**Home book**

**Ver. 0.0.0.0.1**  
  
design, development and production by Dmytro Zaverukha

**I. User Guide**

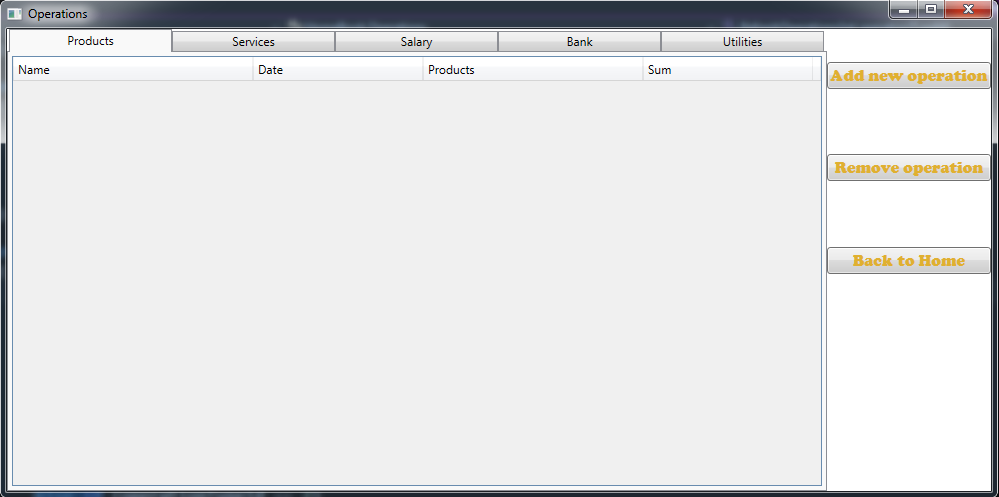
1. When user starts the application the first window that showing is Home, where three boxes (buttons) are present (see pic.1.1):

* Operations – allow user to see all added operations by him;
* Reports – allow user to get operations filtered by some options;
* Settings – allow the user to setup all necessary item that used when operations added.



pic. 1.1

2. Click on the first button “Operations” redirect user to another window where all added and available operations are present (pic 2.1).



pic. 2.1

This is a window which presented as tab control with five tabs for all types of operations that used in the application.

This tabs are:

* Product operation;
* Service operations;
* Salary;
* Bank operations;
* Utility operations.

In the upper right corner displayed something like sidebar with three buttons:

* Add new operation;
* Remove operation
* Back to home.

The first one (“Add new operations”) allows adding operation with a specific type. It’s open separate window with a form which used to add some properties to operation.  
  
It opens “Add operation” window with that type as currently selected tab in “Operations” window. To explain, for example, to add salary operation user need to select “Salary” tab and then click on “Add new operation” button. This will open the appropriate window with specific to this type of operation fields.

So, “Add new operation” button allows adding operation with specific type from each tab respectively.

2.1. From selected first tab “Product operations” click on “Add new operation” buttons. The “Add operation” window is open (pic. 2.2). As product operations are chosen in opened window displayed next fields:

* Type – allow choosing from drop-down another type of operation different from currently opened;
* Date – choose specific date of added operation;
* Operation name – short description of added operations.

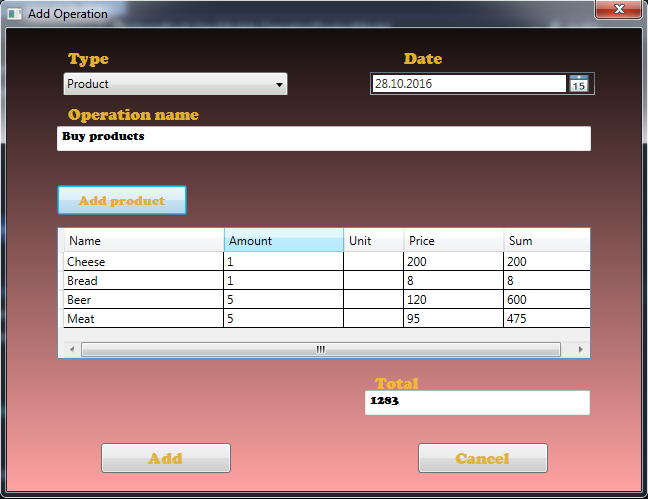
These three fields are global, so it’s available on all windows for adding different operations.

And specific to product operation fields are:

- Add product – button that needs to add product to this operation;

- Table – information of added to operation products;

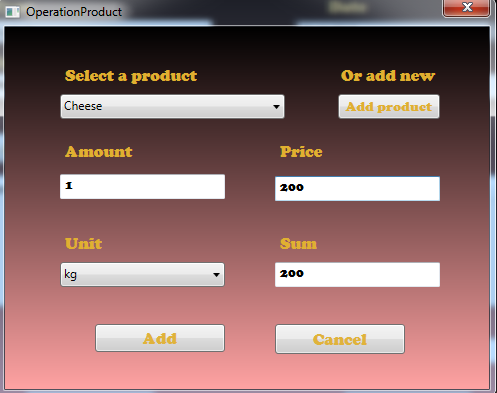
- The total – field that shows the total sum off all added products. Read only, because this sum is automatically calculated by application behavior.



pic. 2.2

Focus more on “Add product” button. This button is an open separate window (pic. 2.3) from which user can add some product to operation with specific fields. As this is product operation than on this form presents next fields:

* Select a product – allows to choose from drop-down some product name;
* Add product – button that needs to add brand new product to this operation and to application at all;
* Amount – number of choose product amount;
* Price – price of the product;
* Unit – dropdown allows to select specific product unit, e.g. kg;
* Sum – sum calculated for selected product. Read only.



pic. 2.3

**Note: all fields required. You can’t allow adding operation product if some of the fields will be not filled.**

And,detailed more on “Add product”. This button also opens another window (pic. 2.4) where you can add new product typing the name of this product to a text field or cancel to add. The field is also required!

  
pic. 2.4

If all fields will be filled “Add” button is starting to be enabled and you can add your product to operation.

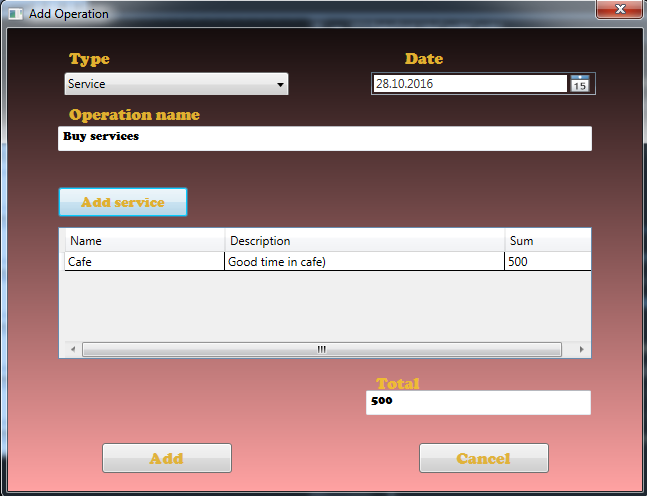
After click on “Add” button you return to “Add operation” window wherein the product table where already added previously product is displayed with appropriate fields. As all fields are filled, “Add” buttons is enabled – you can add this product operation and see it on main “Operations” window with all info about added operation.

**Note: all fields on “Add operation” form are also available!**  
 **2.2.** Select second tab (“Services”) and click “Add new button” to add services operations.

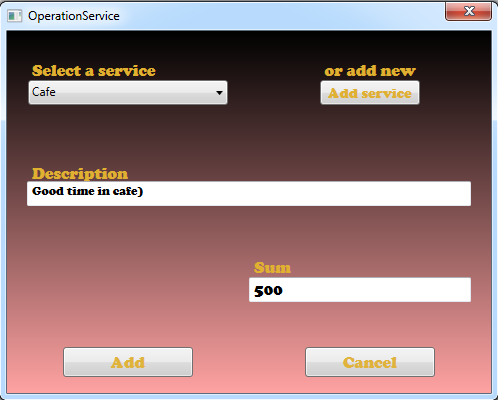
The opened window (pic. 2.5) is pretty the same as “Add product” window this no need to more explanations. See 2.1 part of this guide.

The difference in the window, that opensby clicking on “Add service” button. It’s also having like in operation product window dropdown where you can select some service and “Add service” that open the window to add brand new service (pic.2.6). Different fields are:

* Description – short description of added operation service;
* Sum **–** sum of adding service. In this case not ready only, it’s available and required to add some for selected service.

****

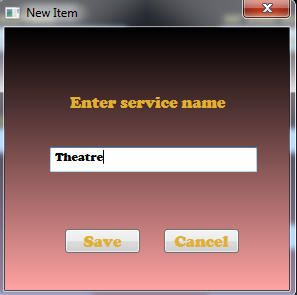
pic. 2.5



pic. 2.6

**Note: all fields on “Add operation service” window are required.**

After adding all necessary services (pic.2.7) you can add this operation which will be immediately presented on main “operations” window.

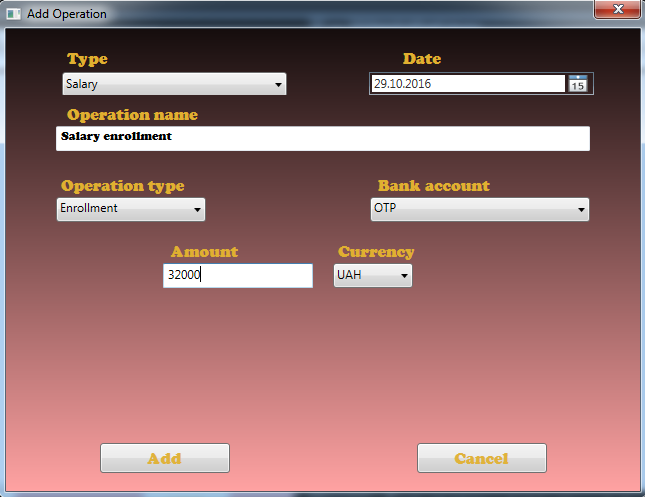


pic. 2.7

2.3. “Salary” tab allows adding salary operations from this opened window (pic.2.8):

The fields’ specific to salary operation is**:**

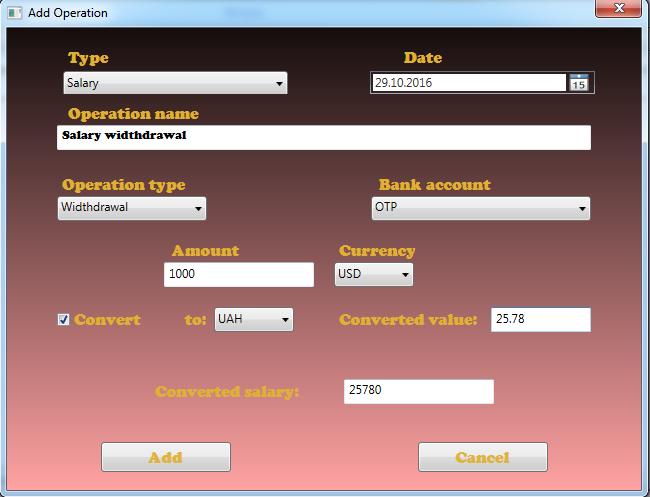
* Operation type – dropdown that allows you to select the type of added salary operation: Enrollment or Withdrawal;
* Bank account – another dropdown to select from bank account (salary card) which related to only salary operations;
* Amount – amount of enrollment or withdrawal salary;
* Currency dropdown –currency of enrollment or withdrawal salary.



pic. 2.8

Other fields on this window for add salary operation is available when you select Withdrawal type (pic.2.9). It shows fields with not required options to convert salary amount to some another currency. After a check for Convert checkbox and type on Converted value, app behavior calculates the appropriate amount of converted salary.

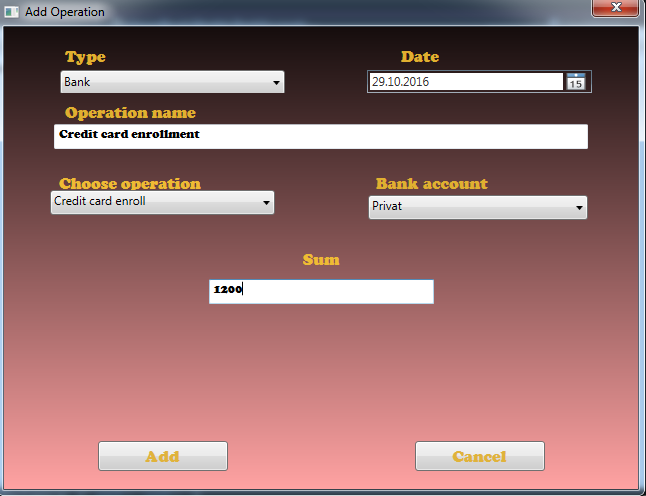
If Convert checkbox will be checked – this “converted options” fields will be also required.



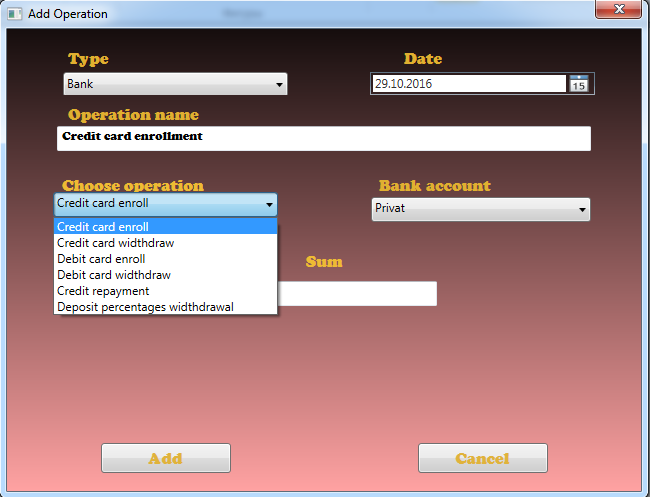
pic. 2.9

2.4. From “bank” tab you can open window (pic. 2.10) from where Bank operations are added. Specific to this type of operation fields are:

* Choose operation dropdown – allow you to choose different type bank operation (pic.2.11), e.g. Credit card enrollment or Credit repayment;
* Bank account - another dropdown to select from a bank account which related to selected type of bank operation. For example: if Credit card enrolls – that in bank account you may select available credit cards;
* Sum – sum of bank operation, for example, the amount of credit repayment.



pic. 2.10

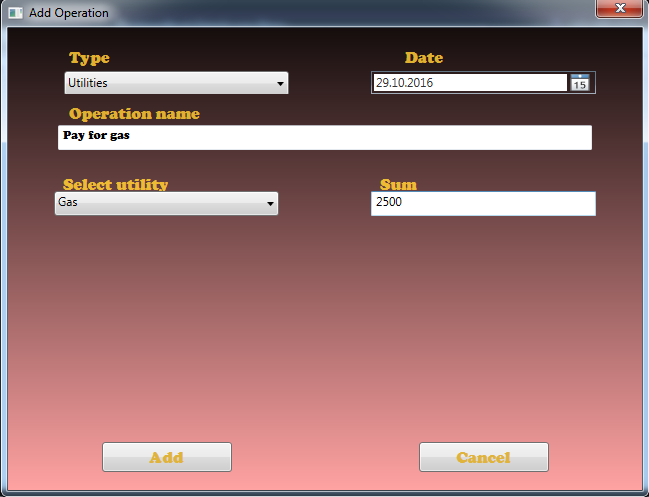


pic. 2.11

2.5. From “Utility” tab you can open window to add utility operation with specific fields (pic.2.12):

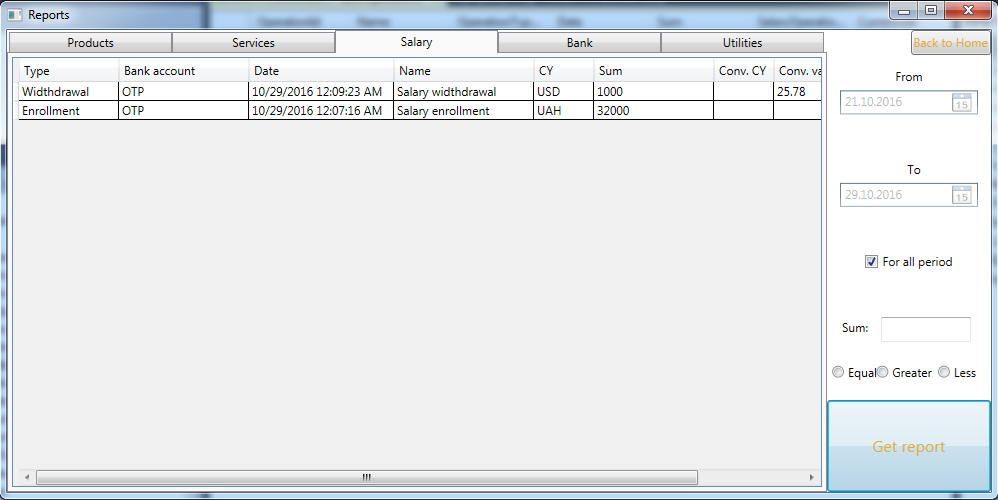
- Select utility dropdown – allows you to choose type of utility (Gas, Electricity, etc.);

- Sum – sum of utility operation.



pic. 2.12

3. From home, window clicks “reports” button to open a separate window (pic.3.1).

In appeared window, you can see the same design as in “Operations” window. As “Reports” allows getting operations with specific filters, this tab controls will be the same, with that five tabs for Product, Services, Salary, Bank and Utility operations. The reported data is displayed in each tab in the same fields as on “operations” window.  


pic. 3.1

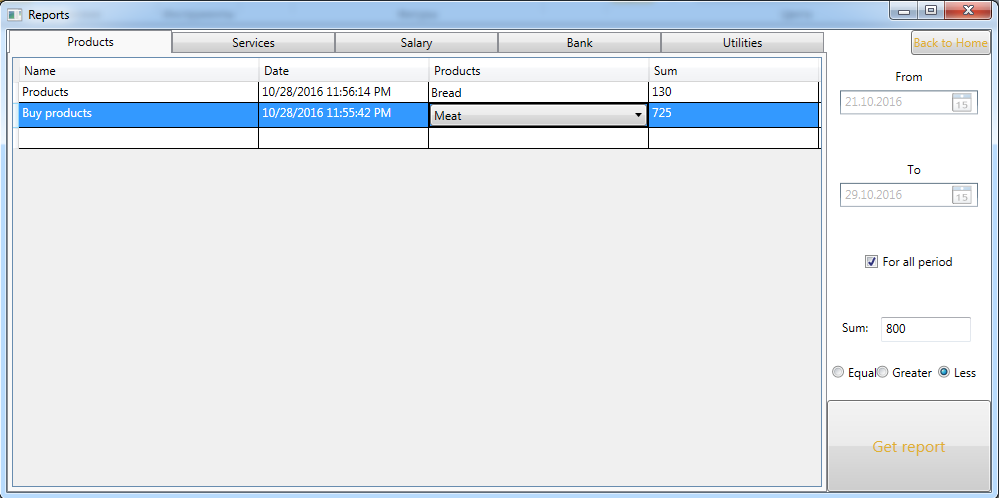
So, this not needs to take more time to explain of that.

Different is a right sidebar, where presents options to the filter of requested reports. It allows you get operations:

- For some date from or date to;

- get operations for all period;

- get operations filtered by the sum. You can select three options for this filter: equal, greater or less, and reported data will contain operations filtered respectively.  
  
After choosing all necessary option clicks on “Get report” button to request the application to get back and show appropriate data (pic.3.2).



pic. 3.2

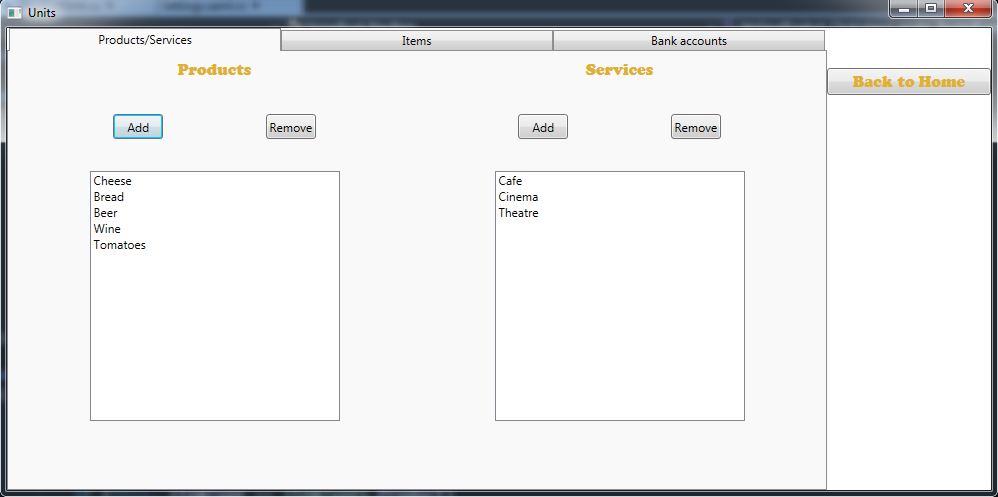
4. Clicking on last “Settings” button from the main window, you will be redirected to the window, where you allow to add product or services, some item, for example, currencies or add a bank account to use all of it in adding operations.

So, this window separates by three tabs:

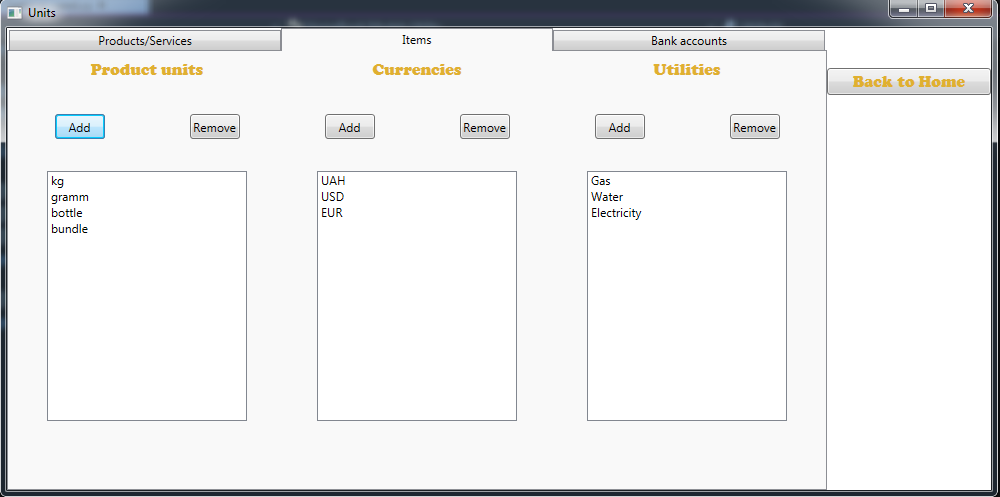
- Product/Services (pic.4.1) – allows you to add brand new products or services or remove that;

- Items (pic.4.2) – where you allow adding or removing product units, currencies, and utilities;

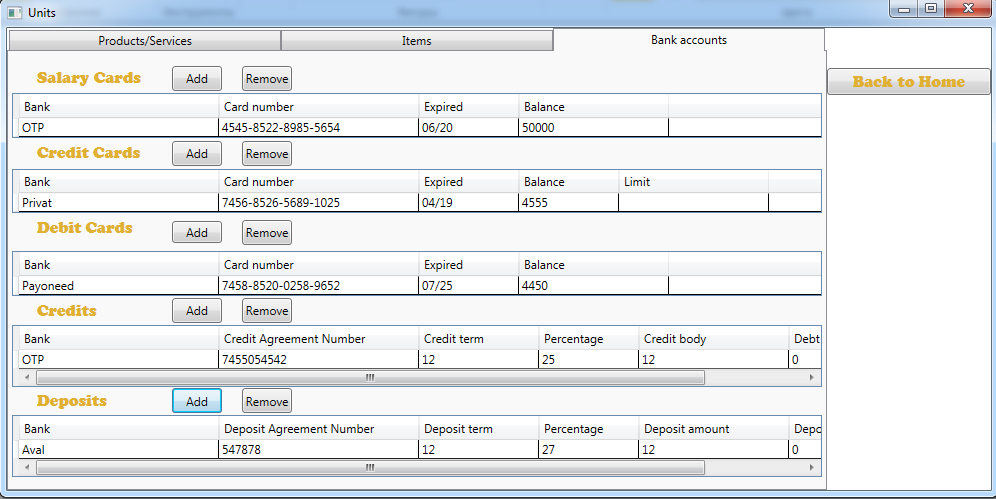
- Bank accounts (pic.4.3) – allow you to add or remove specific types of bank accounts as Salary cards, Credit cards, Debit cards, Credits, and Deposits.



pic. 4.1



pic. 4.2



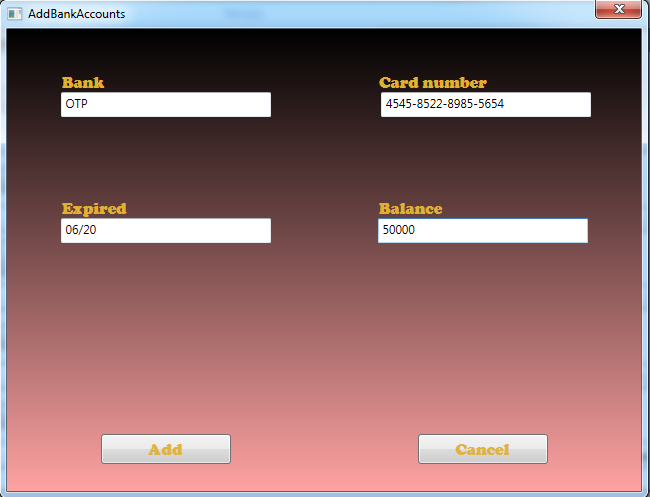
pic. 4.3

Near each of bank accounts table, you can see two buttons that allow you add or remove that respectively.

Add button open separate window with specific to the type of bank accounts fields.

4.1. When you click on Add for Salary cards window opens with next fields (pic.4.4):

* Bank – name of bank where you open your salary card;
* Card number – number of salary card
* Expired – expiration of your salary card;
* Balance – balance available on salary card.

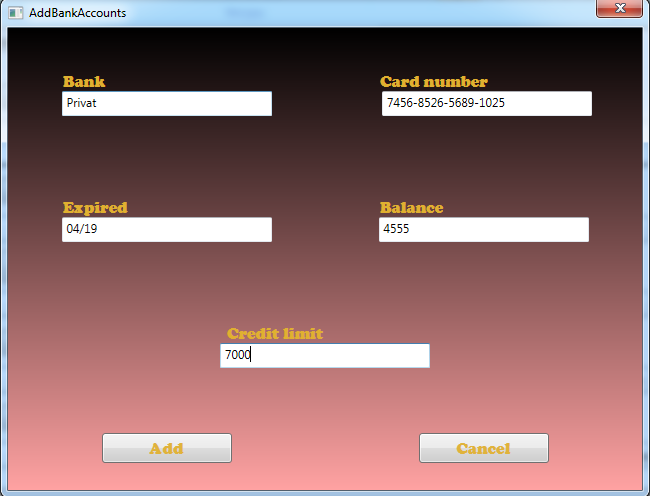


pic. 4.4

**Note: all fields required. Add button will always state as inactive if some of the fields will be not filled!**

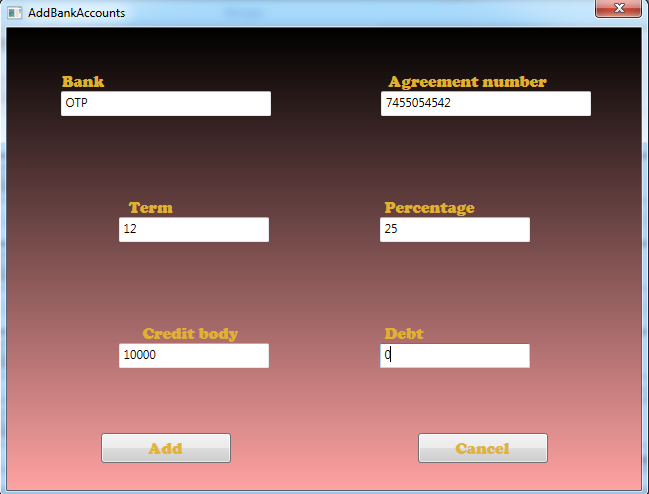
Some fields are general for a different type of bank account.

4.2. For Credit cards in “add bank counts” window specific is “Credit limit” field, which needs to type there limit the amount of credit card (pic.4.5).



pic. 4.5

4.3. The window for Debit cards is the same as for salary cards, so this no need more explanations.  
4.4. Credits add window (pic.4.6) has:

  
pic. 4.6

- Agreement number – number of added credit;

- Term – period for which you take the credit;

- Percentage – percentage of credit amount;

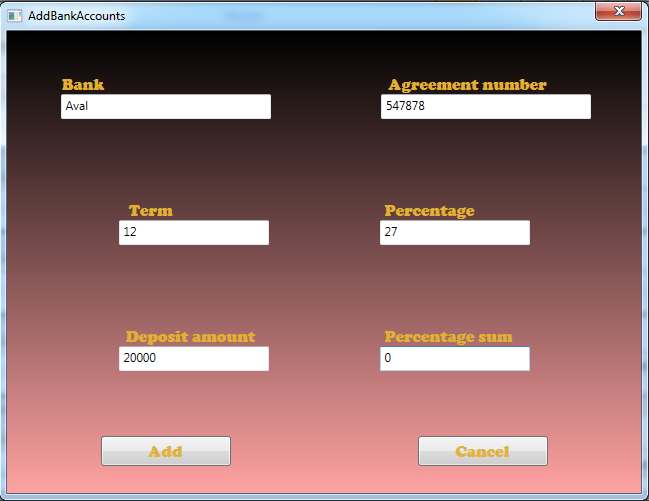
- Credit body – amount of money that taken as credit;

- Debt – remainder of credit, that needs to be repayment;

4.5. Deposit adds window has the same fields as on Credit (pic.4.7), but latest differ:

- Deposit amount – amount of taken deposit;

- Percentage sum – sum of available deposit percentage that accrued on the deposit amount.



pic. 4.7

**Note: All fields presented on a different form for adding bank account are also required and need to be filled in order to add this bank account to the application.**

**II. User Guide**

The application is created used Visual Studio 2015 Community Edition.

The solution contains one project “HomeBook”. Next the short describe of presented parts using in the project:

1. **Client side**
   1. **Windows**

The programming part is windows that cover all application functionality:

* MainWindow.xaml – the first window loaded after running the application. Contain three buttons to direct user to “Operations”, “Reports”, “Settings” windows;
* Operations.xaml – window for loading all added users operations. Include TabControl with five TabItems for Product, Service, Salary, Bank and Utility Operations. For showing, operations used Data grid element.

Right side of the window contains sidebar with three buttons for add, delete selected operation and redirect to the main window;

* Reports.xaml – used for getting and showing filtered operations. Has right sidebar with filter options to get operations by date from, to, for all period, filter by the sum of operation and options to set equal, greater or less than.  
  Window contain elements the same as Operations.xaml: TabControll with the same five tabs and Datagrid to showing operations with the same columns;
* Settings.xaml – window with also TabControl and three tabs: Product/Services, Items, and BankAccounts.

Product/Services used for added products and services that used for adding same naming operations.

Items – for adding products units, currencies, and utilities.

Bank Accounts – for adding credit, debit, salary cards and also deposits and credits.

* AddOperations.xaml – window with form and different fields for adding new operations;
* NewItem.xaml – window with a form for adding new items such as product, service, currency and so on. Used on adding operations, settings windows;
* OperationProduct.xaml – window for adding operation product to currently created operation;
* OperationService.xaml - window for adding operation service to currently created operation;
* BankAccounts.xaml – window with a form for adding bank accounts: credits, salary cards, etc.

All the windows described above use the same XAML grid system with rows and columns for a better view on different screen sizes. Some of the windows are using fixed size, because not necessary for resize it. Also, they have the same design including background color, font, font size.

Some of the window needs to be more explained.

In Operations.xaml “Add new operation” button has function behind that open appropriate window to add operations. These appropriate windows refer to the currently selected tab. So, to open, for example, salary operations, need to select Salary tab and click on that button. Function in controller only check for an index of the currently selected tab and pass this index to the new instance of AddOperation class constructor.

The constructor of AddOperation class takes this index and assign to the property with OperationModel type. And inside OperationModel property as OprationTypeId property is changed that another property that responsible for some operation type set to true and app[rprpiate window is open.

Also, in add operations function to the instance of the AddOperation class add the new instance of the delegate that takes a function to refresh operations after adding one. This delegate is called when in AddOperation class will be called add operations function which raises the Refresh event that belongs to this delegate and RefreshOperations functions in Operation.xaml.cs will be called and appropriate data grid will be updated and contained the new operations.

This template are using also in OperationProduct.xaml.cs, OperationService.xaml.cs and in NewItem.xaml.cs.

Another one thing to consider is that in AddOperation window that belongs to product and service operations there is Total Sum field which bound property Sum is automatically calculated using the sum of all added operation products or service to the data grid.

Also, in OperationProduct window like described above Sum property is automatically calculated when all Amount and Price fields are filled using the model INotifyPropertyChanged interface implementation.

Note that AddOperation window is one form for all five operation types. Some form elements are hide/showed relates to which type of operation is currently open.

These elements are displayed or not depending on which of the related to the type of the operation bool property are setting to true and binding to Visibility attribute of the element.

The same implementation is referred to AddBankAccount window where all elements for all types of a bank account are presented and showing/hidden depending on the type.

So, keep in mind that AddOperation and AddBankAccount windows are one for all five types of operation and all five types of bank accounts.

Also, in Salary and Bank operations on the window presented one element (Combobox) that belongs to the bank accounts, where the user can select some bank account as salary cards or debit/credit cards to use in adding new operation. After Add operation function was called and appropriate operation object was sending to the server in Repo.cs in AddOperation function will be called another function AdjustBankAccount, which in turn check for type of the added operation and type of salary or bank operation type and add or subtracts the operation sum to the bank account card balance, and also check if the sum of the operations is not greater that card balance. So, in Settings in the Bank account tab, all the updated bank account will be changed.

Remove operations function is to send the id of the selected in some of the data grids operation to delete from the database. On the successful result, it updates the appropriate data grid where this operation was selected with a new request to get operations. So, in the data grid, this operation will be deleted too.

All the important aspects of the current implementation and some code are described above…I hope!

* 1. **ViewModels.**

As application in main part developed using MVVM design pattern, the solution includes ViewModels folder where you can find c# classes used for different windows views to present some of the objects as products, bank accounts, etc:

* BankAccountModel.cs – this model describes Bank accounts for different it types and bound to AddBankAccount window view. All the fields are bound to appropriate fields in window form. The most important is BankAccountId, which after changing change type of the bank account; AllFieldsFilled – used as Boolean to check if all the window forms are filled by the user and bound to Add button to enabled/disabled this one to prevent adding not fully filled form and to save not completed object to the server;
* BankAccountTypeModel.cs – contains two (BankAccountTypeID and TypeName) fields to describe bank account type;
* OperationModel.cs – class that describe operation object and bound to AddOperation window view. Contains many properties that explain different fields for different type of operations. OperationTypeId is the property which required for operation type and changing this change the operation types as set true to one of the appropriate Booleans as IsProductOperations, IsSalary and so on, to shown the right form on the window. Contains the same as in BankAccountModel.cs property AllFieldsFilled to check for filled all the required fields.
* OperationProductModel.cs – describe operation product object and bound to OperationProduct window form. Set function of two properties Amount and Price include function (CalcSum) to automatically calculate the sum of the added operation product for Sum property and bind to the same field on the form;
* OperationServiceModel.cs – describe service that added to the service operation. Contain the known AllFields filled and properties to explain different service object fields;
* OperationTypeModel.cs – contain two files (OperationtypeId and Name) to describe the type of added operation;
* ReportModel.cs – describe the object that sending to the server as a request for reporting operation. Contains fields that explain report filter options.

As application use MVVM design pattern all models described above implemented INotifyPropertyChanged, so all changes provided on UI or behind automatically updates the model's property to the object or to the view.

* 1. **Converters**

In some cases, in the application used converters built-in BooleanToVisibilityConverter to convert bool type of some properties of some MVVM models that bound to form element to Visibility type of that element and own implemented converter BoolTurnConverter.cs that included in Converters folder to return some Boolean property value to the opposite.

* 1. **Extensions**

Also in some window view used binding property AllFieldsFilled to check if all the required fields are filled and to prevent sending to the server object without all necessary properties. So, to check all necessary properties are filled used special functions that check if the property contains some data and check also for the type of that property.

This functions are call extensions and stored in Extensions.cs in Extensions folder.

Extension class includes four static boolean type properties to check if some properties contain data:

* IsCleanText – to check string property;
* IsCleanNumber that has three overloads to check double?, int? and int.

1. **Server side**
   1. **Models**

Next folder used in the project is Models which includes the models as entities to describe all object that sent to the server side of the application and used by Entity Framework to store them to the database.

This folder contains more models that in ViewModels as this is not the same and not all the models that used in the application are bound to window forms.

It's unnecessary to describe all the fields of the Entities, its contain all the properties that required for different objects that used in adding operations or some items. So, Models folder include:

* BankAccount.cs – for bank accounts;
* BankAccountType.cs – for bank account types;
* BankOperationType.cs – for type of bank operations;
* Currency.cs – for add currencies;
* Operations.cs - for operation object;
* OperationProduct.cs – for adding operation product;
* OperationService.cs – for adding operation service;
* OperationType.cs – for type of the added operation;
* Product.cs – for adding product. Difference from OperationProduct model in that Product is the single object contain id and name of the product and OperationProduct are contain the Product itself and price, amount and sum of the added product to operation;
* ProductUnit.cs – for adding unit that used in product operations for adding operation product;
* RequestResult.cs – this model contain two properties bool Result and string Message that used as response object that returned from server side functions to UI;
* SalaryOperationType.cs – used for operation type of salary operation;
* Service.cs – for adding services. It’s the same difference from OperationService as between Product and OperationProduct;
* Utility.cs – for adding utilities for used in utility operations.
  1. **Data access**

Next server side folder in the application is DataAccess. Classes included int this folder are used for access, save and getting all data that stored in the application database. This folder contains:

* HomeBookContext.cs – this class used by EF is described all presented database tables. So, the name of the class property is the same name of the database table;
* Repo.cs – contain all the functions that used to get, save and update data to the database directly using LINQ and EF. These functions are not difficult and big, so not need to deep dive in explanation of their body. You may see in Repo.cs functions using context as an instance of HomeBookContext and from there using the property of this instance as database tables. To work with data in functions used Language Integrated Query (LINQ) with Lambda function as a part of Entity Framework.
  1. **Migrations**

In application used migrations feature of Entity Framework updates some changes provided in HomeBookContext.cs and so, on the database structure.

These migrations configured in Configuration.cs in Migrations folder.  
Configuration.cs contain class Configuration, that has a constructor that config the migrations and Seed function that initial save to database some data. These initial data in app stored in OperationsTypes, BankOperationsTypes, BankAccountTypes and SalaryOperationsTypes.

These data are added by the user so it’s added here.

* 1. **Enumerable**

In all application used only one enum for item types to indicate the type of added item type. As you, items used in the app are products, services, currencies, product units, and utilities.

So, in Enums folder there is only one enum ItemTypes.cs.