KOÇ UNIVERSITY
College of Engineering

DEPARTMENT OF COMPUTER ENGINEERING

COMP 291 SUMMER PRACTICES I

Mertcan AKARDERE

Internship Company and Department:
MBIS Danışmanlık
03.08.2015 – 28.08.2015

Supervisor: Selim Öndünç

TABLE OF CONTENTS

Table of Contents

1. Introduction

I did my internship at MBIS Danışmanlık. MBIS is a company that implements SAP's enterprise software for Turkish companies to speed their workflow up and make them more systematic by digitalizing everything. At the time of my internship, main focused SAP software was SAP R/3. That was not the latest SAP software, but many middle sized Turkish companies preferred this due to monetary limitation.

SAP R/3 is an enterprise software that incorporates database, tools to create scripts that edit and view the database, and various tools to produce neatly designed outputs to display data for the end-users. This means companies can use SAP R/3 all throughout their workflow to speed every step of the process up, and decrease response times since every data is presented in a single program.

SAP heavily encourages customization by allowing users to create new scripts that view, pull, edit or add data from the database in many ways and add an interface for these scripts so they can be easily used by every employee of the company. It even allows adding new buttons with custom codes and images to create new look for certain windows in the database. It also allows creating new variable types to minimize the risk of faulty inputs to the database. And it also has a support for multi-language programs by allowing creating strings for multiple languages.

My task in this internship was learning how to customize a SAP program to satisfy the demands of a customer company by doing previous tasks that were given to MBIS Danışmanlık. These tasks included creating different employee types and an interface to add new employee data, showing an invoice in SmartForms and creating an outline for shipping information.

SAP can be customized by using the ABAP language which SAP R/3 natively supports. Therefore I made use of ABAP language to accomplish these tasks. ABAP is a programming language that both support interacting with the database and the forms and the interface within SAP. The codes that interact with the database are similar to programming with SQL and the codes that interact with the interface are similar to programming with C# in Microsoft Visual Studio. Thus I mainly made use of ABAP language to do my queries in the database and create scripts that interact with the forms and buttons I created. I also had to learn how to create an easy to learn and use interface within SAP R/3 to make forms that could be used by every employee of a customer company. This process was similar to web design with PHP. I also had to learn creating SmartForms. SmartForms is a way of displaying data neatly on SAP which is also printable so it is crucial for adapting to uses in the office environment, and it is also heavily customizable with scripts that can adapt the formation therefore making it simple to create an outline for every situation. Designing the outline of SmartForms was similar to Microsoft Visual Studio and creating scripings within SmartForms was like making pseudo-code.

I decided to work with MBIS due to my desire to work with a SAP product. SAP products are based on abstract structures yet it has modules to specialize on every need of companies and all the modules can be customized for all business fields. This structure makes the software very malleable to fit the needs of every company. Thus it can be incorporated by almost any company. This can be observable by the stats; 87% of the Forbes Global 2000 companies use SAP (1). I consider working for SAP as one of my career goals; therefore it was a great opportunity to have a SAP experience in a work environment.

2. Company Description

MBIS Danışmanlık is a company that focuses on implementing and keeps supporting the ERP software, databases and servers for mainly Turkish companies. Their main focus is implementing SAP products as they were found by SAP consultants and they are gold partners with SAP.

MBIS Danışmanlık expanded their services by developing new software products that would help companies in their workflow. These products include; generating online invoices to reduce the cost of delivering physical invoices, a system to keep track of projects that is cloud based, a mobile confirmation system for SAP which makes confirming transactions by manager faster by not requiring them to log in to the SAP server, system to keep track of imports, weighing scale that directly connects to SAP server and many other software products that increase the utilities of SAP products.

MBIS developers are divided to modular developers and ABAP programmers. ABAP programmers write the main components of functions while modular developers fit these functions and prepare the database for customers. I worked on ABAP programming to perform previous customer requests of the company to learn the ropes with a very realistic experience.

MBIS currently has about 150 employees. In 17 years they managed to conclude 500 projects whilst serving 350 different customers and get recognition in Turkey and internationally.

3. ABAP Programming in SAP

3.1.Problem Statement

I had several tasks that were given to me to complete during my internship in MBIS Danışmanlık. Some of these tasks were given as a demand to MBIS Danışmanlık by their customer companies, as one of their responsibilities is keeping the support of the SAP products that they had implemented. I had to provide solutions and outputs that would satisfy the customers to practice my knowledge on SAP in an as close to the real work experience environment as possible. During my internship, I tried to develop interfaces that would not confuse the end-user but still the focus was more on the accuracy of the data than the visual appeal.

My main tasks included doing queries to pull data from the database, displaying them on an ALV format, creating interface for end-users to interact with the database, creating structures, tables, and data types and displaying information in SmartForms. All my tasks were completed by myself with the exception of little guidance from my supervisor.

My projects were considered done only when I had successfully displayed desired outputs accurately and with an efficient code.

3.2.Tools and Techniques Used

Natively ABAP programming language is used within the SAP R/3 system. ABAP is a relational database management system like SQL, but it was designed specifically to perform report creation. Relational database is very beneficial for the task of outputting reports as all the data stored is usually connected. Although there are some abstractions, it still feels outdated to current programming languages by forcing the defining the type while defining.

SAP R/3 was used as the main software. All of the programming, database interactions and testing were accessible via the SAP R/3. This software is a complete pack both for developer and the end-user. This limits the errors that can happen during transitions. This was a quite an old SAP software, thus the interface was old and not end-user friendly; it required prior knowledge about the program to be used effectively.

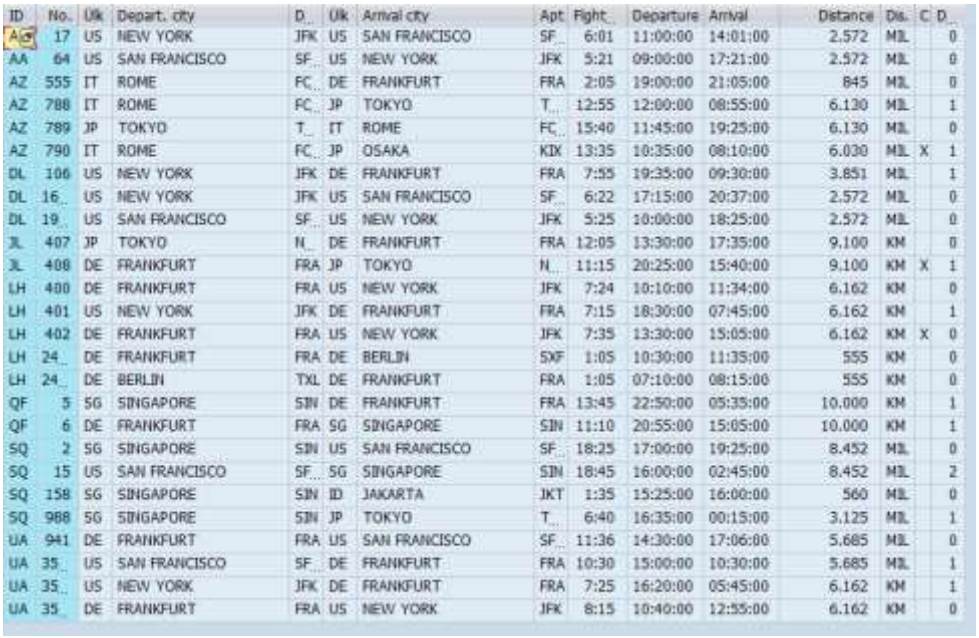
I used a laptop provided by the place I worked as an intern. The software was installed as it was the only requirement to run the program. I also connected and used the practice server within the company.

SAP R/3 is natively incorporates a relational database design. Thus I used ABAP language to interact with this database via the use of relational database and query based language. This allowed displaying tables of data gathered by database with ease as it provides functions to pull data from the database into tables with set conditions. It also provides outputting formats like ALV and SmartForms to display the data in a neat fashion.

3.3.Detailed Explanation

My first project was a practice to learn variable types in ABAP, do simple queries in the database and then displaying them. I used DATA to declare variable and I used BEGIN and END to define multiple variable in a single DATA. I could define variable types by specifying them like coding it as “TYPE i” which would make the variable integer. But when dealing with table from the database I used LIKE and the field name. This would ensure the variable type is exactly the same without needing to check, and it would also ensure longevity of the program as it would adapt to the changes of the field’s data type, and also it creates an abstraction barrier in declaring the type which is always a good programming practice.

After practicing with variable, I did some queries to pull data from the practice database (Figure 1). I used the same keywords in the SQL, as I was able to pull all the fields in the spfli table by using SELECT * FROM spfli INTO TABLE it_spfli. it_spfli was an internal table I previously declared as the type spfli by using the keyword TYPE TABLE OF to easily have a matching type fields.



ID	No.	Clk	Depart. city	D...	Clk	Arrival city	Apt	Flight...	Departure	Arrival	Distance	Dis...	C	D...
AA	17	US	NEW YORK	JFK	US	SAN FRANCISCO	SF...	6:01	11:00:00	14:01:00	2.572	MIL		0
AA	64	US	SAN FRANCISCO	SF...	US	NEW YORK	JFK	5:21	09:00:00	17:21:00	2.572	MIL		0
AZ	555	IT	ROME	FC...	DE	FRANKFURT	FRA	2:05	19:00:00	21:05:00	845	MIL		0
AZ	788	IT	ROME	FC...	JP	TOKYO	T...	12:55	12:00:00	08:55:00	6.130	MIL		1
AZ	789	JP	TOKYO	T...	IT	ROME	FC...	15:40	11:45:00	19:25:00	6.130	MIL		0
AZ	790	IT	ROME	FC...	JP	OSAKA	KIX	13:35	10:35:00	08:10:00	6.030	MIL	X	1
DL	106	US	NEW YORK	JFK	DE	FRANKFURT	FRA	7:55	19:35:00	09:30:00	3.851	MIL		1
DL	16...	US	NEW YORK	JFK	US	SAN FRANCISCO	SF...	6:22	17:15:00	20:37:00	2.572	MIL		0
DL	19...	US	SAN FRANCISCO	SF...	US	NEW YORK	JFK	5:25	10:00:00	18:25:00	2.572	MIL		0
JL	407	JP	TOKYO	N...	DE	FRANKFURT	FRA	12:05	13:30:00	17:35:00	9.100	KM		0
JL	408	DE	FRANKFURT	FRA	JP	TOKYO	N...	11:15	20:25:00	15:40:00	9.100	KM	X	1
LH	400	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:24	10:10:00	11:34:00	6.162	KM		0
LH	401	US	NEW YORK	JFK	DE	FRANKFURT	FRA	7:15	18:30:00	07:45:00	6.162	KM		1
LH	402	DE	FRANKFURT	FRA	US	NEW YORK	JFK	7:35	13:30:00	15:05:00	6.162	KM	X	0
LH	24...	DE	FRANKFURT	FRA	DE	BERLIN	SXF	1:05	10:30:00	11:35:00	555	KM		0
LH	24...	DE	BERLIN	TXL	DE	FRANKFURT	FRA	1:05	07:10:00	08:15:00	555	KM		0
QF	5	SG	SINGAPORE	SIN	DE	FRANKFURT	FRA	13:45	22:50:00	05:35:00	10.000	KM		1
QF	6	DE	FRANKFURT	FRA	SG	SINGAPORE	SIN	11:10	20:55:00	15:05:00	10.000	KM		1
SQ	2	SG	SINGAPORE	SIN	US	SAN FRANCISCO	SF...	18:25	17:00:00	19:25:00	8.452	MIL		0
SQ	15	US	SAN FRANCISCO	SF...	SG	SINGAPORE	SIN	18:45	16:00:00	02:45:00	8.452	MIL		2
SQ	158	SG	SINGAPORE	SIN	ID	JAKARTA	JKT	1:35	15:25:00	16:00:00	560	MIL		0
SQ	988	SG	SINGAPORE	SIN	JP	TOKYO	T...	6:40	16:35:00	00:15:00	3.125	MIL		1
UA	941	DE	FRANKFURT	FRA	US	SAN FRANCISCO	SF...	11:36	14:30:00	17:06:00	5.685	MIL		0
UA	35...	US	SAN FRANCISCO	SF...	DE	FRANKFURT	FRA	10:30	15:00:00	10:30:00	5.685	MIL		1
UA	35...	US	NEW YORK	JFK	DE	FRANKFURT	FRA	7:25	16:20:00	05:45:00	6.162	KM		1
UA	35...	DE	FRANKFURT	FRA	US	NEW YORK	JFK	8:15	10:40:00	12:55:00	6.162	KM		0

Figure 1 – Displaying an example table from the practice database in an ALV format.

My second project was creating an interactive program that took an input from the user and did a query based on that input. First I had to create input textboxes so that user could enter their parameters. For that I used SELECTION-SCREEN to define a layout then give it a name with WITH FRAME TITLE. Then I had to define what boxes would there be available for an input. For that I had to use PARAMETERS and then write the names of the variables and define their types (Figure 2). Again for their types I used LIKE for the same reasons.

```
26    END OF gs_siparis.
27
28  SELECTION-SCREEN BEGIN OF BLOCK B1 WITH FRAME TITLE text_t01.
29
30    SELECT-OPTIONS: S_ERDAT for vbak-erdat.
31    PARAMETERS: P_AUART like vbak-auart,
32                P_VKORG like vbak-vkorg,
33                P_VTWEG like vbak-vtweg.
34
35
36    SELECTION-SCREEN END OF BLOCK B1.
37    *erdat in s_erdat AND auart EQ p_auart AND vkorg EQ p_vkorg AND
38
39    WRITE: /(10) 'SatışBelg.', (12)'Yaratan Kul.', (10)'Belge Trh.', (4)'Tür', (4)'Org.', (21)'Net Değer' CENTERED,
40     (5)'PBirm', (6) 'Kalem', (18) 'Malzeme' CENTERED, (10) 'Parti', (17) 'Hedef Miktarı' CENTERED, (3) 'ÖB.'.
41
42    START-OF-SELECTION.
```

Figure 2 – Creating a simple form that takes 4 inputs from the user. Title is text_t01, as setting it as a separate string allows easier translation in multi-language programs.

Then I used the variables I set for the SELECTION-SCREEN as conditions for the database query by using the WHERE keyword (Figure 3). INNER JOIN combines the two tables in the database whenever the matching fields of the both tables selected with ON keyword have the same value.

```
SELECT a~vbeln a~ernam a~audat a~erdat a~auart a~vkorg a~vtweg a~netwr a~waerk b~posnr b~matnr b~charg b~zmeng b~zieme
  FROM vbak as a INNER JOIN vbap as b
    ON a~vbeln eq b~vbeln
  INTO CORRESPONDING FIELDS OF gs_siparis
 WHERE a~erdat in s_erdat AND a~auart EQ p_auart AND a~vkorg EQ p_vkorg AND a~vtweg EQ p_vtweg.
 WRITE:/ gs_siparis-vbeln, gs_siparis-ernam, gs_siparis-audat, gs_siparis-auart, gs_siparis-vkorg, gs_siparis-netwr, gs_sipari
  clear gs_siparis.
ENDSELECT.
```

Figure 3 – Code for doing a query on two tables from the database based on the inputs filled with a form by the user, then displaying the results.

SAP conveniently creates input textboxes that are best suited for the desired input since it knows the variable type from the declaration of the SELECTION-SCREEN. So dates are automatically converted to date format regardless of the notation of the user, and it can change the order of the day and month similar to language settings to be customized to fit every user in a company. After declaring these 4 inputs, SAP created a form and even length of the textboxes were at a reasonable size (Figure 4). Since my first input was for date, user could also select the date from a simple calendar interface thanks to SAP.

Figure 4 – The default selection-screen that SAP had created after it was given to create one with the given variable types. Default titles matched the variable names, the length and certain features of the textboxes were set by the SAP.

After I gave inputs that would get me a decent amount of results, I saw that I was successfully able to do a query based on the inputs I had given (Figure 5). The program I had created could be seen as a simple search program as the user had no way of doing any changes in the database but only view the data in the given tables.

Seçim Ekranı ve Ayrıntıların Yazımı

SatışBelg.	Yaratan Kul.	Belge Trh.	Tür	Org.	Net Değer	PBirs	Kalem	Malzeme	Parti	Hedef Miktarı	ÖB.
25	BPUSER	25.01.2011	TA	1100	2.377,05	USD	000010	02MS024MM7053EC		0,000	M
25	BPUSER	25.01.2011	TA	1100	2.377,05	USD	000020	02MA030VV6031AEC		0,000	M
25	BPUSER	25.01.2011	TA	1100	2.377,05	USD	000030	02TM05OB01006		0,000	ADT
25	BPUSER	25.01.2011	TA	1100	2.377,05	USD	000040	02TM12DK01001		0,000	ADT
25	BPUSER	25.01.2011	TA	1100	2.377,05	USD	000050	02TM01IS01009		0,000	ADT
25	BPUSER	25.01.2011	TA	1100	2.377,05	USD	000060	02TM01IS01018		0,000	ADT
25	BPUSER	25.01.2011	TA	1100	2.377,05	USD	000070	02TM865GD01542		0,000	ADT
25	BPUSER	25.01.2011	TA	1100	2.377,05	USD	000080	02TM865GD01138		0,000	ADT
26	BPUSER	25.01.2011	TA	1100	2.323,51	USD	000010	02MS024MM7053EC		0,000	M
26	BPUSER	25.01.2011	TA	1100	2.323,51	USD	000020	02MA030VV6031AEC		0,000	M
26	BPUSER	25.01.2011	TA	1100	2.323,51	USD	000030	02TM05OB01006		0,000	ADT
26	BPUSER	25.01.2011	TA	1100	2.323,51	USD	000040	02TM12DK01001		0,000	ADT
26	BPUSER	25.01.2011	TA	1100	2.323,51	USD	000050	02TM01IS01009		0,000	ADT
26	BPUSER	25.01.2011	TA	1100	2.323,51	USD	000060	02TM01IS01018		0,000	ADT
26	BPUSER	25.01.2011	TA	1100	2.323,51	USD	000070	02TM865GD01542		0,000	ADT
26	BPUSER	25.01.2011	TA	1100	2.323,51	USD	000080	02TM865GD01138		0,000	ADT
27	TEST1	27.01.2011	TA	1100	2.323,51	USD	000010	02MS024MM7053EC		0,000	M
27	TEST1	27.01.2011	TA	1100	2.323,51	USD	000020	02MA030VV6031AEC		0,000	M
27	TEST1	27.01.2011	TA	1100	2.323,51	USD	000030	02TM05OB01006		0,000	ADT
27	TEST1	27.01.2011	TA	1100	2.323,51	USD	000040	02TM12DK01001		0,000	ADT
27	TEST1	27.01.2011	TA	1100	2.323,51	USD	000050	02TM01IS01009		0,000	ADT
27	TEST1	27.01.2011	TA	1100	2.323,51	USD	000060	02TM01IS01018		0,000	ADT
27	TEST1	27.01.2011	TA	1100	2.323,51	USD	000070	02TM865GD01542		0,000	ADT
27	TEST1	27.01.2011	TA	1100	2.323,51	USD	000080	02TM865GD01138		0,000	ADT

Figure 5 – The example output for the project after a cerain inputs were given to the program. The titles I manually had written can be seen on top the variables.

Before moving on to the next project, I wanted to learn how to make the interface more end-user friendly. So I named every input textboxes with a more descriptive title to lead the users to entering the correct data on the form (Figure 6). I did this using the dictionary functionality of SAP, which meant the displayed titles of the textboxes could show different titles for users with different languages.

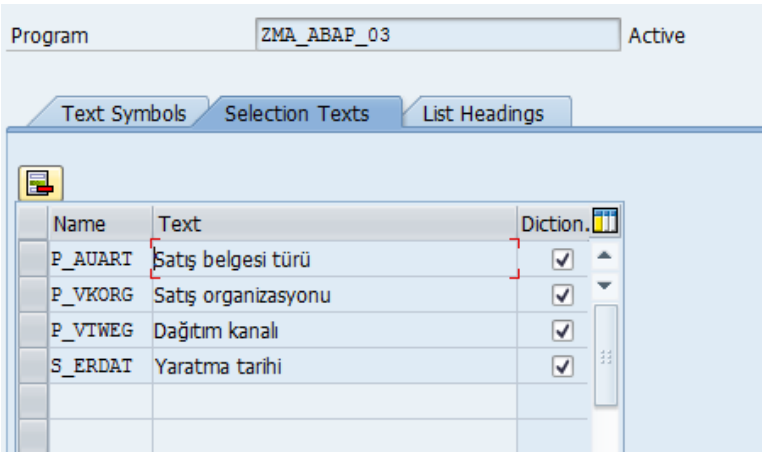
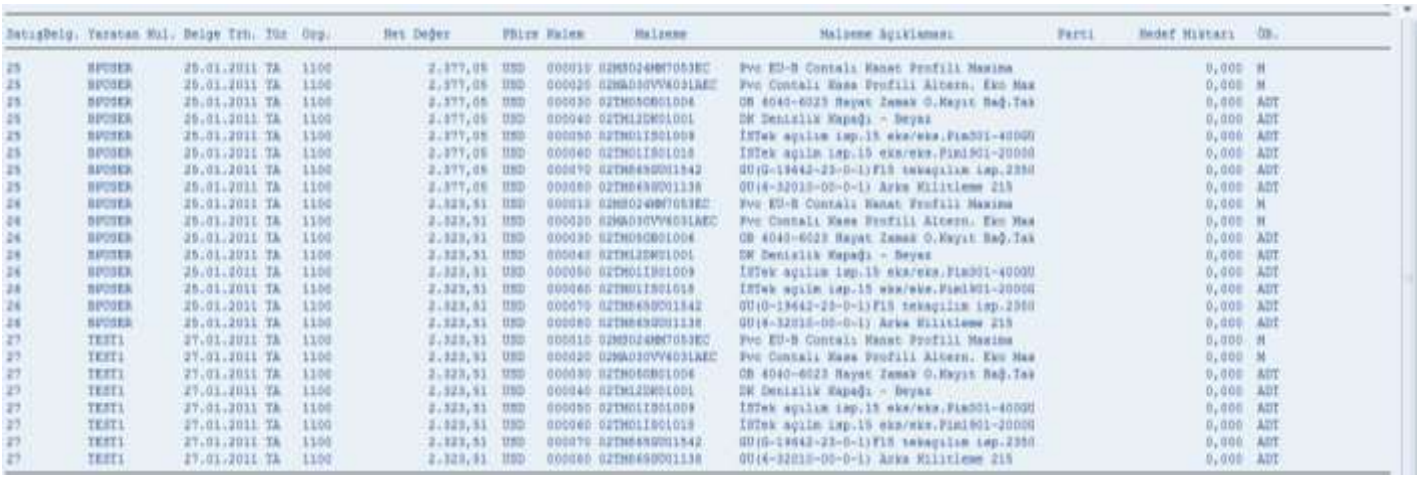


Figure 6 – Every input textbox of the selection-screen is set a title using the dictionary functionality to support developing a multi-language interface.

Lastly I had set title for every field in the output table in ALV format by hand. This method was tedious, bad at adapting to changes, and not scalable. Therefore I learnt how to show every field with its title at the top. After these visual changes, program looked more tidy and professional (Figure 7).



My third project was creating a raffle game where the user would pick from 4 different raffle games from radio buttons, and then a selection of columns to be played would be displayed to be picked by the user in a dropdown list. After selection the game and the amount, user can start the game and the randomly selected numbers for each game would be displayed as the winning numbers.

Thus I started by creating the interactive components of the program. I added

Figure 7 – The sample output for the program after the visual changes. The titles are given at the of every field and every field's length matches the data length of the field.

declaration (Figure 8).

Ideally I would create a single dropdown list and since each game allowed different amounts of columns to be played, and it should have changed the available options for the dropdown list. But initially I did not know how to do this, so I created 4 dropdown lists, one for each game, and I made the one visible that matched the selected game and made the other lists invisible.

```
51  *---------------------------------------------------------------------*
52  *Selection-Screen
53  *---------------------------------------------------------------------*
54
55  PARAMETERS: rb_game1 RADIOBUTTON GROUP a DEFAULT 'X' USER-COMMAND rb,
56              rb_game2 RADIOBUTTON GROUP a,
57              rb_game3 RADIOBUTTON GROUP a,
58              rb_game4 RADIOBUTTON GROUP a.
59
60
61
62  PARAMETERS: list1 TYPE c AS LISTBOX VISIBLE LENGTH 20
63  MODIF ID G1,
64
65  list2 TYPE c AS LISTBOX VISIBLE LENGTH 20
66  MODIF ID G2,
67
68  list3 TYPE c AS LISTBOX VISIBLE LENGTH 20
69  MODIF ID G3,
70
71  list4 TYPE c AS LISTBOX VISIBLE LENGTH 20
72  MODIF ID G4.
73
74
75
76  *---------------------------------------------------------------------*
77  *At Selection Screen
78  *---------------------------------------------------------------------*
```

Figure 8 – The code for creating 4 radio buttons where the first one is selected by default and 4 lists with type of character at 20 length.

Then I handled generating random numbers to determine winning numbers for the raffle games. The SAP had a function for generating random numbers so I made use of that function (Figure 9). I called this function with CALL FUNCTION keyword and repeated generating new numbers with the DO keyword. I kept a manual count on the number of generated random numbers since I would need to generate new numbers as long as the given number was already among the winning numbers of that column.

Initially I called this function 4 times, for each game, due to the differences in the game rules like the highest possible number in the game, but later I changed it by creating variables for the different game rules and change them whenever user selected a new game.

```
IF x_game EQ 1. " GAME TYPE
   do gwa_list1-key TIMES. " SELECTED COLUMN NUMBER
     ADD 1 TO X_COUNTER.
     z_output = x_counter.   CONDENSE z_output.
     CONCATENATE z_output  ' Kolon:' INTO z_output SEPARATED BY '.'.
     CONDENSE z_output.   WRITE: z_output.    CLEAR  z_output.
     DO. " REPEAT UNTIL STOPPED
       CALL FUNCTION 'QF05_RANDOM_INTEGER' "RANDOM GENERATOR
        EXPORTING
          RAN_INT_MAX         = 49
          RAN_INT_MIN         = 1
        IMPORTING
          RAN_INT             = x_luckynumber
        EXCEPTIONS
          INVALID_INPUT       = 1
          OTHERS              = 2
                .
       IF SY-SUBRC <> 0. ENDIF."successful
       gb_repeat = 'F'. " case of REPEATING NUMBER in the output
            LOOP AT gt_output INTO gwa_output.
                 IF gwa_output EQ x_luckynumber.
                   gb_repeat = 'T'. " ROLL AGAIN!
                 ENDIF.
            ENDLOOP.
       IF gb_repeat EQ 'F'.
           APPEND x_luckynumber TO gt_output. " add unique number to out put
           ADD 1 TO x_counter2.
              IF x_counter2 GE 6. " if done, stop rolling
                x_counter2 = 0.
                  EXIT.
              ENDIF.
       ENDIF.
       ENDDO.
SORT gt_output ASCENDING. " SORT THE OUTPUT
     LOOP AT gt_output INTO gwa_output.
       WRITE: gwa_output.
     ENDLOOP.
     clear gt_output.
     ULINE.
   ENDDO.
 ENDIF.

IF x_game EQ 2.
   do gwa_list2-key TIMES.
     ADD 1 TO X_COUNTER.
```

Figure 9 – Code for generating selected amount of columns game results for the game 1 where the maximum possible number is 49, and there are 6 winning numbers for each column. This code also make sure there are no repeating numbers in the winning numbers. Also winning numbers are in ascending order for visual appeal.

After my code was working accurately, I made it visually appealing for the end-users by adding name to every radio button and short summary of the game rules in the parenthesis (Figure 10). I also added a title for every winning number sequence in the output, and draw a line between each winning number to make it easier to read (Figure 11). The numbers were also in ascending order to make it easier for checking the winning numbers.

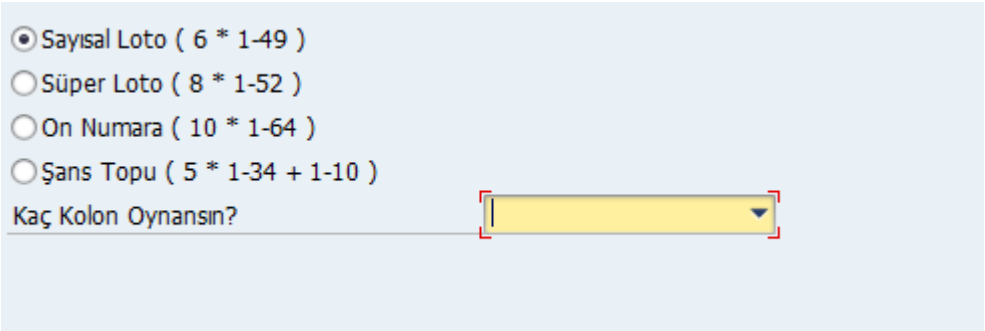


Figure 10 – First interface for the raffle game. Radio buttons have descriptive names but dropdown list doesn't have a default value.

Figure 11 – First output format for the raffle game. Numbers are in ascending order and have a line between them to make it easier to read the winning numbers.

My initial code for this project was very inefficient due to my limited knowledge on ABAP language. But my supervisor showed me how to update the values of a dropdown list, and gave me task to program this in a more efficient way.

Therefore I implemented a dropdown list that would update its contents based on the selected radio button. It would also choose the default value for the dropdown list based on the game.

To make the program more efficient, I also declared variable for the game rules, like the highest possible number and the amount of winning numbers (Figure 12). Thus I was able to use a single random number generator function by updating the game rules after a game was selected.

```
12  FORM SET_PM .
13
14      IF rb_game1 = 'X'.
15        x_dd = 8.
16        x_game = 1.
17        LIST = 8.
18      ELSEIF rb_game2 = 'X'.
19        x_dd = 4.
20        x_game = 2.
21        LIST = 4.
22      ELSEIF rb_game3 = 'X'.
23        x_dd = 6.
24        x_game = 3.
25        LIST = 6.
26      ELSEIF rb_game4 = 'X'.
27        x_dd = 4.
28        x_game = 4.
29        LIST = 4.
30      ENDIF.
31
32  ENDFORM.                    " SET_PM
```

Figure 12 – The game rules change based on the selected radio buttons.

My next project was creating 3 different employee types in the database with some similar and some different fields and even one field that had a custom designed variable type. As continuation of the project I had to create a form to create new employees and a view with extra features.

So I started by creating tables for each employee type and adding the desired fields for each of them (Figure 13). I also had to declare variable types for each of the fields. Most of these could be found within already defined variable types. But I was asked to create a specific type that could only take B or M as input, which stood for white collar or blue collar respectively, so I created a new variable type that allowed 1 char and its value range had B and M as fixed values. I also set their descriptions for future reference and instruction for the end-user in the later applications.

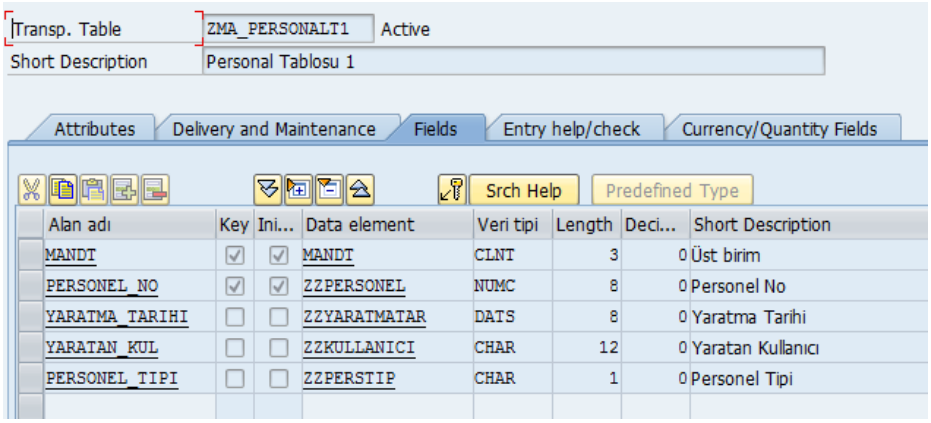


Figure 13 - A personal table with 5 fields with their data types and short descriptions.

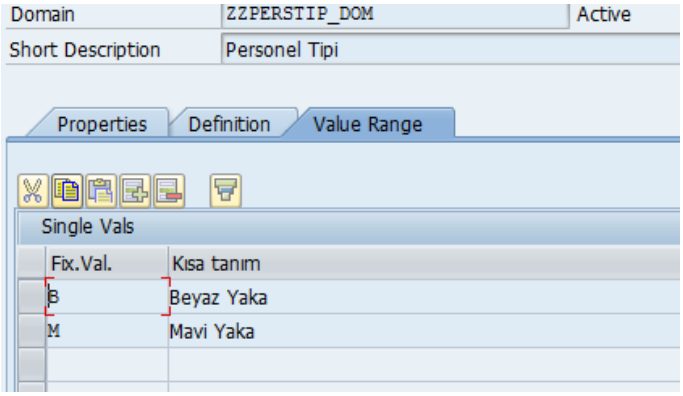


Figure 14 – Employee Type data type has B and M as its fixed value within its value range.

After I was done declaring the tables for all employee types, I created some example employee data in the database to test if everything was in order. Then I started working on a form that would allow creating any type of employee.

First I declared selection-screen with all available fields in the employee tables, but I made only the NOT NULL and KEY fields obligatory to fill in the form by using the OBLIGATORY keyword.

```
9   REPORT  ZMA_ABAP_ADDPERSONAL.
10
11   data: wa_pers1 type zma_personalt1.
12    data: wa_pers2 type zma_personalt2.
13     data: wa_pers3 type zma_personalt3.
14
15  SELECTION-SCREEN BEGIN OF BLOCK B1 WITH FRAME TITLE text-t01.
16
17    PARAMETERS: P_PERSNO TYPE ZZPERSONEL VALUE CHECK OBLIGATORY,
18  *              P_CREDAT TYPE ZZYARATMATAR OBLIGATORY, " sy-datum?
19  *              P_CRETOR TYPE ZZKULLANICI OBLIGATORY,
20                P_PERSTY TYPE ZZPERSTIP VALUE CHECK OBLIGATORY,
21                P_PNAME  TYPE ZZPERSADI VALUE CHECK,
22                P_PSNAME TYPE ZZPERSSOYADI VALUE CHECK,
23                P_PBIRTH TYPE ZZDOGUMTARIHI VALUE CHECK,
24                P_POS    TYPE ZZPOSIZYON VALUE CHECK,
25                P_YONNO  TYPE ZZPERSNOYONETICI VALUE CHECK.
26
27   SELECTION-SCREEN END OF BLOCK B1.
28
29  START-OF-SELECTION.
```

Figure 16 – The code fore creating a form to add new employee data to the database. Every value is checked to fit the field's data type and not null fields need to be filled. Otherwise user is informed of their misuse.

Since I had declared the data types for the selection-screen, SAP also had a feature to allow personal type to be selected with a pop-up since it had fixed values.

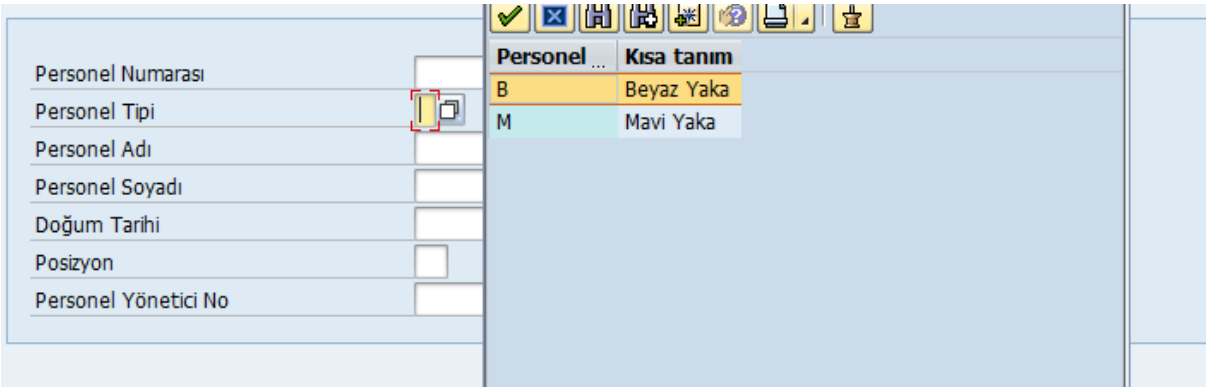


Figure 17 – Interface for creating a new employee data for the database is to the left. A pop-up that appears when user clicks on the button next to the personal type is to the right. The available choices and their description are presented which makes selecting the correct data for the database for the end-user is easier.

My next project was creating a view in an ALV with extra features. These features included creating a BOX field on the left of the table for making selection easier for the end-user when using the view, adding new buttons to the interface while using this particular ALV, creating hyperlinks within a field, and also adding a transition for double clicking on a certain field.

After writing the code for simple queries I started adding the codes for transactions. I checked these conditions by using the WHEN keyword then writing their conditions. For double clicking I used WHEN '&IC1' which works whenever user double clicks, then I checked the index of the field that was double clicked on, find its field name, then see if it matches VBELN, and if it matches perform the transition. Transition to another segment of SAP is done by the TRANSITION keyword and SKIP FIRST SCREEN basically performs pressing of the Return key after the page loads.

Then I created a custom button called MALZEMEHK. This button would make a transition to MM03 with the material that was selected when it was pressed to learn more about the material. WHEN 'MALZEMEHK' worked when the custom button was clicked.

```
152
153  CASE r_ucomm.
154
155      WHEN '&IC1'. " WHEN USER DOUBLE CLICKS
156        READ TABLE IT_INFO INDEX RS_SELFIELD-TABINDEX INTO WA_INFO.
157  * ---------- VBELN için HOTSPOT ----------
158        IF  rs_selfield-fieldname EQ 'VBELN'.
159           set PARAMETER ID 'VF' FIELD WA_INFO-VBELN.
160           call TRANSACTION 'VF03' AND SKIP FIRST SCREEN.
161        ENDIF.
162
163  * ---------- Malzeme Hakkında Butonu için Transaction ----------
164      WHEN 'MALZEMEHK'. " WHEN USER CLICKS CUSTOM MALZEMEHK BUTTON.
165        READ TABLE IT_INFO INDEX RS_SELFIELD-TABINDEX INTO WA_INFO.
166
167           set PARAMETER ID 'MAT' FIELD WA_INFO-matnr.
168           call TRANSACTION 'MM03'." AND SKIP FIRST SCREEN.
169
170      WHEN 'SMARTFR'. " WHEN USER CLICKS CUSTOM MALZEMEHK BUTTON.
171        READ TABLE IT_INFO INDEX RS_SELFIELD-TABINDEX INTO WA_INFO.
172
173           INCLUDE ZMA_ABAP_06_SF. "  ============================  SMART FORM OLUŞTURMA  ============
174  ENDCASE.
175
176    rs_selfield-refresh = 'X'.
177    rs_selfield-col_stable = 'X'.
178    rs_selfield-row_stable = 'X'.
179
```

Figure 18 – The code for transactions. These transactions occur when user double clicks on a vbeln field, malzemehk button. Both of these transitions take the value of the user's selected row's particular field to fill the textboxes that appear after the transition. There's also a button for creating a smartforms.

I created a field called BOX by adding a box fieldname in the formatting of the ALV. I also programmed the hotspot VBELN which means users can transition to details of document just by clicking on its number.

I also had a button for creating SmartForms. I created a SmartForm for this table where the only important fields would be displayed total sum of values would be printed at the top (Figure 19).

After buttons, the box, the ability to form SmartForms and the transitions were added, this procuded a pretty advanced output (Figure 20).

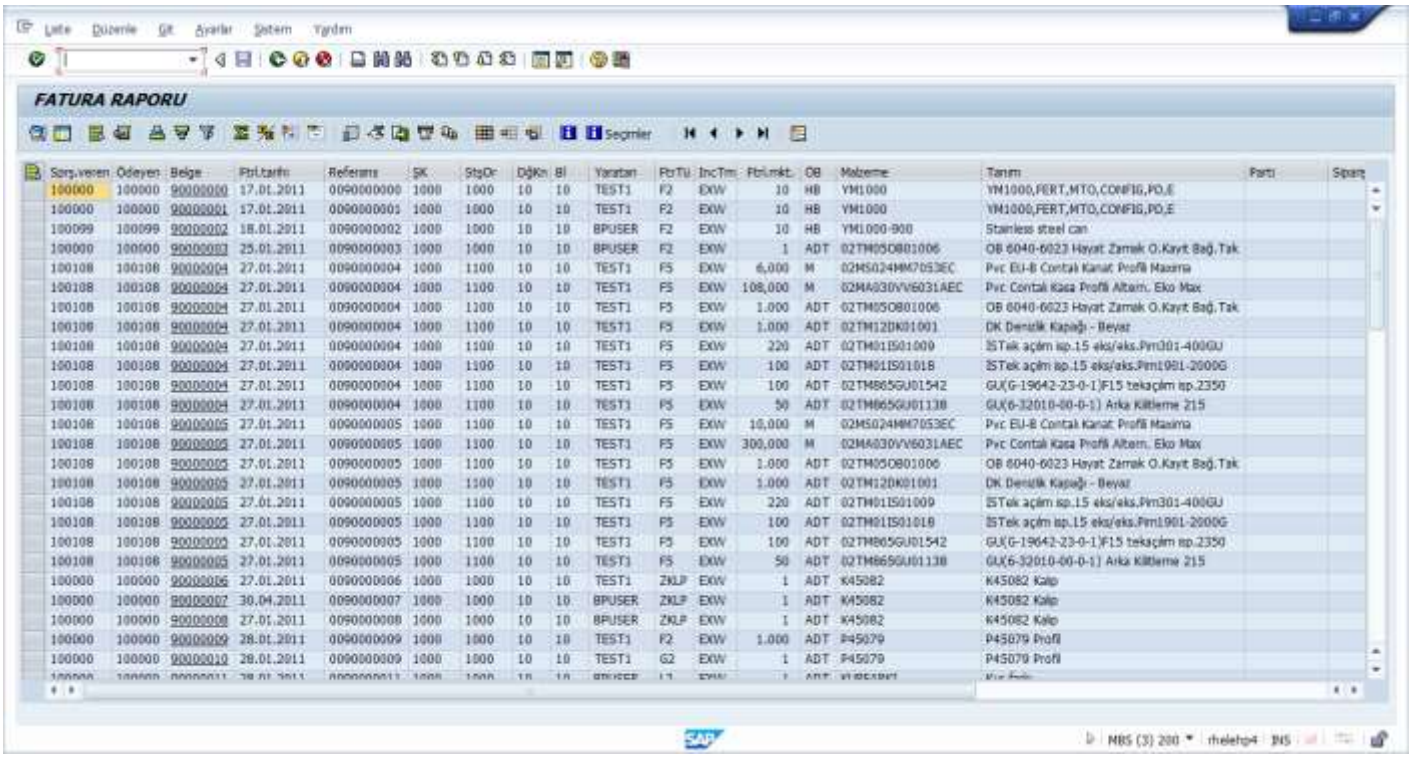


Figure 20 – Output ALV format with all of the features. Belge field is underlined because of the hotspot. The box is the leftmost field with no titles.

Tür: Spr#.proforma ftr.

Fatura No : 90000004
Odeyen : VIAS
Faturalama Tarihi: 27.01.2011

Net Deger : 522,61 USD

Kalem	Metin	Miktar	OB	Net Deger	Mazleme
10	Pvc EU-B Contalı Kanat Profili Maxima	6,000	M	10,82	02MS024MM7053EC
20	Pvc Contalı Kasa Profili Altern. Eko Max	108,000	M	99,79	02MA030VV6031AEC
30	OB 6040-6023 Hayat Zamak O.Kayıt Bağ.Tak	1.000	ADT	154,50	02TM05OB01006
40	DK Denizlik Kapaklı - Beyaz	1.000	ADT	257,50	02TM12DK01001
50	#S Tek açılım isp.15 eks/eks.Pim301-400GU	220	ADT	0,00	02TM01IS01009
60	#S Tek açılım isp.15	100	ADT	0,00	02TM01IS01018

Figure 19 – The SmartForm generated by the program that lists prodcuts, amount, cost and material. The details about the invoice and the total cost is also noted at the top of the form.

My last project was reaching the invoice table within the SAP database, and generating shipping information in SmartForms. This was a task given to MBIS Danışmanlık as a request in the past, so I was asked to satisfy these old demands.

First I created a new structure in the database, so I could easily pull all the relative fields from the invoice but not the others. Then I had to find the location of the invoice within the database. It was the LBBIL_INVOICE table that could get the invoice data and transfer it to SmartForms. After these steps, it only took simple queries to retrieve data.

I had to carefully organize everything to design the desired output since it required much different information to be displayed at different locations (Figure 21)

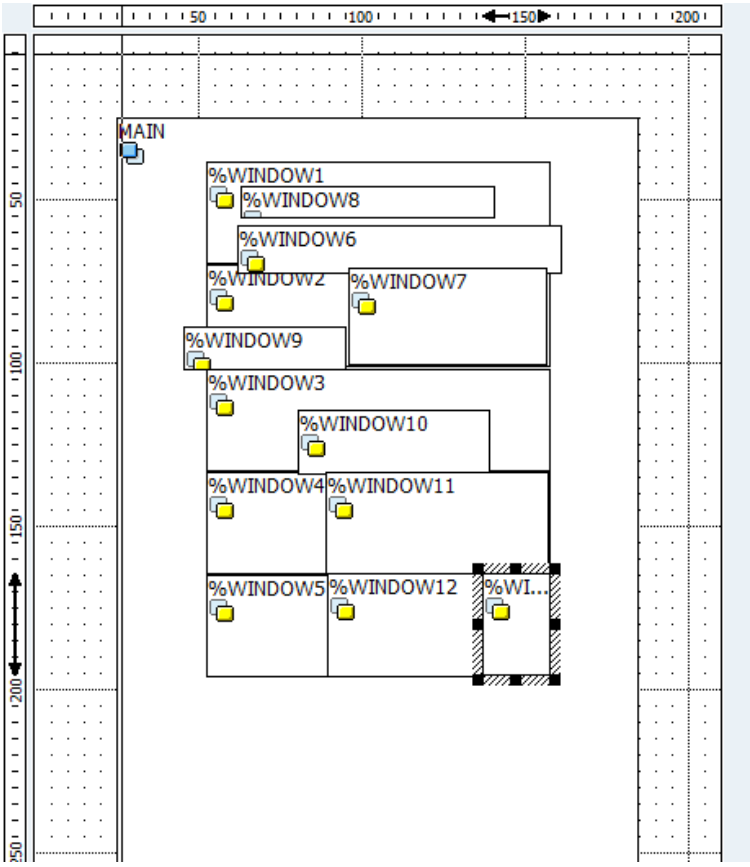


Figure 21 – Design windows of SmartForms. Most of the windows are there to position different texts since their location were given by the customer company.

One of the required information that needed to be displayed was barcodes. But to display barcodes I had to change writing type. But directly changing would make all the writing on the page in barcodes, therefore I had to create a new style that used barcodes as the writing type. Apparently there were many algorithms that formed barcodes, so I asked and learnt which one I should use for this project. It was one of the common barcode algorithms in Turkey. After I had 2 styles for this SmartForms, I only had to choose the second style for the windows that included barcode.

After displaying my query result in the corresponding windows and in the appropriate styles, I had made an output similar to the desired output by the customer company (Figure 22). I only had to make few more adjustments to make it look tidier.

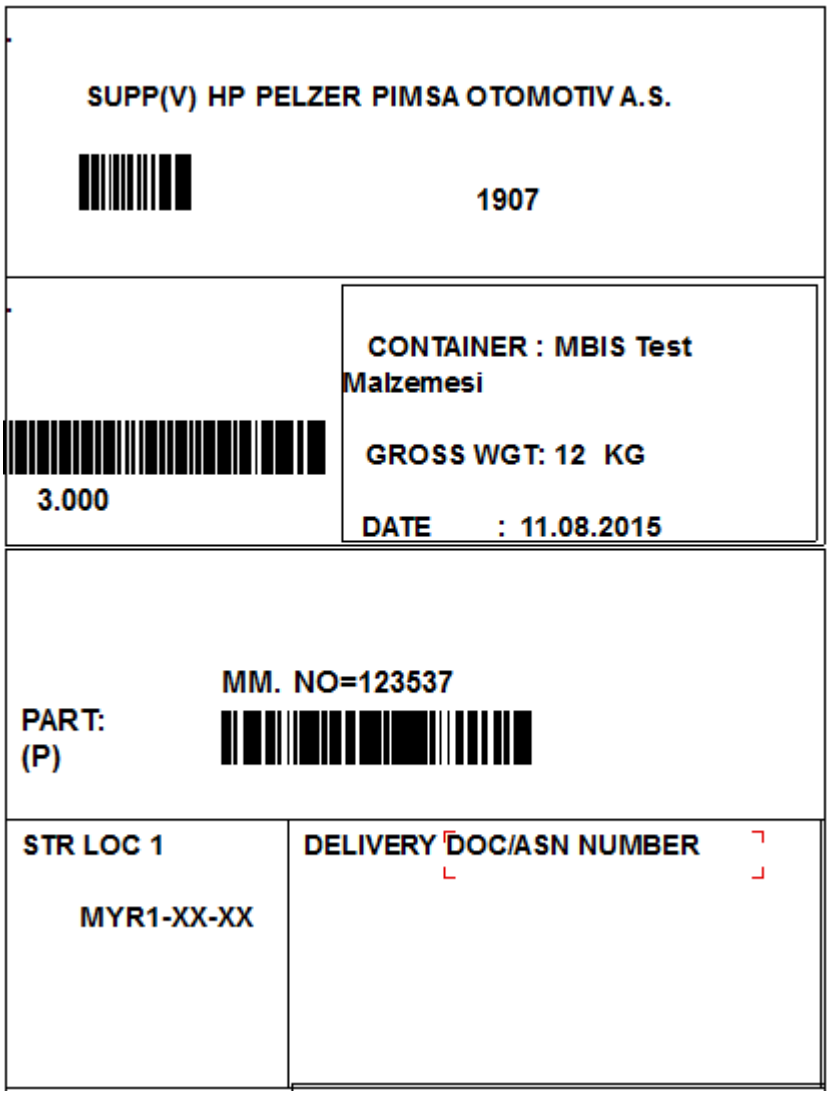


Figure 22 – Sample shipping information based on the data pulled by the SAP's invoice table. The formatting is done as requested by the customer company. Barcodes are also shown.

Finally I was also tasked with using the logic functionality within SmartForms to test my knowledge further. This logic functionality would make the form adapt to organize differently depending on the input therefore broadening its possible usages. First I did queries within SmartForms to define and fill some internal tables (Figure 23). Then I set few conditions and resulting text based on certain field's values (Figure 24).

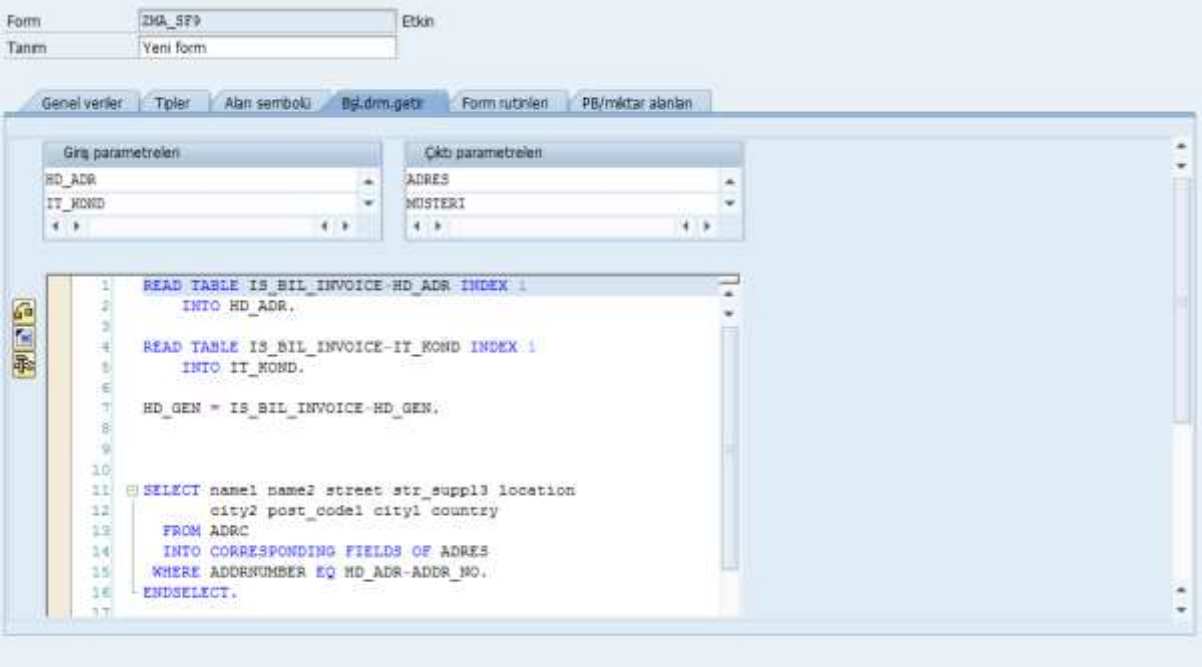


Figure 23 – Queries within SmartForms. The input and ouput parameters are set on the top. Output parameters can be referenced any other place in the SmartForms. Therefore I will be able to access data by sending an internal table as an output

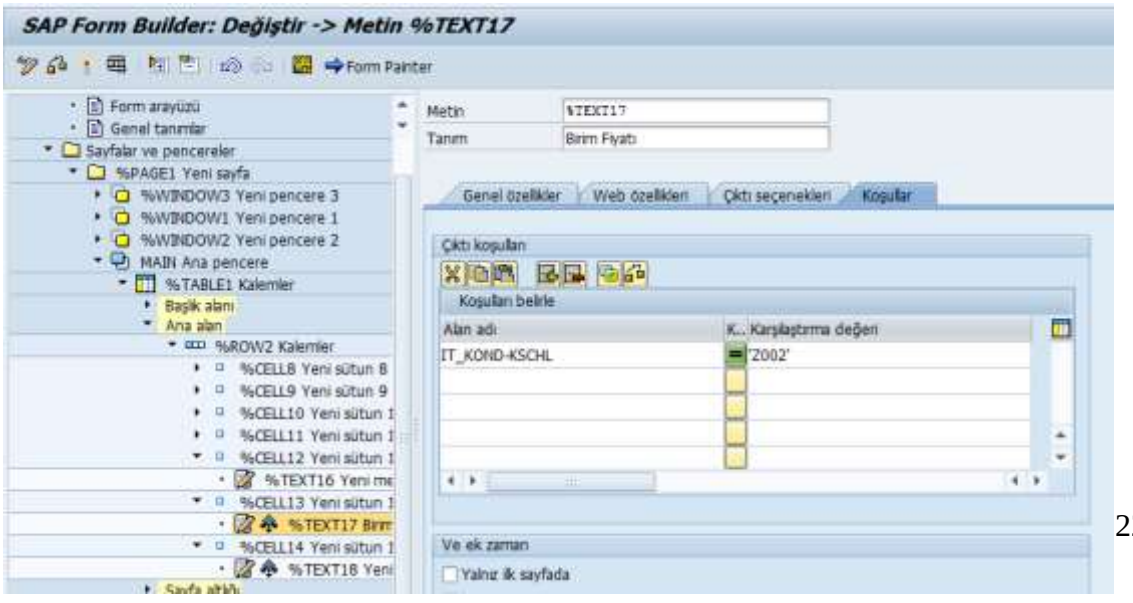


Figure 24 – Condition in SmartForm. It's condition that only return true whenever the field KSCHL field is equal to Z002. Whenever the condition holds true, the objects under the condition are visible.

I was also tasked with displaying sum of the costs of the materials within the invoice. I computed the sum within logic part of the SmartForms as it also allowed adding codes and queries within it.

Even though my result was accurate, the result was shown as an integer. Therefore I had to show these numbers in the currency format. I converted these numbers to currency by using the SAP function SPELL_AMOUNT again within the SmartForms coding part.

As a result I was able show the details from the query I did within SmartForms and display the sum of the bill in the currency format (Figure 25)

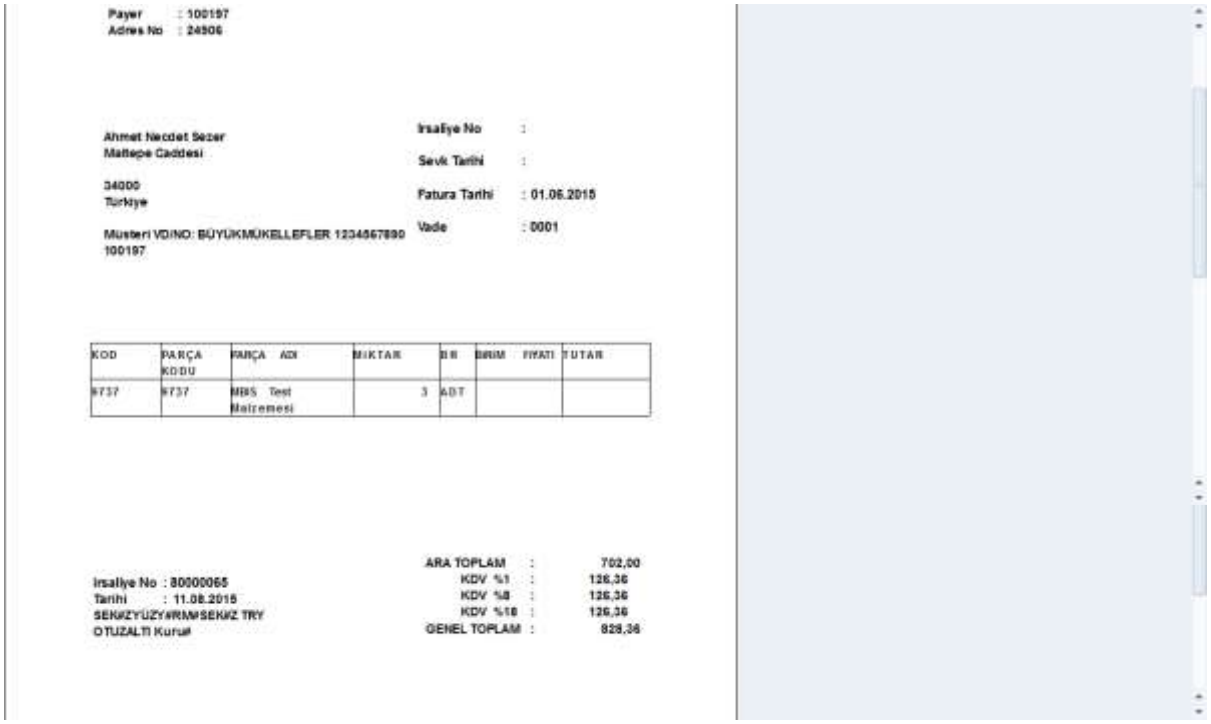


Figure 25 – The output billing information. The details are given at the top, the materials are listed in the middle, and the sum of costs are given at the bottom of the page.

3.4.Results

My main criteria for success were satisfying the demands for me with accurate information and efficient code. Although most applications had a visual aspect, it was not the main focus. I didn't move on to other projects without successfully completing the one, therefore I'd claim all my completed projects as a success. Because even when I failed to meet these conditions, like when I wrote an inefficient code for the raffle game with four dropdown lists, I started over to fix my mistakes and accomplish the task efficiently.

Since the visual appeal was not the focus, there were confusing parts for the end-user. There were many missing fields in the outputs since it was only a practice database. But I implemented every demanded feature for every project I had taken.

4. Conclusions

Since I was very familiar with programming concepts ABAP language was easy to learn for me. Especially COMP 202 course was helpful in my work since I knew concepts of data storing.

Similar to my in-class assignments; output formatting was specifically clear. Unlike to my in-class assignments, the way I solved assignment had no rules; I could any code or system that was provided by software. Since my tasks were not set after I learnt how to complete them, like in-class, I had to find information by asking for help from my colleagues or search the resources on the internet. This sometimes resulted in inefficient approaches to the problem, so later on we discussed other ways to solve them with my supervisor.

Unlike my classes in the university, we had to make end-user friendly executable projects so that a person without the knowledge of the SAP software could use the program. This taught me how to make easy to use interfaces and to include descriptive titles.

This was my first time using SQL type, query based language. I think this experience has given me immense insight on the subject of this practice as well as database management.

Before my internship I did not look bright to do database related work as I believed type definitions would be hard to manage. During my internship I have come to realize, defining by “Like” makes it easy to manage.

Since I have come to conclusion how SAP was malleable to all business fields, I look forward to using more SAP products as I love abstract designs that appeals to everything.

Appendix

(including relevant material such as catalogues, product specifications, papers, codes)

References

Each information, figure, table, etc. that does not belong to you (has been found online, taken from some other document, etc.) **must** be referenced, or you risk being penalized due to plagiarism.