Faculty of Business and Management

Team project

«BookTheBook »

Developed by:

Second grade students of Business Informatics, HSE

Kulish Andrey, Arutyunyan Avetik, Chundyshko Artem

Project supervisor:

Efremov S.G.

**Moscow 2016**

**«BookTheBook» Bookshop**

1. **Annotation**

The main task of our project was to create a desktop application for Windows OS that is capable of implementing tasks like: searching the necessary book in the store and reserve it for pickup. For this aim we have chosen the C# programming language and the Microsoft Visual Studio 2015.

1. **Github**

During the development we have used a web-based [Git](https://en.wikipedia.org/wiki/Git_(software)" \o "Git (software)) repository hosting service, which offers all of the [distributed revision control](https://en.wikipedia.org/wiki/Distributed_revision_control) and [source code management](https://en.wikipedia.org/wiki/Source_code_management) (SCM) functionality «Github»

Our central repository address:

https://github.com/aakulish/BookTheBook

1. **Team members and their tasks**

* Kulish Andrey – main logic, links between forms and code, database queries
* Arutyunyan Avetik – database management, secondary logic, logo design
* Chundyshko Artem – main design, UI, unit tests

1. **Classes**

**BTB\_Data**

Book.cs - a class of the Book database entity, which contains book properties.

Cart.cs – a class with constructor which is aimed to ease transportation information between UI Windows.

Catalog.cs – a class with constructor which is aimed to ease transportation information about books from DB queries to UI Main Window.

Category.cs - a class of the Category database entity, which contains category properties.

Context.cs – class based on dbContext, which instances implement the work with database.

Journal.cs - a class of the Journal database entity, which contains journal properties, such as number of books in stock or expected (after normalization Journal entity has coupling n:1).

OrderListClass.cs – a class which contains fields to create a final sales order and transport it to CartWindow.

Repository.cs – a class where methods of queries to DB are contained (6 queries)

Shop.cs - a class of the Shop database entity, which contains shop properties.

ShopList.cs – a class with constructor which is aimed to ease transportation information about shops from DB queries to UI Main Window.

**BTB\_WF\_UI**

BookInfo.cs - a class which provides user with opportunity to look detailed information about book and add it to Cart after choosing available shop.

CartForm.cs – a class which displayed form of Cart and allows user to confirm its order filling in personal contact information.

DownloadImage.cs – a class which includes method of downloading images from web server and saving it in bmp format.

EmailSender.cs – based on INotify interface, send email confirmation to customer using API.

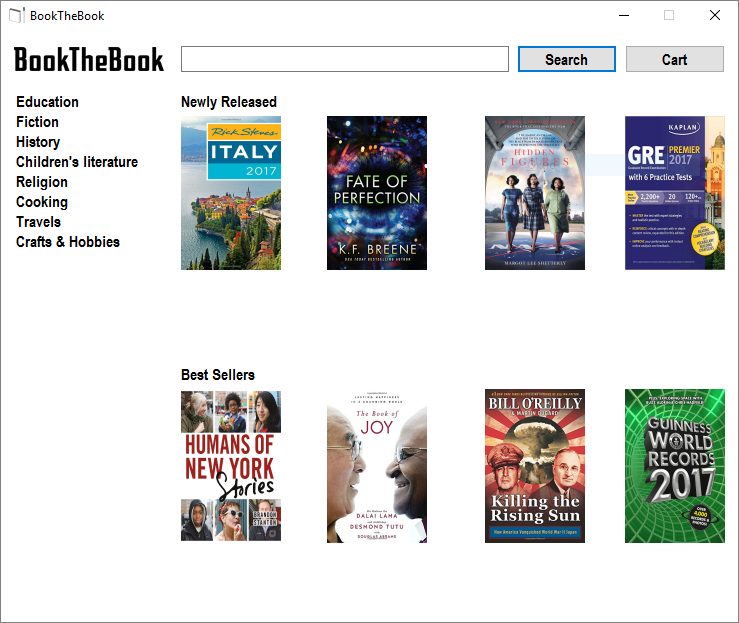
Form1.cs – main form of application where newly released books, best sellers are shown. Also shown categories of books and search string.

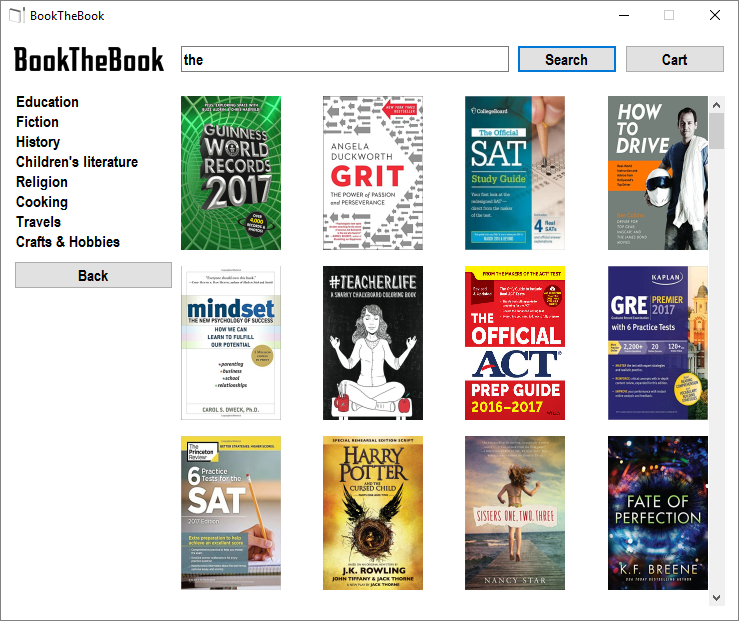
INotify – interface for EmailSender and SMS classes. Contains method which get list of items in order and send confirmations through different channels.

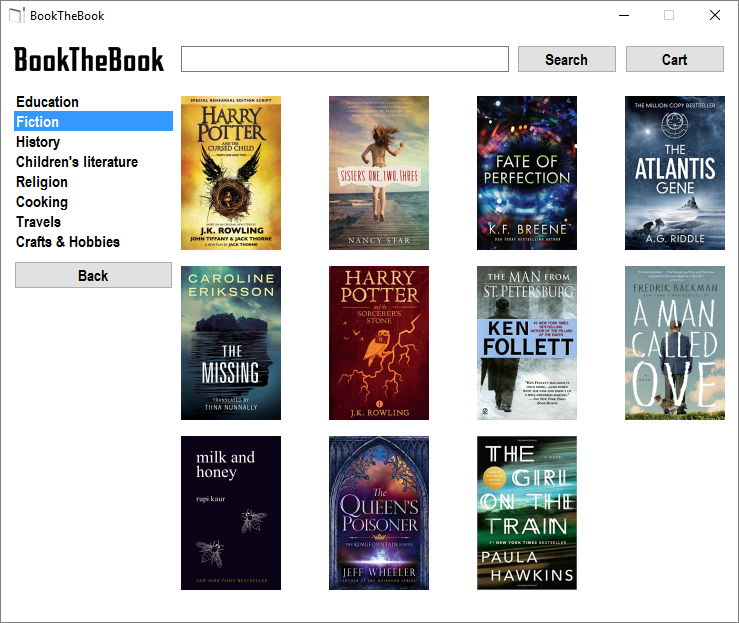
Program.cs – standard class of application. Used for delegate.

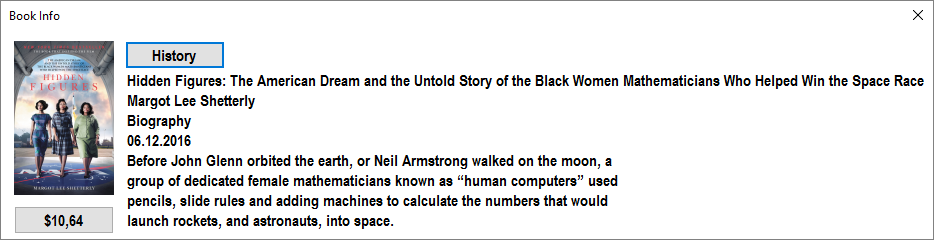
SMS.cs - based on INotify interface, send SMS confirmation to customer using API.

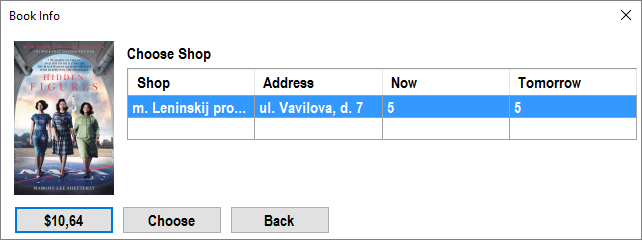
1. **Program interface**

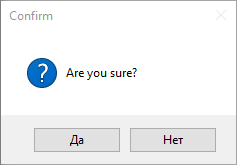
The application, when launched, shows the main window to the user. Here, the user is presented with “Newly Released” and “Best Sellers” suggestions that change from time to time.

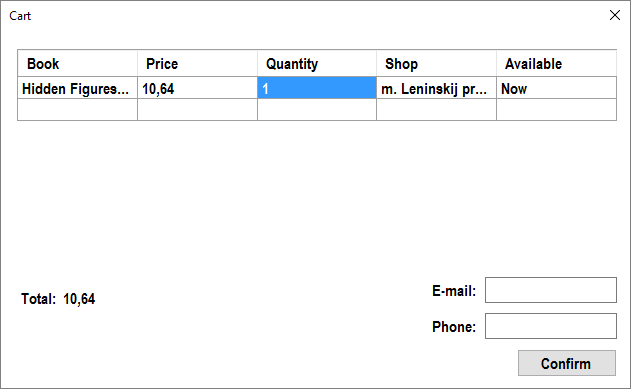
The user is able to enter any text into the TextBox located on top of the window for the program to search for in the available database. Clicking “Search” while having some text in the TextBox performs such behavior. After a search is performed, a “Back” button becomes visible, allowing the user to go back to the suggestions page.

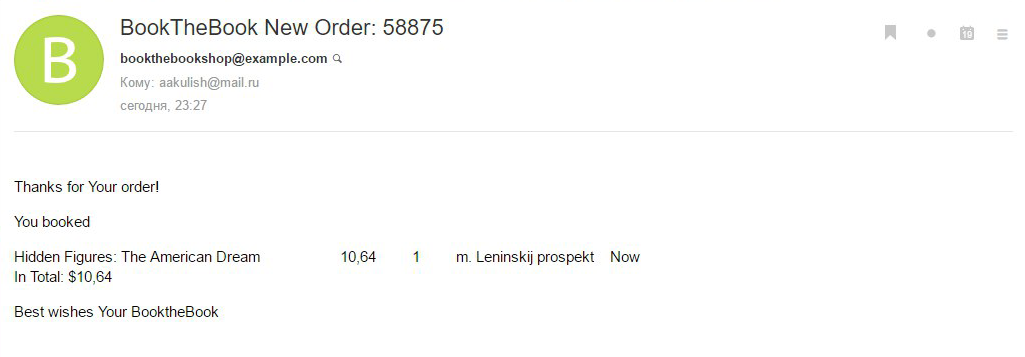
The user is also able to double click on any category on the left side of the main window to view books based on category. As with the search behavior, the “Back” button becomes visible.

Every picture is clickable: a click opens a new window called “Book Info” that contains some information about the book. From here, the user is able to either click the Category button (in this case, it’s “History”, but the text inside the button changes depending on what category the book is in) which closes the info window and switches the main window to the according category page, or click the Price button

Clicking on the price button prompts the user to choose the shop by picking a row in a DataGridView and then pressing “Choose”.

The user is asked to confirm if the order should take place.

After the user’s confirmation, the cart is updated. Here, the user may change the quantity of each object that has been added to the cart.

To place the order, the user enters E-mail and Phone data. Here is a sample of what the user receives

1. **Possible Development**

Hard to find the certain book that you need? It is quite common situation almost for every booklover, especially when the book is a rare one. Moreover, it could be easily sold out, so the trip to the bookshop would be the waste of time. These factors make our project rather topical. That would mean that future development needs to be set. It is possible to expand the application database by adding new online shops (now Amazon is the only one) and create partnership with offline bookshops. Nevertheless, there is an idea, to unite all online shops database and establish distributing service. What is more, the feature of different formats e-books purchasing via BookTheBook could be added.

As the application only implements a model of placing orders and reservations a logical step will be to implement orders database. Also, for the working with databases many-to-many relationships should be executed to allow book entity to have many shops. Furthermore, the UI can be improved by adding a constant user profile with login and password.