Materials management

Description of the "Materials management" task

The information system being created using the **IsFusion** platform must support very basic supply chain execution capabilities.

For simplicity, let's define one type of document in our system that increases the stock balance — a receipt from the supplier; and one type of document that does the opposite — a shipment for a wholesale to a customer.

Defining domain logic

The information system will consist of a set of modules, each implementing a logically isolated piece of functionality. Each module can use the functionality of other modules, which involves special syntax constructions for defining module dependencies.

Based on our task, let's define the list of modules to be implemented: stock module, item module, legal entity module, receipt module, shipment module, current balance module. We will separately define the main module that will be executed and will basically be a compound solution. The composition of modules can be different and is determined by the developer depending on the need to re-use the functionality elsewhere.

Defining a stock

Let's create a module where we will define a stock instance and its attributes.

```
MODULE
Stock;
```

Let's define the concept of a stock and its attributes: name, address.

```
CLASS
Stock 'Ware
house';

name 'Name'
 = DATA STR
ING[100]
(Stock) IN
base;
address 'Ad
dress' = DA
TA STRING[1
50]
(Stock) IN
base;
```

Defining an item

Let's create a module in which we'll define the concept of an item and its attributes.

```
MODULE
 Item;
```

Let's define the concept of an item and its attributes: name, barcode.

```
CLASS
Item 'Pro
duct';

name 'Nam
e' = DATA
STRING[100
](Item) IN
 base;
barcode '
Barcode'
= DATA BP
STRING[13
](Item) IN
 base;
```

Let's set the wholesale price for an item.

```
salePrice 'Wholesale price' = DATA NUMERIC[17,2] (Item) IN base;
```

Defining a legal entity

Let's create a module where we will define a legal entity instance and its attributes. In the system, legal entities will act as suppliers and customers.

```
MODULE LegalEntity;
```

Let's define the concept of a legal entity and its attributes: name, legal address, Tax ID.

```
CLASS LegalEntity 'Organization';

name 'Name' = DATA STRING[100] (LegalEntity) IN base;
address 'Address' = DATA STRING[150] (LegalEntity) IN base;
inn 'TIN' = DATA BPSTRING[9] (LegalEntity) IN base;
```

We define the uniqueness of the Tax ID for the legal entity.

```
legalEntityINN = GROUP AGGR LegalEntity legalEntity BY inn (legalEntity);
```

The legalEntityINN property binds an organization and a Tax ID one-to-one and allows to find a legal entity by a Tax ID. The expression of the property can be interpreted as follows: when grouping legal entities by Tax ID (innLegalEntity property), each group must contain a non-repeating legal entity.

Defining a receipt

Let's create a module where we'll define all instances and attributes required for defining the logic of a supplier receipt.

```
MODULE Receipt;
```

Let's define the use of functionality from other modules in the Receipt module.

```
REQUIRE Stock, Item, LegalEntity;
```

We define the concepts that determine the logic of a supplier receipt. Let's work on the premise that all documents (both receipts and shipments) in the system consist of a header and an item specification. Accordingly, let's define the concepts of a receipt header and receipt line.

```
CLASS Receipt '
Receipt';
CLASS
ReceiptDetail '
Receipt line';
```

Each receipt line contains a link to the document header, so in the end, the header and the subset of lines with links to this document together define the receipt from the user perspective. The **NONULL** parameter indicates that the link must be defined. The **DELETE** parameter specifies that if the main Receipt object is deleted, all ReceiptDetail lines linking to it will also be deleted. By default, when an object is deleted, all links to it are nullified. This way, without the **DELETE** parameter, the system will show an error message about an undefined link.

```
receipt 'Line document'
= DATA Receipt
(ReceiptDetail) NONULL DE
LETE;
```

Let's define the line number in a receipt.

```
index 'Line number'
(ReceiptDetail d) =
        PARTITION SUM
 1 IF d IS
ReceiptDetail
        ORDER d BY
receipt(d);
```

The use of the name of an object class in expressions is similar to using its identification number (id) created by the system for all objects by an automatic counter. In this case, the use of the ORDER receiptDetail construct helps sort the lines of the receipt by the order of ascension of their id, i.e. basically in the order of their creation.

Here, the PARTITION instruction uses the BY block that groups objects by a certain attribute. The calculation of the expression cumulative total is performed in each group. In this case, the line number is determined only within this line's document (receipt(d) property).

We define a set of key attributes of a receipt header: number, date, supplier and its name, the stock where the product is received and its name. The name of the supplier and the stock will be needed in the future for displaying them on the form.

```
number 'Receipt number' = DATA BP
STRING[10] (Receipt);
date 'Receipt date' = DATA DATE
(Receipt);

supplier 'Supplier' = DATA
LegalEntity (Receipt);
nameSupplier 'Supplier name'
(Receipt r) = name(supplier(r));

stock 'Warehouse' = DATA Stock
(Receipt);
nameStock 'Warehouse name'
(Receipt r) = name(stock(r));
```

We define a set of key attributes of a receipt line: item and its name, quantity, supplier price, supplier amount (price multiplied by quantity).

```
item 'Product' = DATA Item (ReceiptDetail);
nameItem 'Product name' (ReceiptDetail d) =
name(item(d));

quantity 'Quantity' = DATA NUMERIC[16,4]
(ReceiptDetail);
price 'Supplier price' = DATA NUMERIC[17,2]
(ReceiptDetail);
sum 'Supplier amount' (ReceiptDetail d) =
quantity(d) * price(d);
```

Defining a shipment

Let's create a module where we will define all instances and attributes required for a wholesale shipment.

```
MODULE
Shipment;
```

We define the use of functionality from other modules in the Shipment module.

```
REQUIRE Stock, Item,
LegalEntity;
```

Similarly to the receipt, we define the shipment header and lines, as well as a link in the line to the header and its number.

```
CLASS Shipment 'Shipment';
CLASS ShipmentDetail 'Shipm
ent line';

shipment 'Line document' =
DATA Shipment
(ShipmentDetail) NONULL DEL
ETE;
index 'Line number'
(ShipmentDetail d) =
        PARTITION SUM 1 IF
d IS ShipmentDetail
        ORDER d BY shipment
(d);
```

We define a set of attributes for the shipment: number, date, customer and its name, the stock from which the item is shipped, and its name.

```
number 'Shipment number' = DATA BP
STRING[10] (Shipment);
date 'Shipment date' = DATA DATE
(Shipment);

customer 'Customer' = DATA
LegalEntity (Shipment);
nameCustomer 'Customer name'
(Shipment s) = name(customer(s));

stock 'Warehouse' = DATA Stock
(Shipment);
nameStock 'Warehouse name'
(Shipment s) = name(stock(s));
```

We define a set of key attributes of a shipment: item and its name, quantity, sale price, sale amount (price multiplied by quantity).

```
item 'Product' = DATA Item (ShipmentDetail);
nameItem 'Product name' (ShipmentDetail d) =
name(item(d));

quantity 'Quantity' = DATA NUMERIC[16,4]
(ShipmentDetail);
price 'Selling price' = DATA NUMERIC[17,2]
(ShipmentDetail);
sum 'Sale amount' (ShipmentDetail d) =
quantity(d) * price(d);
```

We implement the auto filling of the item sale price in the shipment with the value of the wholesale price defined by the user for item (salePrice property). Auto filling for the shipment line should work when the item is changed (WHEN CHANGED instruction).

```
29 price(ShipmentDetail d) <- salePrice(item(d))
   WHEN CHANGED(item(d));
```

Defining current item balance

The current item balance is defined as a difference between all item receipts and all its shipments.

Let's create a separate module.

```
1 MODULE
  StockItem;
```

We define the use of functionality from other modules in the StockItem module.

```
3 REQUIRE Shipment,
  Receipt;
```

Let's define the calculated property of the current item balance at the stock in quantitative terms.

```
5 receivedQuantity 'Total income' = GROUP SUM quantity(ReceiptDetail d) BY
   item(d), stock(receipt(d));
6 shippedQuantity 'Total expenses' = GROUP SUM quantity(ShipmentDetail d)
  BY item(d), stock(shipment(d));
7 currentBalance 'Current balance' (Item i, Stock s) = receivedQuantity
  (i, s) (-) shippedQuantity (i, s);
```

Let's prohibit the negative item balance. The prohibition will work for any user action resulting in a negative balance. In this case, the user will see a specified message on the screen.

```
9 CONSTRAINT currentBalance(Item i, Stock s) < 0
  MESSAGE 'The balance of the product cannot be
  negative';
```

Defining view logic

In order to be able to work with the created solution, let's add directory forms and a current balances form, and also a set of paired forms for working with documents: a form for listing receipts and a form for editing them, a form for listing shipments and a form for editing them.

First, let's create directory forms.

In the Stock module, we add a form that provides the user with the functionality of creating and deleting stocks, as well as changing their attributes.

```
 8 FORM stocks 'Wareho
   uses'
 9         OBJECTS s
   = Stock
10         PROPERTIES(
   s) name, address, N
11 EW, DELETE
   ;
```

In a similar manner, we'll create an item form in the Item module, and a legal entity form in the LegalEntity module.

```
CLASS
Item 'Product';

name 'Name' = DATA STRING[100](Item) IN base;
barcode 'Barcode' = DATA BPSTRING[13](Item) IN base;
```

```
CLASS
LegalEntity 'Organization';

name 'Name' = DATA STRING[100](LegalEntity) IN base;
address 'Address' = DATA STRING[150](LegalEntity) IN base;
inn 'TIN' = DATA BPSTRING[9](LegalEntity) IN base;
```

Let's create edit forms for a receipt and a shipment. These forms will be used for creating new documents or editing existing ones. The layout of the forms will be similar: two vertical blocks, the top one containing a panel with the header attributes of the document being created/edited, and the lower one containing the document lines in a grid view and their attributes.

In the Receipt module, we should create a receipt edit form. For the form we are building, we specify that it will be used as a default form for creating /editing receipts (the EDIT instruction).

```
FORM receipt 'Receipt'
        OBJECTS r = Receipt PANEL
        PROPERTIES(r) number, date, nameSupplier, nameStock

        OBJECTS d = ReceiptDetail
        PROPERTIES(d) index, nameItem, quantity, price, sum READONLY, NEW, DELETE
GRID
        FILTERS receipt(d) = r

        EDIT Receipt OBJECT r
;
```

Line filtering for the current receipt is performed with the help of the FILTERS receipt(d) == r expression. The FILTERS construct displays an object of a corresponding class on the form if the filter expression returns a value different from NULL. In this case, the receipt line will be displayed on the form if the header of the document to which the line is linked ("receipt" property) equals to the current object of the top block. In other words, only the lines of the created/edited document will be displayed.

In addition, if a filter is specified for objects of this class on the form, then when the user presses the NEW button, the property of the newly added object will be automatically filled in a way that will make this object meet the filter conditions. In this case, when a new receipt line is created, the "receipt" property of this line will be automatically filled with a link to the current header of the receipt.

Let's create an edit form for the shipment in the Shipment module. For the form we are creating, we specify that it will be used as the default form for creating/editing shipments (EDIT instruction).

```
FORM shipment 'Shipment'
        OBJECTS s = Shipment PANEL
        PROPERTIES(s) number, date,
nameCustomer, nameStock

        OBJECTS d = ShipmentDetail
        PROPERTIES(d) nameItem,
quantity, price, sum READONLY, NEW, DELETE
 GRID
        FILTERS shipment(d) = s

        EDIT Shipment OBJECT s
;
```

Visually, supplier receipt and shipment forms will look almost identical and consist of two vertical blocks: one with a table for document headers and one with a table of document lines. Document lines should support visual filtering by documents and their subsets displayed on the form will change when navigating in the top block.

Let's create a receipt form. On this form, we will display all the properties defined above for document headers and their lines. Additionally, we will place automatically defined buttons for creating/editing a receipt using the edit form created above. All properties of document headers and their lines, except the buttons for creating/editing a receipt, should be read-only for editing directly on the form (READONLY operator).

```
FORM receipts 'Receipts'
        OBJECTS r = Receipt
        PROPERTIES(r) READONLY
number, date, nameSupplier, nameStock
        PROPERTIES(r) NEWSESSION NEW,
EDIT, DELETE

        OBJECTS d = ReceiptDetail
        PROPERTIES(d) READONLY index,
nameItem, quantity, price, sum
        FILTERS receipt(d) = r
;
```

Let's create the shipment form in a similar manner.

```
FORM shipments 'Shipments'
        OBJECTS s = Shipment
        PROPERTIES(s) READONLY
number, date, nameCustomer, nameStock
        PROPERTIES(s) NEWSESSION NEW,
EDIT, DELETE

        OBJECTS d = ShipmentDetail
        PROPERTIES(d) READONLY
nameItem, quantity, price, sum
        FILTERS shipment(d) = s
;
```

Next, in the StockItem module, let's create a form for displaying current balances. A form should be a table whose lines contain information about the item (its name and barcode), the name of the stock, and the current balance for this item at this stock. The count of lines on the form will be equal to the count of items entered into the system multiplied by the count of entered stocks. To display only relevant data (i.e. only those items and stocks, for whose intersection the current balance is not NULL), let's add a filter to the form.

```
FORM currentBalanceItemStock 'Current balances'
    OBJECTS si = (s = Stock, i = Item)
    PROPERTIES READONLY name(i), barcode(i), name
(s), currentBalance(i, s)
    FILTERS currentBalance(i, s)
;
```

The OBJECTS si = (s = Stock, i = Item) construct adds an object group with the name si, which is a Cartesian product of Stock and Item class objects.

Finally, let's declare the head module and specify what functionality from other modules will be used in it.

```
MODULE StockAccounting;

REQUIRE Stock, Item, LegalEntity, Receipt, Shipment,
StockItem;
```

In the StockAccounting module, we compose the system menu. Directories should be added to the predefined **masterData** folder of the navigator that we show immediately after the directories. We place the current balance form to the main menu (horizontal window **root**). Links to directory and document forms will be shown on the vertical **toolbar** when the user selects a corresponding **root** folder.

```
NAVIGATOR {
    NEW FOLDER
masterData 'Directories'
FIRST WINDOW toolbar {
        NEW items;
        NEW stocks;
        NEW
legalEntities;
    }
    NEW FOLDER documents
'Documents' AFTER
masterData WINDOW
toolbar {
        NEW receipts;
        NEW shipments;
    }
    NEW
currentBalanceItemStock A
FTER documents;
}
```

The process of creating an information system is completed.

The complete source code (on Github)

```
MODULE Stock;

CLASS Stock 'Wareho
use';

name 'Name' = DATA
STRING[100]
(Stock) IN base;
address 'Address'
= DATA STRING[150]
(Stock) IN base;

FORM stocks 'Wareho
uses'
        OBJECTS s
= Stock
        PROPERTIES(
s) name, address, N
EW, DELETE
;
```

```
MODULE Item;

CLASS Item 'Product';

name 'Name' = DATA STRING[100]
(Item) IN base;
barcode 'Barcode' = DATA BPSTR
ING[13](Item) IN base;

salePrice 'Wholesale price' =
DATA NUMERIC[17,2](Item) IN
base;

FORM items 'Products'
        OBJECTS i = Item
        PROPERTIES(i) name,
barcode, salePrice, NEW, DELETE

;
```

```
MODULE LegalEntity;

CLASS LegalEntity 'Organi
zation';

name 'Name' = DATA STRING[
100](LegalEntity) IN
base;
address 'Address' = DATA
STRING[150](LegalEntity)
IN base;
inn 'TIN' = DATA BPSTRING[
9](LegalEntity) IN base;

legalEntityINN = GROUP AG
GR LegalEntity
legalEntity BY inn
(legalEntity);

FORM legalEntities 'Organ
ization'
        OBJECTS l =
LegalEntity
        PROPERTIES(l)
name, inn, address, NEW,
DELETE
;
```

```
MODULE Receipt;

REQUIRE Stock, Item, LegalEntity;

CLASS Receipt 'Receipt';
CLASS ReceiptDetail 'Receipt line';

receipt 'Line document' = DATA Receipt
(ReceiptDetail) NONULL DELETE;

index 'Line number' (ReceiptDetail d) =
        PARTITION SUM 1 IF d IS
ReceiptDetail
        ORDER d BY receipt(d);

number 'Receipt number' = DATA BPSTRING[10]
(Receipt);
date 'Receipt date' = DATA DATE (Receipt);

supplier 'Supplier' = DATA LegalEntity
(Receipt);
nameSupplier 'Supplier name' (Receipt r) =
name(supplier(r));

stock 'Warehouse' = DATA Stock (Receipt);
nameStock 'Warehouse name' (Receipt r) =
name(stock(r));

item 'Product' = DATA Item (ReceiptDetail);
nameItem 'Product name' (ReceiptDetail d) =
name(item(d));

quantity 'Quantity' = DATA NUMERIC[16,4]
(ReceiptDetail);
price 'Supplier price' = DATA NUMERIC[17,2]
(ReceiptDetail);
sum 'Supplier amount' (ReceiptDetail d) =
quantity(d) * price(d);

FORM receipt 'Receipt'
        OBJECTS r = Receipt PANEL
        PROPERTIES(r) number, date,
nameSupplier, nameStock

        OBJECTS d = ReceiptDetail
        PROPERTIES(d) index, nameItem,
quantity, price, sum READONLY, NEW, DELETE G
RID
        FILTERS receipt(d) = r

        EDIT Receipt OBJECT r
;

FORM receipts 'Receipts'
        OBJECTS r = Receipt
        PROPERTIES(r) READONLY number,
date, nameSupplier, nameStock
        PROPERTIES(r) NEWSESSION NEW, EDIT,
DELETE

        OBJECTS d = ReceiptDetail
        PROPERTIES(d) READONLY index,
nameItem, quantity, price, sum
        FILTERS receipt(d) = r
;
```

```
MODULE Shipment;

REQUIRE Stock, Item, LegalEntity;

CLASS Shipment 'Shipment';
CLASS ShipmentDetail 'Shipment line';

shipment 'Line document' = DATA Shipment
(ShipmentDetail) NONULL DELETE;
index 'Line number' (ShipmentDetail d) =
        PARTITION SUM 1 IF d IS ShipmentDetail
        ORDER d BY shipment(d);

number 'Shipment number' = DATA BPSTRING[10]
(Shipment);
date 'Shipment date' = DATA DATE (Shipment);

customer 'Customer' = DATA LegalEntity
(Shipment);
nameCustomer 'Customer name' (Shipment s) =
name(customer(s));

stock 'Warehouse' = DATA Stock(Shipment);
nameStock 'Warehouse name' (Shipment s) = name
(stock(s));

item 'Product' = DATA Item (ShipmentDetail);
nameItem 'Product name' (ShipmentDetail d) =
name(item(d));

quantity 'Quantity' = DATA NUMERIC[16,4]
(ShipmentDetail);
price 'Selling price' = DATA NUMERIC[17,2]
(ShipmentDetail);
sum 'Sale amount' (ShipmentDetail d) =
quantity(d) * price(d);

price(ShipmentDetail d) <- salePrice(item(d))
WHEN CHANGED(item(d));

FORM shipment 'Shipment'
        OBJECTS s = Shipment PANEL
        PROPERTIES(s) number, date,
nameCustomer, nameStock

        OBJECTS d = ShipmentDetail
        PROPERTIES(d) nameItem, quantity,
price, sum READONLY, NEW, DELETE GRID
        FILTERS shipment(d) = s

        EDIT Shipment OBJECT s
;

FORM shipments 'Shipments'
        OBJECTS s = Shipment
        PROPERTIES(s) READONLY number, date,
nameCustomer, nameStock
        PROPERTIES(s) NEWSESSION NEW, EDIT, DE
LETE

        OBJECTS d = ShipmentDetail
        PROPERTIES(d) READONLY nameItem,
quantity, price, sum
        FILTERS shipment(d) = s
;
```

```
MODULE StockItem;

REQUIRE Shipment, Receipt;

receivedQuantity 'Total income' = GROUP SUM quantity(ReceiptDetail d) BY
 item(d), stock(receipt(d));
shippedQuantity 'Total expenses' = GROUP SUM quantity(ShipmentDetail d)
BY item(d), stock(shipment(d));
currentBalance 'Current balance' (Item i, Stock s) = receivedQuantity
(i, s) (-) shippedQuantity (i, s);

CONSTRAINT currentBalance(Item i, Stock s) < 0 MESSAGE 'The balance of
the product cannot be negative';

FORM currentBalanceItemStock 'Current balances'
    OBJECTS si = (s = Stock, i = Item)
    PROPERTIES READONLY name(i), barcode(i), name(s), currentBalance(i,
s)
    FILTERS currentBalance(i, s)
;
```

```
MODULE StockAccounting;

REQUIRE Stock, Item, LegalEntity, Receipt, Shipment,
StockItem;

NAVIGATOR {
    NEW FOLDER masterData 'Directories' FIRST WINDOW
toolbar {
        NEW items;
        NEW stocks;
        NEW legalEntities;
    }
    NEW FOLDER documents 'Documents' AFTER masterData WI
NDOW toolbar {
        NEW receipts;
        NEW shipments;
    }
    NEW currentBalanceItemStock AFTER documents;
}
```