PESEL, Dostawca_ID,
    Kwota_bez_znizki, Data_dostawy, Znizka, Cena_dostawy)
    VALUES (p_Dostawa_ID, p_Ilosc, p_Czesc_ID, p_PESEL, p_Dostawca_ID,
    p_Kwota_bez_zniki, p_Data_dostawy, v_Znizka, v_Cena_dostawy);

    COMMIT;
    sys.htp.p('Dodano nową dostawę do tabeli Dostawy.');
EXCEPTION
    WHEN OTHERS THEN
        sys.htp.p('Wystąpił błąd: ' || SQLERRM);
        ROLLBACK;
END;
/

```

```

create or replace PROCEDURE DodajSzczegolyZamowienia (
    p_Szczegoly_zamowienia SzczegolyZamowienia_Object
) AS
    v_Przewidywany_czas_realizacji NUMBER;
    BAD_LOC EXCEPTION;
    BAD_KWOTA EXCEPTION;
BEGIN
    SELECT COUNT(*)

```

```

    INTO v_Przewidywany_czas_realizacji
    FROM Magazyn_produkto
    WHERE Lokacja_ID = p_Szczegoly_zamowienia.Lokacja_ID;

    IF v_Przewidywany_czas_realizacji = 0 THEN
        RAISE BAD_LOC;
    END IF;

    IF p_Szczegoly_zamowienia.Kwota <= 0 THEN
        RAISE BAD_KWOTA;
    END IF;

    SELECT COUNT(*)
    INTO v_Przewidywany_czas_realizacji
    FROM Harmonogram_urlopow hu
    JOIN Pracownicy p ON hu.PESEL = p.PESEL
    JOIN Zatrudnienie z ON p.id_zatrudnienia = z.id_zatrudnienia
    WHERE z.Nazwa_stanowiska = 'Pracownik magazynu'
        AND SYSDATE BETWEEN hu.Od_kiedy AND hu.Do_kiedy;

    IF v_Przewidywany_czas_realizacji = 0 THEN
        INSERT INTO Szczegoly_zamowienia (Szczegoly_zamowienia_ID, Znizka,
Lokacja_ID, Zamowienie_ID, Kwota, Rodzaj_platnosci,
Przewidywany_czas_realizacji)
        VALUES (p_Szczegoly_zamowienia.Szczegoly_zamowienia_ID,
p_Szczegoly_zamowienia.Znizka, p_Szczegoly_zamowienia.Lokacja_ID,
p_Szczegoly_zamowienia.Zamowienie_ID, p_Szczegoly_zamowienia.Kwota,
p_Szczegoly_zamowienia.Rodzaj_platnosci, SYSDATE);
    ELSE
        INSERT INTO Szczegoly_zamowienia (Szczegoly_zamowienia_ID, Znizka,
Lokacja_ID, Zamowienie_ID, Kwota, Rodzaj_platnosci,
Przewidywany_czas_realizacji)
        VALUES (p_Szczegoly_zamowienia.Szczegoly_zamowienia_ID,
p_Szczegoly_zamowienia.Znizka, p_Szczegoly_zamowienia.Lokacja_ID,
p_Szczegoly_zamowienia.Zamowienie_ID, p_Szczegoly_zamowienia.Kwota,
p_Szczegoly_zamowienia.Rodzaj_platnosci, SYSDATE + 3);
    END IF;

    COMMIT;
    sys.htp.p('Dodano nowy rekord do tabeli Szczegoly_zamowienia.');

```

EXCEPTION

```

        WHEN NO_DATA_FOUND THEN
            sys.htp.p('Brak danych w tabeli Harmonogram_urlopow.');
        WHEN VALUE_ERROR THEN
            sys.htp.p('Błąd: Nieprawidłowe typy danych.');
        WHEN BAD_LOC THEN
            sys.htp.p('Podane Lokacja_ID nie istnieje w tabeli
Magazyn_produkto.');
        WHEN BAD_KWOTA THEN
            sys.htp.p('Błędna kwota. Kwota musi być większa od zera.');

```

```

        WHEN OTHERS THEN
            sys.htp.p('Wystąpił błąd: ' || SQLERRM);
            ROLLBACK;
    END;
/

create or replace PROCEDURE InsertClientDetails(
    p_client_data IN client_info
) AS
    v_error BOOLEAN := FALSE;
BEGIN
    IF p_client_data.Nazwa_klienta IS NULL OR p_client_data.Nazwa_klienta =
    '' THEN
        DBMS_OUTPUT.PUT_LINE('Nazwa_klienta nie może być pusta');
        RAISE_APPLICATION_ERROR(-20202, 'Nazwa_klienta nie może być pusta');
        v_error := TRUE;
    ELSIF p_client_data.Email IS NULL OR INSTR(p_client_data.Email, '@') = 0
THEN
        DBMS_OUTPUT.PUT_LINE('Zły format email');
        RAISE_APPLICATION_ERROR(-20202, 'Zły format email');
        v_error := TRUE;
    ELSIF p_client_data.Telefon IS NULL OR LENGTH(p_client_data.Telefon) NOT
BETWEEN 9 AND 11 THEN
        DBMS_OUTPUT.PUT_LINE('Telefon powinien mieć pomiędzy 9 i 11
znaków');
        RAISE_APPLICATION_ERROR(-20202, 'Telefon powinien mieć pomiędzy 9 i
11 znaków');
        v_error := TRUE;
    ELSE
        INSERT INTO Klienci (Klient_ID, Nazwa_klienta, Miejscowosc, Email,
        Telefon, Typ_klienta, id_kod_adresu)
        VALUES (p_client_data.Klient_ID, p_client_data.Nazwa_klienta,
        p_client_data.Miejscowosc, p_client_data.Email, p_client_data.Telefon,
        p_client_data.Typ_klienta, p_client_data.id_kod_adresu);

        COMMIT;
        apex_application.g_print_success_message:='Sukces. ';
    END IF;

    IF v_error THEN
        RAISE_APPLICATION_ERROR(-20201, 'Validation failed');
    END IF;
EXCEPTION
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20202, 'An error occurred: ' || SQLERRM);
        ROLLBACK;
END;

```

```

/
create or replace PROCEDURE InsertEmployeeDetails(
    emp_data IN employee_info
) AS
    v_error BOOLEAN := FALSE;
BEGIN
    IF emp_data.Imie IS NULL OR emp_data.Nazwisko IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('Imię lub nazwisko jest puste dla pracownika o PESELU ' || emp_data.PESEL);
        RAISE_APPLICATION_ERROR(-20202,'Imię lub nazwisko jest puste dla pracownika o PESELU ' || emp_data.PESEL);
        v_error := TRUE;
    ELSIF INSTR(emp_data.Email, '@') = 0 THEN
        DBMS_OUTPUT.PUT_LINE('Email dla pracownika ' || emp_data.PESEL || ' nie zawiera "@."');
        RAISE_APPLICATION_ERROR(-20202,'Email dla pracownika ' || emp_data.PESEL || ' nie zawiera "@."');
        v_error := TRUE;
    ELSIF emp_data.Stanowisko IS NULL THEN
        DBMS_OUTPUT.PUT_LINE('Stanowisko jest puste dla pracownika o PESELU ' || emp_data.PESEL);
        RAISE_APPLICATION_ERROR(-20202,'Stanowisko jest puste dla pracownika o PESELU ' || emp_data.PESEL);
        v_error := TRUE;
    ELSE
        INSERT INTO Zatrudnienie(
            id_zatrudnienia, Nazwa_stanowiska, Pensja, Data_zatrudnienia,
            Umowa_nazwa, Dzial_nazwa, Opis_zmiany
        )
        VALUES (
            emp_data.id_zatrudnienia, emp_data.Stanowisko, emp_data.Pensja,
            emp_data.Data_zatrudnienia,
            emp_data.Umowa_nazwa, emp_data.Dzial_nazwa, emp_data.Opis_zmiany
        );
    END IF;
    IF v_error = FALSE THEN
        INSERT INTO Pracownicy(
            PESEL, Imie, Nazwisko, id_kod_adresu, Linia_ID, id_zatrudnienia,
            Data_urodzenia, Miejscowosc,
            Email, Telefon, Wyksztalcenie
        )
        VALUES (
            emp_data.PESEL, emp_data.Imie, emp_data.Nazwisko,
            emp_data.id_kod_adresu, emp_data.Linia_ID,
            emp_data.Data_urodzenia, emp_data.Miejscowosc,
            emp_data.Email, emp_data.Telefon, emp_data.Wyksztalcenie
        );
    END IF;
END;
/

```

```

        emp_data.id_zatrudnienia, emp_data.Data_urodzenia,
emp_data.Miejscowosc, emp_data.Email,
        emp_data.Telefon, emp_data.Wyksztalcenie
    );
    COMMIT;
    apex_application.g_print_success_message:='Dodano pracownika';
END IF;

IF v_error THEN
    ROLLBACK;
    RAISE_APPLICATION_ERROR(-20201, 'Błąd walidacji');
END IF;

EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        DBMS_OUTPUT.PUT_LINE('Wystąpił błąd: ' || SQLERRM);
        RAISE_APPLICATION_ERROR(-20202,'Wystąpił błąd: ' || SQLERRM);
END;
/

```

```

create or replace PROCEDURE OdejmijIloscZMagazynu(
    p_produkt_id INT,
    p_ilosc INT
) IS
    v_ilosc_w_magazynie INT;
    v_nowa_ilosc INT;
    v_produkty Produkty_Collection := Produkty_Collection();
    BAD_PROD EXCEPTION;
    BAD_NUMB EXCEPTION;
BEGIN
    SELECT Ilosc_w_magazynie INTO v_ilosc_w_magazynie
    FROM Magazyn_produktof
    WHERE Produkt_ID = p_produkt_id;

    IF v_ilosc_w_magazynie = 0 THEN
        RAISE BAD_PROD;
    END IF;

    if p_ilosc < 0 THEN
        RAISE BAD_NUMB;
    END IF;

```

```

v_nowa_ilosc := v_ilosc_w_magazynie - p_ilosc;

IF v_nowa_ilosc < 0 THEN
    v_produkty.EXTEND();
    v_produkty(v_produkty.LAST) := Produkt_Object(p_produkt_id, p_ilosc
- v_ilosc_w_magazynie);

FOR i IN 1..v_produkty.COUNT LOOP
    INSERT INTO Brakujace_produkty (Produkt_ID, Brakujaca_ilosc)
    VALUES (v_produkty(i).Produkt_ID,
v_produkty(i).Brakujaca_ilosc);
    END LOOP;

sys.hpt.p('Wystąpił niedobór w magazynie dla produktu o ID: ' ||
p_produkt_id || '. Brakująca ilość: ' || (p_ilosc - v_ilosc_w_magazynie));
ELSE
    UPDATE Magazyn_produkto
    SET Ilosc_w_magazynie = v_nowa_ilosc
    WHERE Produkt_ID = p_produkt_id;

    sys.hpt.p('Pomyślnie odjęto ' || p_ilosc || ' sztuk z magazynu dla
produktu o ID: ' || p_produkt_id);
    END IF;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        sys.hpt.p('Nie znaleziono informacji o produkcie o ID: ' ||
p_produkt_id || ' w magazynie.');
    WHEN BAD_PROD THEN
        sys.hpt.p('Podane ID produktu nie istnieje w bazie danych
Produkty.');
    WHEN BAD_NUMB THEN
        sys.hpt.p('Ilosc musi byc wieksza od 0');
    WHEN OTHERS THEN
        sys.hpt.p('Wystąpił inny błąd.');
END;
/

```

```

create or replace PROCEDURE SprawdzIloscWMagazynie(
    p_Produkt_ID IN INT,
    p_ilosc IN INT
) IS
    ilosc_w_magazynie INT;
    roznica INT;

```

```

brakujace_produkty ListaBrakujacychProduktow :=
ListaBrakujacychProduktow();
brakujacy_produkt BrakujacyProdukt;
BAD_NUMB EXCEPTION;
BEGIN
  SELECT Ilosc_w_magazynie INTO ilosc_w_magazynie
  FROM Magazyn_produkty
  WHERE Produkt_ID = p_Produkt_ID;

  IF p_ilosc < 0 THEN
    RAISE BAD_NUMB;
  END IF;

  IF ilosc_w_magazynie >= p_ilosc THEN
    sys.hip.p('Wystarczająca ilość w magazynie');
  ELSE
    roznica := p_ilosc - ilosc_w_magazynie;
    sys.hip.p('Brakuje ' || roznica || ' produktu');

    brakujacy_produkt := BrakujacyProdukt(p_Produkt_ID, roznica);
    brakujace_produkty.EXTEND();
    brakujace_produkty(brakujace_produkty.COUNT) := brakujacy_produkt;
  END IF;
EXCEPTION
  WHEN NO_DATA_FOUND THEN
    sys.hip.p('Nie znaleziono produktu o podanym ID.');
  WHEN BAD_NUMB THEN
    sys.hip.p('Ilość nie może być mniejsza od zera');
  WHEN OTHERS THEN
    sys.hip.p('Wystąpił inny błąd.');
END;
/

```

```

--FUNKCJE
create or replace FUNCTION CheckCustomerData
RETURN SYS_REFCURSOR
AS
  v_cursor SYS_REFCURSOR;

```

```

BEGIN
    OPEN v_cursor FOR
        SELECT Klient_ID, Nazwa_klienta, Miejscowosc, Email, Telefon,
    Typ_klienta, id_kod_adresu
        FROM Klienci;

        RETURN v_cursor;
END;
/

```

```

create or replace FUNCTION CheckPartStorageFullness
RETURN VARCHAR2
AS
    v_result VARCHAR2(32767);
BEGIN
    FOR storage_info IN (
        SELECT LOKACJA_ID, CZESC_ID, ILOSC_W_MAGAZYNIE, MAX_POJEMNOSC,
               ILOSC_W_MAGAZYNIE / MAX_POJEMNOSC * 100 AS PartStorageFullnes
        FROM Magazyn_czesci
    )
    LOOP
        v_result := v_result ||
            'Lokacja: ' || storage_info.LOKACJA_ID || ', część: ' ||
storage_info.CZESC_ID ||
            ', zapełnienie: ' || ROUND(storage_info.PartStorageFullnes, 2)
|| '%' || '<br>';
        v_result := v_result || '</br>';
    END LOOP;

    RETURN v_result;
END;
/

```

```

create or replace FUNCTION CheckProductStorageFullness
RETURN VARCHAR2
AS

```

```

    v_result VARCHAR2(32767);
BEGIN
    FOR storage_info IN (
        SELECT LOKACJA_ID, LOKACJA_NAZWA, ILOSC_W_MAGAZYNIE,
MAKSYMALNA_PODAJNOSC, PRODUKT_ID,
            ILOSC_W_MAGAZYNIE / MAKSYMALNA_PODAJNOSC * 100 AS
ProductStorageFullnes
        FROM Magazyn_produkow
    )
    LOOP
        v_result := v_result ||
            'Lokacja: ' || storage_info.LOKACJA_ID || ', produkt: ' ||
storage_info.PRODUKT_ID ||
            ', zapełnienie: ' || ROUND(storage_info.ProductStorageFullnes,
2) || '%' || '<br>';
        v_result := v_result || '<br>';
    END LOOP;

    RETURN v_result;
END;
/

```

```

create or replace FUNCTION IloscPotrzebnychCzesci(
    produkt_id_p INT,
    ilosc_sztuk_p INT
) RETURN SYS_REFCURSOR IS
    ilosc_czesci_cur SYS_REFCURSOR;
    BAD_PROD EXCEPTION;
    v_produkt_count INT;
BEGIN

    OPEN ilosc_czesci_cur FOR
        SELECT pc.Czesc_ID, pc.wymagana_ilosc_czesci * ilosc_sztuk_p AS
ilosc_potrzebna
        FROM Produkt_czesci pc
        WHERE pc.Produkt_ID = produkt_id_p;

    RETURN ilosc_czesci_cur;
END;
/

```

```

create or replace FUNCTION ObliczBrakujaceCzesci(
    czesc_id_p INT,

```

```

    ilosc_potrzebna_p INT
) RETURN SYS_REFCURSOR IS
    ilosc_w_magazynie INT;
    brakujaca_ilosc INT;
    cur SYS_REFCURSOR;
BEGIN
    OPEN cur FOR
        SELECT CASE
            WHEN Ilosc_w_magazynie >= ilosc_potrzebna_p THEN 0
            ELSE ilosc_potrzebna_p - Ilosc_w_magazynie
        END AS brakujaca_ilosc
    FROM Magazyn_czesci
    WHERE Czesc_ID = czesc_id_p;

    RETURN cur;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        RAISE_APPLICATION_ERROR(-20001, 'Nie znaleziono informacji o danej
części w magazynie.');
    WHEN OTHERS THEN
        RAISE_APPLICATION_ERROR(-20002, 'Wystąpił inny błąd.');
END;
/

```

```

create or replace FUNCTION WyswietlIloscProdukcji RETURN VARCHAR2 AS
    TYPE ProduktInfo IS RECORD (
        Produkt_ID Produkty.Produkt_ID%TYPE,
        Nazwa_produktu Produkty.Nazwa_produktu%TYPE
    );

    TYPE CzescInfo IS RECORD (
        Czesc_ID Produkt_czesci.Czesc_ID%TYPE,
        Nazwa_czesci Czesci.Nazwa_czesci%TYPE,
        Minimalne_dzielenie NUMBER
    );

    TYPE ProduktCzesciInfo IS TABLE OF CzescInfo INDEX BY BINARY_INTEGER;
    TYPE WszystkieProduktyInfo IS TABLE OF ProduktInfo INDEX BY
BINARY_INTEGER;

    v_output VARCHAR2(32000);
    v_produkty WszystkieProduktyInfo;
BEGIN
    SELECT DISTINCT p.Produkt_ID, p.Nazwa_produktu

```

```

BULK COLLECT INTO v_produkty
FROM Produkty p;

FOR i IN 1 .. v_produkty.COUNT LOOP
    v_output := v_output || 'Produkt: ' || v_produkty(i).Nazwa_produktu ||
', Produkt ID: ' || v_produkty(i).Produkt_ID || '</br>';

DECLARE
    v_czesci ProduktCzesciInfo;
BEGIN
    SELECT pc.Czesc_ID, c.Nazwa_czesci, FLOOR(MIN(m.Ilosc_w_magazynie /
pc.wymagana_ilosc_czesci)) AS minimalne_dzielenie
    BULK COLLECT INTO v_czesci
    FROM Produkt_czesci pc
    JOIN Magazyn_czesci m ON pc.Czesc_ID = m.Czesc_ID
    JOIN Czesci c ON m.Czesc_ID = c.Czesc_ID
    WHERE pc.Produkt_ID = v_produkty(i).Produkt_ID
    GROUP BY pc.Czesc_ID, c.Nazwa_czesci;

    FOR j IN 1 .. v_czesci.COUNT LOOP
        v_output := v_output || 'Część: ' || v_czesci(j).Nazwa_czesci || ',',
        Część ID: ' || v_czesci(j).Czesc_ID || ', Liczba produktów do wyprodukowania
        z części: ' || v_czesci(j).minimalne_dzielenie || '</br>';
    END LOOP;
    v_output := v_output || '</br>';
END;
END LOOP;

RETURN v_output;
END;
/

```

```

create or replace FUNCTION WyswietlKlientow RETURN KlienciList IS
    v_klienci_list KlienciList := KlienciList();
    v_klient KlientType;
    CURSOR c_klienci IS
        SELECT Klient_ID, Nazwa_klienta, Miejscowosc, Email, Telefon,
    Typ_klienta, id_kod_adresu
        FROM Klienci;
BEGIN
    FOR k IN c_klienci LOOP
        v_klient := KlientType(
            Klient_ID => k.Klient_ID,
            Nazwa_klienta => k.Nazwa_klienta,
            Miejscowosc => k.Miejscowosc,

```

```
        Email => k.Email,
        Telefon => k.Telefon,
        Typ_klienta => k.Typ_klienta,
        id_kod_adresu => k.id_kod_adresu
    );
    v_klienci_list.EXTEND;
    v_klienci_list(v_klienci_list.LAST) := v_klient;
END LOOP;

RETURN v_klienci_list;
END;
/
```