From CPI to CPIX

Gerrit de Bolster, Statistics Netherlands

1. Abstract

For the data collection of the Consumer Price Index (CPI) a stand-alone Maniplus 4.8 application was developed in 2013. Based on a list of shops it uses several dialogs with lookup's and functionality to edit the price and weight or size (quantity) of a pre-defined set of products that can be found in the selected shop. As an experiment this Maniplus 4.8 application was converted to a Blaise 5 questionnaire (CPIX) to be used with the data entry program (DEP.exe). The extended and powerful functionality of Blaise 5 made it possible to support more or less the same functions available in Maniplus 4.8 in a Blaise 5 questionnaire. However, the layout was adapted for Blaise 5. The paper will show how the Maniplus 4.8 functions were converted to Blaise 5 fields and rules.

2. History

In June 2013 the office of Statistics Netherlands in the Caribbean Netherlands (the islands Bonaire, Saba and St. Eustatius) started to use Windows 8 tablets for its data collection by CAPI. The first survey was the “Omnibus” survey built in Blaise 4.8 by the Blaise team. It was quite successful and this success got known throughout the office. As a result the statistical department responsible for the CPI contacted the Blaise team and asked them to create also a Blaise solution for the CPI to be used on the tablets as the PAPI solution they were using until then was costing a lot of effort and error prone. The Blaise team agreed as in this way more knowledge about using Blaise for tablets was gathered and, because of the way the CPI was designed, a Maniplus application was created. Although several updates have been issued since its introduction at the end of 2013 the application is still working fine.

3. The CPI

The Consumer Price Index (CPI) is a well-known survey performed by most of the National Statistical Institutes. In Caribbean Netherlands it is a monthly survey. A small group of interviewers gets every month a list of shops and for each shop a list of products with quantities and prices as was collected the previous month. They have to check if the products are still available and if so if the price or quantity (kilo, liter, pieces, etc.) has changed. If so they have to report the changes. If the product is not available anymore they have to report this too and search for alternative and add it to the list. They also have to report if the shop is closed (permanent or temporarily). Every interviewer receives its own unique list.

4. Three different implementations

The Maniplus 4.8 application built by the Blaise team (and which is still in production) is a stand-alone implementation. The application is encapsulated by a shell (also built in Maniplus 4.8) which takes care of the export of completed monthly results and of the import of a new list of shops and products for the next month. As in 2014 a CAPI system called CAPI Logistics System (CPLS, presented during the IBUC 2015 in Beijing) came available the Maniplus stand-alone application was adapted to run with the CPLS. This implementation was never taken into production because there was no need for it as the original stand-alone implementation is still running fine.

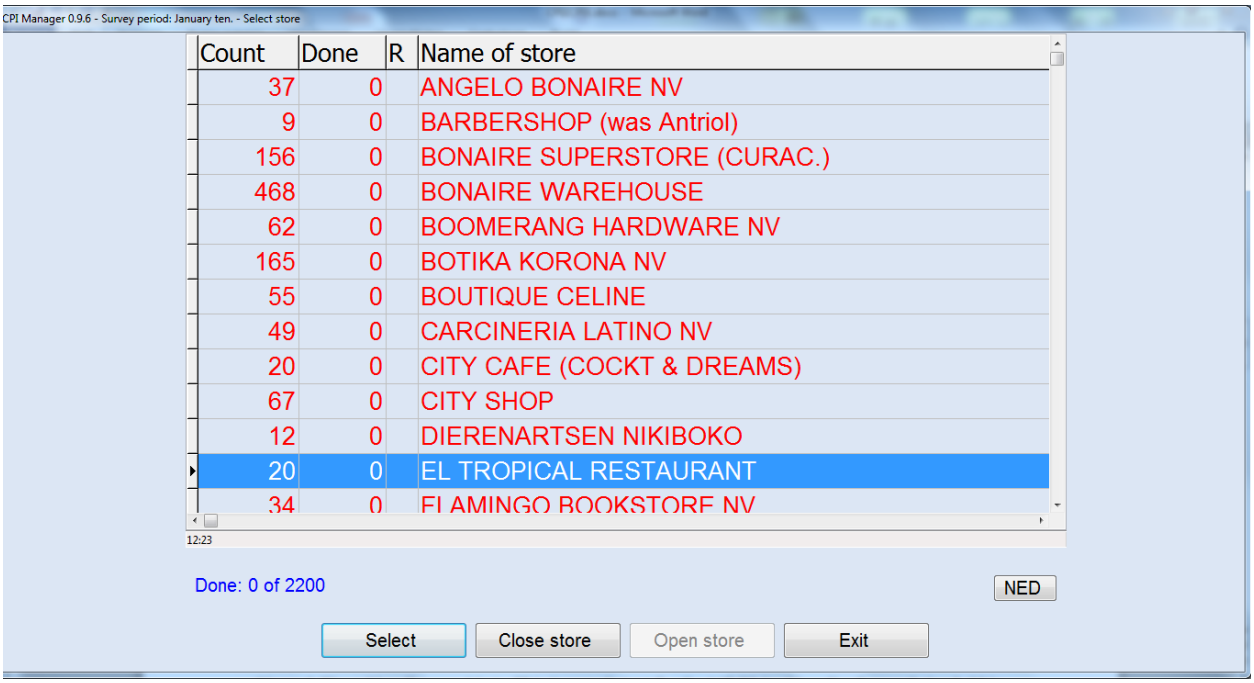
To learn more about Blaise 5 a third implementation was created: the CPIX. As Maniplus was not available in Blaise 5 the challenge was to build a Blaise 5 CAPI questionnaire with similar functionality as the Maniplus 4.8 applications.

5. The Maniplus 4.8 implementations

5.1 The stand-alone implementation

Starting the application from the shell a lookup appears with the list of shops and counters indicating the total number of products listed for that shop and how many products already had been completed (see fig 1). As the islands are relatively small the name of the shop is enough for the (experienced) interviewers to know which shop was meant, an address is not necessary. The application is bi-lingual (Dutch/English), the language can be switched by pressing a button on the bottom right hand corner.

Figure 1. The list of shops

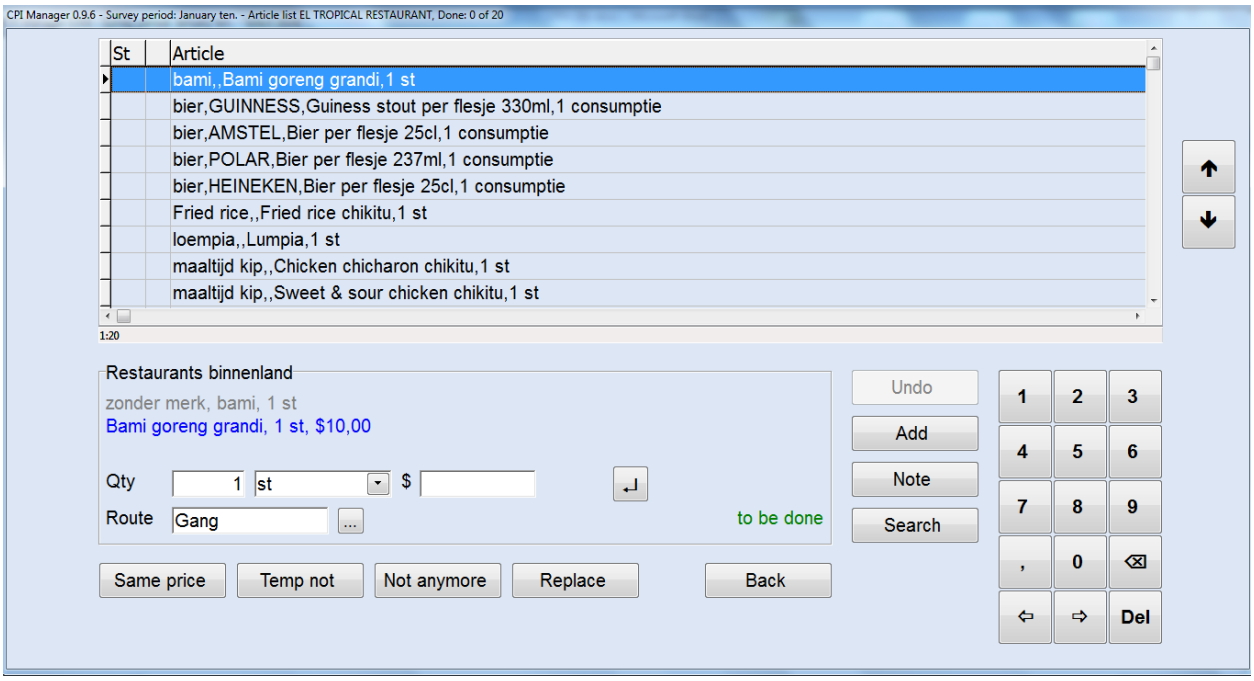


Using the buttons at the bottom of the screen the selected shop can be marked as “closed” or marked as “opened” again. To go to the list of the products of the focused shop the interviewer should click on the “Select” button.

The list of products (see fig.2) contains the type of product, the brand, the volume/weight and the price (in US $ being the currency of the islands). The first three are shown in a lookup in the dialog, the price is shown with more information below the lookup. The first column (marked “St”) will contain a symbol indicating the change in one of the properties of the product. The interviewer can use input lines or the buttons at the bottom of the screen to report the changes. It is also possible to add a product or make some notes.

The product list is sorted according to the best route through the shop. The interviewer can edit the location of the product if the shop has changed it. As it is running on a tablet without a separate keyboard a small numeric keyboard was added to the dialog.

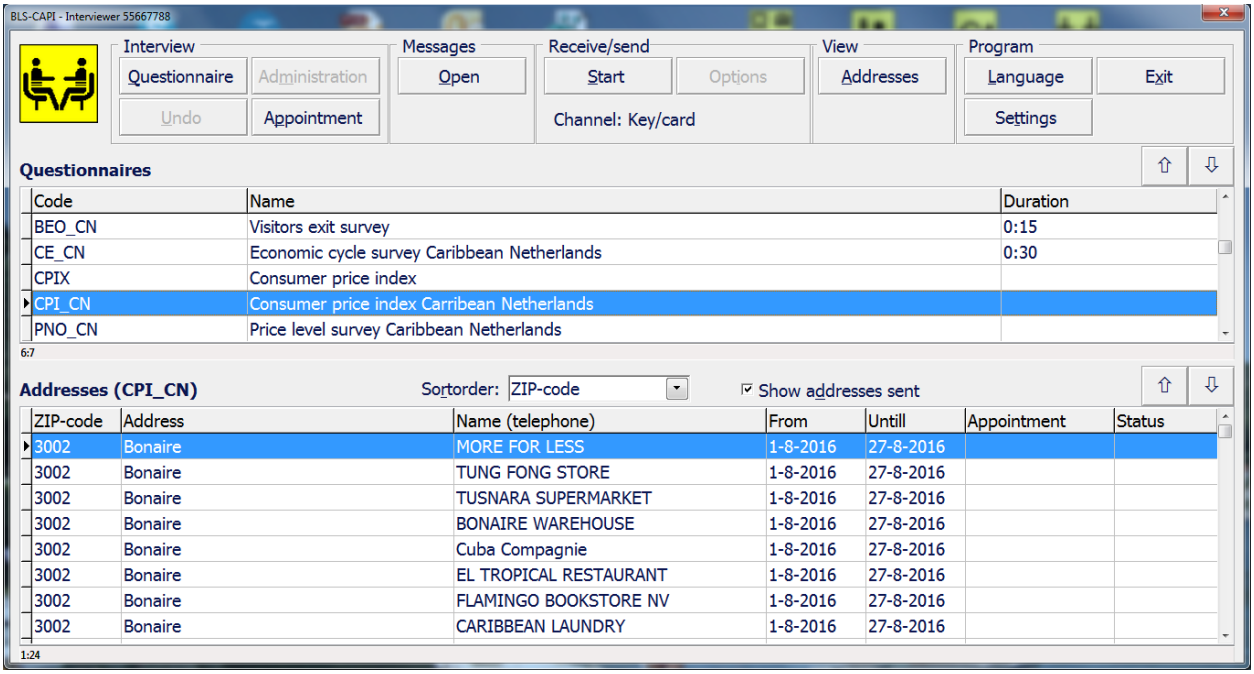
Figure 2. The list of products



5.2 The CPLS implementation

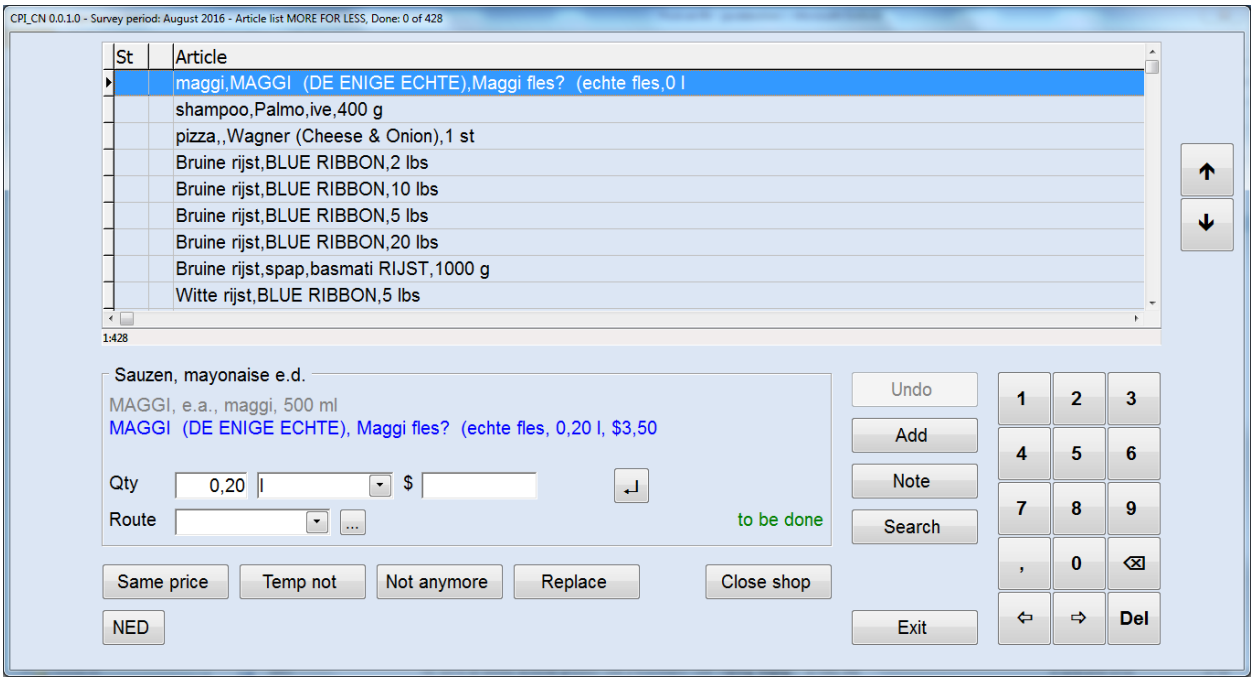
The main difference between the CPLS implementation is that the list of shops has moved to the list of survey cases (addresses) in the main window of the CPLS (see fig. 3).

Fig 3. The CPI in the main window of the CPLS



After opening the instrument with the button “Questionnaire” (in the top left of the window) the product list appears in a screen very similar to that of the stand-alone implementation (see fig. 4). Its functionality is also the same.

Fig 4. The list of products in the CPLS



Another difference between the Maniplus implementations is that the export of completed results of the previous month and the import of a new list for the next month is done automatically by the data communication process of the CPLS. In the stand-alone solution separate means are used to download and upload the files with data to the office in the Netherlands.

6. The Blaise 5 CPIX implementation

The CPIX implementation was also created for the CPLS. As a consequence the list of shops is now shown as the list of addresses in the main window. The real challenge was the product dialog including the product list and the sub-dialogs connected to it. To be able to offer similar functionality as in the Maniplus implementations an alternative had to be found for these dialogs working within a Blaise 5 instrument. The choice was made to use a table embedded in the text of a field with integrated images containing actions. This table as part of the field text (see fig. 5) could be generated on the fly making it very flexible. Images have been chosen instead of buttons because symbols could be used making these type of “buttons” very compact and language independent (see fig. 6). Furthermore the use of text fills in this so called inline table makes it even more flexible. Leaving these referred fields empty will result in empty space on the screen. This is e.g. used with the action of closing a shop (see fig 7).

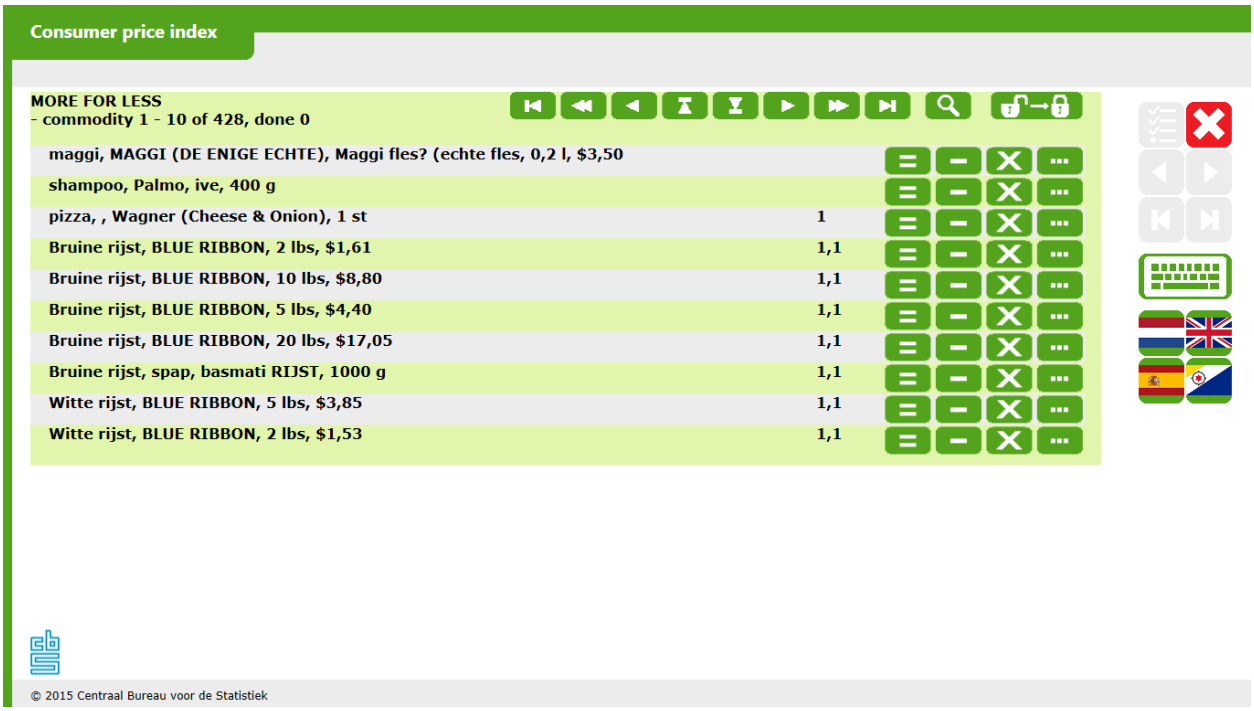
With the buttons on the top of the screen you can navigate through the product list. The products are pre-filled in an array of blocks in the record. The trick behind it is that the OnClick events of these buttons (see fig. 8) contain actions that are filling fields with a start value. In the rules the products matching these values are stored in the fields connected to the text fills (“^{aVariant[x]}”) in the inline table. Clicking on the arrows “Up” or “Down” the list will focus on the previous or next product yet to be edited (status is empty). The navigation buttons on the right-hand side are defined in the resource library and are disabled

because the questionnaire does not consist of consecutive screens between which the interviewer can browse.

Figure 5. The inline table

```
679   aVariants
680     "<table><column width=525><column width=*><row><cell>^ShopName<br>- ^{aHeader}</cell>
681      <cell>^{aNavigation}</cell></row></table>
682      <table><column width=20><column width=*><column width=70><column width=225>
683      <row background=#FFECECEC><cell>^{aStatus[1]}</cell><cell>^{aVariant[1]}</cell><cell>^{aLocation[1]}</cell><cell>
684      ^{aImg1[1]} ^{aImg2[1]} ^{aImg3[1]} ^{aImg4[1]}</cell></row>
685      <row><cell>^{aStatus[2]}</cell><cell>^{aVariant[2]}</cell><cell>^{aLocation[2]}</cell><cell>
686      ^{aImg1[2]} ^{aImg2[2]} ^{aImg3[2]} ^{aImg4[2]}</cell></row>
687      <row background=#FFECECEC><cell>^{aStatus[3]}</cell><cell>^{aVariant[3]}</cell><cell>^{aLocation[3]}</cell><cell>
688      ^{aImg1[3]} ^{aImg2[3]} ^{aImg3[3]} ^{aImg4[3]}</cell></row>
689      <row><cell>^{aStatus[4]}</cell><cell>^{aVariant[4]}</cell><cell>^{aLocation[4]}</cell><cell>
690      ^{aImg1[4]} ^{aImg2[4]} ^{aImg3[4]} ^{aImg4[4]}</cell></row>
691      <row background=#FFECECEC><cell>^{aStatus[5]}</cell><cell>^{aVariant[5]}</cell><cell>^{aLocation[5]}</cell><cell>
692      ^{aImg1[5]} ^{aImg2[5]} ^{aImg3[5]} ^{aImg4[5]}</cell></row>
693      <row><cell>^{aStatus[6]}</cell><cell>^{aVariant[6]}</cell><cell>^{aLocation[6]}</cell><cell>
694      ^{aImg1[6]} ^{aImg2[6]} ^{aImg3[6]} ^{aImg4[6]}</cell></row>
695      <row background=#FFECECEC><cell>^{aStatus[7]}</cell><cell>^{aVariant[7]}</cell><cell>^{aLocation[7]}</cell><cell>
696      ^{aImg1[7]} ^{aImg2[7]} ^{aImg3[7]} ^{aImg4[7]}</cell></row>
697      <row><cell>^{aStatus[8]}</cell><cell>^{aVariant[8]}</cell><cell>^{aLocation[8]}</cell><cell>
698      ^{aImg1[8]} ^{aImg2[8]} ^{aImg3[8]} ^{aImg4[8]}</cell></row>
699      <row background=#FFECECEC><cell>^{aStatus[9]}</cell><cell>^{aVariant[9]}</cell><cell>^{aLocation[9]}</cell><cell>
700      ^{aImg1[9]} ^{aImg2[9]} ^{aImg3[9]} ^{aImg4[9]}</cell></row>
701      <row><cell>^{aStatus[10]}</cell><cell>^{aVariant[10]}</cell><cell>^{aLocation[10]}</cell><cell>
702      ^{aImg1[10]} ^{aImg2[10]} ^{aImg3[10]} ^{aImg4[10]}</cell></row>
703      </table>"
704                                        : SET OF (_yes),EMPTY,NODK,NORF
```

Figure 6. The list of products in the CPIX



The number of products shown on one page has been limited to 10 without any real reason, it could have been more or less too. The navigation line on the top of the screen also contains a button to search for a product or to close the shop. A closed shop can be re-opened again. The mechanism works with a field set

to “0” (opened) or “1” (closed). Beneath the top line there is a line indicating which set of the products (“commodity”) is shown and how many are already completed (“done”).

Figure 7. Closing the shop

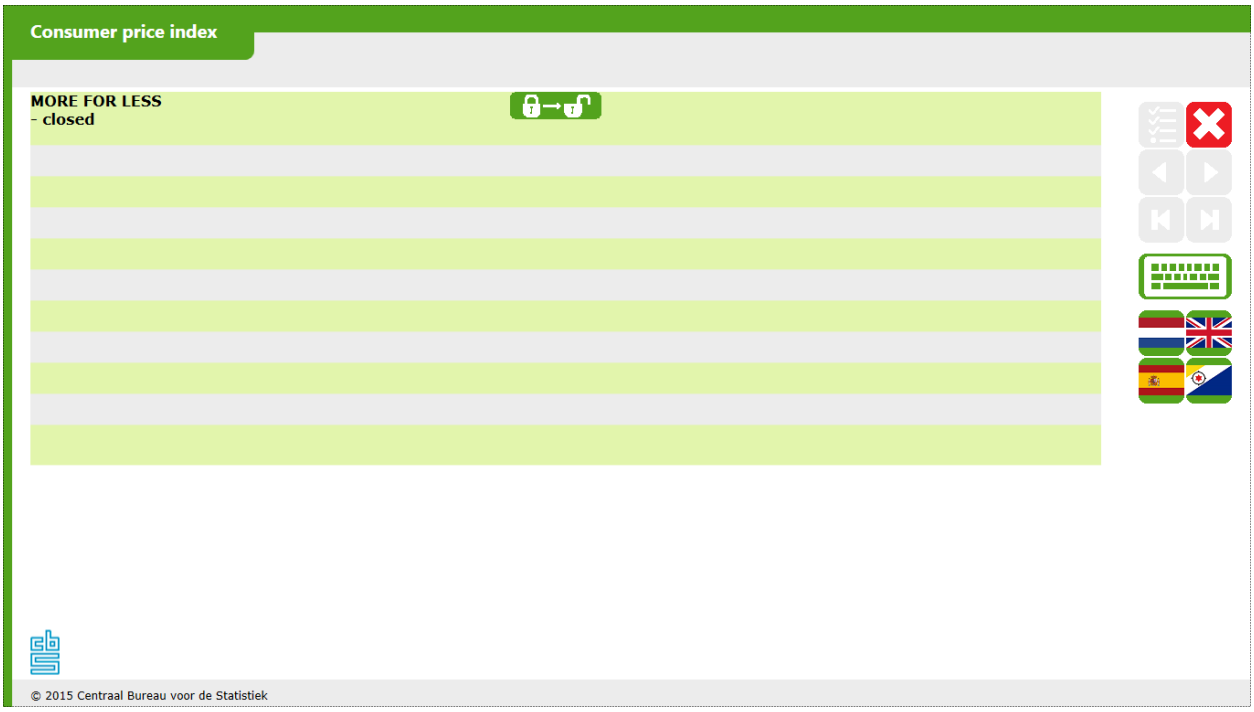


Figure 8. The navigation field with OnClick events

```
899   IF ShopClosed <> 1 THEN
900     aNavigation := '<img source=FirstSmall OnClick=\"{Action AssignField(aNr, \'-6\')}\">'+
901                    ' <img source=PreviousSetSmall OnClick=\"{Action AssignField(aNr, \'-5\')}\">'+
902                    ' <img source=PreviousSmall OnClick=\"{Action AssignField(aNr, \'-4\')}\">'+
903                    ' <img source=JumpUp OnClick=\"{Action AssignField(aJump, \'-1\')}\">'+
904                    ' <img source=JumpDown OnClick=\"{Action AssignField(aJump, \'1\')}\">'+
905                    ' <img source=NextSmall OnClick=\"{Action AssignField(aNr, \'-3\')}\">'+
906                    ' <img source=NextSetSmall OnClick=\"{Action AssignField(aNr, \'-2\')}\">'+
907                    ' <img source=LastSmall OnClick=\"{Action AssignField(aNr, \'-1\')}\">'+
908                    '<space count=3><img source=Search OnClick=\"{Action GotoURI(\''+REPLACE(aKBopen,'\\','\\\\')+
909                    '\');AssignField(SearchVariant.aSearchNext,\'0\');SetParallel(\'SearchVariant\')}\">'+
910                    '<space count=4><img source=Lock OnClick=\"{Action AssignField(ShopClosed,\'1\')}\">'
911   ELSE
912     aNavigation := '<img source=Unlock OnClick=\"{Action AssignField(ShopClosed,\'0\');AssignField(aNrCur,\'0\')}\">'
913     aNrCur := 0
914   ENDIF
```

On every line a product is shown. In the column behind it the location of the product in the shop is listed. This is the field “Route” in the Maniplus implementations. The OnClick event of the 4 buttons at the end of each line contain actions to edit the product as in the Maniplus implementations: [=] for [Same price], [-] for [Temp not] and [X] for [Not anymore]. The symbol on the button will appear in the first column of the line. With the fourth button […] an new screen is opened to edit the product. This can lead to new symbols: [>] for more expensive, [»] for much more expensive, [<] for less expensive, [«] for much less expensive, [~] for changed but not possible to determine if the product is more or less expensive and [#] in case only the description has changed (see fig. 9).

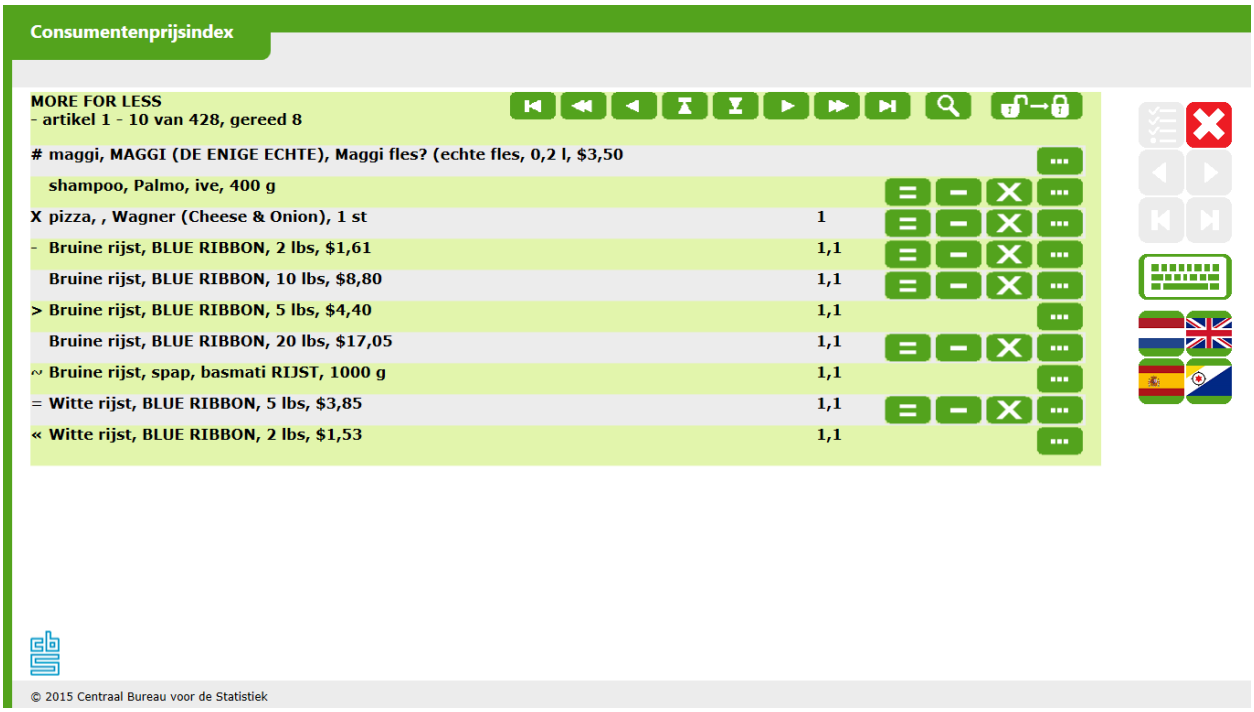
The edit screen is created as a parallel. This parallel contains the array of blocks with the pre-filled products and fields to store the changes. When activated through the edit button […] in the main window the parallel will only show the data of the related product. This is done by looping through the array and only showing the fields if the array number matches the product sequence number. If the numbers do not match the fields are opened with "KEEP" and therefore not visible:

Listing 1. Implementation Source Code

```
FOR x := 1 TO 999 DO
  IF x = aProdNr THEN
    Commodity[x].ASK(x)
  ELSEIF x <= NrMax THEN
    Commodity[x].KEEP(x)
  ENDIF
ENDDO
```

The field NrMax prevents that the loop goes beyond the maximum number of products. In the instrument a maximum of 999 products per shop is supported.

Figure 9. Status symbols



In the edit screen (see fig. 10) all kind of properties of the product can be changed. As several of these properties are descriptions or names or numeric values an on-screen keyboard was needed as the instrument should be able to run on a touch screen tablet without a keyboard. Currently it is not possible to create an on screen keyboard within a Blaise 5 questionnaire. This has to do with the different controls used for different type of fields. Instead of an integrated keyboard a C# program was developed based on the WOSK project using an XAML definition for the keyboard. Included in the OnClick event of the edit button […] it opens automatically with the edit screen using a GotoUri action.

To get several fields on the same line the “grouping” function is used to create a table for that line. To organize the fields “Brand” and “Variant” being next to the field “Notes” a table within a table was created (see fig. 11). The fields in the edit screen are all critical. As soon the focus jumps to another field the rules are invoked so that a line with changes in red is shown. At the end of this line the status symbol that will be filled in the first column in the main screen is added. The green rectangle image with an arrow under the field “Notes” activates the action “Save”. It forces the instrument to make a round trip to the server so all the fields are updated.

Figure 10. The edit screen with the keyboard

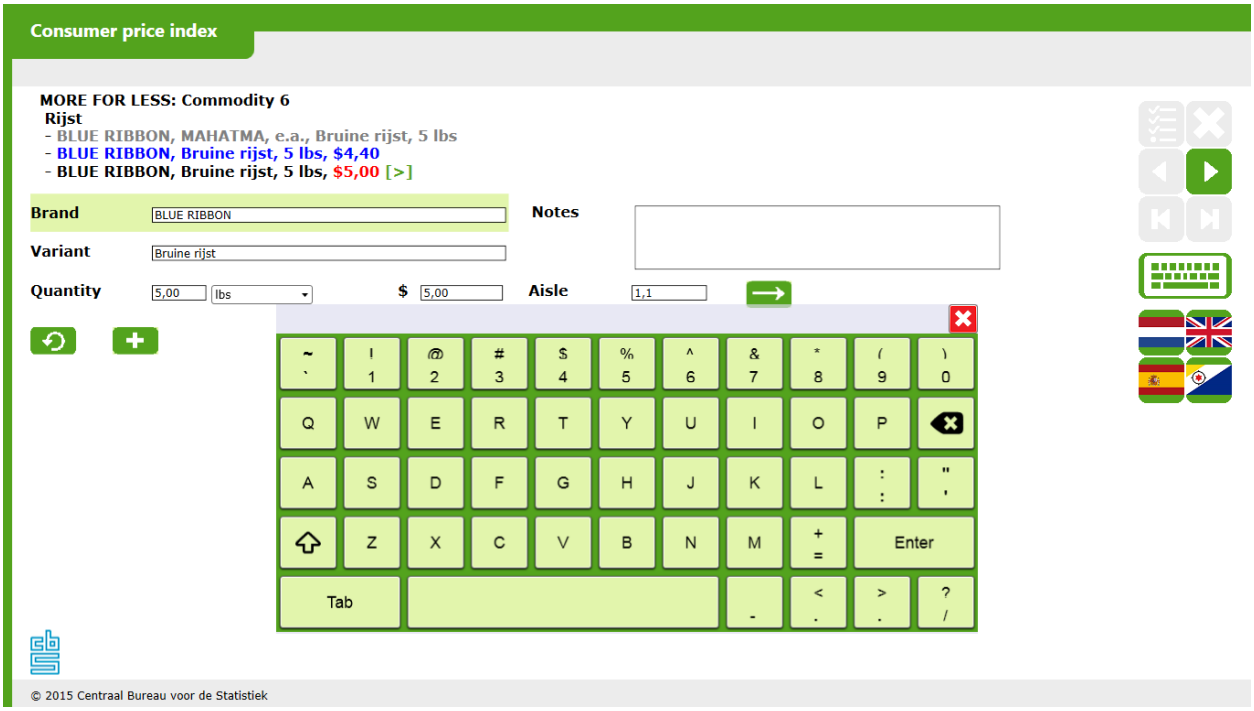
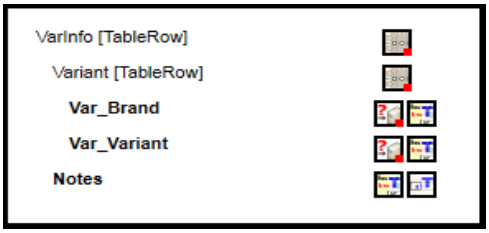


Figure 11. A table within a table



The unit of the quantity is store as a dropdown list. There are 7 groups of units of more or less the same type (e.g. weight, liquid, length, etc.). The texts are imported from an external file based on the number of the group:

Listing 2. Group Source Code

```
FOR x := 1 TO 7 DO
  FOR y := 1 TO 20 DO
    IF extUnits.SEARCH(STR(x)+STR(y,2)) THEN
      extUnits.READ
      IF   SUBSTRING(Product.Var_Unit,1,1)   =   SUBSTRING(extUnits.TagStr,1,1)
THEN
        aUnit[y] := extUnits.ValStr
      ENDIF
    ENDIF
  ENDDO
ENDDO
```

The primary key of the external consists of a 3 digit number. The first digit is the group number.
It is only possible to change the unit to one available in the group. It is e.g. not allowed to switch from kilograms to centimeters.

The curled arrow image contains an action to undo all changes. It sets the value of the auxiliary field "aResetStatus" to 1. This value causes in the rules to activate 3 fields "aConfirm", "aOK" and "aCancel" in the place of the [+] field (called "aNew") which is used to create a new product to replace the current one:

Listing 3. aResetStatus Source Code

```
aReset.ASK
IF aResetStatus = 1 THEN
  aConfirm.ASK
  aOK.SHOW
  aCancel.SHOW
ELSE
  aNew.SHOW
ENDIF
```

The field "Confirm" is a so-called QuestionTextOnly field displaying a text asking for confirmation. The two fields "aOK" (image [✓]) and "aCancel" (image [X]) are images with actions in the OnClick event changing the value of the field "aResetStatus". Based on this value the original pre-filled values of the product are shown again by emptying the fields with the changes.

When the field "aNew" (image [+]) is clicked the value of a field called "aNewStatus" is changed to 1 using the AssignField action in the OnClick event. This value causes in the rules to increment the maximum number of products with 1 at the end and focus on this product which has still empty properties. After filling in the properties of the product and clicking on the field "aOK" (image [✓]) the value of the field is changed to 2 using the AssignField action in the OnClick event and in the rules the array of products (now with one extra product) is sorted on the location field using the EXCHANGE function of Blaise.

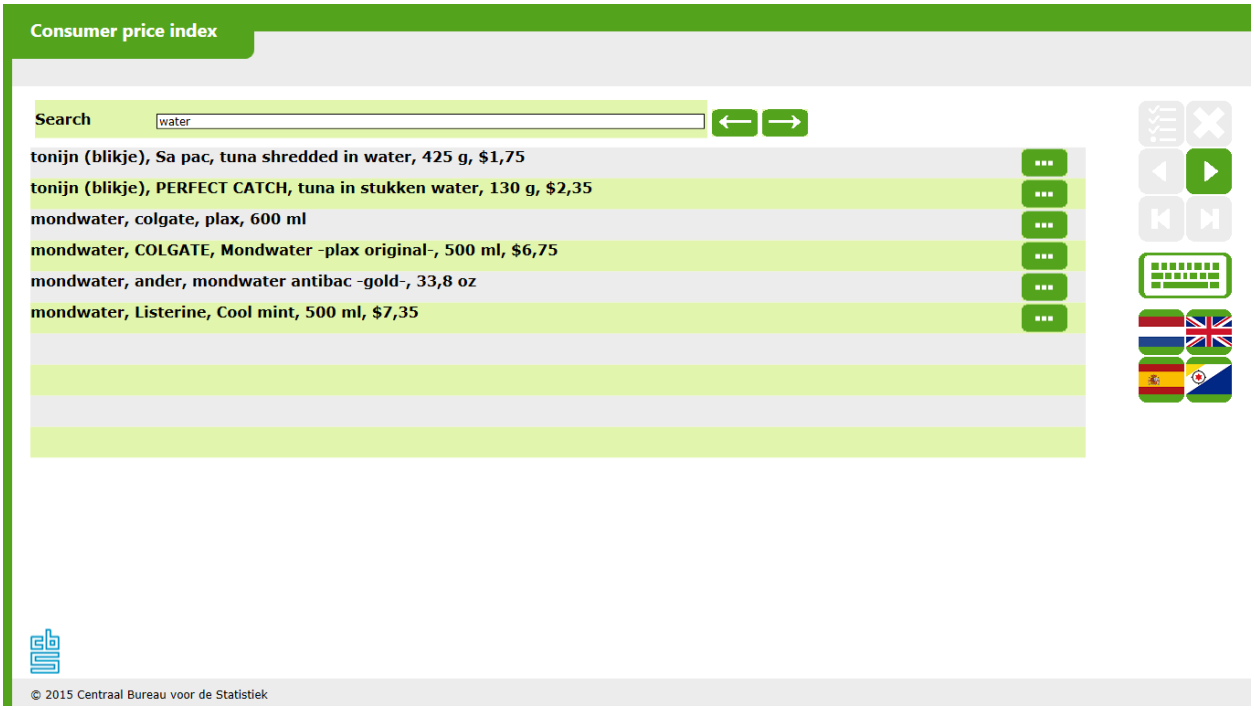
The location of the product in the edit screen can also be changed using the text field with the label "Aisle". When changed in the rules the array with products is sorted on that text using the EXCHANGE function.

Clicking on the arrow button on the right hand side of the screen will make the main screen appear again using the NextPage action. It focuses on the product that has been relocated by assigning the new

sequence number of the product to the field which is used by the main screen to load the subset of products into the table. This (image) button is defined in resource library together with the buttons for the keyboard and the languages. Using different resource sets these buttons can also appear on the left hand side of the screen for left-handed interviewers. This is controlled by a setting in the CPLS. The CPIX is only designed to use in landscape position although it is possible to create a layout for portrait too.

It is also possible to search for the product based on a text in the description. Therefor a search screen was created that is also defined as a parallel (see fig. 12). Clicking on the magnifying glass in the main screen will jump to the search screen using a SetParallel action. After filling in the search text a click on one of the arrow buttons will cause the table to be filled with the products which description matches the search text. With the two arrow buttons you can browse (up and down) through the screens if there is more than one. To jump to a product it suffice to click on the […] button next to the product. The OnClick event contains actions which will close the on screen keyboard with a GotoUri action that invokes a small C# program that kills the task (in case the keyboard was opened), assign the sequence number of the product to the field which is used by the main screen to load the subset of products into the table and a SetParallel action to jump to the main parallel.

Figure 12. The search screen



7. Conclusion

With Blaise 5 it is now possible to create very exotic instruments. Although not always easy the CPIX proves the richness and power of Blaise 5. The only things you need are a creative mind and a solid knowledge of the Blaise language and layout functions. The creative mind is given to you by birth ☺ (or not ☹), the solid knowledge is a matter of learning and practicing.