Stwórz tabele

1. Tabela z samochodami

kolumny: ID, marka, przebieg, cena, status ( używany/nowy CHAR U/N )

2. Tabela z klientami

ID (klucz), imie, nazwisko, pesel (klucz) tabela ma mieć 2 klucze.

3. Tabela z zamówieniami

kolumny ID, klient\_id, Data zamówienia, id\_samochodu

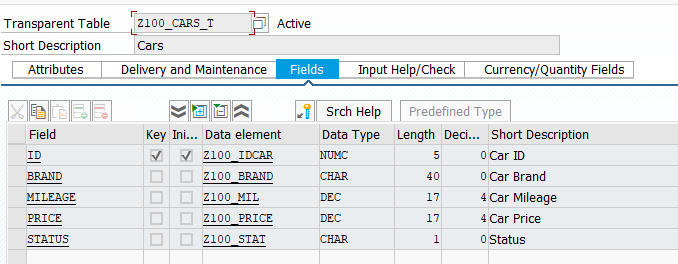
4. Tabela z wypozyczeniami

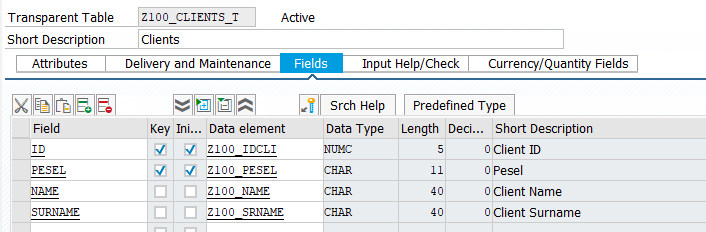
kolumny ID, klient\_id, data\_wypozyczenia, data\_zwrotu, id\_samochodu

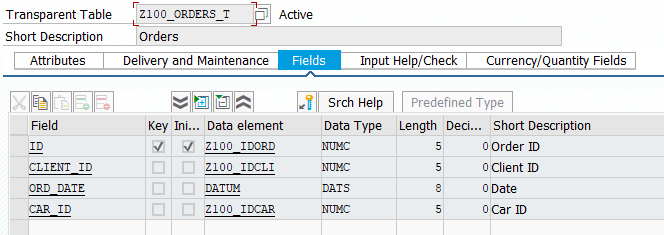
5. Tabela z logami ZMJ\_LOGGER\_T,

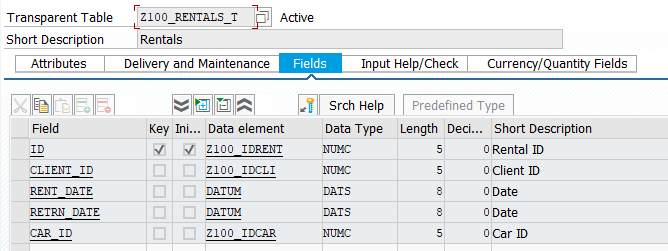
ID ( unikalny sposób np. CLIENT+ciag 10 losowych znaków, CAR+ciag 10 losowych znaków), operation\_name ( np add\_client, edit\_client, remove\_car etc. ) ,

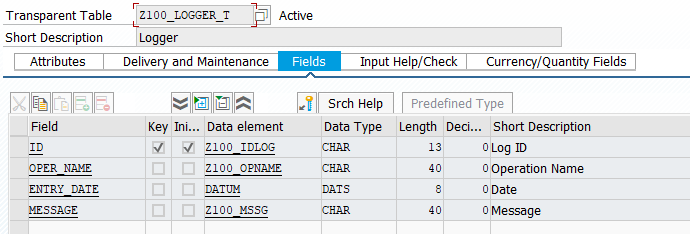
data\_wpisu, message ( tutaj treść wpisu )











ZADANIE:

Raport **ZMJ\_MANAGE\_CLIENT\_DATA** plik ( client\_data )

Wymagania:

1.Radiobutton służący do dodania klientów do bazy danych - ma za zadanie operując na pliku wejściowym typu CSV wyciągnać z niego dane i umieścic w tabelii z klientami, w pliku znajdują się kolumny ID, imie, nazwisko, pesel. ( kazda aktualizacje zapisz w tabeli

ZMJ\_LOGGER\_T )

2. Radiobutton służący do aktualizacji danych tabeli klientów również

ma za zadanie operując na pliku wejściowym typu CSV wyciągnać z niego dane i zaktualizować je

w tabelii z klientami na podstawie ID oraz peselu.

( kazda aktualizacje zapisz w tabeli ZMJ\_LOGGER\_T ) pamiętaj o error handlingu przypadków kiedy podane ID w pliku nie istnieje

( wtedy message w tabeli ZMJ\_LOGGER\_T powinienen być czymś w stylu ( odczytany pracownik z ID

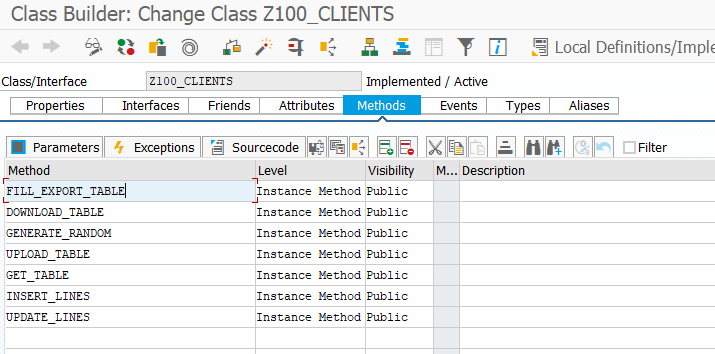
nie istnieje i nie mógł zostać edytowany )

3. Radiobutton sluzacy do exportu do pliku ( csv ) rekordów z tabeli klientów razem

z wpisami z tabeli z zamówieniami i wypozyczeniami ( jeśli jakikolwiek klient je ma )

ROZWIĄZANIE:

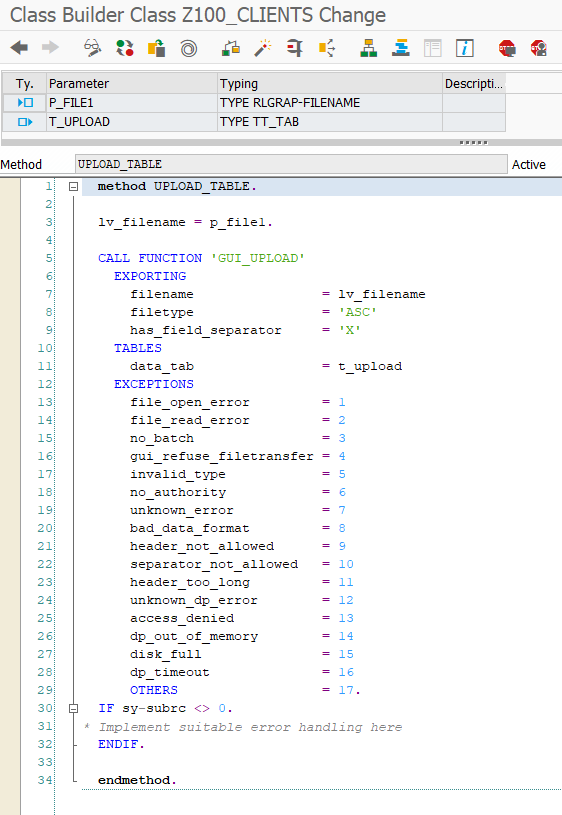
KLASA **Z100\_CLIENTS**



Metody:

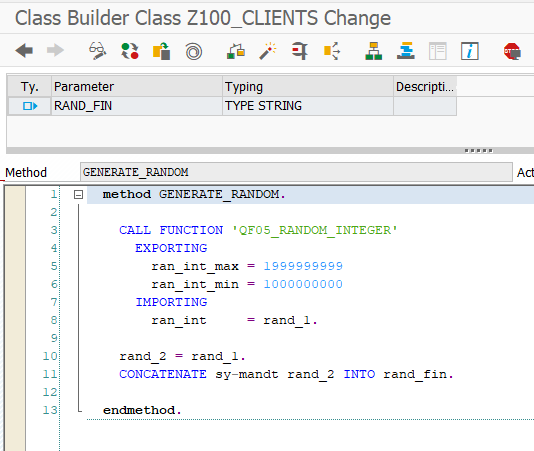
1 UPLOAD TABLE - ładowanie pliku CSV (czyli woła Funkcje GUI Upload)

Importuje ścieżke pliku P\_FILE1 i exportuje tabele z danymi z pliku



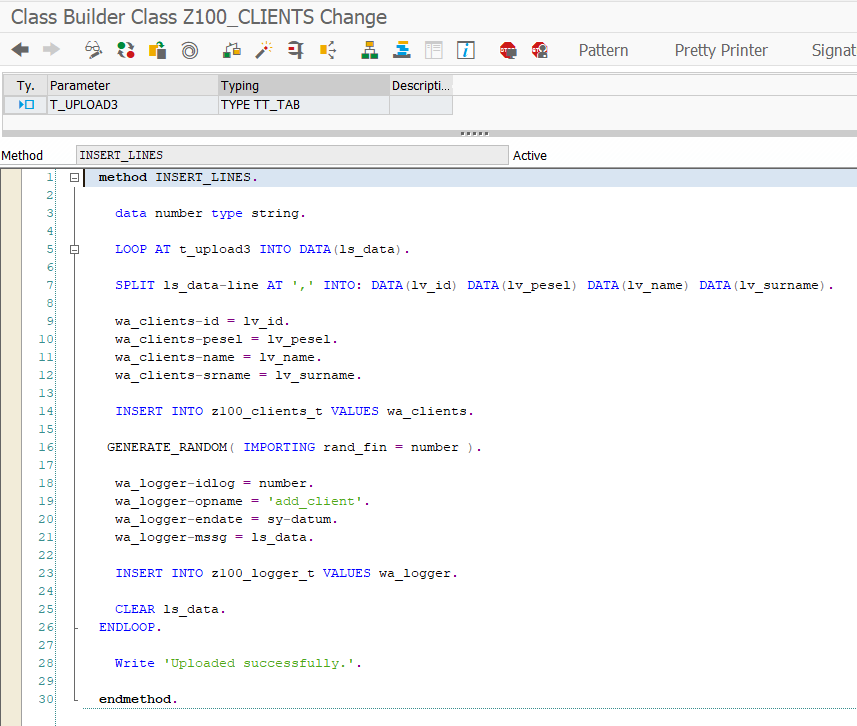
2 GENERATE\_RANDOM

Tworzy randomową liczbę Client+10 losowych znaków która potem będzie użyta w tabeli Logger



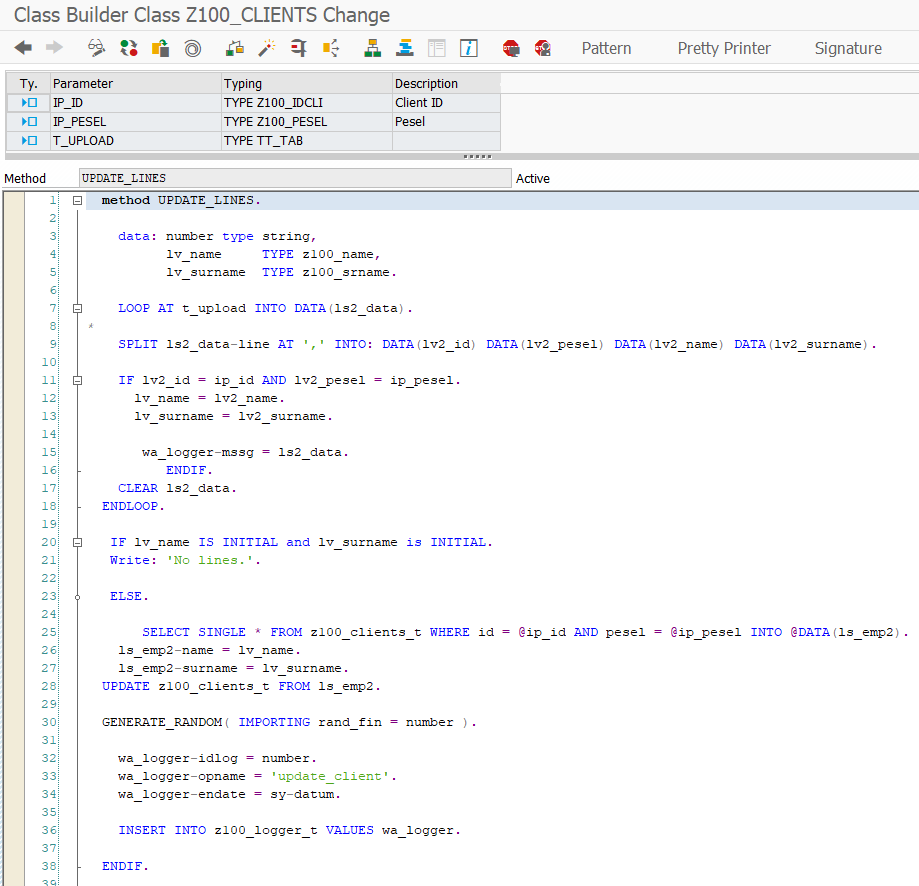
3 INSERT\_LINES

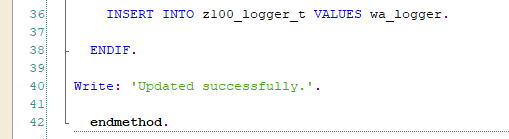
Importuje tabele z metody UPLOAD i na jej podstawie tworzy linie w tabelach Z100\_CLIENTS\_T oraz Z100\_LOGGER\_T (czyli wypełnia task 1). Przy okazji wywołuje też GENERATE\_RANDOM żeby się wykręciła losowa liczba do loggera



3 UPDATE\_LINES

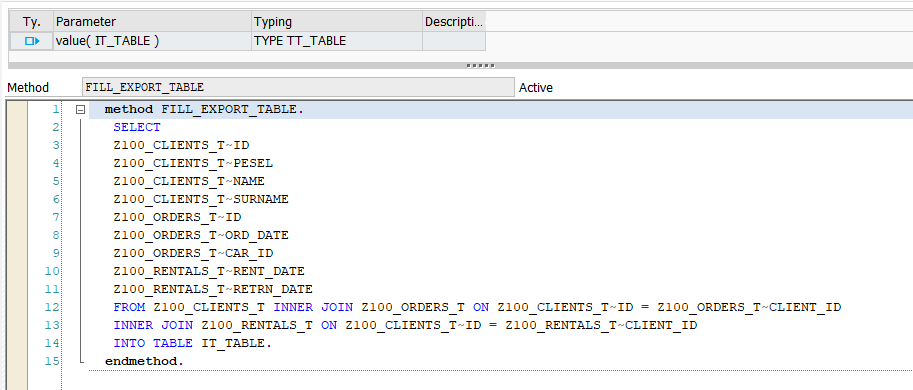
Importuje Client ID oraz Client Pesel oraz Tabele z UPLOAD -> robi update tej linii która ma dany ID i Pesel (czyli wypełnia task 2) + tworzy nowe linie w tabeli LOGGER





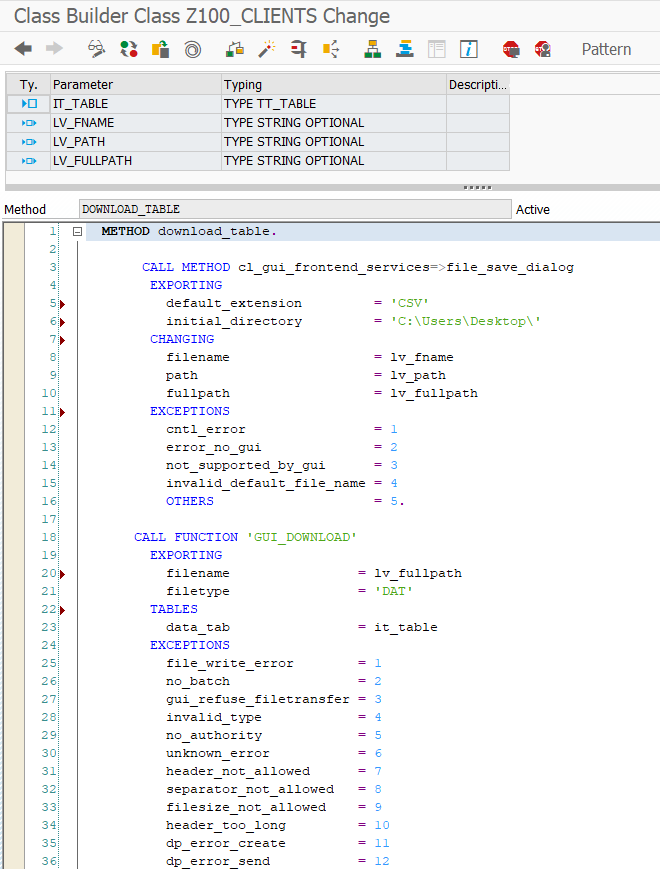
4 FILL\_EXPORT\_TABLE

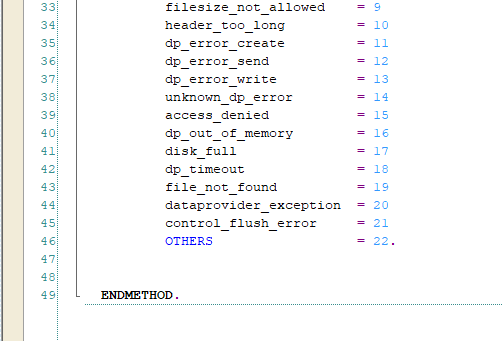
Tworzy internal table która ma być potem Exportowana do CSV - jest to JOIN trzech tabel Clients + Orders + Rentals



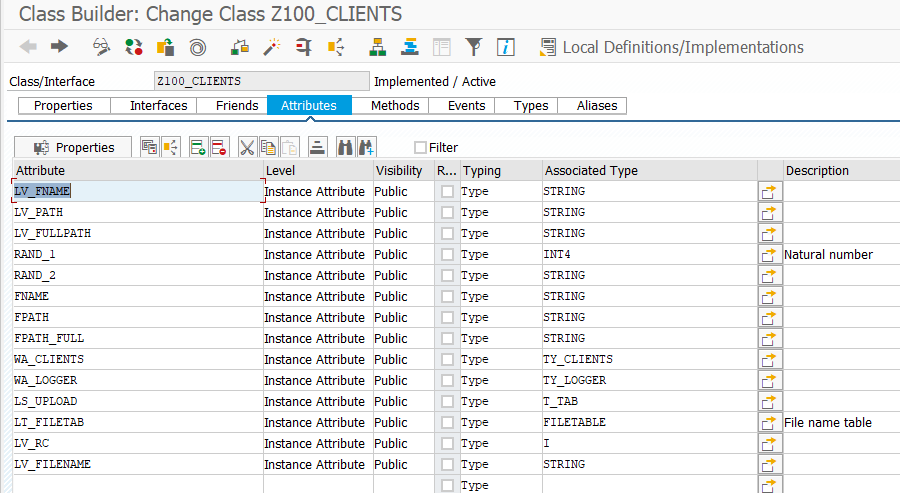
5 DOWNOLAD\_TABLE

Importuje tabele z poprzedniej metody i generuje z niej CSV która mozna zapisać na dysku

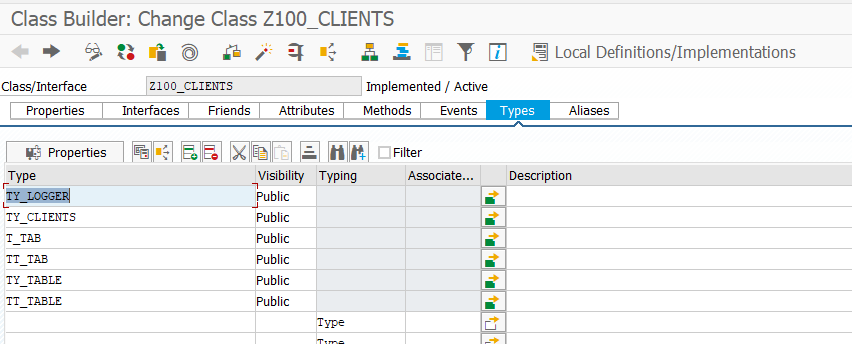


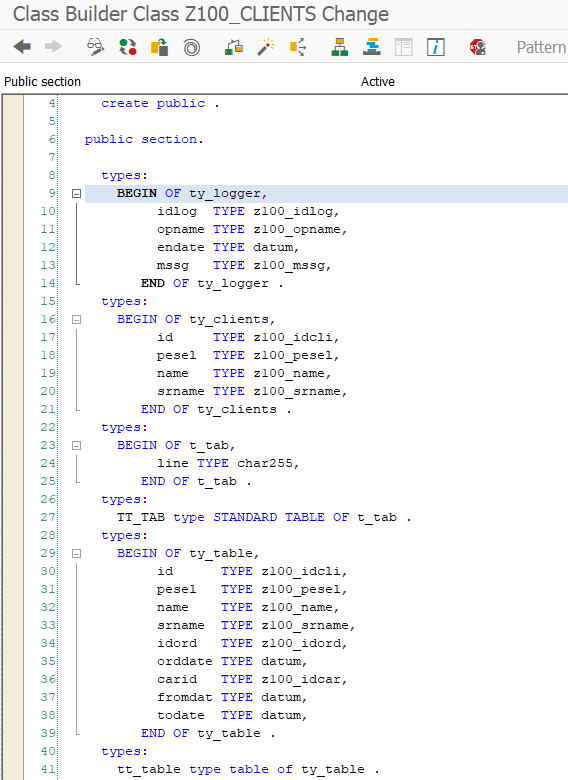


Attributes:



Types:





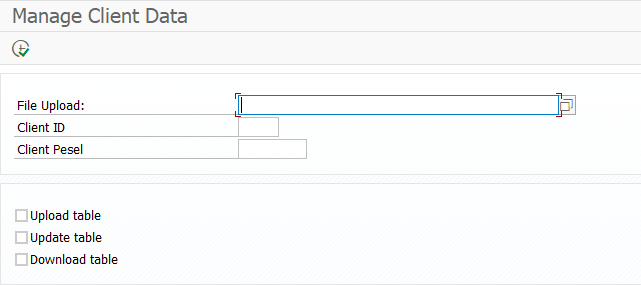
RAPORT:

Kod:

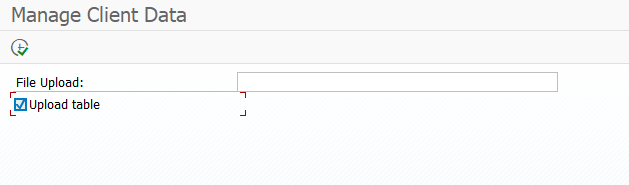
REPORT z100\_manage\_client\_data.  
  
TYPE-POOLS truxs.  
TABLES z100\_clients\_t.  
  
TYPES: BEGIN OF t\_tab1,  
         line TYPE char255,  
       END OF t\_tab1.  
  
TYPES: BEGIN OF ty\_table,  
         id      TYPE z100\_idcli,  
         pesel   TYPE z100\_pesel,  
         name    TYPE z100\_name,  
         srname  TYPE z100\_srname,  
         idord   TYPE z100\_idord,  
         orddate TYPE datum,  
         carid   TYPE z100\_idcar,  
         fromdat TYPE datum,  
         todate  TYPE datum,  
       END OF ty\_table.  
  
DATA: t\_upload1  TYPE STANDARD TABLE OF t\_tab1,  
      ls\_upload  TYPE t\_tab1,  
      lt\_filetab TYPE filetable,  
      lv\_rc      TYPE i,  
      it\_tab2    TYPE TABLE OF ty\_table.  
  
SELECTION-SCREEN BEGIN OF BLOCK b1 WITH FRAME TITLE TEXT-000.  
PARAMETERS: p\_file  TYPE   rlgrap-filename MODIF ID eid,  
            p\_id    TYPE   z100\_idcli MODIF ID wid,  
            p\_pesel TYPE z100\_pesel MODIF ID uid.  
SELECTION-SCREEN END OF BLOCK b1.  
  
SELECTION-SCREEN BEGIN OF BLOCK b2 WITH FRAME TITLE TEXT-001.  
PARAMETERS:  
  p\_chx1 AS CHECKBOX USER-COMMAND pay MODIF ID sc1,  
  p\_chx2 AS CHECKBOX USER-COMMAND sur MODIF ID sc2,  
  p\_chx3 AS CHECKBOX USER-COMMAND sta MODIF ID sc3.  
SELECTION-SCREEN END OF BLOCK b2.  
  
AT SELECTION-SCREEN OUTPUT.  
  LOOP AT SCREEN.  
    IF p\_chx1 EQ abap\_true.  
      p\_chx2 = abap\_false.  
      p\_chx3 = abap\_false.  
      IF screen-group1 EQ 'SC1' OR screen-group1 EQ 'EID'.  
        screen-active = 1.  
      ELSE.  
        screen-active = 0.  
      ENDIF.  
      MODIFY SCREEN.  
    ENDIF.  
  
    IF p\_chx2 EQ abap\_true.  
      p\_chx1 = abap\_false.  
      p\_chx3 = abap\_false.  
      IF screen-group1 EQ 'SC2' OR screen-group1 EQ 'EID' OR screen-group1 EQ 'WID' OR screen-group1 EQ 'UID'.  
        screen-active = 1.  
      ELSE.  
        screen-active = 0.  
      ENDIF.  
      MODIFY SCREEN.  
    ENDIF.  
  
    IF p\_chx3 EQ abap\_true.  
      p\_chx1 = abap\_false.  
      p\_chx2 = abap\_false.  
      IF screen-group1 EQ 'SC3'.  
        screen-active = 1.  
      ELSE.  
        screen-active = 0.  
      ENDIF.  
      MODIFY SCREEN.  
    ENDIF.  
  ENDLOOP.  
  
AT SELECTION-SCREEN ON VALUE-REQUEST FOR p\_file.  
  
  CALL METHOD cl\_gui\_frontend\_services=>file\_open\_dialog  
    CHANGING  
      file\_table              = lt\_filetab  
      rc                      = lv\_rc  
    EXCEPTIONS  
      file\_open\_dialog\_failed = 1  
      cntl\_error              = 2  
      error\_no\_gui            = 3  
      not\_supported\_by\_gui    = 4  
      OTHERS                  = 5.  
  
  p\_file = lt\_filetab[ 1 ]-filename.  
  
START-OF-SELECTION.  
  
  DATA go\_client TYPE REF TO z100\_clients.  
  CREATE OBJECT go\_client.  
  
  IF p\_chx1 IS NOT INITIAL.  
  
    CALL METHOD go\_client->upload\_table  
      EXPORTING  
        p\_file1  = p\_file  
      IMPORTING  
        t\_upload = t\_upload1.  
  
    CALL METHOD go\_client->insert\_lines  
      EXPORTING  
        t\_upload3 = t\_upload1.  
  
  ENDIF.  
  
  IF p\_chx2 IS NOT INITIAL.  
  
    CALL METHOD go\_client->upload\_table  
      EXPORTING  
        p\_file1  = p\_file  
      IMPORTING  
        t\_upload = t\_upload1.  
  
    CALL METHOD go\_client->update\_lines  
      EXPORTING  
        ip\_id    = p\_id  
        ip\_pesel = p\_pesel  
        t\_upload = t\_upload1.  
  
  ENDIF.  
  
  
  IF p\_chx3 IS NOT INITIAL.  
  
    CALL METHOD go\_client->fill\_export\_table  
      IMPORTING  
        it\_table = it\_tab2.  
  
    IF it\_tab2 IS NOT INITIAL.  
  
      CALL METHOD go\_client->download\_table  
        EXPORTING  
          it\_table = it\_tab2.  
    ELSE.  
      WRITE 'No data available.'.  
  
    ENDIF.  
  
  ENDIF.

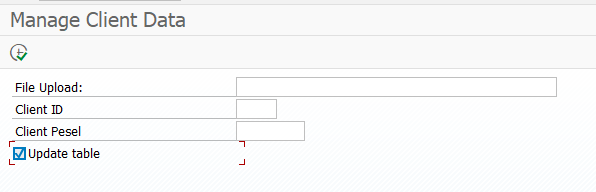
Działanie programu:

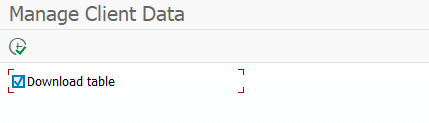
Wywołanie programu:



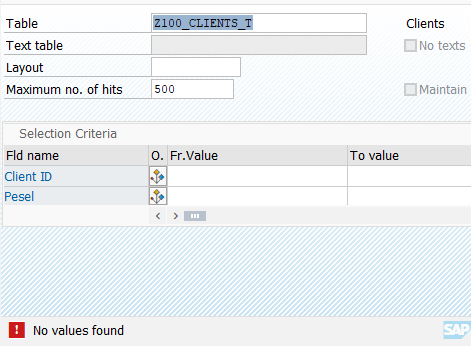
Po kolei zatickane:

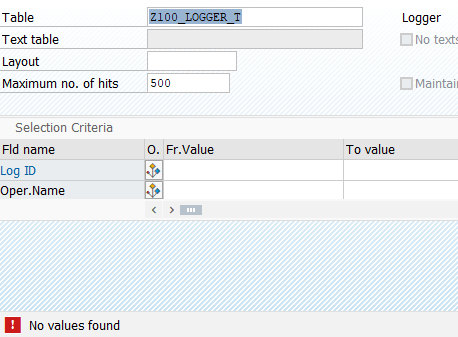




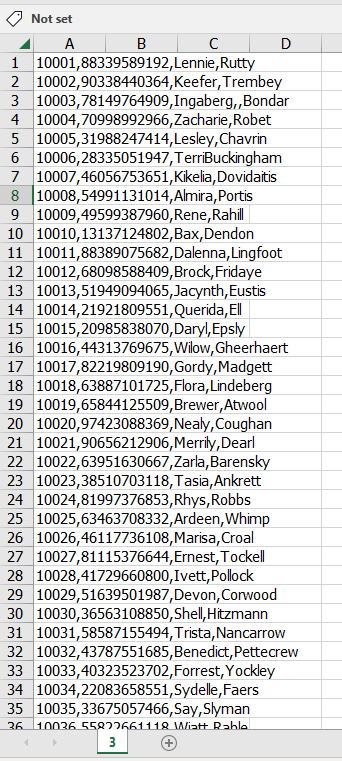


Przed wywołaniem programu

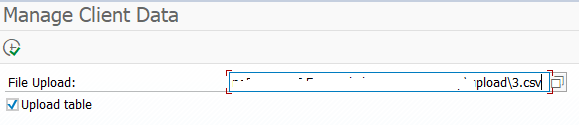


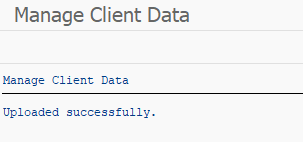


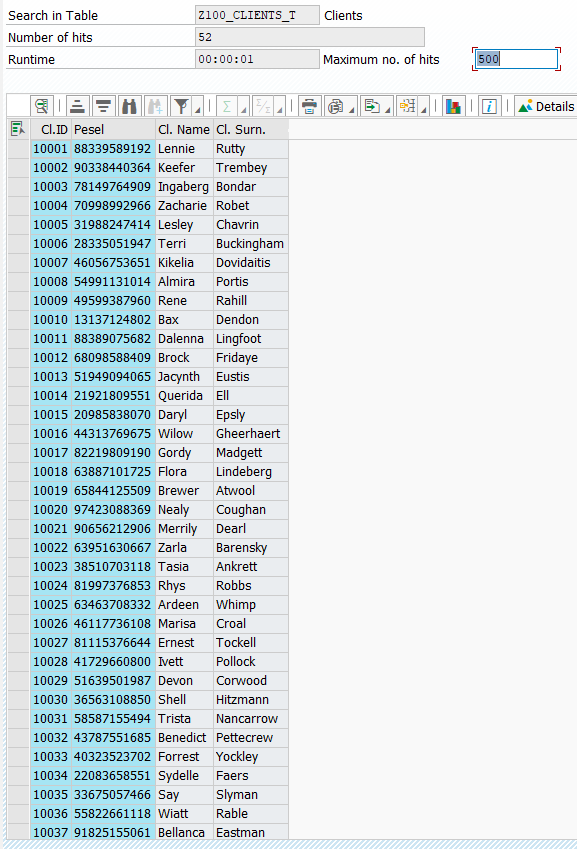
Plik do załadowania (52 linie):

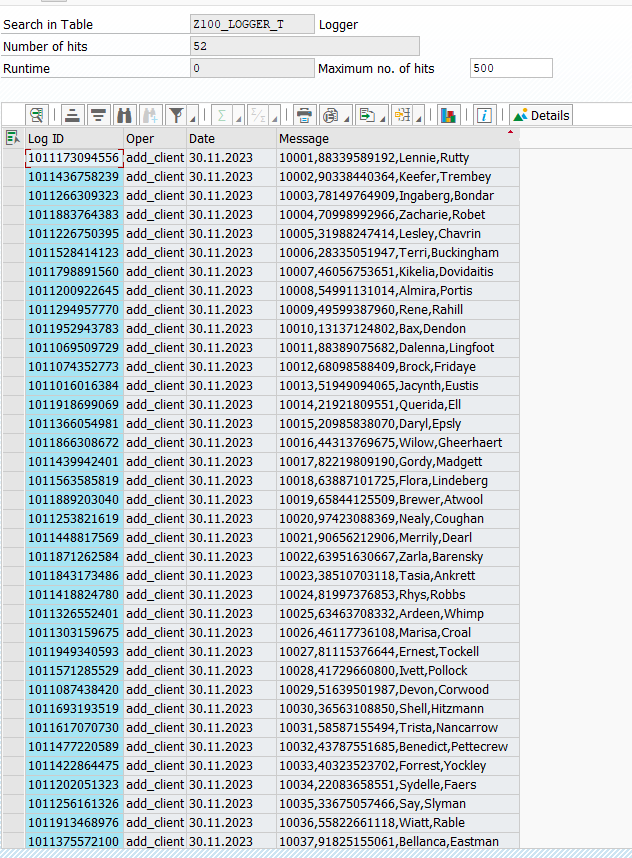




TASK1:  


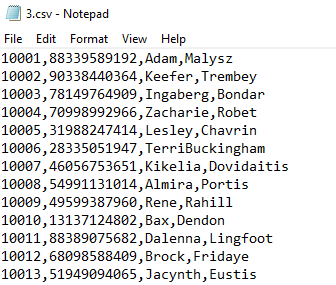




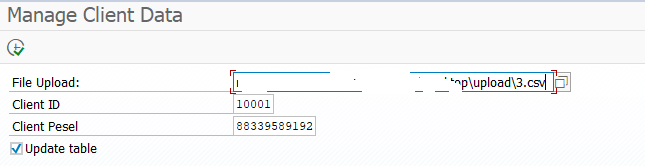


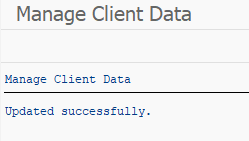
TASK2:

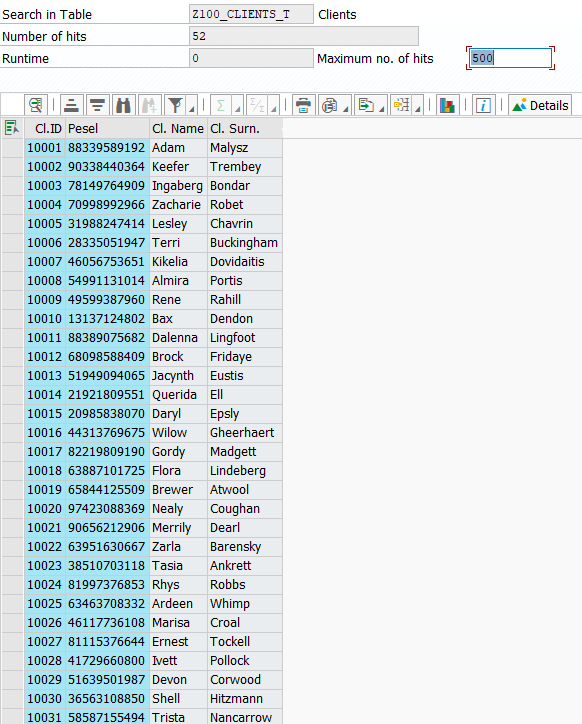
Zmieniamy plik w pierwszej linii na Adam Małysz:



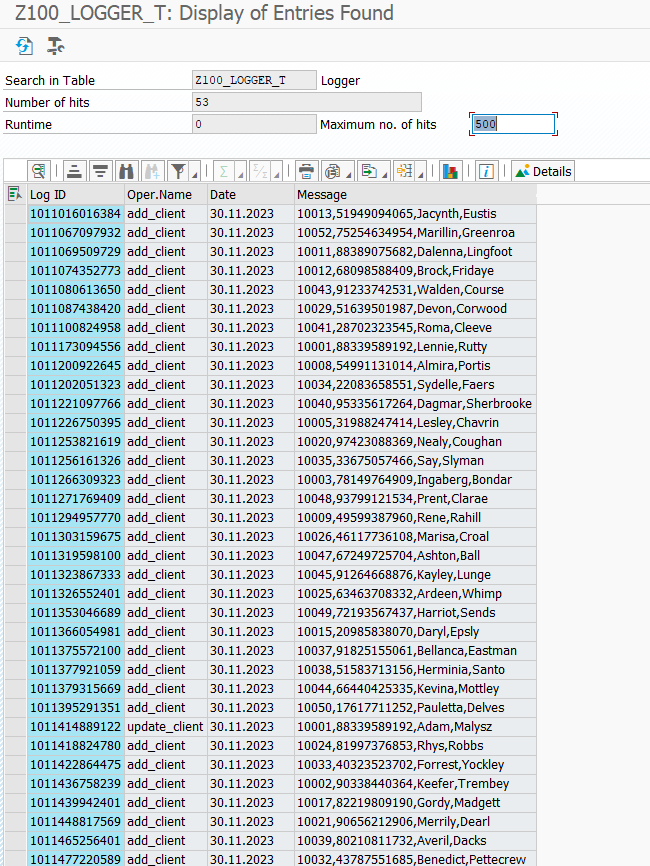
Uruchamiamy:



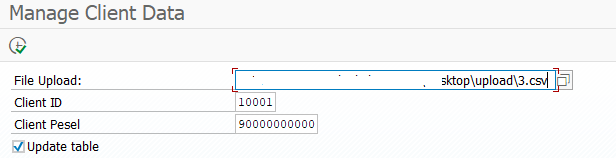


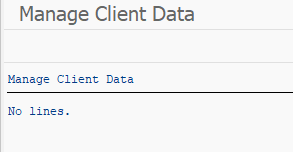


Logger -> dodana linie update\_client (na środku screenu)



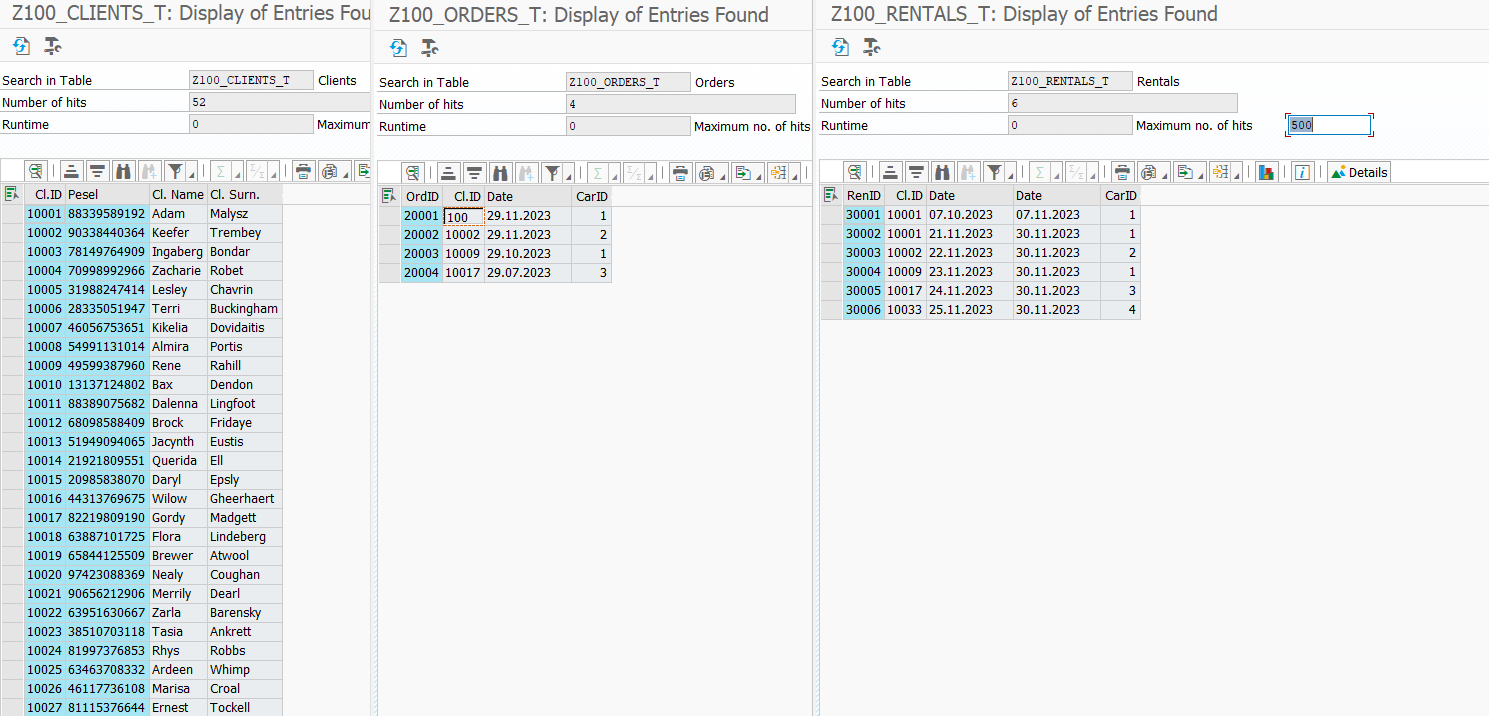
Uruchomienie ze złym peselem:





TASK3:

Tabele przed wywołaniem:



Uruchomienie:

