Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Электротехнический факультет

Кафедра «Информационные технологии и автоматизированные системы» направление подготовки: 09.03.04 Программная инженерия

О Т Ч Е Т

по учебной практике

Выполнил студент гр. РИС-19-1б

\_\_\_\_\_**\_\_\_\_**\_Наумов А.Н. \_**\_\_\_\_\_\_**

(фамилия, имя, отчество)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(подпись)

Проверил:

*\_\_\_старший преподаватель кафедры ИТАС Кузнецов Д.Б.\_\_*

*(должность, ФИО руководителя по практической подготовке от кафедры)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(оценка) (подпись)*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*(дата)*

Пермь 2021

**Содержание**

**Введение**3

1. **Анализ предметной области**4

1.1 Ознакомление с интерфейсом SAP 4

1.2Ознакомление с языком программирования ABAP 6

1.3Анализ поставленной задачи9

**2 Технология реализации**11

2.1 Создание таблиц бд11

2.2 Разработка пользовательского приложения13

**Заключение**17

**Список используемой литературы**18

**Приложение** 19

# 

**Введение**

**ABAP** (Advanced Business Application Programming) – внутренний язык высокого уровня программирования в среде SAP. Углубляться в историю особенно не хочется, но отмечу лишь, что язык синтаксис языка наиболее близок к языку программирования COBOL. Язык был создан в 1980 году для работы с системой SAP R/2, позднее унаследован в SAP R/3. Сохранён и в последующих версиях приложений SAP, наряду с Java стал языком создания приложений для SAP NetWeaver Application Server.

Язык позволяет работать с внутренними структурами данных, интерфейсами пользователя, транзакциями, отчётами, интерфейсами загрузки, выгрузки данных (word, excel, pdf, …); стоит отметить, что можно использовать объектно-ориентированные конструкции (ABAP Objects); существует множество технологий связи с другими системами (BAPI, RFC, …) для выгрузки и загрузки данных (или же сторонней обработки).

**Цель:**

Разработка программного продукта для работы с таблицами баз данных (система учёта продаж автомобильного салона).

**Задачи:**

1. Проанализировать поставленную задачу (поиск информации, обработка поставленной задачи).
2. Разработать визуальную часть для удобства пользователя.
3. Разработать программную часть.

# 

# **1 Анализ предметной области**

Предметной областью является создание система учёта продаж автомобильного салона в среде разработки SAP на языке программирования ABAP. Так как основной целью не является построение базы данных, это означает, что для примера работы программы будет достаточно двух таблиц: Автомобили и покупатели.

Таблицы будут созданы, используя встроенные в язык программирования инструменты.

**1.1 Ознакомление с интерфейсом SAP**

Одним из главных отличий ABAP от других языков программирования, таких как C++, Java и Python является то, что код программы никогда не хранится во внешних файлах, код находится в базе данных SAP и при запуске программы требуется войти в сеть предприятия, работающего с данной средой. На рисунке 1 изображено окно входа в систему.

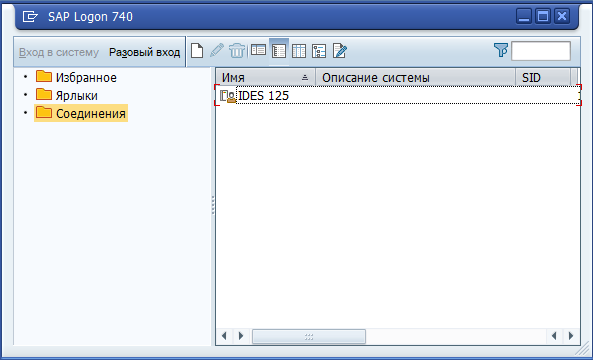


Рисунок 1 - Начальный экран

После входа в систему открывается окно с репозитарием, в котором создаются пакеты, программы, классы, интерфейсы и другие виды объектов, для хранения и использования на сервере. Окно с репозитарием изображено на рисунке 2. В данном случае создается пакет, в котором будет храниться код программы и глобальные классы и таблицы бд.

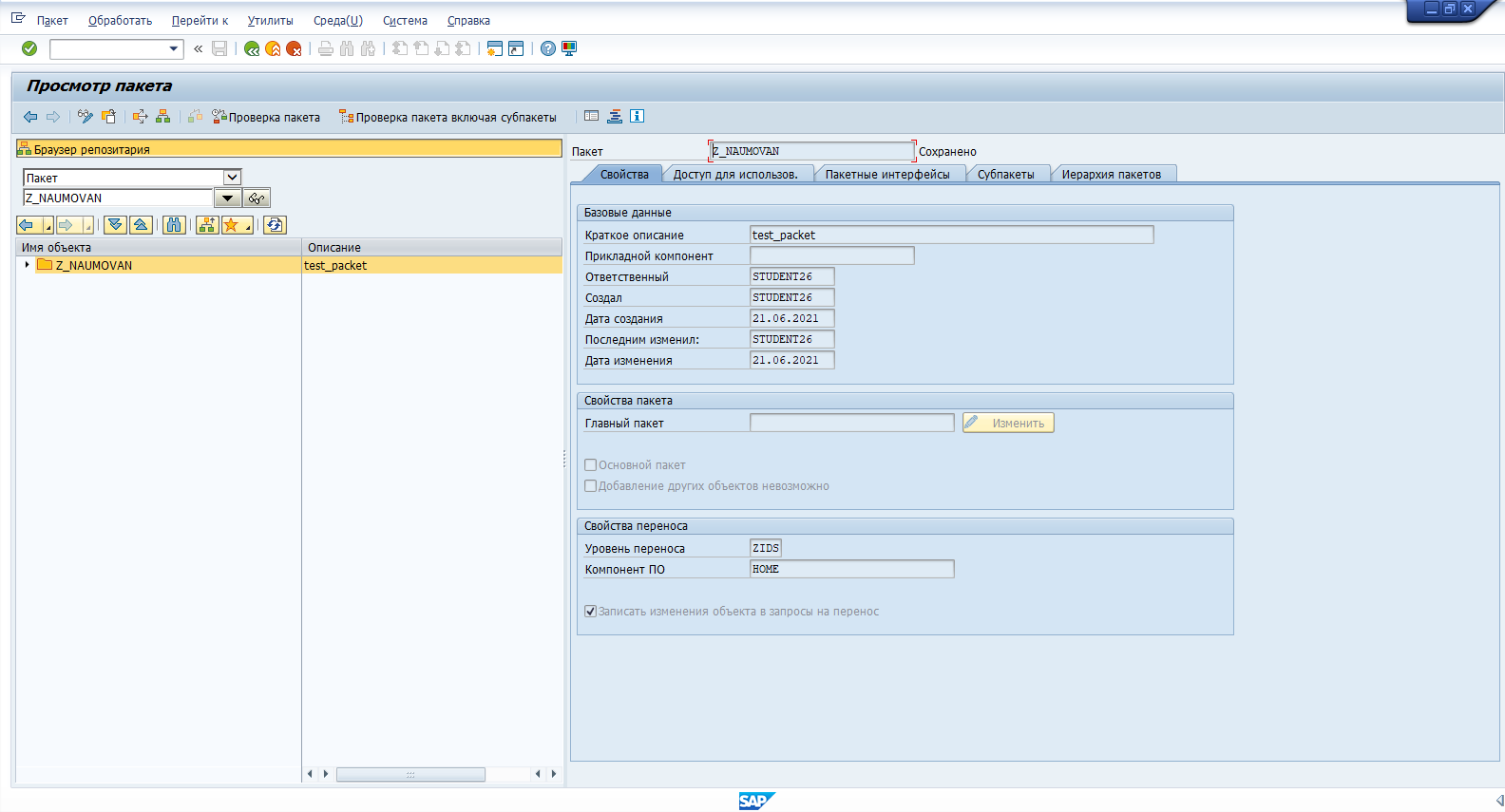


Рисунок 2 - Окно репозитария.

Структура пакета похожа на структуру хранения файлов, в данном случае хранятся несколько папок: Объекты словаря, Библиотека классов, Программы, Группы функций и Includes. Структура пакета изображена на рисунке 3.

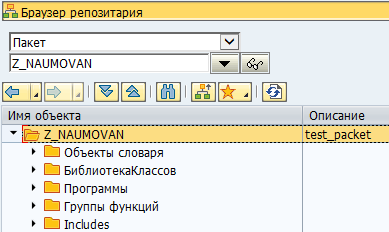


Рисунок 3 - Структура пакета.

Объекты словаря содержат таблицы, типы таблиц, структуры данных и элементы данных, так как программа подразумевает использование таблиц, то в пакете будут созданы таблицы и необходимое количество элементов данных. Классы, программы будут созданы по такой же логике.

Как итог, прошло ознакомление с интерфейсом среды разработки Sap и можно приступить к изучению самого языка программирования ABAP.

* 1. **Ознакомление с языком программирования ABAP**

ABAP – проприетарный язык программирования высокого уровня, используемый в решениях компании SAP. Как и в других языках программирования весь код тут разветвляется на исходный код, написанный программистом для реализации поставленной задачи на данном языке и внутренний код, который является частью ядра SAP, написанный для контролирования логики и реакции на события. Перейдем к типам данных. В языке ABAP типы данных мало чем отличаются от их реализации в других языках программирования, делятся на полные и неполные, различие их состоит в том, что полные типы данных имеют заранее определенную фиксированную длину. В список полных стандартных типов данных входят:

1. D – тип для даты. Формат ГГГГММДД, длиной 8.
2. T – тип для времени. Формат ЧЧММСС, длиной 6.
3. I – тип для целого числа, длиной 4.
4. F – тип для числа с плавающей точкой, длиной 8.
5. STRING – тип для символьной строки переменной длины.

Неполные стандартные типы данных:

1. С – тип для символьной строки, для которой нужно указать длину.
2. N – тип для числовой символьной строки, для которой нужно указать длину.
3. X – тип последовательности байтов, для которой нужно указать длину.

Стандартные типы данных можно использовать в программе для описания локальных типов данных, которые могут быть более полными или сложными, чем лежащие в их основе стандартные типы. Локальные типы данных существуют только в отдельно взятом пакете и, соответственно, могут использоваться только в ней. Для описания типов используется оператор TYPES. Циклы и ветвления смысла рассматривать нет, здесь они такие же, как и в популярных языках программирования. Конструкции ветвления кода также присутствуют: с условием IF, CASE, DO и WHILE. Синтаксис изображен на рисунке 4.

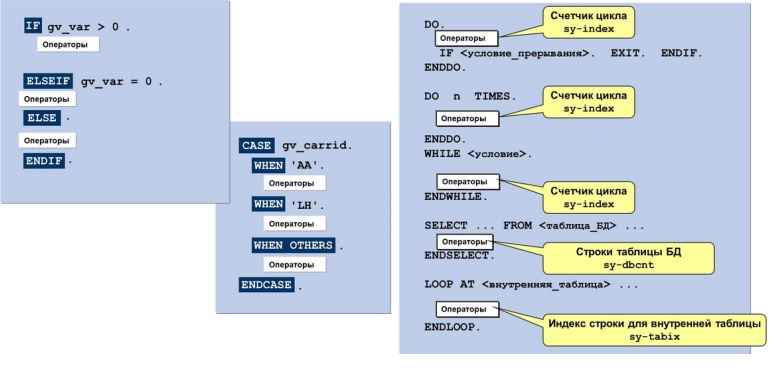


Рисунок 4 - Варианты ветвления кода.

Для модуляризации программы используются функции, которые объявляются в основном коде программы с помощью конструкции FORM и ENDFORM и вызываются с помощью команды PERFORM. Пример конструкции изображен на рисунке 5.

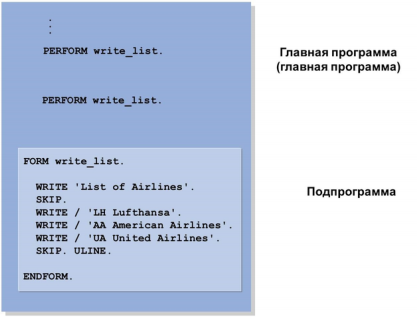


Рисунок 5 - Пример функции.

Так же для работы и масштабирования программы могут потребоваться классы для работы с данными. В ABAP-редакторе, который является IDE (Интегрированной средой разработки) есть возможность определения глобальных и локальных классов. Локальные классы определяются с помощью описания класса в отдельном файле и вызываемом с помощью INCLUDE. Вся работа с данными происходит в глобальных классах, которые создаются с помощью встроенного конструктора, изображенного на рисунке 6.

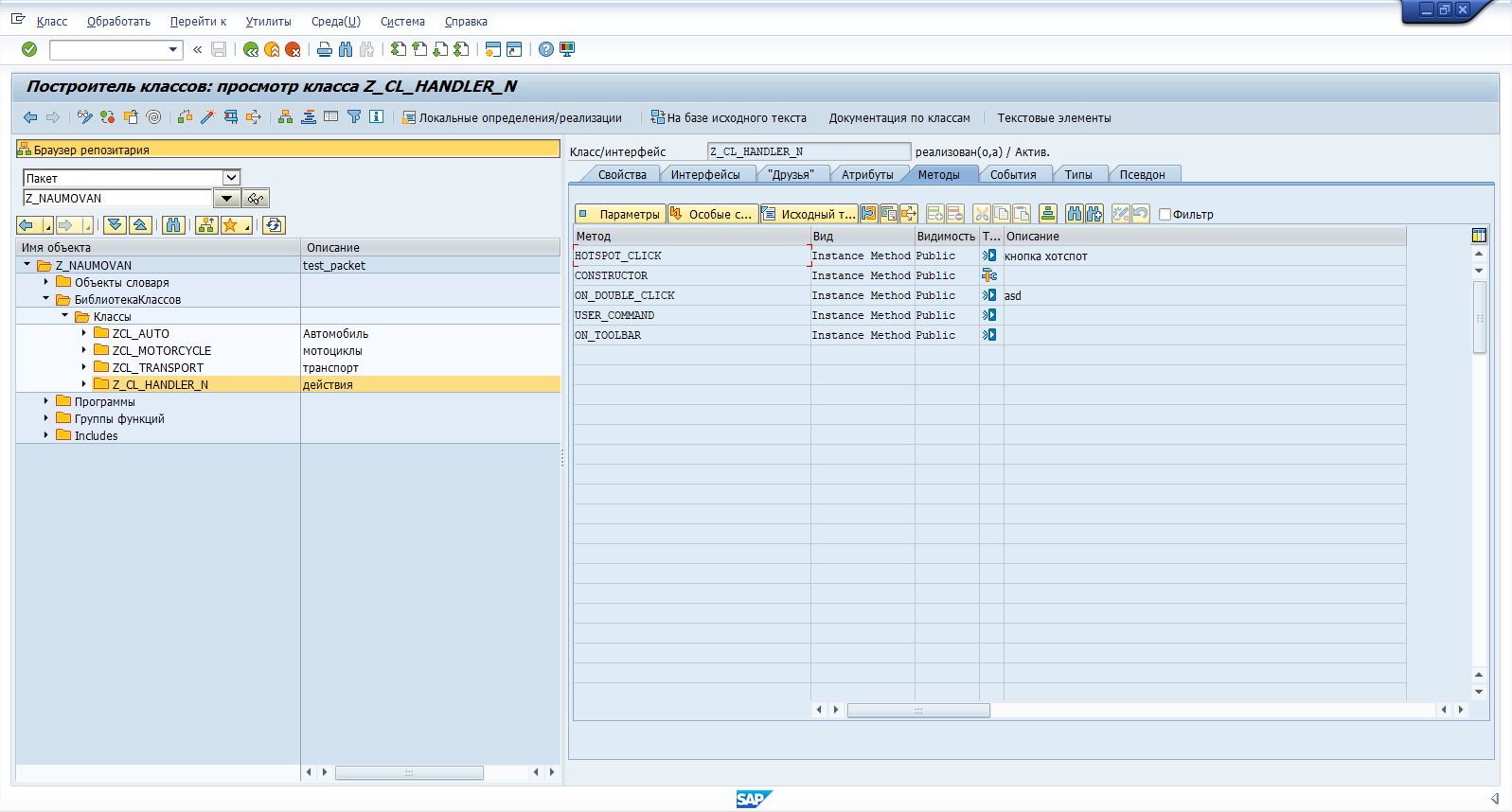


Рисунок 6 - Конструктор классов.

В окне конструктора классов можно увидеть вкладки, в которых и описывается вся логика программы.

Выводиться результаты работы программы будут на экраны. Экран – это локальный объект, который может содержать в себе структуру с полями ввода/вывода, кнопками и другими элементами экрана, помимо этого в экране также содержится логика обработки. В нашем случае экран будет содержать только контейнер для вывода ALV (ABAP LIST VIEWER). Данный элемент экрана используется для вывода таблицы на экран.

По итогу был изучен синтаксис языка ABAP и реализация элементов для выполнения задания.

* 1. **Анализ поставленной задачи и способы ее реализации на языке ABAP**

Создание учета продажи автомобилей в среде разработки SAP – задача на реализацию учебной практикой от выбранного предприятия. Так как в первую очередь язык ABAP предназначен для работы с таблицами, то требуется создать схему отношений для работы с данными. В нашем случае достаточно будет двух таблиц – это таблица с автомобилями и покупателями. Модель данных изображена на рисунке 7.

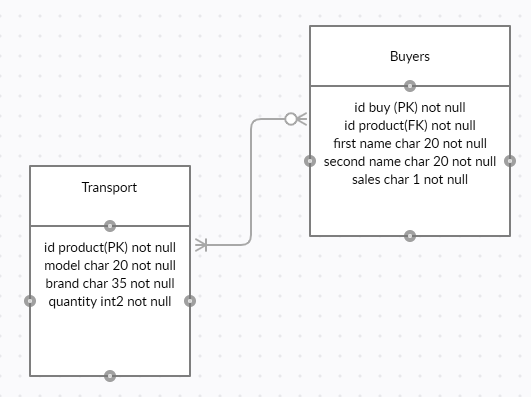


Рисунок 7 - Модель данных.

Для работы с данными было решено использовать классы, чтобы лучше ознакомиться с языком программирования на практике, всего будет создано 2 класса. Родительский класс, содержащий в себе методы отрисовки ALV и обращения к классу событий. От родительского класса наследуется класс автомобили, в котором будут содержаться методы обработки данных из таблицы автомобилей.

Результаты работы программы так же было решено выводить в отдельное окно для красивого вывода таблиц и углубления в изучении языка.

**Вывод:** По результатам анализа был изучен интерфейс программы Sap, синтаксис языка ABAP, проанализирована поставленная задача и выбран способ её реализации.

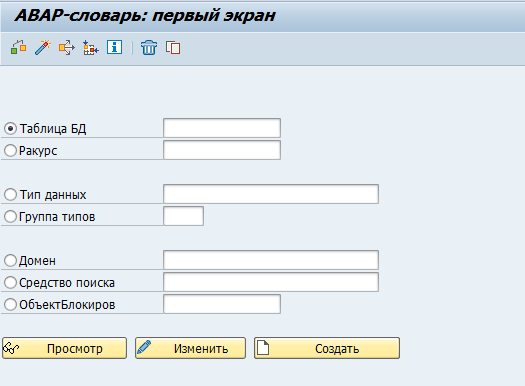
# **Технология реализации**

Процесс реализации приложения состоит из двух этапов:

1. создание таблиц бд;
2. написание кода на Abap, работа с интерфейсом.

**2.1 Создание таблиц бд.**

Процесс создания таблиц бд прост: первым делом нужен abap-словарь. Для этого необходимо либо через поле транзакции (транзакция se11), либо через навигатора по объектам вызвать abap-словарь.(рисунок 8 )

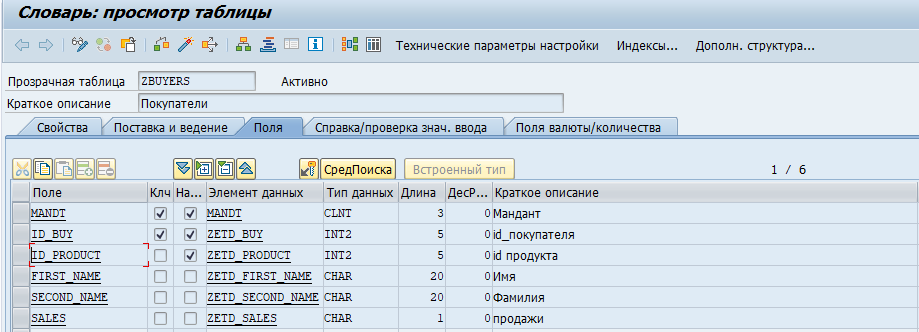


*Рисунок 8 - Abap-словарь.*

Сам словарь обладает достаточно широким функционалом: здесь создаются таблицы, ракурсы, типы данных и др., а также есть возможность работы с уже существующими типами.

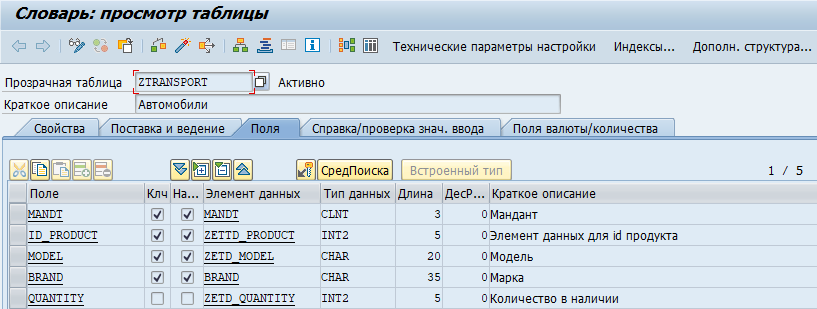
При нажатии кнопки «создать» открывается следующий экран создание таблицы, в котором указываются различные настройки таблицы: класс поставки (для чего создается, к примеру, класс A представляющий из себя прикладную таблицу, в которой хранятся основные и переменные данные), а также указываются необходимые поля таблицы: имена полей, ключи, элементы и типы данных, длина, краткое описание.

Создание полей таблицы покупателей (таблица ZBUYERS) отражено на рисунке 4.



*Рисунок 9 – таблица ZBUYERS.*

Для того чтобы отследить изменения в таблицах покупателей и автомобилей создается ключевое поле ID\_PRODUCT. Создание полей таблицы автомобилей (таблица ZTRANSPORT) отражено на рисунке 5.



*Рисунок 10 – таблица ZTRANSPORT.*

**2.2 Разработка пользовательского приложения.**

Так как таблицы уже определены в системе, то какого-то подключения извне не требуется и при обращении к таблицам в коде используются SQL- команды.

Процесс создания интерфейса основывается на взаимодействии действующих экранов, на которых отображаются параметры ввода, кнопки, таблицы и др.

Логика работы экранов базируется на сменяемости двух модульных блоков: PBO (Process Before Output), в котором прописываются команды до срабатывания экрана, и PAI (Process After Input), в этом модуле прописываются команды после ввода каких-либо данных на экране.

Вызов экрана выполняется при помощи команды Call screen.

Для взаимодействия с таблицей автомобилей создан класс ZCL\_TRANSPORT. В самом классе создано множество методов для обработки событий на экране, основными из которых является выборка данных из таблицы и действие двойного клика по полю таблицы.

На листинге 1 представлена реализация методов выборки из таблицы и двойного клика.

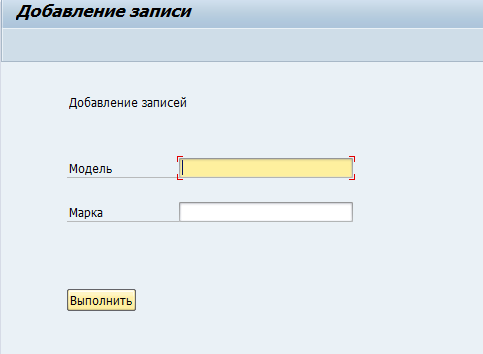
*Листинг 1 - реализация методов выборки из таблицы и двойного клика.*

METHOD select\_table\_transport.  
     SELECT \* FROM  
       ztransport  
       INTO CORRESPONDING FIELDS OF TABLE mt\_ztransport  
       WHERE ztransport~model IN mr\_model AND ztransport~brand IN mr\_brand.  
     GET REFERENCE OF mt\_ztransport INTO go\_ref\_table.  
     GET REFERENCE OF 'ZTRANSPORT' INTO go\_ref\_tab\_name.  
   ENDMETHOD.

  METHOD on\_double\_click.  
     DATA: lv\_mess   TYPE string,  
          lv\_row    TYPE char10,  
          lv\_column TYPE char15.  
     WRITE e\_row-index TO lv\_row.  
     WRITE e\_column TO lv\_column.  
     lv\_mess = |Двойной клик: Строка: { lv\_row } колонка { lv\_column }|.  
     MESSAGE 'lv\_mess' TYPE 'I'.  
     go\_auto2->select\_table\_buyers( EXPORTING e\_row = e\_row ).  
     go\_auto2->show\_alv( ).  
   ENDMETHOD.

Программа состоит из двух экранов: на первом отражена возможность добавления автомобилей в таблицу, а на втором сменяемость таблиц автомобилей и покупателей с помощью двойного клика по полю таблицы.

На рисунке 6 отражен функционал первого экрана.



*Рисунок 11 – экран добавление записи в таблицу ZTRANSPORT.*

На листинге 2 представлен пример реализации экрана добавления записей в таблицу.

*Листинг 2 – добавление записей в таблицу*.

PROCESS BEFORE OUTPUT.  
 MODULE STATUS\_0100.

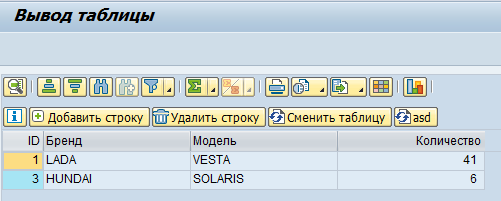
MODULE status\_0100 OUTPUT.  
    SET PF-STATUS 'STATUS\_100'.  
   SET TITLEBAR 'TITLE\_100'.  
   CREATE OBJECT go\_auto.  
   go\_auto->run( ).  
 ENDMODULE.

PROCESS AFTER INPUT.  
 MODULE USER\_COMMAND\_0100\_EC at EXIT-COMMAND.  
 MODULE USER\_COMMAND\_0100.

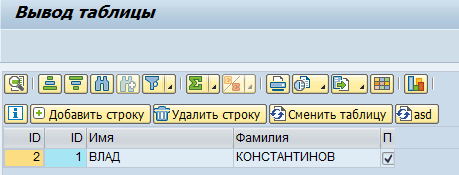
MODULE user\_command\_0100\_ec INPUT.  
   CASE ok\_code.  
     WHEN 'BACK'.  
       LEAVE TO SCREEN 0.  
     WHEN 'EXIT'.  
       LEAVE PROGRAM.  
   ENDCASE.  
 ENDMODULE.

MODULE user\_command\_0200\_ec INPUT.  
   CASE ok\_code.  
     WHEN 'BACK'.  
       LEAVE TO SCREEN 0.  
     WHEN 'EXIT'.  
       LEAVE PROGRAM.  
   ENDCASE.  
 ENDMODULE.

На рисунке 7, 8 отображена реализация второго экрана.



*Рисунок 12 – вывод таблицы автомобилей*

**

*Рисунок 13 – вывод таблицы покупателей.*

При двойном клике на поле таблицы автомобилей открывается таблица покупателей. В таблице покупателей отражается вся необходимая информация (id\_product указывает на тип автомобиля), а галочка в последнем поле указывает на то, что автомобиль был куплен.

На листинге 3 представлен пример работы экрана вывода таблиц.

*Листинг 3 – вывод таблиц*

PROCESS BEFORE OUTPUT.  
   MODULE status\_0200.  
   MODULE show\_alv.

MODULE status\_0200 OUTPUT.  
   SET PF-STATUS 'STATUS\_200'.  
   SET TITLEBAR 'TITLE\_200'.  
 ENDMODULE.

MODULE show\_alv OUTPUT.  
   go\_auto->show\_alv( ).  
 ENDMODULE.

MODULE user\_command\_0200\_ec INPUT.  
   CASE ok\_code.  
     WHEN 'BACK'.  
       LEAVE TO SCREEN 0.  
     WHEN 'EXIT'.  
       LEAVE PROGRAM.  
   ENDCASE.  
 ENDMODULE.

**Вывод:**

В конечном счёте создан программный продукт на языке программирования Abap, с использованием функциональных модулей языка и таблиц бд.

**ЗАКЛЮЧЕНИЕ**

В процессе выполнения практического проекта были использованы среда разработки Sap, язык программирования ABAP.

Были выполнены следующие задачи: проведён анализ предметной области, разработка визуализации программы, реализация программы для работы с таблицами баз данных.

Цель данной практической работы, а именно, разработка программного продукта для работы с таблицами баз данных (система учёта продаж автомобильного салона) была достигнута.

**СПИСОК ИСТОЧНИКОВ**

1. Кале Вивек. Внедрение SAP R/3: Руководство для менеджеров и инженеров. 2006 – 505 c.

2. BC400 – Введение в ABAP – инструментальные средства. SAP NetWeaver. 2005 – 361 c.

3. BC401 – ABAP Объекты. SAP NetWeaver. 2009 – 448 c.

4.Хомоненко А. Д. Базы данных: Учебник для высших учебных заведений/  В.М. Цыганков, М. Г. Мальцев, под ред. проф. А. Д. Хомоненко— СПб.:  КОРОНА принт, 2002 — 672 с.

5.Дейт К. Дж. Введение в системы баз данных, 8-е издание.: Пер. с англ. — М.: Издательский дом “Вильямс”, 2005. — 1328 с.

**ПРИЛОЖЕНИЕ**

Листинг 1

## REPORT zclass\_on\_class. TABLES: sscrfields. \*CLASS lcl\_handler DEFINITION DEFERRED. DATA:   gs\_zmotorcycle  TYPE ztransport,   go\_cont         TYPE REF TO cl\_gui\_custom\_container,   go\_alv          TYPE REF TO cl\_gui\_alv\_grid,   gt\_ztransport   TYPE TABLE OF ztransport,   gs\_transport    TYPE ztransport,   go\_event        TYPE REF TO z\_cl\_handler\_n,   go\_ref\_table    TYPE REF TO data,   go\_ref\_tab\_name TYPE REF TO data,   gt\_bsis         TYPE TABLE OF zbuyers. SELECT-OPTIONS: so\_model FOR gs\_transport-model NO INTERVALS NO-EXTENSION,                 so\_brand FOR gs\_transport-brand NO INTERVALS NO-EXTENSION. DATA: ls\_table TYPE ztransport. DATA: go\_auto      TYPE REF TO zcl\_motorcycle,       is\_transport TYPE  ztransport. DATA : ok\_code TYPE sy-ucomm. INITIALIZATION.   SELECTION-SCREEN FUNCTION KEY 1.   DATA: gv\_funk TYPE smp\_dyntxt.   sscrfields-functxt\_01 = 'Добавить запись'. AT SELECTION-SCREEN OUTPUT. AT SELECTION-SCREEN.   IF sy-ucomm <> 'CMD' .

## CASE zcl\_motorcycle=>set\_select\_param(        EXPORTING        ir\_model = so\_model[]        ir\_brand = so\_brand[] ) .       WHEN 'E'.         MESSAGE 'Emp' TYPE 'I'.       WHEN 'N'.         MESSAGE 'UcI' TYPE 'I'.     ENDCASE.   ENDIF.   CASE sy-ucomm.     WHEN 'FC01'.       CALL SCREEN 100."Экран добавления записи   ENDCASE. START-OF-SELECTION.   CREATE OBJECT go\_auto.   go\_auto->run( ).   gt\_ztransport = go\_auto->get\_data( ).   go\_auto->set\_object( EXPORTING io\_auto = go\_auto ).   CALL SCREEN 200. \*&---------------------------------------------------------------------\* \*&      Module  STATUS\_0100  OUTPUT \*&---------------------------------------------------------------------\* \*       text \*----------------------------------------------------------------------\* MODULE status\_0100 OUTPUT.   SET PF-STATUS 'STATUS\_100'.   SET TITLEBAR 'TITLE\_100'.   CREATE OBJECT go\_auto.   go\_auto->run( ). ENDMODULE.

## \*&---------------------------------------------------------------------\* \*&      Module  USER\_COMMAND\_0100  INPUT \*&---------------------------------------------------------------------\* \*       text \*----------------------------------------------------------------------\* MODULE user\_command\_0100 INPUT.   zcl\_motorcycle=>write\_to\_bd( gs\_zmotorcycle ). ENDMODULE. \*&---------------------------------------------------------------------\* \*&      Module  STATUS\_0200  OUTPUT \*&---------------------------------------------------------------------\* \*       text \*----------------------------------------------------------------------\* MODULE status\_0200 OUTPUT.   SET PF-STATUS 'STATUS\_200'.   SET TITLEBAR 'TITLE\_200'. ENDMODULE. \*&---------------------------------------------------------------------\* \*&      Module  USER\_COMMAND\_0200  INPUT \*&---------------------------------------------------------------------\* \*       text \*----------------------------------------------------------------------\* MODULE user\_command\_0200 INPUT. ENDMODULE. \*&---------------------------------------------------------------------\* \*&      Module  USER\_COMMAND\_0100\_EC  INPUT \*&---------------------------------------------------------------------\* \*       text \*----------------------------------------------------------------------\* MODULE user\_command\_0100\_ec INPUT.   CASE ok\_code.     WHEN 'BACK'.

LEAVE TO SCREEN 0.  
    WHEN 'EXIT'.  
      LEAVE PROGRAM.  
  ENDCASE.  
ENDMODULE.  
\*&---------------------------------------------------------------------\*  
\*&      Module  USER\_COMMAND\_0200\_EC  INPUT  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
MODULE user\_command\_0200\_ec INPUT.  
  CASE ok\_code.  
    WHEN 'BACK'.  
      LEAVE TO SCREEN 0.  
    WHEN 'EXIT'.  
      LEAVE PROGRAM.  
  ENDCASE.  
ENDMODULE.  
\*&---------------------------------------------------------------------\*  
\*&      Module  SHOW\_ALV  OUTPUT  
\*&---------------------------------------------------------------------\*  
\*       text  
\*----------------------------------------------------------------------\*  
MODULE show\_alv OUTPUT.  
  go\_auto->show\_alv( ).  
  
ENDMODULE.

## 