UDC: 004.4

PROGRAMMATIC SOLUTIONS FOR BUSINESSES THAT HAVE MULTIPLE ACCOUNTS

ПРОГРАМНІ РІШЕННЯ ДЛЯ БІЗНЕСУ, ЯКИЙ МАЄ КІЛЬКА АККАУНТІВ

[1]Pohorieltsev P.M., [2]Boltach S.V., [3]Popel O.V.
[1]Погорєльцев П.М., [2]Болтач С.В., [3]Попель О.В.

[1,2, 3] Odesa National Technological University, Odesa, Ukraine
ORCID: [2]https://orcid.org/0000-0003-1902-2405, [3]https://orcid.org/0000-0001-8236-7794
E-mail: [2]boltach.svetlana@gmail.com, [1]pashafotograph@gmail.com, [3]ksenjapopel2580@gmail.com

DOI: https://doi.org/10.15673/atbp.v16i2.2844

Abstract. *Since ancient times, people have used various types of monetary units to perform various trade transactions. A large amount of money is stored in certain computer systems. Accounting for such units is no less important than accounting for physical money, and because of this, banks are developing client applications. However, almost all client banks have one significant drawback for business users, namely the lack of a centralized place to track information on all their accounts in different banks. To make sure that a large number of people need to view accounts in different banks, it is enough to look at the statistics of the NBU (National Bank of Ukraine) and UkrStat (State Statistics Service). It is not uncommon for entrepreneurs to open not only several accounts, but also in different banks. Which makes it difficult to keep records without some kind of system where you can see all transactions in one place. This work is devoted to the development of a software solution for a business that has several accounts. Also, this development can be used by individuals who have several accounts in different banks. This work covers some of the practical challenges that a business may face and offers one of the solutions to them by creating a desktop application that allows you to monitor multiple bank accounts in one place, providing vital data such as transaction history and recent balances, allowing you to add personal notes and store data locally on the operator's machine. Software development is carried out according to the SDLC model.*

Анотація. *З давніх часів люди використовували різні види грошових одиниць для здійснення різноманітних торгових операцій. Велика сума грошей зберігається в певних комп'ютерних системах. Облік таких одиниць не менш важливий, ніж облік фізичних грошей, і тому банки розробляють клієнтські додатки. Однак майже всі банки-клієнти мають один істотний недолік для бізнес-користувачів, а саме відсутність централізованого місця для відстеження інформації про всі їхні рахунки в різних банках. Щоб переконатися у необхідності перегляду рахунків у різних банках великій кількості людей, достатньо переглянути статистику НБУ (Національний банк України) та Укрстату (Державна служба статистики). Нерідкі випадки, коли підприємці відкривають не тільки кілька рахунків, але і в різних банках. Це ускладнює ведення записів без якоїсь системи, де ви можете бачити всі транзакції в одному місці. Дана робота присвячена розробці програмного рішення для бізнесу, який має кілька облікових записів. Також цією розробкою можуть скористатися фізичні особи, які мають декілька рахунків у різних банках. Ця робота охоплює деякі практичні проблеми, з якими може зіткнутися бізнес, і пропонує одне з їх вирішення шляхом створення настільної програми, яка дозволяє відстежувати кілька банківських рахунків в одному місці, надаючи життєво важливі дані, такі як історія транзакцій і останні баланси, дозволяючи ви можете додавати особисті нотатки та зберігати дані локально на машині оператора. Розробка програмного забезпечення здійснюється за моделлю SDLC.*

Keywords: Finance, Banking, Business, Desktop Program, Tracking, WinUI, SDLC.
Ключові слова: фінанси, банківська справа, бізнес, настільна програма, відстеження, WinUI, SDLC.

I. INTRODUCTION

Since the ancient times, at least, as far back, as the Ancient Greece, there existed some sort of business in societies. In Ancient Greece, it was craftsmanship and guilds, that produced specific products, that not everyone was able to make, and to exchange their goods with other people, money was used.

Money is a very valuable tool in any society, especially – in our modern capitalistic world. But as the time went on,

and our societies progressed, developing into modern governmental structures with control and regulations over every aspect of human life, as well as increased speed of transactions in businesses, due to the development of technologies, and overall wellbeing being far better than even a century ago, said businessmen were facing a problem.

That problem is – keeping track of their incomes, expenses and balances, that they have at their disposal. Previously, they were able to just count up their assets, like golden coins, or livestock, but as the business grows, it becomes harder and harder to keep up. At some point, the physical money became less frequently used, and bank checks and other means were used in business-to-business interactions, and this made tracking of money more difficult, since you now had to account for more types of "Money", or wealth.

This all leads us to current days, when many businesses use some sort of banking account, acquiring, ecommerce and so on. Some companies may not even have a cent in physical money, and all their wealth is stored in various banks, in a safe and 24/7 accessible place, that allows you to send and receive funds with ease. So, what problems businessmen nowadays may even have?

The biggest problem is access to all of this information from a single place, with the possibility to share it with managers on accountants, that may require such details to process customer's requests.

And this is where this paper comes into play. If we somehow could solve this problem, we'd increase the productiveness of businesses, and provide a possibility to increase customers satisfaction as the result.

Let us take a more in-depth look into the problem in the next section.

II. LITERATURE ANALYSIS

This section will analyze the problems at hand, take a look at statistical data, and come up with steps to fix, or at the very least, make said problems less prominent.

It is worth mentioning that in this paper we will be looking at the Ukrainian market, since this is where the paper's team is located at, and it's a good place to start, due to what the Ukrainian market is, as will be explained later on.

2.1. Analyzing the market

If we take a look at data from the National Bank of Ukraine and National Service of Statistics of Ukraine (Ukrstat), we can see that there are about 2.9 million business clients in banks of Ukraine, and 1.956 million active businesses, at the beginning of year 2022 (newer data is not yet available, probably due to the ongoing invasion into Ukraine's territory). From this, we can conclude that every business makes 1.48 bank clients. [1] [2]

This means that every other business in Ukraine has at least 2 accounts, more specific research and inquiries are required to be more specific, but we can use other data from those infographics to also conclude that every business has 2.86 accounts in banks, regardless of the fact if they are in the same bank, or in different ones.

Having this data, we can be rest assured, that, at the very least, there are businesses that may be experiencing some issues, while working with multiple banks.

2.2. Analyzing the problem of having multiple banking accounts

As we can see – some businesses, have multiple banking accounts, and working with multiple accounts, even in one bank, is no trivial task. As an accountant – you have to make sure that every transaction has gone into the right account, and when you are making financial reports, you have to make sure to properly combine data from two sources, otherwise – you might be at risk of being called out on committing tax fraud.

And when a company decides to use multiple banks, they are making this work even harder. The main issue is, of course, that there's no centralized source of information, like, for example – one application in which you can add multiple accounts. This causes you, as a manager, or an accountant to use multiple applications, sites, or some other form of program to access information. This causes operators to be distracted, frustrated, their computers get cluttered with different sites, or applications, and the information is now distributed across multiple sources, this may slow down search and confirmation of payment, for example.

Those problems are a great start for some requirements for the software (so called SRS, Software Requirements Specification). For example – we can say that the software has to provide the following functionality:

- Ability to add multiple accounts;
- Ability to add accounts from different banking services;
- Ability to get Balances and Transactions history;
- Ability to view said information for all accounts, or per account (a.k.a. — filtering);
- Ability to search in history of transactions, by comment/description of payment, account, and date time of transaction;
- Ability to search in balances by date and account;

2.3. Analyzing the banking services

Let us analyze what banking services currently provider their customers with. For example – we can analyze PrivatBank and Monobank services, to see, if businesses have some tools for multi-account, and multi-banking.

Starting from PrivatBank. They provide API for legal entities and private entrepreneurs, and Autoclient application for desktop "Автоклієнт". [3]

Autoclient application, is basically a wrapper for API, so we can just take a look at the API, to find out, what is possible for a client of PrivatBank service. We can see that it's possible to fetch data for as many accounts, as we have the keys for. But only for PrivatBank.

Monobank's API provides functionality to get information about accounts, for both – regular users (even non-business accounts), and for businesses. This is great for tracking personal expenses, but it also doesn't allow us to get data from different banks. [4]

This means that those banks do not provide one API for all banks, neither does the National Bank of Ukraine (NBU), and from this, we can only conclude that there is no universal aggregation service, even just for Ukraine. [5]

But, since API documentation is public for said banks, as well as some other (for example – A-Bank), this means that it's theoretically possible to create an application, that allows to fulfil requirements we have outlined in section 2.2.

III. OBJECT, SUBJECT, AND METHODS OF RESEARCH

The object of research of this paper is a solution to some problems of workers of companies, that use multiple banking services, or accounts, with which workers have to interact.

The subject of research is difficulties themselves, that we have outlined in section 2 of this paper.

Methods of research include: software development (as the main tool for solving the problems at hand), analysis of workers' routines and of available tools to solve them.

IV. RESULTS

4.1 Creation of the application

Creating some application is a complicated process. After specifying the requirements, software engineers, like building engineers, have to create a "Plan" of the construction. In software development, this plan is called the architecture.

Architecture describes how the system will be built, or how it was built. It has to be precise, and verbose, specifying the technological stack of the application. For example – which programming languages and frameworks will be used to create the product. Summarize how it operates, describing the flow of data, some complex workflows, etc.

To develop this application, we used the following languages: C#, XAML, JSON. During the development of the application, we used WinUI 3 SDK (Software Development Kit) made by Microsoft, which allows us to create native Windows applications. [6]

To describe the architecture, we have to use diagrams. In figure 1 you can see how the app works in general. The application is composed of UI elements, processing logic (not shown on the diagram), and a database. The user is interacting with the application, and the operating system (OS) reports actions to the app. The app itself also makes external requests (blue section) to banking services, to fetch and store data from them (like balances, transactions history, etc.).

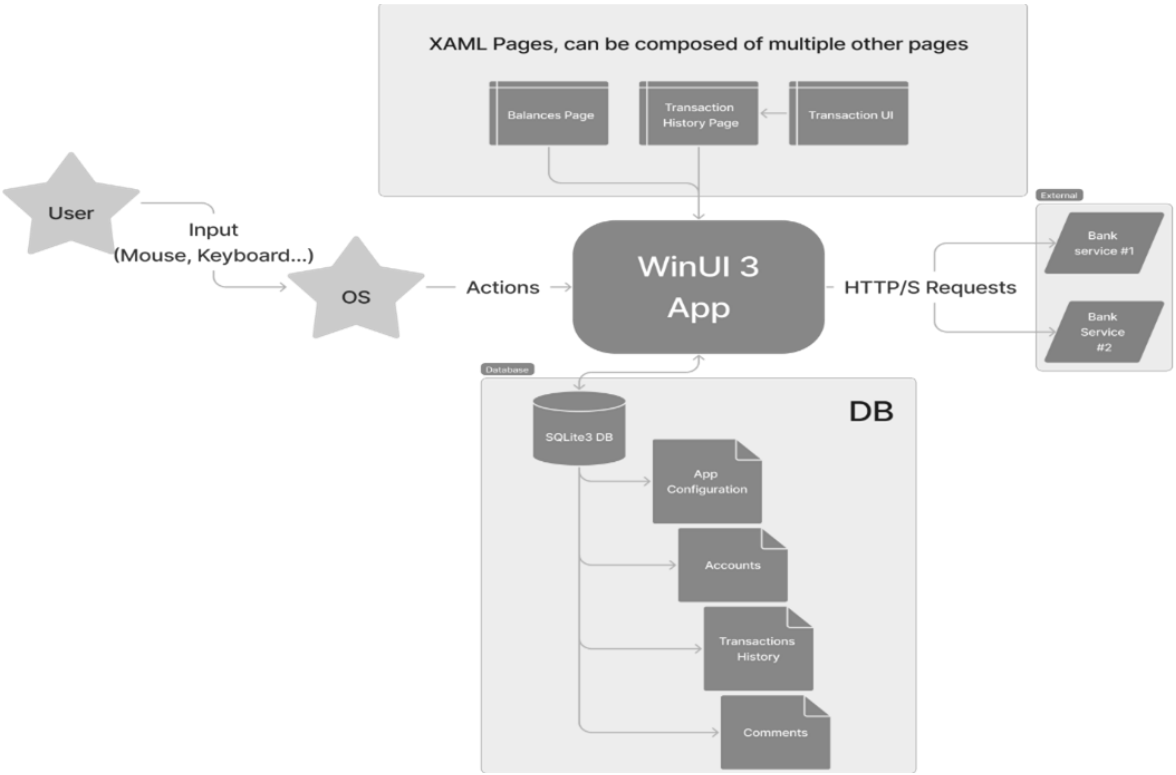

Fig 1. The overall architecture
Рис 1. Загальна архітектура

On figure 2 you can see the structure of the database, shown as a mind map. Colors indicate to which database does this field reference. For example – in the Transaction Comments table there are two colored fields – Transaction ID and Comment ID, this means that it contains reference from two tables – Comments and Transactions, creating a Many-To-Many connection between the two.

Let us also introduce some information on how the app is made. Architecture can only say so much about the product, and it's the developer's job to make concrete solutions (code) from those "blueprints".

The application is based on the MVVM, a pattern which stands for Model-View-ViewModel. And basically, means that every View, visual element of the app, for example: a page, a transaction record, is related to a ViewModel, the intermediary object, that combines a Model and a View, and the Model itself – the data, represented by some object in

the program. [7]

For example – let's take a look at how the Accounts page looks and works. As a user, we get to see a page with a button to add a new account, a list of accounts, and for each account we have a set of actions (Edit, Delete). In the program, however, we have to decompose our application.

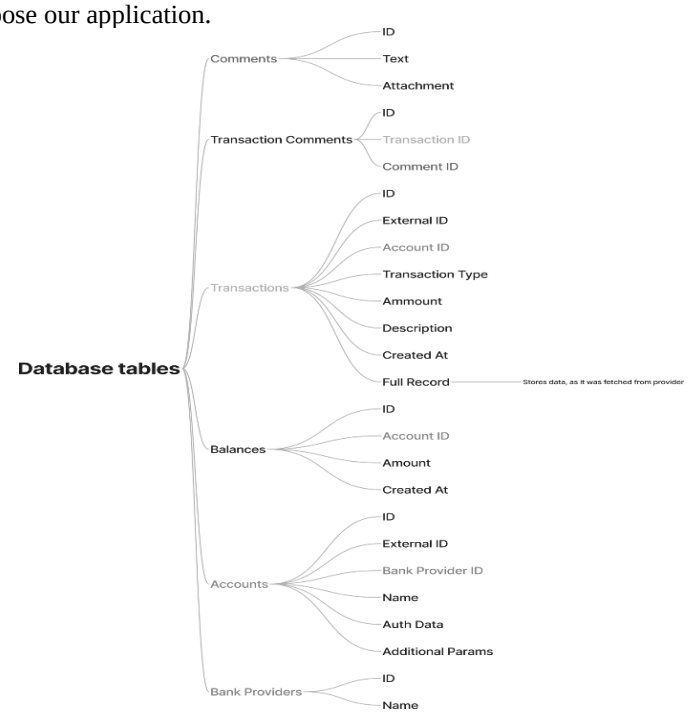

Fig. 2 – Structure of the database
Рис. 2 – Структура бази даних

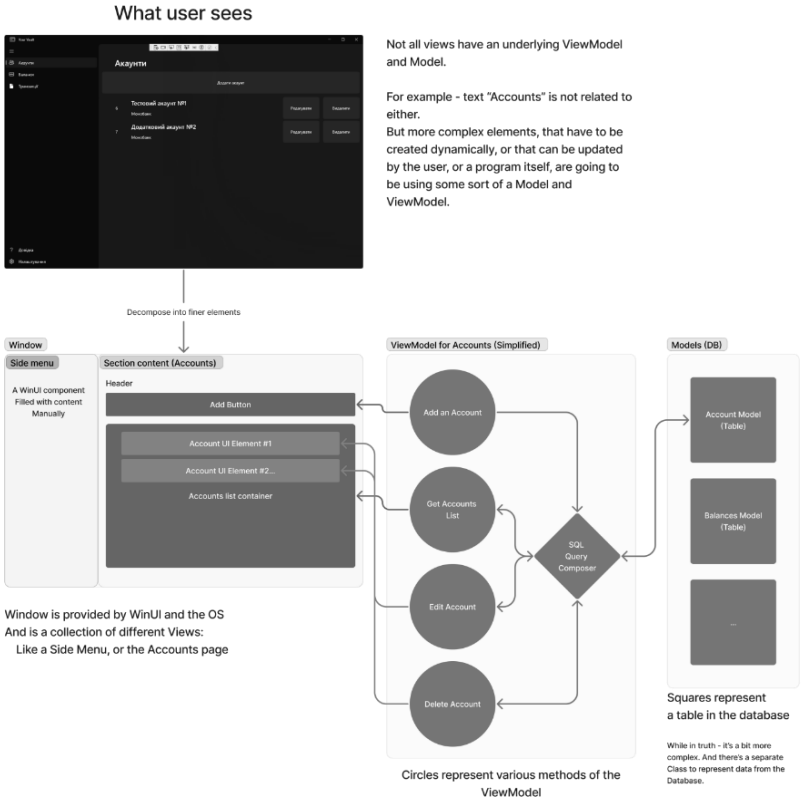

Fig. 3 – A schematic overview of how the app operates
Рис. 3 – Схематичний огляд роботи програми

More complicated relations of Views Models and ViewModels are difficult to express without some visual aid, so on the figure 3 you can see their connections in the application.

In the application, there are also things that user cannot see, but that are crucial to the operation of the app. In our case, it's the scheduler that fetches data from banking services, and updates the database, and in turn – the content of the Views.

Showing this visually would be of no benefit, so, let us instead show the code, which makes this happen.

In the current form, it works as shown in figures 4 and 5.

When initializing the app - we initialize our component (App), then - the database, and we configure services;

```
36  public partial class App : Application
37  {
38
39      private static Timer timer;
        3 references
40      public static Microsoft.UI.Dispatching.DispatcherQueue DispatcherQueue { get; private set; }
41
        1 reference
42      public App()
43      {
44          this.InitializeComponent();
45
46          DataAccess.InitializeTables();
47
48          ConfigureServices();
49
50          timer = new Timer(TimeSpan.FromSeconds(60).TotalMilliseconds);
51          timer.Start();
52          timer.AutoReset = true;
53          timer.Elapsed += new ElapsedEventHandler(UpdateTransactions);
54      }
```

After this - we set up a timer to run every 60 seconds (later this will be configurable, if so desired), that will execute the UpdateTransactions method.

Fig. 4 – Setting up the application
Рис. 4 – Налаштування програми

This setup is required, since without it, the timer would have to be run by the user. By running a timer in the App class initialization method, we ensure the operation of it, so, user won't be frustrated because they forgot to start the fetch scheduler.

But to entirely comprehend how this code works, we have to look at figure 5, which shows and explains the code of the UpdateTransactions method.

In layman's terms, this code goes through all the accounts, and asks the banking service "Do you have some new transactions", by sending a request (or a series of requests), if there are some new transactions – it stores them to the database, then – updates the balance, if required, and exists.

UpdateTransactions method does the following:
It get's services (special containers that allow us to interact with the database from different places of the app, while having same content everywhere).

Then - we get all the accounts, and iterate trough each of them.

```
56  1 reference
57  private static void UpdateTransactions(object sender, ElapsedEventArgs e)
58  {
59      var _accountsService = (IAccountService)((App)Current).ServiceProvider.GetService(typeof(IAccountService));
60      var _transactionService = (ITransactionService)((App)Current).ServiceProvider.GetService(typeof(ITransactionService));
61      var _transactionUpdateService = (ITransactionUpdateService)((App)Current).ServiceProvider.GetService(typeof(ITransactionUpdateService));
62      var _balanceService = (IBalanceService)((App)Current).ServiceProvider.GetService(typeof(IBalanceService));
63
64      var accounts = _accountsService.GetAccounts();
65      foreach(var account in accounts)
66      {
67          if(account.bankProvider.Name == "МоноБанк")
68          {
69              var integration = new MonoBankIntegraton(account);
70              List<Models.Transaction> transactions;
71              try {
72                  transactions = integration.GetNewTransactions();
73              }
74              catch (Exception ex)
75              {
76                  Console.WriteLine(ex.Message);
77                  continue;
78              }
79
80              foreach(var transaction in transactions)
81              {
82                  _transactionService.AddTransaction(transaction, false);
83              }
84              if (transactions.Count > 0) {
85                  var lastTranasction = transactions.First();
86                  _balanceService.InsertBalance(new Models.Balance(0, account.ID, lastTranasction.CurrentAmount, lastTranasction.CreatedAt));
87              }
88              App.DispatcherQueue.TryEnqueue(() => _transactionService.UpdateTransactions());
89          }
90          else
91          {
92              Console.Write("Unsupported bank provider");
93          }
94      }
95      App.DispatcherQueue.TryEnqueue(() => _transactionUpdateService.SetUpdateTime());
96  }
```

If it's a supported provider - we execute the processing code (lines 67-88), if it's not - we just log that this provider is not supported (lines 89-92), and we do so, for every account in our list.

On lines 67-88 we call the integration class for this account, and fetch new transactions. (lines 68-77).
Then - we add them all to our database (lines 79-82).
And if we have more than 0 transactions - we add a new balances from the latest transaction. (lines 83-86)

Fig. 5 – Showcase and explanation of UpdateTransactions
Рис. 5 – Демонстрація та пояснення UpdateTransactions

4.2 Deployment of the product

Deploying the product means making it available to the end user, in a way they can use it.

Deployment steps for this product include: creating a release, pushing the package file to GitHub; User has to download the package, install it and run the application.

On the repository page, there exists a “Releases” section, in which you can find different versions of the application. For example – RC 3 (Release Candidate 3), in this release – you can get all the required files and the source code of the project at the moment of release. [8]

As a user – updating is a simple process of getting a newer (or older) version of a program, and hitting “Update” (or “Reinstall”) in the newly opened window. See figure 6.

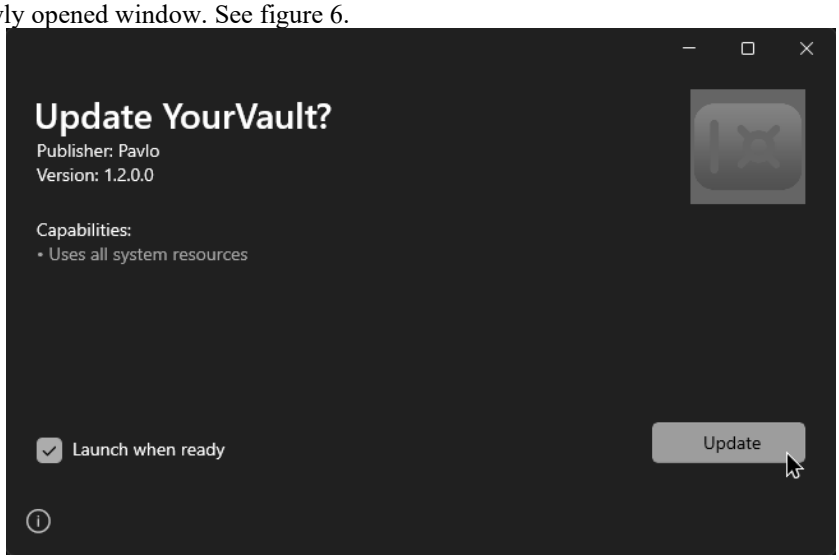

Fig. 6 – App Installer window, allowing user to update the app
Рис. 6 – Вікно інсталятора програми, що дозволяє користувачеві оновлювати програму

After installing, or updating the app – users are free to explore it, for example by opening the Help section of the app – they can find out how to use the app with the banking service they use. You can see how the help section looks for Monobank in figure 7.

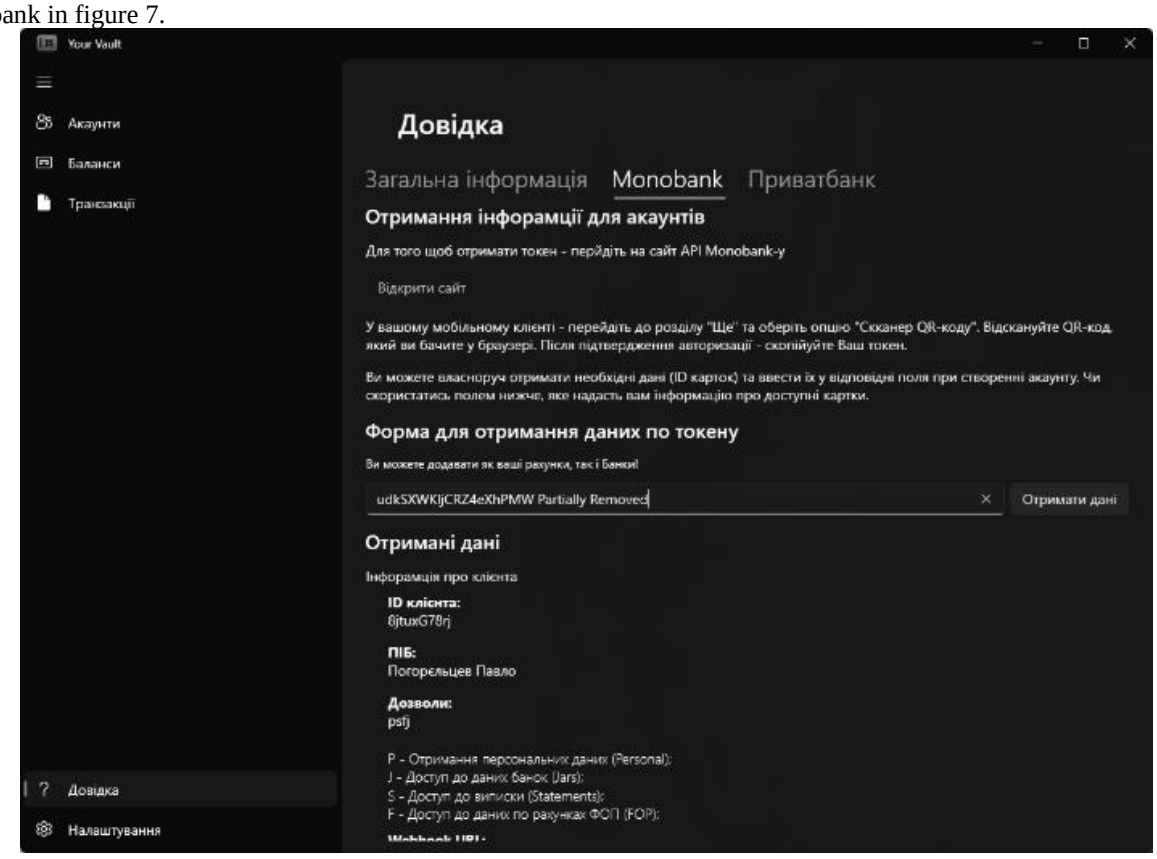

Fig. 7 – Help section for Monobank
Рис. 7 – Розділ довідки для Monobank

V. CONCLUSIONS

In conclusion – we can say that technologies always push humanity forward, providing new opportunities, better quality of life and other benefits, but with them – come difficulties, however, we can resolve those difficulties by creating newer products, or technologies.

In this concrete example – we have created an application that can help small businesses thrive, since their workers will be relieved in one aspect of their work – keeping up with incoming and outgoing transactions.

By using GitHub as both – development, distribution and communication platform, we can improve the imperfections of this product, and make it even better for everyone using it.

Список використаних джерел

1. НБУ, «Кількість клієнтів та рахунків в банках» 2023: вебсайт. URL: https://bank.gov.ua/ua/news/all/kilkist-kliyentiv-ta-rahunkiv-v-bankah-2021-rik. [дата звернення: 02.12.2023].
2. УкрСтат, «Кількість суб'єктів господарювання за видами економічної діяльності» 10.10.2023: вебсайт. URL: https://www.ukrstat.gov.ua/operativ/operativ2012/fin/osp/ksg/ksg_u/arch_ksg_u.htm. [дата звернення: 02.12.2023].
3. PrivatBank. (n.d.). Інструкція для роботи з додатком «Автоклієнт»: вебдокумент. URL https://docs.google.com/document/d/e/2PACX-1vS8rx2WKg69o6JvG5L4AhSXcU6vxXcJph6WK84qJcAYDBvsNYEob57jDMQhbosjc9gRS5bOTqTXf0vb/pub#h.nqpje6ikfhcq [дата звернення: 02.12.2023].
4. Monobank. Monbank open API: вебсайт. URL: https://api.monobank.ua/docs/ [дата звернення: 02.12.2023].
5. National Bank of Ukraine. National Bank of Ukraine, Main Page: вебсайт. URL: https://bank.gov.ua/en/ [дата звернення: 02.12.2023].
6. Microsoft corporation. Windows UI Library in the Windows App SDK (WinUI 3). (Microsoft) : вебсайт. URL: https://learn.microsoft.com/en-us/windows/apps/winui/winui3/ [дата звернення: 24.12.2023].
7. Microsoft corporation. Model-View-ViewModel (MVVM). (Microsoft): вебсайт. URL: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm [дата звернення: 24.12.2023].
8. П. М. Погорєльцев, «Репозиторій YourVault,» 10.11.2023: вебсайт. URL: https://github.com/makisukurisu/YourVault. [дата звернення: 24.12.2023].

References

[1] National Bank Of Ukraine. (n.d.). Кількість клієнтів та рахунків в банках. (2023) [Online]. Available: https://bank.gov.ua/ua/news/all/kilkist-kliyentiv-ta-rahunkiv-v-bankah-2021-rik
[2] Ukrstat. (n.d.). Кількість суб'єктів господарювання за видами економічної діяльності. (2023) [Online]. Available: https://www.ukrstat.gov.ua/operativ/operativ2012/fin/osp/ksg/ksg_u/arch_ksg_u.htm
[3] PrivatBank. (n.d.). Інструкція для роботи з додатком «Автоклієнт». [Online]. Available:https://docs.google.com/document/d/e/2PACX-1vS8rx2WKg69o6JvG5L4AhSXcU6vxXcJph6WK84qJcAYDBvsNYEob57jDMQhbosjc9gRS5bOTqTXf0vb/pub#h.nqpje6ikfhcq
[4] Monobank. (n.d.). Monbank open API. [Online]. Available: https://api.monobank.ua/docs/
[5] National Bank of Ukraine. (n.d.). National Bank of Ukraine, Main Page. [Online]. Available: https://bank.gov.ua/en/
[6] Microsoft corporation. (2023, 06 22). Windows UI Library in the Windows App SDK (WinUI 3). (Microsoft). [Online]. Available: https://learn.microsoft.com/en-us/windows/apps/winui/winui3/
[7] Microsoft corporation. (2022, 11 04). Model-View-ViewModel (MVVM). (Microsoft). [Online]. Available: https://learn.microsoft.com/en-us/dotnet/architecture/maui/mvvm
[8] Pohorieltsev, P. (n.d.). makisukurisu/YourVault - GitHub repository. (GitHub). [Online]. Available: https://github.com/makisukurisu/YourVault/

Отримана в редакції 13.05.2024. Прийнята до друку 24.05.2024. Received 13 May 2024. Approved 24 May 2024. AvailableinInternet 30 July 2024