Explanatory note

**Program name:** VK Info Analyzer

**Topic of the project**: The user is supposed to grant access to his VK profile, so the application can get some information. The program analyzes the information, based on user’s preferences in groups and public pages. Finally, the app generates an informative chart and diagram block.

**Annotation**: this is a Standalone application for analyzing profile data in vk.com. The program firstly must be started by signing up with VK profile. User can choose from Facebook and VK profiles, though Facebook part is not developed yet. After signing up the user has options to analyze profiles from his own friends list or by entering a unique ID of a profile. After picking the profile “Analyze” button should be pressed, and after some time (querying and calculations) the result is shown in the main window. The result is based on favorite groups and public pages. Only pages and groups with more than 5000 subscribers are analyzed. The app has list of all such pages in VK grouped by categories. Also user has options to view profile photo and add profile to a database – some kind of a favorite list, where only analyzed profiles can be displayed.

**Central repository address:** <https://github.com/menovikov/VKAnalyzer.git>

**Team members:**

* Novikov Maksim – main logic, tests, api
* Bondarenko Georgy – UI and UX, api, calculations logic
* Prokhorov Nikita - Database

**Description of classes:**

The main logic of the project is split between 6 classes and 3 windows. The project from the beginning was supposed to handle only VK queries and support only its functions, though our team decided to expand the program’s possibilities.

VkRepository – main class for logic in the application. All queries, which are executed to vk.com api are handled here. So are calculations. The class implements all features of IRepository interface which allows it to be a repository for such app. The class implements Singleton pattern – there is no possibility to create a second instance of the class. This is very important due to the fact that each query gets its own piece of information, which builds up the whole information context. The repository has methods and properties which allow it interact with UI.

FacebookRepository – second supported repository, which doesn’t have logic in it yet. Its existence is needed here to show that our team is capable of working with Facebook api in near future.

IRepository – interface for supported repositories. All APIs usually require authorization – app identification, scope of rights which allow the app to execute queries, information about the user and so on. Though the repository doesn’t implement Singleton pattern: sometimes it might be more convenient to use static methods for queries (the repositories are supposed to interact with each other). In the future this might be revised.

RepositoryFactory – a class that implements factory pattern. It is supposed to create instances of repositories. Now it is not used because the interface doesn’t implement Singleton, so creating is not needed now, but in future it might be important, so the class was added.

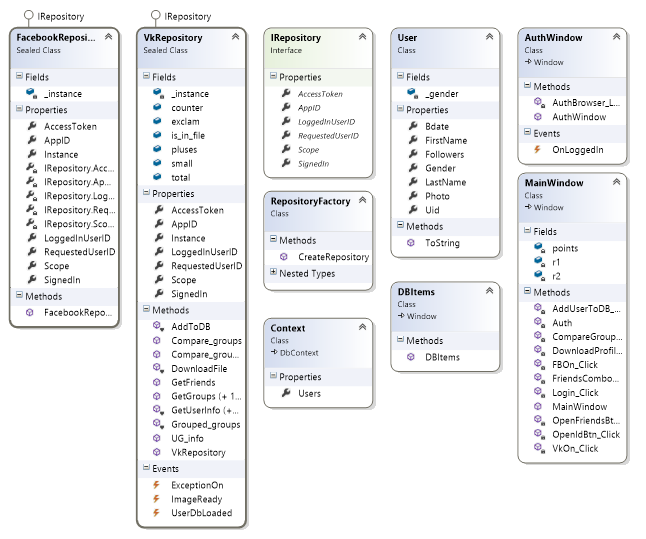
Context – class for interacting with databases. The app creates only 1 table, so there is no execution of relational model. Now this class allows user to see what profiles he analyzed before.

User – class that represents profiles of social network users. It has all fields that a profile has and which are important for analyzing data. This class is used as a data-transfer-object, as a logical class and construction for sql table. Fields are mostly of string type – this is based on the fact that queries work with string values, and there is no need to convert types. The class follows principles of object-oriented programming.

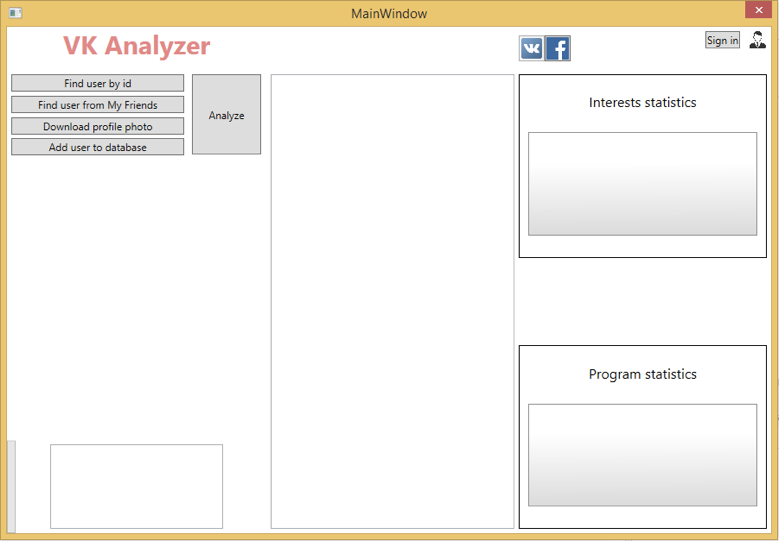
AuthWindow – a window class that has only 1 feature – browser – for authorizing in the system and extracting tokens.

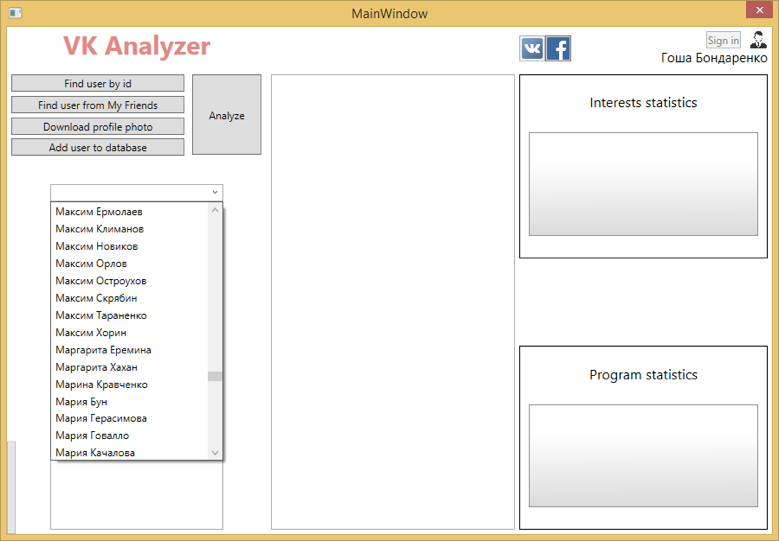
DBItems – window class that has 1 item – data grid – for representing objects (user profiles) which were queried from database after adding 1 more user to it.

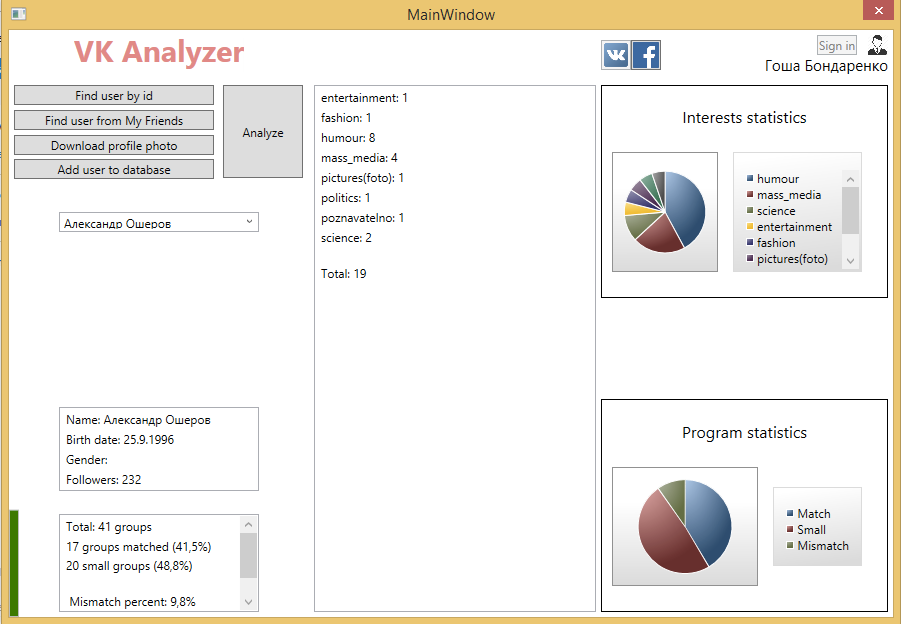
MainWindow – class that is supposed to handle 95% of UI and UX. Unusually it has 2 instances: VkRepository.Instance and Facebook.Repository. This is because factory pattern is not properly implemented yet, so this is a kind of a simple analogue to factory pattern. All other logic that is placed inside MainWindow class deals with regular for a standalone app staff.



**Program interface**

Start interface:

«Sign in» button pressed:

Chose one of your friends, whom profile do you want to analyze:

**1**

**4**

**2**

**3**

**5**

1 - General user’s information.

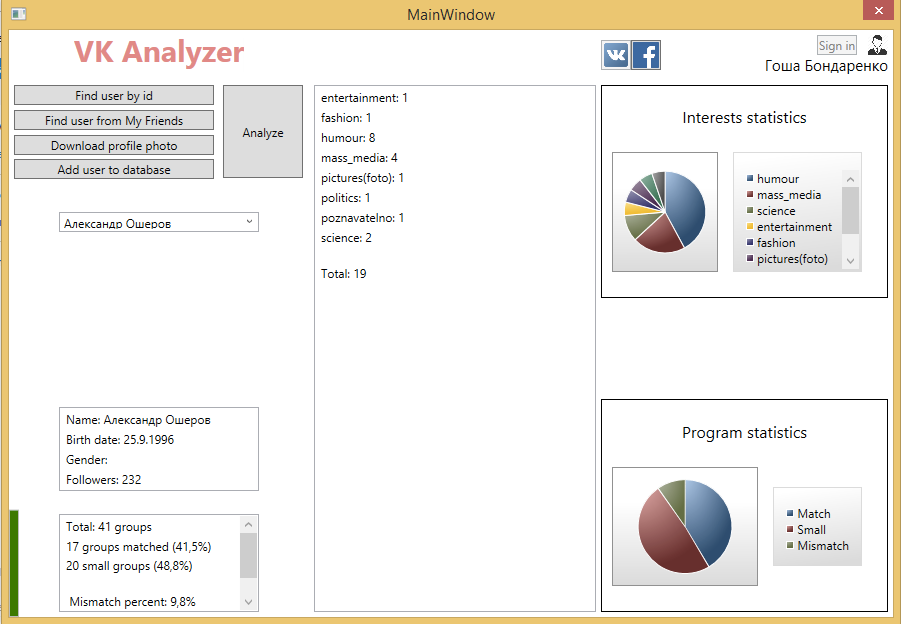
2 - User’s interests (based on data about the pages to which the user is subscribed)

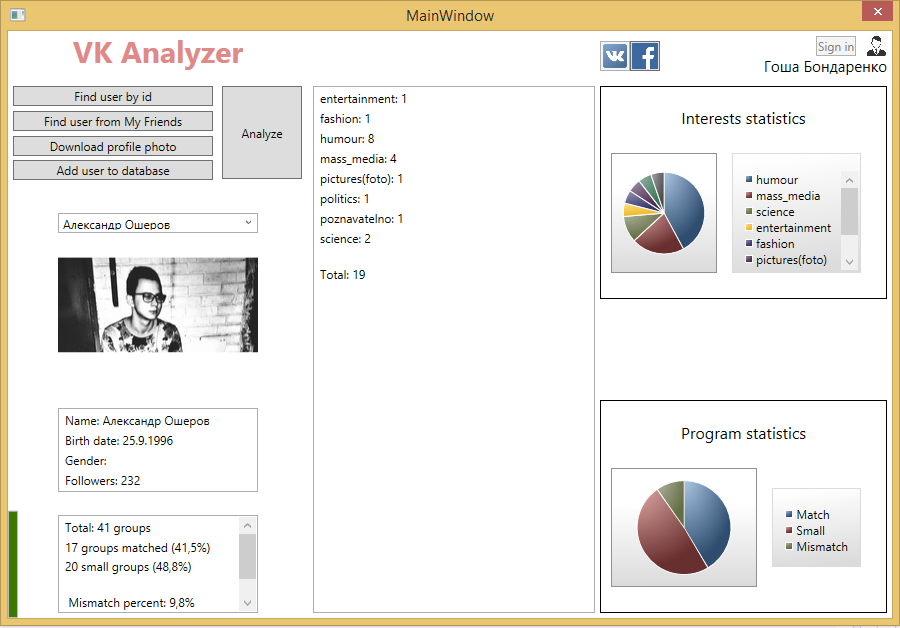
3 - Convenient graphical representation of preferences in the form of a pie chart

4 - Presentation of efficiency of the conclusions\*

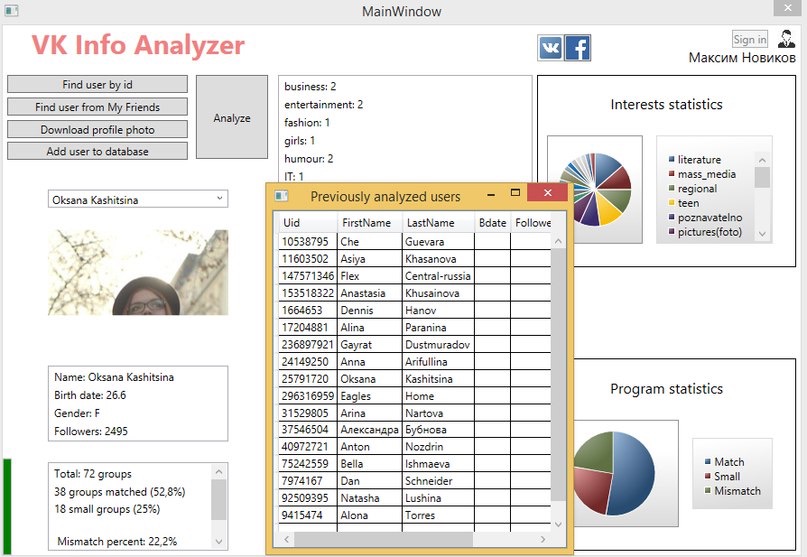
5 - Graphical output of «efficiency» percentage

* Show the total number of groups; sum of groups, that have fit our table with a classified groups (.csv); the number of small groups, we make minor; and «mismatch percent», that represents the «error».

This bar turns green when «mismatch percent» is lower than 30 (promising a kind of accuracy for interests conclusion), and goes red when mismatch percent is too high to be confident in the conclusions.

«Download photo» button:

When you press «Add to DB», the new window (with information about users you added earlier) opens:

Re-open the application. If we try to push one of the buttons on the left side when not signed into your account, we will get a warning window:

