Dokumentacja aplikacji Service Order Wizard

Miłosz Rolewski

24 września 2025

Spis treści

Rozdział 1

Wprowadzenie

Aplikacja Service Order Wizard jest systemem służącym do zarządzania zamówieniami. Dokumentacja obejmuje instrukcje uruchomienia w warunkach developerskich, użycia, architekturę aplikacji oraz opis głównych funkcjonalności i modułów.

1.1 Cel dokumentacji

Celem dokumentacji jest umożliwienie programistom szybkiego zrozumienia działania aplikacji w celu dalszego rozwoju systemu. Dokument ten pozwala również na zrozumienie sposobu jej uruchomienia oraz zasad korzystania z jej funkcji.

1.2 Zakres dokumentacji

Dokumentacja obejmuje:

- Instrukcję instalacji aplikacji,
- Opis interfejsu użytkownika,
- Opis architektury i modułów backendu,
- Przykłady użycia funkcji i metod,

Rozdział 2

Instalacja

2.1 Wymagania wstępne

- System operacyjny: Windows/Linux/macOS
- Node.js / SAP Logon / SAPUI5 / ABAP (zależnie od projektu)
- Skonfigurowana baza danych dostępna przez serwis SAP

2.2 Instrukcja instalacji

1. Pobierz repozytorium z GitHub:

```
git clone https://github.com/uzytkownik/projekt.git
```

2. Skonfiguruj połączenie z serwisem SAP w pliku `manifest.json`.
3. Uruchom aplikację:

```
npm start
```

Rozdział 3

Użycie aplikacji

3.1 Interfejs użytkownika

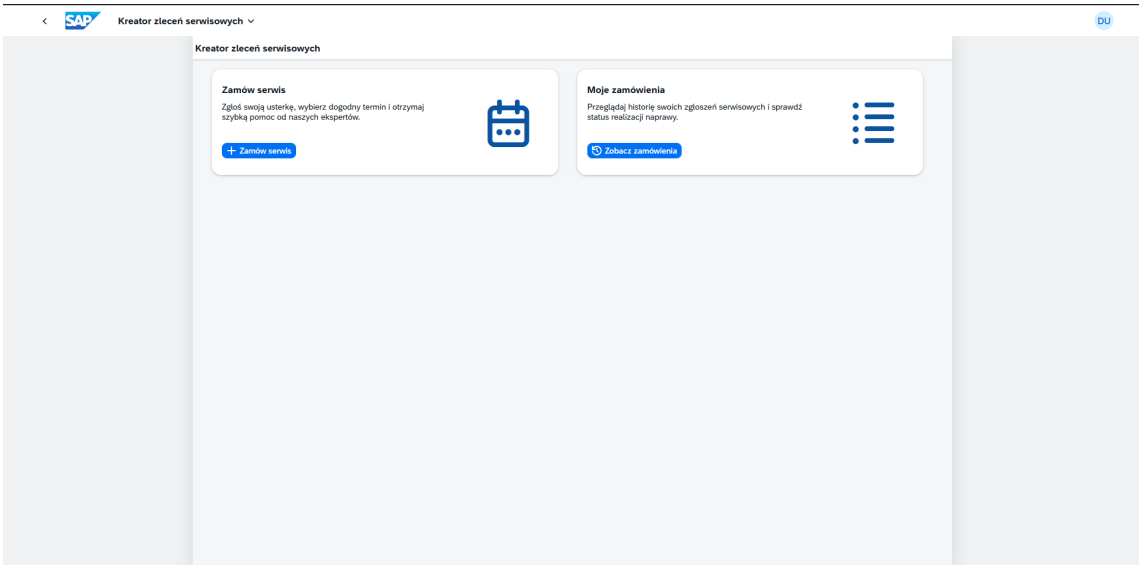
Interfejs użytkownika oferuje 3 główne widoki:

- Widok ekranu głównego,
- Widok tabeli zamówień serwisowych,
- Widok formularz do składania zamówień serwisowych.

Ich zadaniem jest w sposób intuicyjny poprowadzić użytkownika przez cały proces składania zamówienia serwisowego, a także pozwolić na zobaczenie obecnego stanu zamówień w serwisie.

3.1.1 Widok ekranu głównego

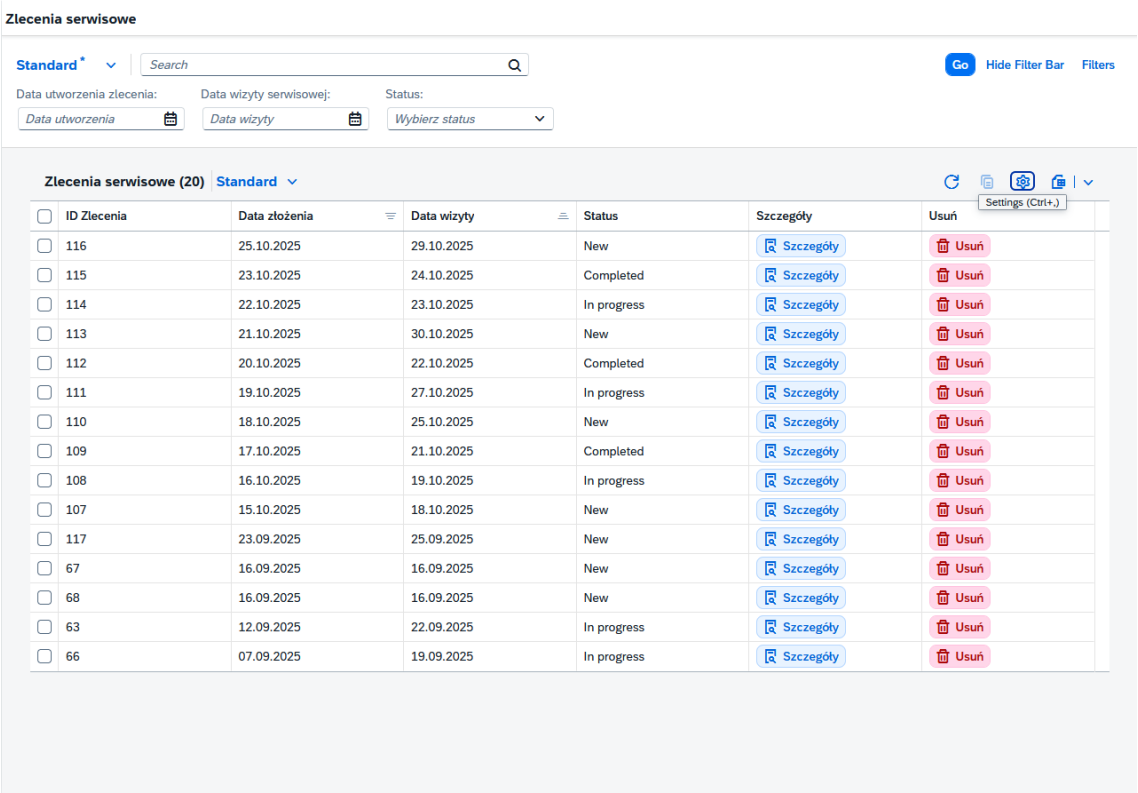
Widok ekranu głównego 3.1 - zawiera karty nawigujące do pozostałych dwóch widoków. Jego główną funkcjonalnością jest nawigacja po aplikacji oraz krótkie wprowadzenie jakie możliwości dają poszczególne widoki.



Rysunek 3.1: Caption

3.1.2 Widok tabeli zamówień serwisowych

Widok tabeli zamówień serwisowych 3.2 - umożliwiający przeglądanie, filtrowanie oraz przeszukiwanie zamówień serwisowych znajdujących się w systemie. Zawiera zarówno nowe zamówienia (Status: New), obecnie realizowane (In progress) i archiwalne, już zrealizowane zamówienia (Completed). Szczegóły każdego z zamówień można zobaczyć poprzez kliknięcie przycisku "Szczegóły"znajdującego się w piątej kolumnie.



ID Zlecenia	Data złożenia	Data wizyty	Status	Szczegóły	Usuń
116	25.10.2025	29.10.2025	New	Szczegóły	Usuń
115	23.10.2025	24.10.2025	Completed	Szczegóły	Usuń
114	22.10.2025	23.10.2025	In progress	Szczegóły	Usuń
113	21.10.2025	30.10.2025	New	Szczegóły	Usuń
112	20.10.2025	22.10.2025	Completed	Szczegóły	Usuń
111	19.10.2025	27.10.2025	In progress	Szczegóły	Usuń
110	18.10.2025	25.10.2025	New	Szczegóły	Usuń
109	17.10.2025	21.10.2025	Completed	Szczegóły	Usuń
108	16.10.2025	19.10.2025	In progress	Szczegóły	Usuń
107	15.10.2025	18.10.2025	New	Szczegóły	Usuń
117	23.09.2025	25.09.2025	New	Szczegóły	Usuń
67	16.09.2025	16.09.2025	New	Szczegóły	Usuń
68	16.09.2025	16.09.2025	New	Szczegóły	Usuń
63	12.09.2025	22.09.2025	In progress	Szczegóły	Usuń
66	07.09.2025	19.09.2025	In progress	Szczegóły	Usuń

Rysunek 3.2: Caption

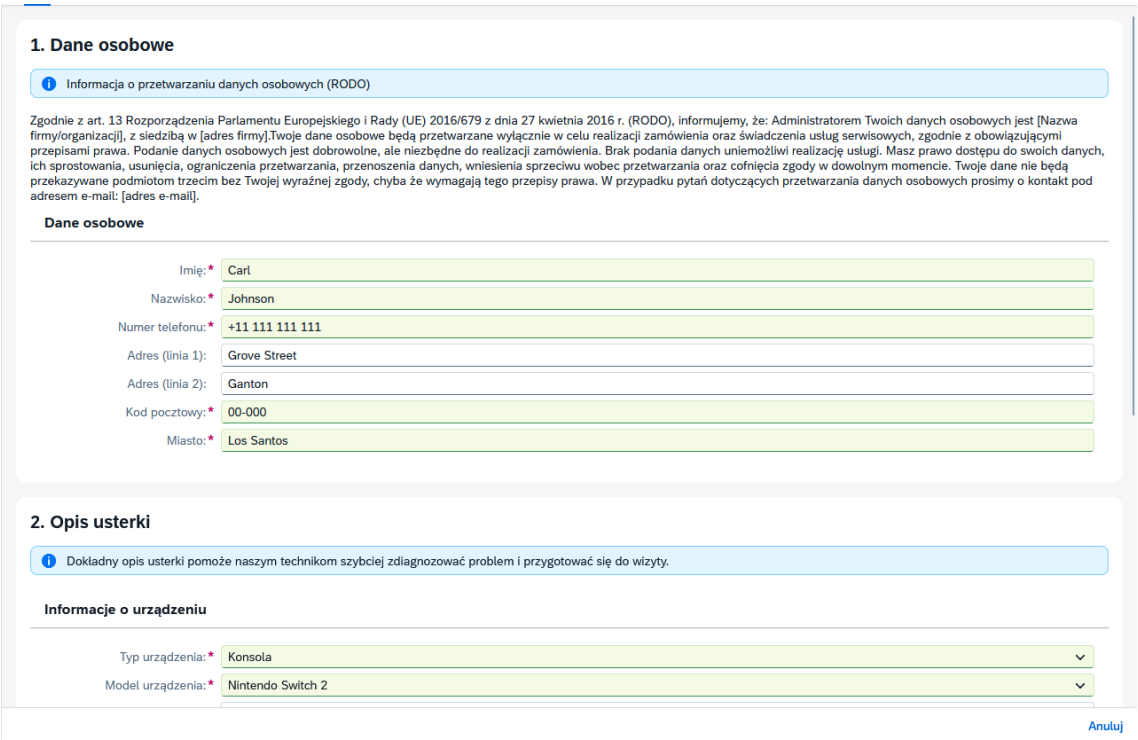
3.1.3 Widok formularza do składania zamówień serwisowych

Widok formularza do składania zamówień serwisowych 3.3 - formularz umożliwiający użytkownikowi złożenie zamówienia serwisowego w intuicyjny sposób. Prowadzi on przez sekwencje kroków pozwalających wypełnić dane potrzebne do zrealizowania zamówienia. Składa się z takich kroków jak:

1. Dane osobowe,
2. Opis usterki,
3. Termin wizyty,
4. Podsumowanie.

Dane w formularzu są sprawdzane przy próbie przejścia do następnego kroku, a w przypadku błędu lub braku danych użytkownik jest o tym powiadamiany przez

odpowiednie komunikaty. Po przejściu wszystkich kroków użytkownik może sprawdzić ponownie swoje dane na stronie recenzji 3.4. I zedytować je w razie błędnych danych. By przesłać formularz, należy wcisnąć przycisk "wyślij"znajdujący się w prawym dolnym rogu strony recenzji. Po pomyślnym przesłaniu zamówienia serwisowego użytkownik zostaje przekierowany do widoku tabeli zamówień serwisowych.



Rysunek 3.3: Widok formularza

3.2 Scenariusz użycia

W tej sekcji przedstawiono przykładowy scenariusz użycia aplikacji przez użytkownika końcowego.

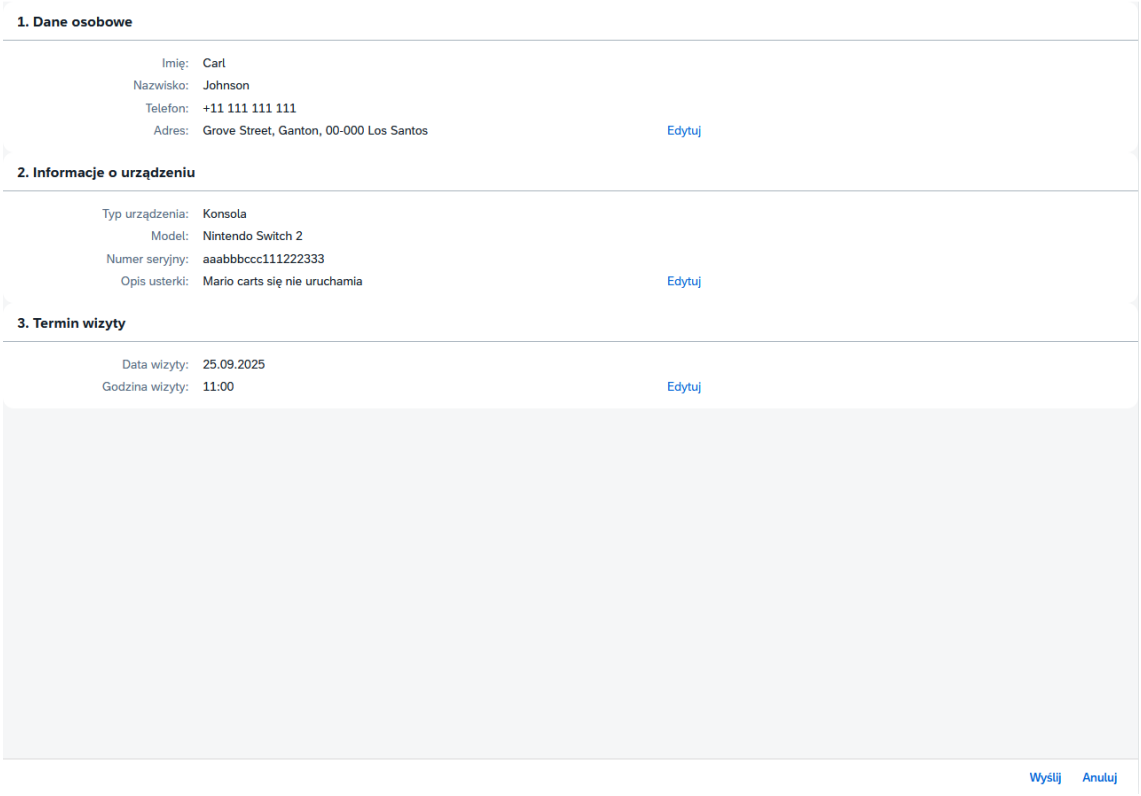
1. Użytkownik po uruchomieniu aplikacji znajduje sie na ekranie głównym - wybiera kartę "Zamów serwis"3.5.

2. Użytkownik wypełnia dane osobowe, jednak podaje nieprawidłowy numer telefonu - otrzymuje komunikat 3.6 o konieczności poprawienia danych przed przejściem do kolejnego kroku.

3. Użytkownik poprawia numer telefonu na poprawny i przechodzi do kolejnego kroku.

4. Użytkownik wybiera z listy predefiniowanych typów i modeli urządzeń swoją konsolę, podaje numer seryjny i opis usterki 3.7. Dane są wypełnione prawidłowo i przechodzi do kolejnego kroku.

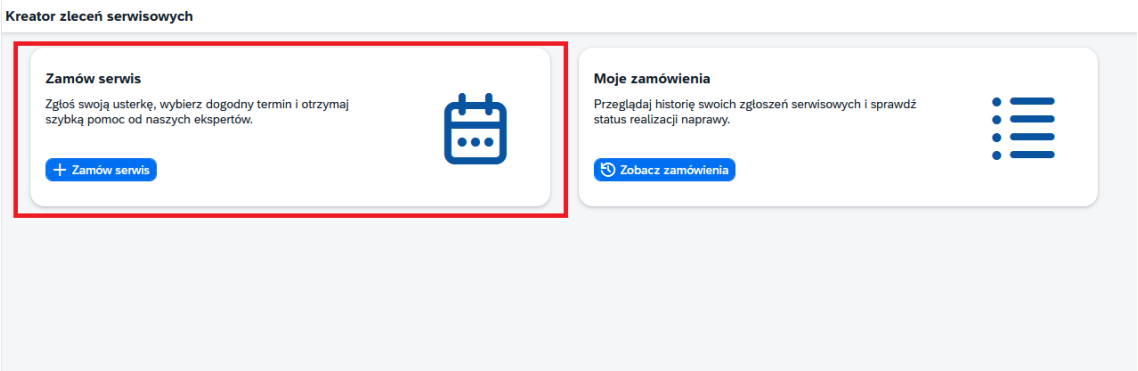
5. Użytkownik wybiera date i godzinę wizyty 3.8.



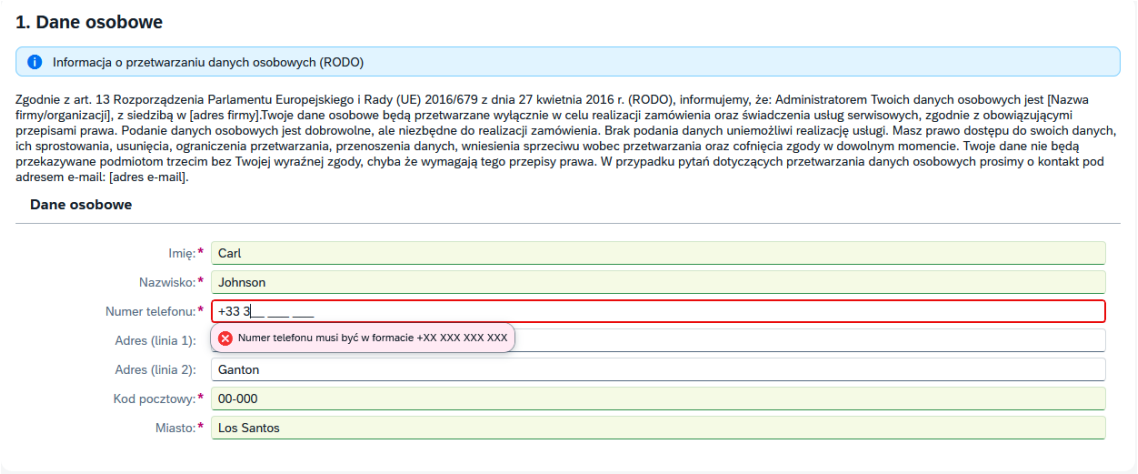
Rysunek 3.4: Strona recenzji danych

6. Użytkownik przechodzi do kroku podsumowania gdzie zostaje poinformowany o dalszych krokach działania aplikacji 3.9. Klika "Review".

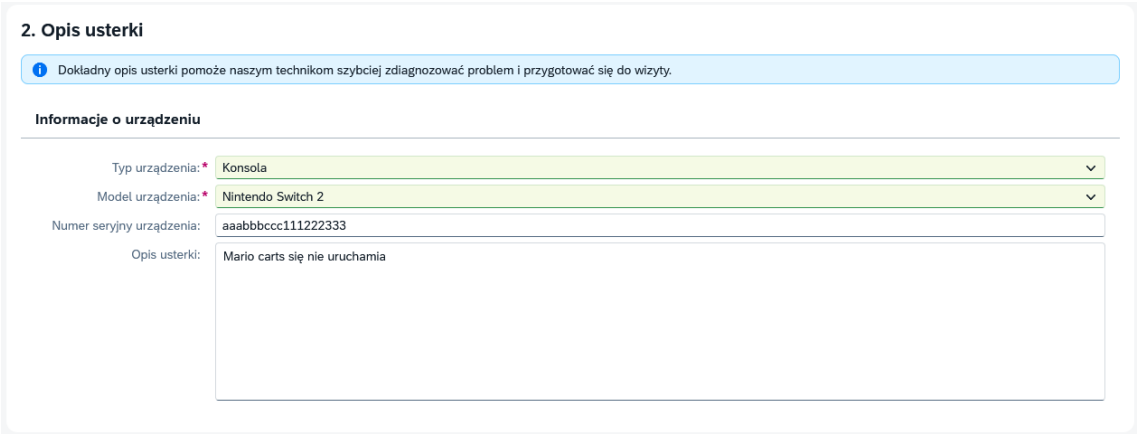
7. Użytkownik sprawdza poprawność swoich danych na ekranie 3.4, a następnie klika wyślij. Po potwierdzeniu akcji zostaje przeniesiony na ekran 3.2.



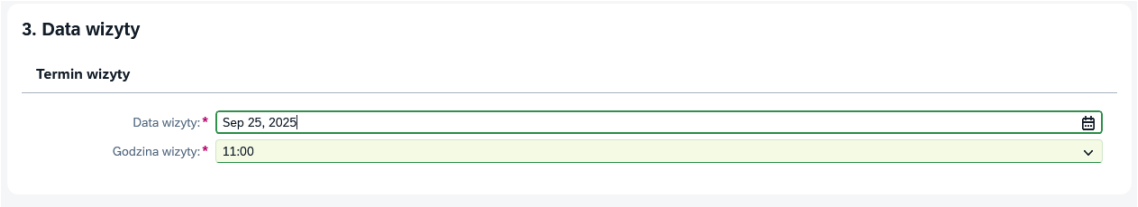
Rysunek 3.5: karta "zamów serwis"



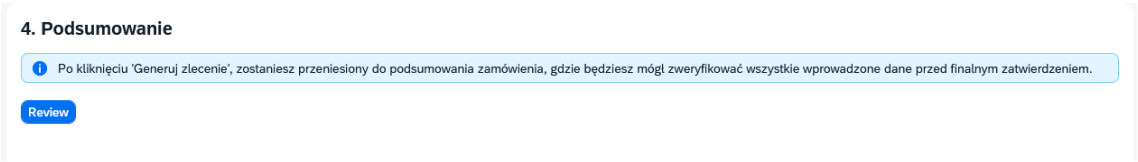
Rysunek 3.6: Krok danych osobowych



Rysunek 3.7: Opis usterki



Rysunek 3.8: Data wizyty



Rysunek 3.9: Podsumowanie

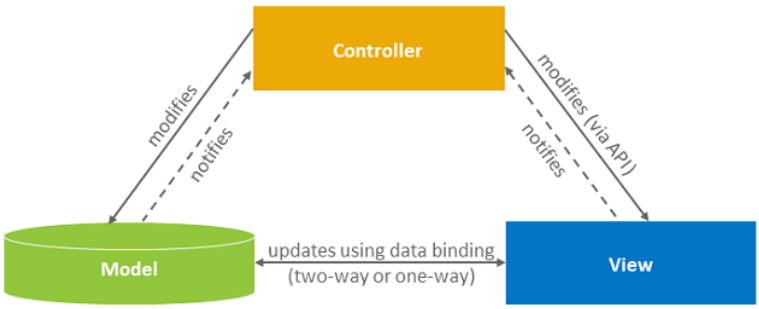
Rozdział 4

Architektura aplikacji

4.1 Frontend

4.1.1 Architektura Model View Controller (MVC)

Przy tworzeniu aplikacji zastosowano architekturę MVC 4.1 powszechnie stosowaną w SAPUI5 w celu oddzielenia warstwy reprezentacji danych od akcji użytkownika na ekranie. Data binging pozwala na efektywne połączenie poszczególnych elementów architektury MVC.



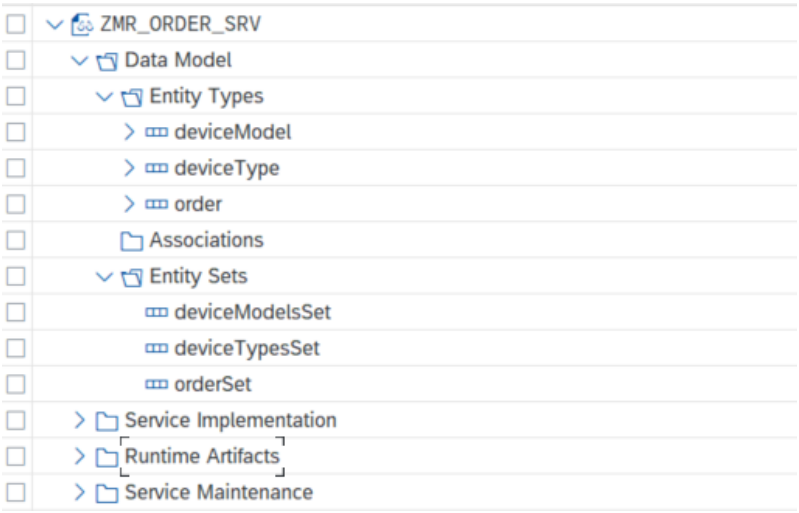
Rysunek 4.1: Diagram architektury MVC [źródło]

4.1.2 Struktura projektu

- `models/` – definicje modeli danych
- `views/` – interfejs użytkownika
- `controllers/` – logika kontrolerów

4.2 Backend

Serwis OData w wersji 2 został zaimplementowany w języku ABAP z wykorzystaniem standardowych narzędzi środowiska SAP. Umożliwia on wystawienie danych zapisanych w wcześniej skonfigurowanej tabeli bazodanowej w postaci usług sieciowych dostępnych przez protokół HTTP. Dzięki temu warstwa frontendowa aplikacji



Rysunek 4.2: Serwis

może w prosty sposób wykonywać operacje typu CRUD (Create, Read, Update, Delete) na danych, komunikując się z backendem w sposób ustandaryzowany i niezależny od technologii użytej po stronie klienta.

Rozwiązanie to znacząco usprawnia wymianę informacji pomiędzy warstwą frontendową a backendową, zapewniając:

- spójny model danych – dzięki mapowaniu struktur ABAP na format OData,
- łatwą integrację – frontend może konsumować dane w formacie JSON lub XML bez dodatkowych konwerterów,
- skalowalność – możliwość łatwego rozszerzania serwisu o kolejne encje lub operacje,
- standaryzację komunikacji – opartą na powszechnie stosowanym protokole OData v2.

4.3 Serwis

W transakcji SEGW utworzono serwis 4.2 oparty na tabelach bazodanowych opisanych w rozdziale 5. Serwis, którego użyto w tej aplikacji korzysta z trzech typów Entity Set:

- orderSer - odpowiada za zarządzania zamówieniami serwisowymi,
- deviceTypesSet - konieczny do obsługi predefiniowanych typów urządzeń,
- deviceModelsSet - konieczny do obsługi predefiniowanych modeli urządzeń dla poszczególnych typów.

Opartych na odpowiadających Entity Types. Każdy Entity Type zawiera wszystkie pola zdefiniowane w odpowiadającej mu tabeli bazodanowej.

Rozdział 5

Bazy danych

W tym rozdziale opisano tabele bazodanowe użyte przy tworzeniu rozwiązania, ich pola oraz relacje między nimi. Należy pamiętać, że każda z tabel jest wyznacznikiem dla serwisu udostępniającego odpowiedni EntitySet.

5.1 Tabele

5.1.1 Tabela zamówień (orders)

Tabela zamówień (orders) 5.1 przechowuje infomracje o zamówieniach serwisowych złożonych w serwisie. Ma które są predefiniowane przez system takie jak:

- MANDT - identyfikator klienta,
- ORDER_ID - dynamicznie wyznaczany dla nowych zamówień dzięki SNRO,
- STATUS - dla nowych zamówień predefiniowany na "New",
- ORDERCREATIONDATE - systemowa data utworzenia zamówenia.

Pozostałe pola są zależne od danych podanych przez użytkownika w trakcie zamawiania serwisu:

- FIRSTNAME - imię
- LASTNAME - nazwisko
- PHONENUMBER - numer telefonu w formacie +00 000 000 000
- ADDRESSFIRSTLINE - pierwsza linia adresu
- ADDRESSSECONDLINE - druga linia adresu
- ADDRESSZIPCODE - kod pocztowy w formacie 00-000
- ADDRESSCITY - miasto
- DEVICETYPE - typ urządzenia
- DEVICEMODEL - model urządzenia

- DEVICESERIALNUMBER - numer seryjny urządzenia
- FAULTDESCRIPTION - opis usterki
- VISITDATE - data wizyty w formacie YYYYMMDD
- VISITTIME - godzina wizyty w formacie HHMM

5.1.2 Tabela typów urządzeń (device_type)

Tabela typów urządzeń 5.2 przechorowuje predefiniowane typy urządzeń wyświetlane podczas wyboru typu przy opisie usterki. Rozwiązanie to pozwala na dynamiczny rozwój aplikacji poprzez dodanie nowych typów. Pola tabeli odpowiadają:

- MANDT - identyfikator klienta
- ID - id typu
- NAME - nazwa typu

5.1.3 Tabela model urządzeń (device_model)

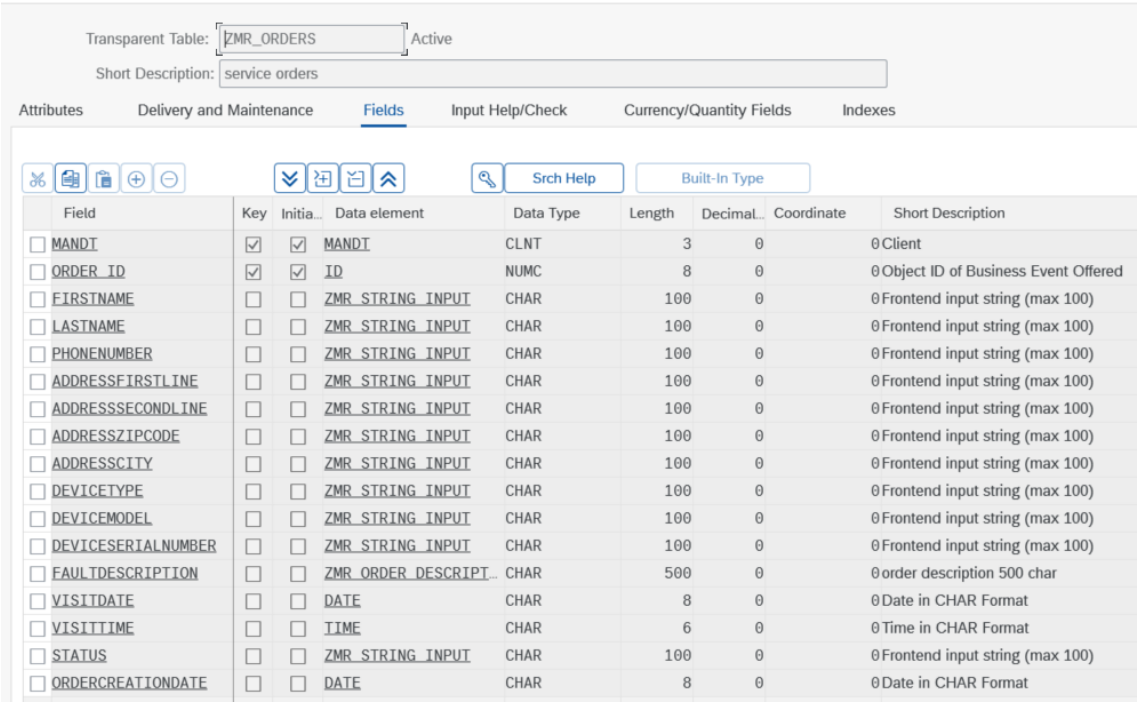
Tabela modeli urządzeń 5.3 zawiera modele dla poszczególnych typów urządzeń zdefiniowanych w tabeli typów urządzeń 5.1.2. Zawiera pola:

- MANDT - indentyfikator klienta
- ID - id modeli
- DEVICE_TYPE_ID - klucz obcy z tabeli typów urządzeń5.1.2
- MODEL_NAME - nazwa modelu

5.2 Relacje

5.2.1 Diagram relacji tabel

Diagram 5.4 przedstawia relacje w tabelach.



Transparent Table: ZMR_ORDERS Active

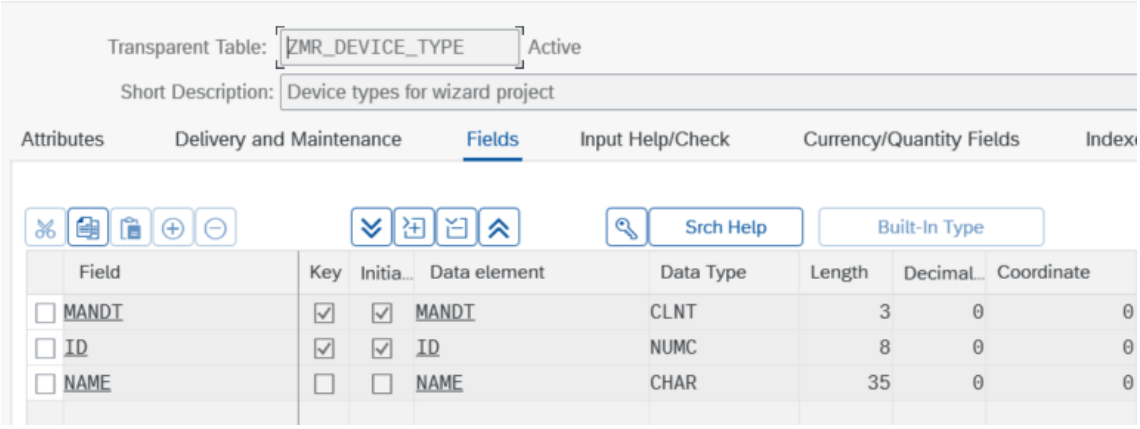
Short Description: service orders

Attributes | Delivery and Maintenance | Fields | Input Help/Check | Currency/Quantity Fields | Indexes

Srch Help | Built-In Type

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate	Short Description
MANDT	☑	☑	MANDT	CLNT	3	0	0	Client
ORDER_ID	☑	☑	ID	NUMC	8	0	0	Object ID of Business Event Offered
FIRSTNAME	☐	☐	ZMR_STRING_INPUT	CHAR	100	0	0	Frontend input string (max 100)
LASTNAME	☐	☐	ZMR_STRING_INPUT	CHAR	100	0	0	Frontend input string (max 100)
PHONENUMBER	☐	☐	ZMR_STRING_INPUT	CHAR	100	0	0	Frontend input string (max 100)
ADDRESSFIRSTLINE	☐	☐	ZMR_STRING_INPUT	CHAR	100	0	0	Frontend input string (max 100)
ADDRESSSECONDLINE	☐	☐	ZMR_STRING_INPUT	CHAR	100	0	0	Frontend input string (max 100)
ADDRESSZIPCODE	☐	☐	ZMR_STRING_INPUT	CHAR	100	0	0	Frontend input string (max 100)
ADDRESSCITY	☐	☐	ZMR_STRING_INPUT	CHAR	100	0	0	Frontend input string (max 100)
DEVICETYPE	☐	☐	ZMR_STRING_INPUT	CHAR	100	0	0	Frontend input string (max 100)
DEVICEMODEL	☐	☐	ZMR_STRING_INPUT	CHAR	100	0	0	Frontend input string (max 100)
DEVICESERIALNUMBER	☐	☐	ZMR_STRING_INPUT	CHAR	100	0	0	Frontend input string (max 100)
FAULTDESCRIPTION	☐	☐	ZMR_ORDER_DESCRIPT...	CHAR	500	0	0	order description 500 char
VISITDATE	☐	☐	DATE	CHAR	8	0	0	Date in CHAR Format
VISITTIME	☐	☐	TIME	CHAR	6	0	0	Time in CHAR Format
STATUS	☐	☐	ZMR_STRING_INPUT	CHAR	100	0	0	Frontend input string (max 100)
ORDERCREATIONDATE	☐	☐	DATE	CHAR	8	0	0	Date in CHAR Format

Rysunek 5.1: Tabela zamówień



Transparent Table: ZMR_DEVICE_TYPE Active

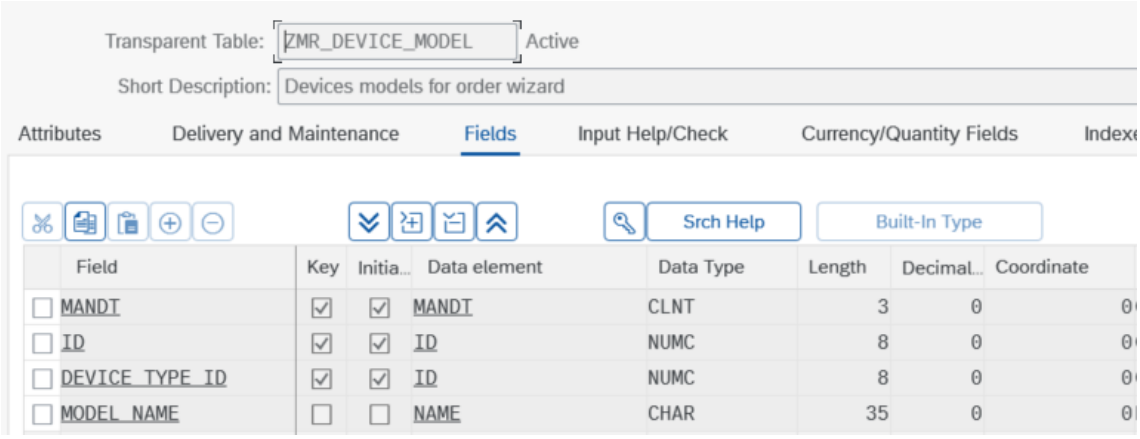
Short Description: Device types for wizard project

Attributes | Delivery and Maintenance | Fields | Input Help/Check | Currency/Quantity Fields | Indexe

Srch Help | Built-In Type

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate
MANDT	☑	☑	MANDT	CLNT	3	0	0
ID	☑	☑	ID	NUMC	8	0	0
NAME	☐	☐	NAME	CHAR	35	0	0

Rysunek 5.2: Tabela typów urządzeń



Transparent Table: ZMR_DEVICE_MODEL Active

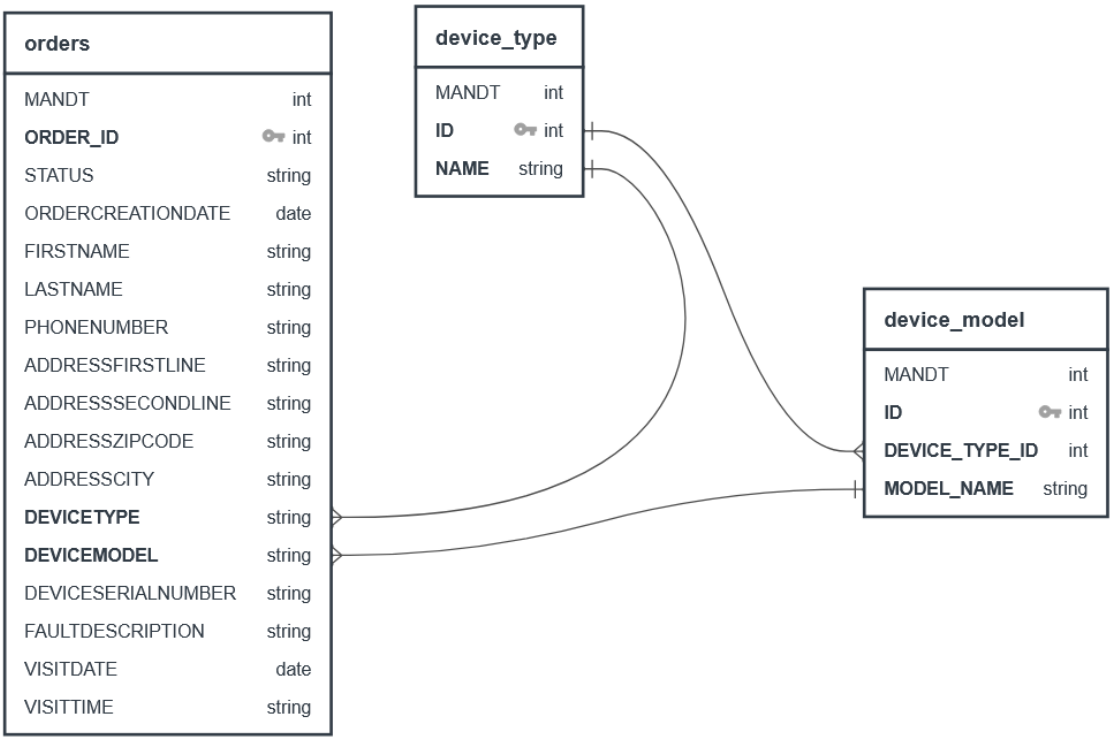
Short Description: Devices models for order wizard

Attributes | Delivery and Maintenance | Fields | Input Help/Check | Currency/Quantity Fields | Indexe

Srch Help | Built-In Type

Field	Key	Initia...	Data element	Data Type	Length	Decimal...	Coordinate
MANDT	☑	☑	MANDT	CLNT	3	0	0
ID	☑	☑	ID	NUMC	8	0	0
DEVICE_TYPE_ID	☑	☑	ID	NUMC	8	0	0
MODEL_NAME	☐	☐	NAME	CHAR	35	0	0

Rysunek 5.3: Tabela modeli urządzeń



Rysunek 5.4: Diagram Relacji

Rozdział 6

Opis klas i metod

6.1 Home.controller

Home.controller odpowiada za nawigację pomiędzy widokami. Wszystkie metody w Home.controller zostały opisane w kodzie według schematu JSDoc.

6.1.1 Przykład metody z Home.controller

Listing 6.1: Metoda odpowiada za nawigację do widoku tworzenia zamówień.

```
/**
 * Handles navigation to the create order view
 * @public
 */
onNavToCreateOrder: function() {
    var oRouter = sap.ui.core.UIComponent.getRouterFor(this);
    oRouter.navTo("RouteCreateOrder");
}
```

6.2 CreateOrder.controller

CreateOrder.controller odpowiada za akcje na ekranie tworzenia zamówień. Wszyskie metody zawarte w tym kontrolerze zostały skomentowane w kodzie według schematu JSDoc

6.2.1 Przykład metody z CreateOrder.controller

Listing 6.2: Metoda validateVisitDateDater odpowiada za weryfikacje wprowadzenia daty wizyty.

```
/**
 * Validates the visit date field in visit date section
 * @returns {boolean} Returns true if validation passes, false
     otherwise
 * @public
 */
validateVisitDateDate: function () {
    var oView = this.getView();
    var oModel = this.getView().getModel("orderData");
    var oVisitDateInput = oView.byId("visitDateInput");
    var bValid = true;
    var oVisitDate = oVisitDateInput.getDateValue();

    if (!oVisitDate) {
        oVisitDateInput.setValueState(sap.ui.core.ValueState.
           Error);
        oVisitDateInput.setValueStateText(this._getText("
           visitDateValidationError"));
        bValid = false;
    } else {
        oVisitDateInput.setValueState(sap.ui.core.ValueState.
           Success);
        // setProperty - datepicker is bugged
        oModel.setProperty("/visitData/visitDate", oVisitDate)
           ;
        console.log("Data wizyty ustawiona:", oVisitDate);
    }
    return bValid;
}
```

6.3 Orders.controller

Orders.controller odpowiada za akcje na ekranie podglądu listy zamówień. Wszyskie metody zawarte w tym kontrolerze zostały skomentowane w kodzie według schematu JSDoc

6.3.1 Przykład metody z Orders.controller

Listing 6.3: Metoda odpowiedzialna z wywolanie przeszukanie wyników po zmianie filtrów

```
/**
 * Handles filter value changes - triggers search in
    SmartFilterBar
 * @public
 */
onFilterChange: function () {
    var oSmartFilterBar = this.byId("smartFilterBar");
    if (oSmartFilterBar) {
        oSmartFilterBar.triggerSearch();
    }
}
```

6.4 Metody komunikacji z serwisem oData

6.4.1 orderset_get_entity

Metoda orderset_get_entity 6.4 odpowiada za pobranie pojedynczego rekordu zamówienia na podstawie klucza OrderId przekazanego w parametrze wejściowym.

- Jeśli rekord o podanym OrderId istnieje, metoda zwraca jego dane w strukturze encji (er_entity).
- Jeśli rekord nie zostanie znaleziony, generowany jest wyjątek biznesowy (/iwbep/cx_mgw_busi_exception) z komunikatem informującym o braku zamówienia.
- W przypadku błędu połączenia z bazą danych lub problemów z zapytaniem SQL metoda zgłasza wyjątek techniczny (/iwbep/cx_mgw_tech_exception).

Metoda umożliwia w ten sposób bezpieczne pobranie danych pojedynczego zamówienia w serwisie OData, zapewniając odpowiednie komunikaty dla błędów biznesowych i technicznych.

6.4.2 orderset_get_entityset

Metoda orderset_get_entityset 6.5 odpowiada za pobranie listy zamówień z tabeli zamówień z uwzględnieniem filtrów, sortowania oraz opcjonalnego wyszukiwania tekstowego.

- Filtrowanie odbywa się na polach: ‘STATUS‘, ‘ORDERCREATIONDATE‘ i ‘VISITDATE‘.
- Sortowanie realizowane jest dynamicznie na podstawie parametrów z requestu.
- Wyniki mogą być dodatkowo przeszukiwane po polach encji, takich jak ‘firstname‘, ‘lastname‘, ‘order_id‘, ‘phonenumber‘, ‘address*‘, ‘devicetype‘, ‘devicemodel‘, ‘deviceserialnumber‘, ‘faultdescription‘ i ‘visittime‘.

- w przypadku braku rekordów zgłaszany jest wyjątek biznesowy (‘/iwbep/cx_mgw_busi_exception‘), a w przypadku problemów z bazą danych wyjątek techniczny (‘/iwbep/cx_mgw_tech_exception‘).

Listing 6.5: orderset_get_entity

```
METHOD orderset_get_entityset.

  " ==== 1. Deklaracje tabel zakres w dla filtrowania ====
  DATA: lra_status                TYPE RANGE OF zmr_orders-
     status,
        wa_status                 LIKE LINE OF lra_status,
        lra_order_creation_date   TYPE RANGE OF zmr_orders-
           ordercreationdate,
        wa_order_creation_date    LIKE LINE OF
           lra_order_creation_date,
        lra_visitdate             TYPE RANGE OF zmr_orders-
           visitdate,
        wa_visitdate              LIKE LINE OF lra_visitdate,
        lv_order_by               TYPE string,
        lt_db_result              TYPE STANDARD TABLE OF
           zmr_orders,
        lt_filtered               TYPE STANDARD TABLE OF
           zmr_orders,
        lo_mc                     TYPE REF TO /iwbep/
           if_message_container.

  " ==== 2. Pobranie parametr w filtrowania i sortowania z
     requestu ====
  DATA(lr_filter) = io_tech_request_context->get_filter( ).
  DATA(lt_filter_select_options) = lr_filter->
     get_filter_select_options( ).
  DATA(lt_order_by) = io_tech_request_context->get_orderby( ).
  DATA(lv_search) = io_tech_request_context->get_search_string(
      ).

  " ==== 3. Wype nienie tabel zakres w dla ka dej kolumny
     ====
  LOOP AT lt_filter_select_options INTO DATA(
     ls_filter_select_options).

    " --- filtr Status ---
    IF ls_filter_select_options-property EQ 'STATUS'.
      LOOP AT ls_filter_select_options-select_options INTO
         DATA(ls_select_option).
        wa_status-sign   = ls_select_option-sign.
        wa_status-option = ls_select_option-option.
        wa_status-low    = ls_select_option-low.
        wa_status-high   = ls_select_option-high.
        APPEND wa_status TO lra_status.
```

```
32        ENDLOOP.
33      ENDIF.
34
35      " --- filtr OrderCreationDate ---
36      IF ls_filter_select_options-property EQ 'ORDERCREATIONDATE
           '.
37        LOOP AT ls_filter_select_options-select_options INTO
             DATA(ls_select_option_date).
38          wa_order_creation_date-sign   = ls_select_option_date-
              sign.
39          wa_order_creation_date-option = ls_select_option_date-
              option.
40          wa_order_creation_date-low    = ls_select_option_date-
              low.
41          wa_order_creation_date-high   = ls_select_option_date-
              high.
42          APPEND wa_order_creation_date TO
              lra_order_creation_date.
43        ENDLOOP.
44      ENDIF.
45
46      " --- filtr Visitdate ---
47      IF ls_filter_select_options-property EQ 'VISITDATE'.
48        LOOP AT ls_filter_select_options-select_options INTO
             DATA(ls_select_option_visit).
49          wa_visitdate-sign   = ls_select_option_visit-sign.
50          wa_visitdate-option = ls_select_option_visit-option.
51          wa_visitdate-low    = ls_select_option_visit-low.
52          wa_visitdate-high   = ls_select_option_visit-high.
53          APPEND wa_visitdate TO lra_visitdate.
54        ENDLOOP.
55      ENDIF.
56
57    ENDLOOP.
58
59    " ==== 4. Obsługa sortowania ====
60    LOOP AT lt_order_by INTO DATA(ls_order_by).
61      IF lv_order_by IS INITIAL.
62        lv_order_by = |{ ls_order_by-property }|.
63      ELSE.
64        lv_order_by = |{ lv_order_by }, { ls_order_by-property
             }|.
65      ENDIF.
66
67      IF ls_order_by-order = 'asc'.
68        lv_order_by = |{ lv_order_by } ASCENDING|.
69      ELSE.
70        lv_order_by = |{ lv_order_by } DESCENDING|.
71      ENDIF.
72    ENDLOOP.
73
```

```
74  " ==== 5. Zapytanie do bazy z filtrowaniem i sortowaniem
        ====
75  TRY.
76      SELECT *
77        FROM zmr_orders
78        INTO TABLE @lt_db_result
79        WHERE status IN @lra_status
80          AND ordercreationdate IN @lra_order_creation_date
81          AND visitdate IN @lra_visitdate
82        ORDER BY (lv_order_by).
83
84      IF sy-subrc <> 0.
85        lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~
              get_message_container( ).
86        lo_mc->add_message_text_only(
87          iv_msg_type = /iwbep/if_message_container=>
                gcs_message_type-error
88          iv_msg_text = |EntitySet not found or empty| ).
89        RAISE EXCEPTION NEW /iwbep/cx_mgw_busi_exception(
              message_container = lo_mc ).
90      ENDIF.
91
92    CATCH cx_sy_open_sql_db INTO DATA(lx_sql).
93      lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~
            get_message_container( ).
94      lo_mc->add_message_text_only(
95        iv_msg_type = /iwbep/if_message_container=>
              gcs_message_type-error
96        iv_msg_text = |Database connection error| ).
97      RAISE EXCEPTION NEW /iwbep/cx_mgw_tech_exception(
            message_container = lo_mc ).
98  ENDTRY.
99
100 " ==== 5. Filtrowanie wynik w w pami ci po search ====
101 IF lv_search IS NOT INITIAL.
102   LOOP AT lt_db_result ASSIGNING FIELD-SYMBOL(<fs_entity>).
103     IF <fs_entity>-lastname CP |*{ lv_search }*|
104       OR <fs_entity>-firstname CP |*{ lv_search }*|
105       OR <fs_entity>-order_id CP |*{ lv_search }*|
106       OR <fs_entity>-phonenumber CP |*{ lv_search }*|
107
108       OR <fs_entity>-addresscity CP |*{ lv_search }*|
109       OR <fs_entity>-addressfirstline CP |*{ lv_search }*|
110       OR <fs_entity>-addresssecondline CP |*{ lv_search }*|
111       OR <fs_entity>-addresszipcode CP |*{ lv_search }*|
112
113       OR <fs_entity>-devicetype CP |*{ lv_search }*|
114       OR <fs_entity>-devicemodel CP |*{ lv_search }*|
115       OR <fs_entity>-deviceserialnumber CP |*{ lv_search }*|
116       OR <fs_entity>-faultdescription CP |*{ lv_search }*|
117       OR <fs_entity>-visittime CP |*{ lv_search }*|.
```

```
118
119         APPEND <fs_entity> TO et_entityset.
120       ENDIF.
121     ENDLOOP.
122   ELSE.
123     et_entityset = lt_db_result.
124   ENDIF.
125
126
127 ENDMETHOD.
```

6.4.3 orderset_create_entity

orderset_create_entity 6.6 – metoda odpowiedzialna za utworzenie nowego zamówienia w tabeli zamówień, generuje unikalny Order ID, zapisuje dane w bazie i obsługuje błędy biznesowe oraz techniczne w przypadku problemów z bazą lub kolizji klucza.

Listing 6.6: Metoda orderset_create_entity

```
1  METHOD orderset_create_entity.
2
3    DATA: ls_order    TYPE zmr_orders,
4          lv_orderid TYPE zmr_orders-order_id,
5          lo_mc      TYPE REF TO /iwbep/if_message_container.
6
7    " Read entry data from request
8    io_data_provider->read_entry_data(
9      IMPORTING
10       es_data = ls_order
11   ).
12
13   CALL FUNCTION 'NUMBER_GET_NEXT'
14     EXPORTING
15       nr_range_nr       = '01'            " range number from
            SNRO
16       object            = 'ZMR_ORD_ID'    " SNRO object name
17     IMPORTING
18       number            = lv_orderid
19     EXCEPTIONS
20       interval_not_found       = 1
21       number_range_not_intern  = 2
22       object_not_found         = 3
23       quantity_is_0            = 4
24       interval_overflow        = 5
25       OTHERS                   = 6.
26
27   IF sy-subrc <> 0.
28     lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~
          get_message_container( ).
29     lo_mc->add_message_text_only(
```

```
30        iv_msg_type = /iwbep/if_message_container⇒
              gcs_message_type-error
31        iv_msg_text = 'Failed␣to␣generate␣new␣Order␣ID␣from␣
              number␣range' ).
32      RAISE EXCEPTION NEW /iwbep/cx_mgw_busi_exception(
            message_container = lo_mc ).
33    ENDIF.
34
35    ls_order-order_id = lv_orderid.
36    ls_order-mandt    = sy-mandt.
37
38
39    " Insert error handling
40    TRY.
41        INSERT zmr_orders FROM ls_order.
42    CATCH cx_sy_open_sql_db INTO DATA(lx_sql).
43        lo_mc = me→/iwbep/if_mgw_conv_srv_runtime~
              get_message_container( ).
44        lo_mc→add_message_text_only(
45          iv_msg_type = /iwbep/if_message_container⇒
                gcs_message_type-error
46          iv_msg_text = 'Sql␣database␣exception' ).
47        RAISE EXCEPTION NEW /iwbep/cx_mgw_tech_exception(
              message_container = lo_mc ).
48    CATCH cx_sy_itab_duplicate_key INTO DATA(lx_dup).
49        lo_mc = me→/iwbep/if_mgw_conv_srv_runtime~
              get_message_container( ).
50        lo_mc→add_message_text_only(
51          iv_msg_type = /iwbep/if_message_container⇒
                gcs_message_type-error
52          iv_msg_text = |Order with ID { lv_orderid } already
                exists| ).
53        RAISE EXCEPTION NEW /iwbep/cx_mgw_busi_exception(
              message_container = lo_mc ).
54    ENDTRY.
55
56
57    er_entity = ls_order.
58  ENDMETHOD.
```

6.4.4 orderset_delete_entity

Metoda orderset_delete_entity 6.7 odpowiada za usunięcie pojedynczego zamówienia z tabeli `zmr_orders` na podstawie klucza `OrderId` przekazanego w parametrze wejściowym.

- Jeśli rekord o podanym `OrderId` istnieje, zostaje usunięty z bazy danych.
- Jeśli rekord nie zostanie znaleziony, generowany jest wyjątek biznesowy (/iwbep/cx_mgw_busi_exception) z komunikatem informującym o braku zamówienia.

- W przypadku błędu połączenia z bazą danych lub problemów z zapytaniem SQL metoda zgłasza wyjątek techniczny (/iwbep/cx_mgw_tech_exception).

Listing 6.7: Metoda orderset_delete_entity

```
METHOD orderset_delete_entity.

  DATA: lv_order_id TYPE zmr_orders-order_id,
        lo_mc       TYPE REF TO /iwbep/if_message_container.

  " --- Pobranie klucza z requesta ---
  READ TABLE it_key_tab INTO DATA(ls_key) WITH KEY name = 'OrderId'.
  IF sy-subrc = 0.
    lv_order_id = ls_key-value.
  ELSE.
    lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~get_message_container( ).
    lo_mc->add_message_text_only(
      iv_msg_type = /iwbep/if_message_container=>gcs_message_type-error
      iv_msg_text = 'OrderId is required' ).
    RAISE EXCEPTION NEW /iwbep/cx_mgw_busi_exception(
      message_container = lo_mc ).
  ENDIF.

  " --- Pr ba usuni cia rekordu ---
  TRY.
      DELETE FROM zmr_orders WHERE order_id = @lv_order_id.

      IF sy-subrc <> 0.
        lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~get_message_container( ).
        lo_mc->add_message_text_only(
          iv_msg_type = /iwbep/if_message_container=>gcs_message_type-error
          iv_msg_text = |Order with ID { lv_order_id } not found| ).
        RAISE EXCEPTION NEW /iwbep/cx_mgw_busi_exception(
          message_container = lo_mc ).
      ENDIF.

    CATCH cx_sy_open_sql_db INTO DATA(lx_sql).
      lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~get_message_container( ).
      lo_mc->add_message_text_only(
        iv_msg_type = /iwbep/if_message_container=>gcs_message_type-error
        iv_msg_text = |Database error while deleting Order { lv_order_id }| ).
      RAISE EXCEPTION NEW /iwbep/cx_mgw_tech_exception(
        message_container = lo_mc ).
```

```
36    ENDTRY.
37
38  ENDMETHOD.
```

6.4.5 devicetypesset_get_entity

Metoda devicetypesset_get_entity 6.8 odpowiada za pobranie pojedynczego rekordu typu urządzenia (`DeviceType`) na podstawie klucza `Id` przekazanego w parametrze wejściowym.

- Jeśli rekord o podanym `Id` istnieje, metoda zwraca jego dane w strukturze encji (`er_entity`).
- Jeśli rekord nie zostanie znaleziony, generowany jest wyjątek biznesowy (/iwbep/cx_mgw_busi_exception) z komunikatem informującym o braku typu urządzenia.
- W przypadku błędu połączenia z bazą danych lub problemów z zapytaniem SQL metoda zgłasza wyjątek techniczny (/iwbep/cx_mgw_tech_exception).

Listing 6.8: metoda devicetypesset_get_entity

```
1  METHOD devicetypesset_get_entity.
2
3    DATA: lv_type_id TYPE zmr_device_type-id,
4          lo_mc      TYPE REF TO /iwbep/if_message_container.
5
6    lv_type_id = it_key_tab[ name = 'Id' ]-value.
7
8    TRY.
9        SELECT SINGLE *
10         FROM zmr_device_type
11         INTO @er_entity
12         WHERE id = @lv_type_id.
13
14       IF sy-subrc <> 0.
15         lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~
              get_message_container( ).
16         lo_mc->add_message_text_only(
17           iv_msg_type = /iwbep/if_message_container=>
                gcs_message_type-error
18           iv_msg_text = |DeviceType with ID { lv_type_id } not
                found| ).
19         RAISE EXCEPTION NEW /iwbep/cx_mgw_busi_exception(
              message_container = lo_mc ).
20       ENDIF.
21
22     CATCH cx_sy_open_sql_db INTO DATA(lx_sql).
23       lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~
            get_message_container( ).
24       lo_mc->add_message_text_only(
```

```
25          iv_msg_type = /iwbep/if_message_container⇒
               gcs_message_type-error
26          iv_msg_text = |Database error while reading DeviceType
                with ID { lv_type_id }| ).
27        RAISE EXCEPTION NEW /iwbep/cx_mgw_tech_exception(
             message_container = lo_mc ).
28    ENDTRY.
29
30  ENDMETHOD.
```

6.4.6 devicetypesset_get_entityset

Metoda devicetypesset_get_entityset 6.9 odpowiada za pobranie wszystkich rekordów typu urządzenia (`DeviceType`) i zwrócenie ich w postaci listy encji (`et_entityset`).

- Jeśli w tabeli nie ma żadnych rekordów, metoda generuje wyjątek biznesowy (/iwbep/cx_mgw_busi_exception) z komunikatem informującym o braku typów urządzeń.
- W przypadku błędu połączenia z bazą danych lub problemów z zapytaniem SQL metoda zgłasza wyjątek techniczny (/iwbep/cx_mgw_tech_exception).

Listing 6.9: metoda devicetypesset_get_entityset

```
1  METHOD devicetypesset_get_entityset.
2
3    DATA lo_mc TYPE REF TO /iwbep/if_message_container.
4
5    TRY.
6        SELECT *
7          FROM zmr_device_type
8          INTO TABLE @et_entityset.
9
10       IF et_entityset IS INITIAL.
11         lo_mc = me→/iwbep/if_mgw_conv_srv_runtime~
              get_message_container( ).
12         lo_mc→add_message_text_only(
13           iv_msg_type = /iwbep/if_message_container⇒
                gcs_message_type-error
14           iv_msg_text = 'No␣device␣types␣found' ).
15         RAISE EXCEPTION NEW /iwbep/cx_mgw_busi_exception(
              message_container = lo_mc ).
16       ENDIF.
17
18     CATCH cx_sy_open_sql_db INTO DATA(lx_sql).
19       lo_mc = me→/iwbep/if_mgw_conv_srv_runtime~
            get_message_container( ).
20       lo_mc→add_message_text_only(
21         iv_msg_type = /iwbep/if_message_container⇒
              gcs_message_type-error
```

```
22          iv_msg_text = 'Database␣error␣while␣reading␣device␣
               types' ).
23        RAISE EXCEPTION NEW /iwbep/cx_mgw_tech_exception(
             message_container = lo_mc ).
24    ENDTRY.
25
26  ENDMETHOD.
```

6.4.7 devicemodelsset_get_entity

Metoda devicemodelsset_get_entity 6.10 odpowiada za pobranie pojedynczego rekordu modelu urządzenia (`DeviceModel`) na podstawie klucza `Id` przekazanego w parametrze wejściowym.

- Jeśli rekord o podanym `Id` istnieje, metoda zwraca jego dane w strukturze encji (`er_entity`).
- Jeśli rekord nie zostanie znaleziony, generowany jest wyjątek biznesowy (/iwbep/cx_mgw_busi_exception) z komunikatem informującym o braku modelu urządzenia.
- W przypadku błędu połączenia z bazą danych lub problemów z zapytaniem SQL metoda zgłasza wyjątek techniczny (/iwbep/cx_mgw_tech_exception).

Listing 6.10: devicemodelsset_get_entity

```
1  METHOD devicemodelsset_get_entity.
2
3    DATA: lv_model_id TYPE zmr_device_model-id,
4          lo_mc       TYPE REF TO /iwbep/if_message_container.
5
6    lv_model_id = it_key_tab[ name = 'Id' ]-value.
7
8    TRY.
9        SELECT SINGLE *
10         FROM zmr_device_model
11         INTO @er_entity
12         WHERE id = @lv_model_id.
13
14       IF sy-subrc <> 0.
15         lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~
              get_message_container( ).
16         lo_mc->add_message_text_only(
17           iv_msg_type = /iwbep/if_message_container=>
                gcs_message_type-error
18           iv_msg_text = |DeviceModel with ID { lv_model_id }
                not found| ).
19         RAISE EXCEPTION NEW /iwbep/cx_mgw_busi_exception(
              message_container = lo_mc ).
20       ENDIF.
21
```

```
22      CATCH cx_sy_open_sql_db INTO DATA(lx_sql).
23        lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~
             get_message_container( ).
24        lo_mc->add_message_text_only(
25          iv_msg_type = /iwbep/if_message_container=>
               gcs_message_type-error
26          iv_msg_text = |Database error while reading
               DeviceModel with ID { lv_model_id }| ).
27        RAISE EXCEPTION NEW /iwbep/cx_mgw_tech_exception(
             message_container = lo_mc ).
28    ENDTRY.
29
30  ENDMETHOD.
```

6.4.8 devicemodelsset_get_entityset

Metoda devicemodelsset_get_entityset 6.11 odpowiada za pobranie zbioru rekordów modeli urządzeń (`DeviceModel`) na podstawie filtrów przesłanych w zapytaniu OData.

- Filtry są przetwarzane dynamicznie i wykorzystywane do zbudowania klauzuli WHERE dla zapytania do tabeli `zmr_device_model`.
- Jeśli dla podanych filtrów nie zostaną znalezione żadne rekordy, generowany jest wyjątek biznesowy (/iwbep/cx_mgw_busi_exception) z odpowiednim komunikatem.
- W przypadku problemów z połączeniem do bazy danych lub błędów SQL zgłaszany jest wyjątek techniczny (/iwbep/cx_mgw_tech_exception).

Listing 6.11: Metoda devicemodelsset_get_entityset

```
1   METHOD devicemodelsset_get_entityset.
2
3     DATA: lv_where TYPE string,
4           lo_mc    TYPE REF TO /iwbep/if_message_container.
5
6     TRY.
7         " Pobranie warunku WHERE z filtr w OData
8         lv_where = io_tech_request_context->get_osql_where_clause
             ( ).
9
10        " SELECT z dynamicznym WHERE -> od razu do et_entityset
11        SELECT *
12          FROM zmr_device_model
13          INTO TABLE @et_entityset
14          WHERE (lv_where).
15
16        " Je li brak wynik w
17        IF et_entityset IS INITIAL.
18          lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~
               get_message_container( ).
```

```
19          lo_mc->add_message_text_only(
20            iv_msg_type = /iwbep/if_message_container=>
                gcs_message_type-error
21            iv_msg_text = 'No device models found for given
                filter' ).
22          RAISE EXCEPTION NEW /iwbep/cx_mgw_busi_exception(
              message_container = lo_mc ).
23        ENDIF.
24
25      CATCH cx_sy_open_sql_db INTO DATA(lx_sql).
26        lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~
            get_message_container( ).
27        lo_mc->add_message_text_only(
28          iv_msg_type = /iwbep/if_message_container=>
              gcs_message_type-error
29          iv_msg_text = 'Database error while reading device
              models' ).
30        RAISE EXCEPTION NEW /iwbep/cx_mgw_tech_exception(
            message_container = lo_mc ).
31    ENDTRY.
32
33  ENDMETHOD.
```

Listing 6.4: orderset_get_entity

```
METHOD orderset_get_entity.
  DATA: lv_order_id TYPE zmr_orders-order_id,
        lo_mc       TYPE REF TO /iwbep/if_message_container.

  lv_order_id = it_key_tab[ name = 'OrderId' ]-value.

  TRY.
      SELECT SINGLE *
        FROM zmr_orders
        INTO @er_entity
        WHERE order_id = @lv_order_id.

      IF sy-subrc <> 0.
        lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~
           get_message_container( ).
        lo_mc->add_message_text_only(
          iv_msg_type = /iwbep/if_message_container=>
             gcs_message_type-error
          iv_msg_text = |Order with ID { lv_order_id } not
             found| ).
        RAISE EXCEPTION NEW /iwbep/cx_mgw_busi_exception(
           message_container = lo_mc ).
      ENDIF.

    CATCH cx_sy_open_sql_db INTO DATA(lx_sql).
      lo_mc = me->/iwbep/if_mgw_conv_srv_runtime~
         get_message_container( ).
      lo_mc->add_message_text_only(
        iv_msg_type = /iwbep/if_message_container=>
           gcs_message_type-error
        iv_msg_text = |Order with ID { lv_order_id } not found
           | ).
      RAISE EXCEPTION NEW /iwbep/cx_mgw_tech_exception(
         message_container = lo_mc ).
  ENDTRY.

ENDMETHOD.
```