

University of Engineering and Technology

Hanoi, 1 October, 2015

Service-oriented Architecture

Library Management Project Report

**Members (Group 4):**

* Lê Đình Minh (12020609)
* Vương Tùng Long (12020620)
* Lý Phương Anh (12020453)

Table of Contents

[I. Problem 2](#_Toc431651413)

[II. Problem solving 2](#_Toc431651414)

[1. The related theories, algorithms, technologies 2](#_Toc431651415)

[a. Service-oriented architecture (SOA) 2](#_Toc431651416)

[b. Service 2](#_Toc431651417)

[c. Application Program Interfaces (APIs) 2](#_Toc431651418)

[2. Our solution 3](#_Toc431651419)

[III. APIs Specification 4](#_Toc431651420)

[IV. Application 9](#_Toc431651421)

[1. Web browser (http://uetlib.herokuapp.com/books ) 9](#_Toc431651422)

[2. Desktop application 10](#_Toc431651423)

[a. Technology 10](#_Toc431651424)

[b. Design Pattern 10](#_Toc431651425)

[c. User Interface Design 11](#_Toc431651426)

[d. Call API from server and get response 14](#_Toc431651427)

[3. Mobile application (Android application) 15](#_Toc431651428)

[a. Technology 15](#_Toc431651429)

[b. User Interface Design 15](#_Toc431651430)

[c. Call API from server and get response 16](#_Toc431651431)

[V. References 17](#_Toc431651432)

# Problem

Build a simple library management application.

Application have two kind of user: student and librarian.

* Librarian use desktop application to show book list, edit book properties, add and delete book.
* Student can see the book list and details via website

# Problem solving

## The related theories, algorithms, technologies

### Service-oriented architecture (SOA)

A **service-oriented architecture** (**SOA**) is an architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network. The principles of service-orientation are independent of any vendor, product or technology.

### Service

A service is a key factor in SOA, services can be understood as a type module perform certain functionalities. One of the purpose of SOA is to help applications can communicate with each other without knowing the techniques inside. To accomplish that SOA defines a standard interface clearly and independently of system platforms, and can reuse. Thus, SOA is a higher level of application development, focus on functionalities and standard interface to help cover the technical complexity underneath. The abstract is the core of the service sheet concept, it enables enterprises to integrate existing components into new applications and components can be shared or reused in various fields several of the company without having to modify the source code or restructure the system.

### Application Program Interfaces (APIs)

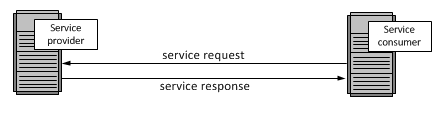
APIs/information hiding allow for the creation of a minimal interface that is relatively stable that can be used by other software systems to access or manipulate the underlying systems or data. This allows for enhancements to the underlying systems or data without disturbing the software systems that use the API.

## Our solution

We using “**web services**”. A web service is a collection of open protocols and standards used for exchanging data between applications or systems. Software applications written in various programming languages and running on various platforms can use web services to exchange data over computer networks like the Internet in a manner similar to inter-process communication on a single computer.

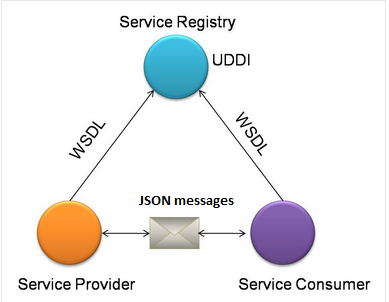
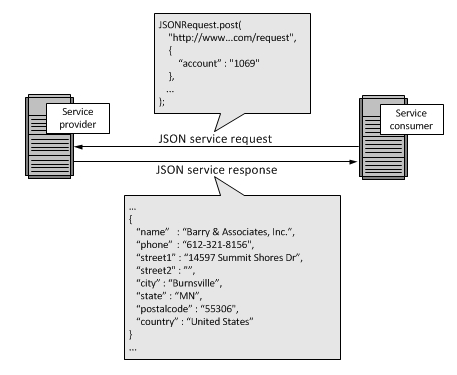
The technology of Web Services is the most likely connection technology of service-oriented architectures. The following figure illustrates a basic service-oriented architecture. It shows a service consumer at the right sending a service request message to a service provider at the left. The service provider returns a response message to the service consumer. The request and subsequent response connections are defined in some way that is understandable to both the service consumer and service provider.

We create API to make application communicate easily and implemented it using Web Services JSON.



When applying to our web application, we follow this flow:

* Generate an URL for each request.
* Use HTTP Request to query data. The main HTTP methods we use are: GET, POST, PUT, DELETE
* JavaScript Object Notation (JSON)

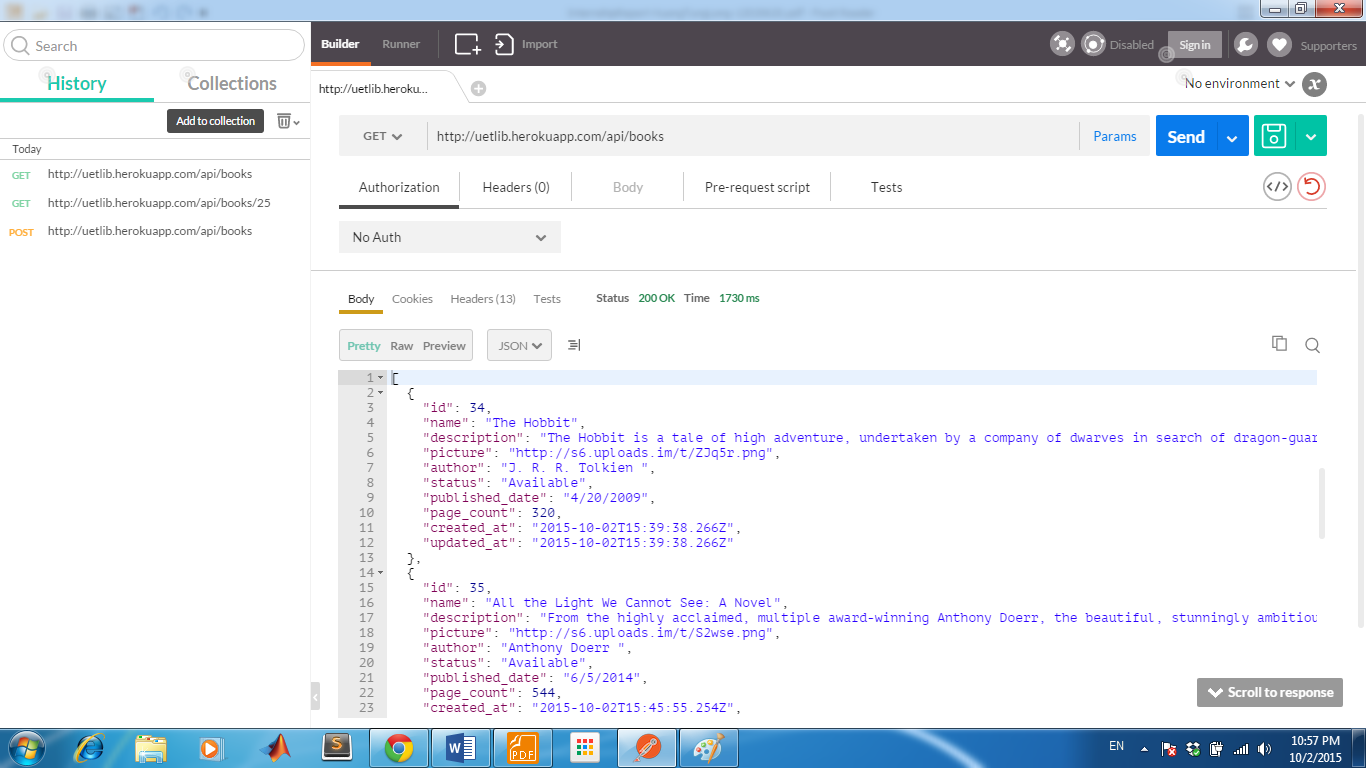


# APIs Specification

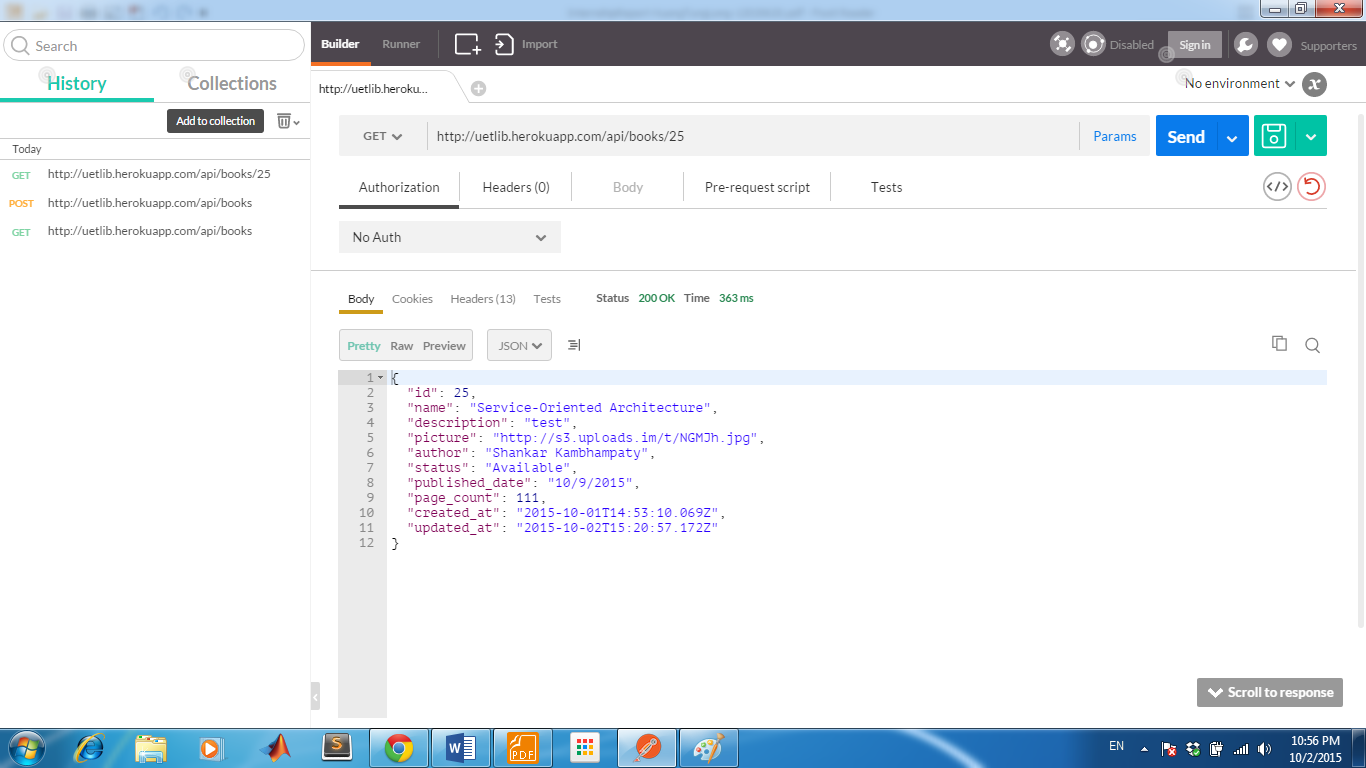
|  |  |  |  |
| --- | --- | --- | --- |
| **Function** | **Link** | **Params in JSON** | **Method** |
| Get information of all book store on our system. | <http://uetlib.herokuapp.com/api/books> | None | GET |
| Get information of a book store on our system by id | <http://uetlib.herokuapp.com/api/books/42>  (EX. Id = 42) | None | GET |
| Create a new book | <http://uetlib.herokuapp.com/api/books> | {  "name": "Service-Oriented Architecture",  "description": "test",  "picture": "http://s3.uploads.im/t/NGMJh.jpg",  "author": "Shankar Kambhampaty",  "status": "Available",  "published\_date": "10/9/2015",  "page\_count": 111  } | POST |
| Update a book by id | <http://uetlib.herokuapp.com/api/books/42>  (EX. Id = 42) | {  "name": "Service-Oriented Architecture",  "description": "test",  "picture": "http://s3.uploads.im/t/NGMJh.jpg",  "author": "Shankar Kambhampaty",  "status": "Available",  "published\_date": "10/9/2015",  "page\_count": 111  }  **Just put attributes you want to update** | PUT |
| Delete a book by id | <http://uetlib.herokuapp.com/api/books/42>  (EX. Id = 42) | none | DELETE |

**Example test using API with postman (an extension of chrome)**

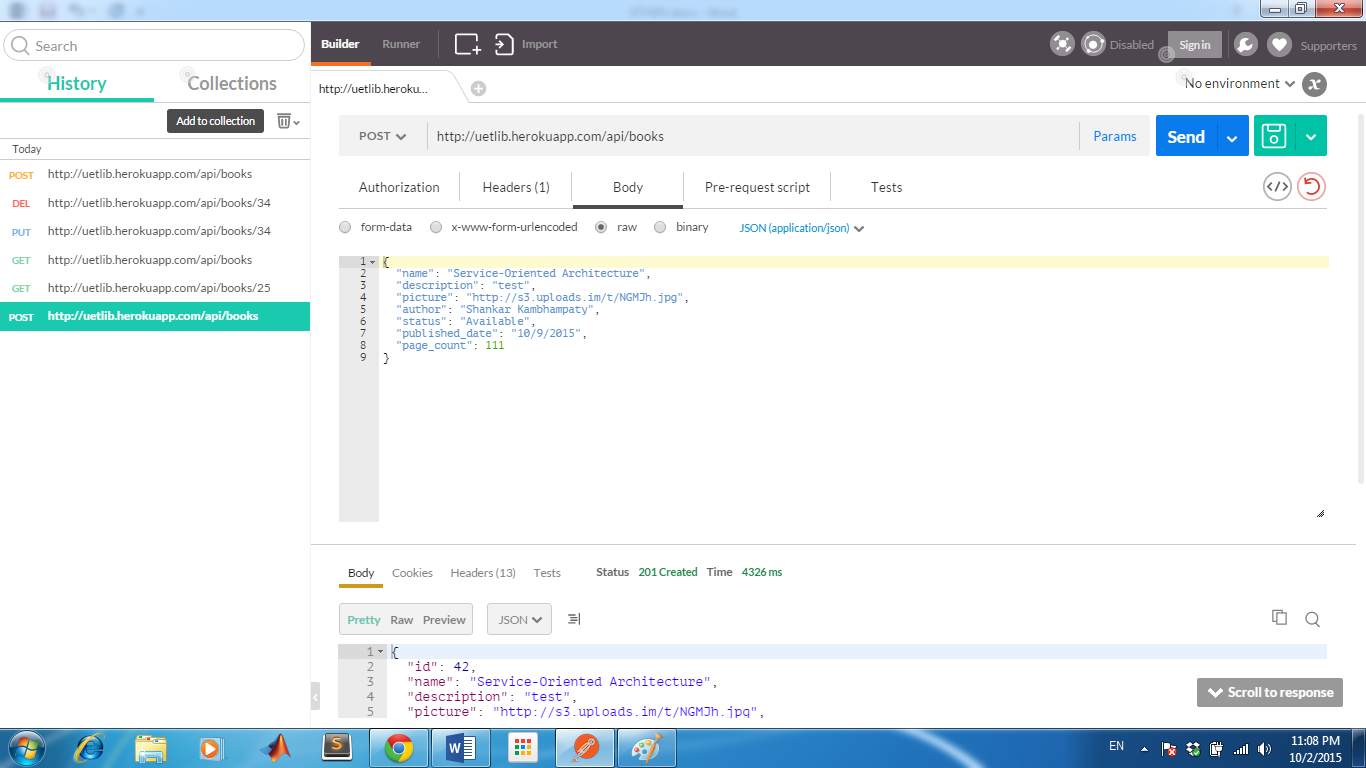
* Get information of all books store on our system with method **GET**



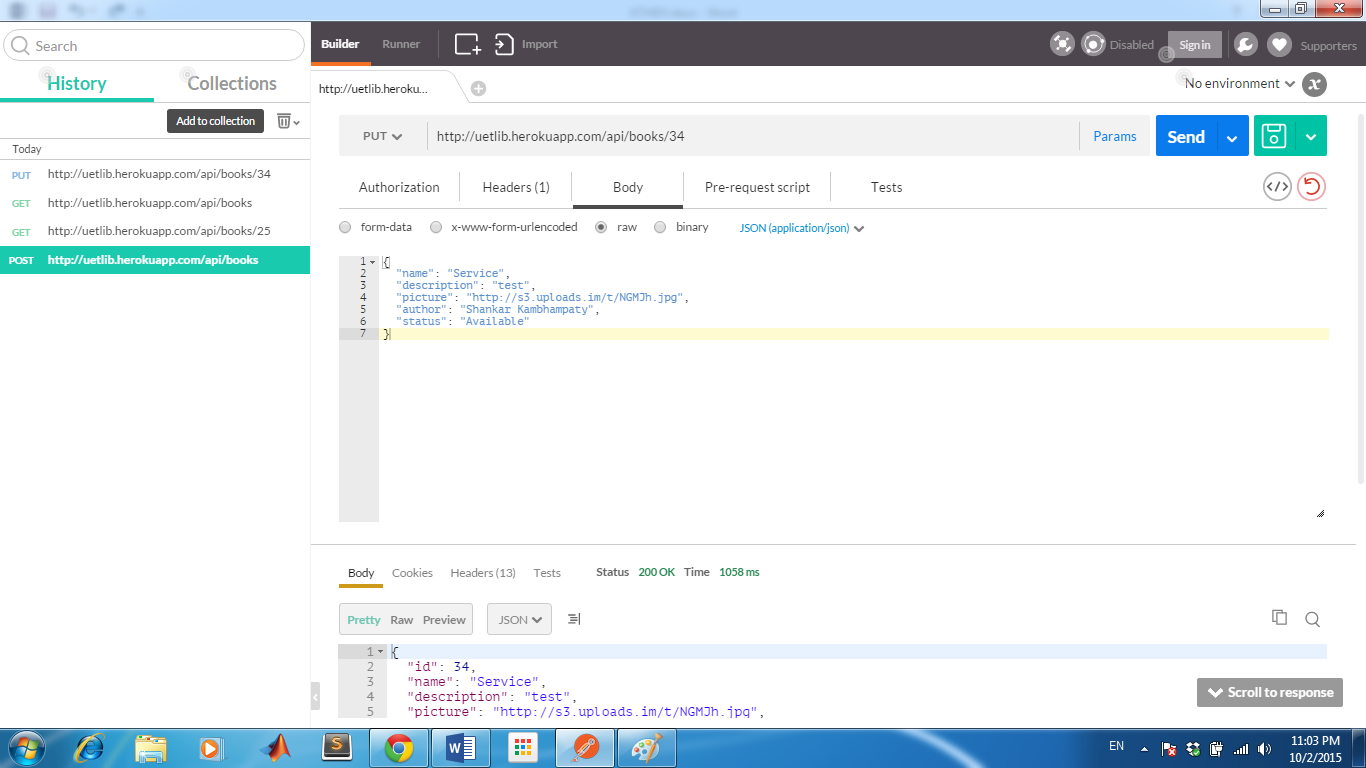
* Get information of a book store on our system by id with method **GET**



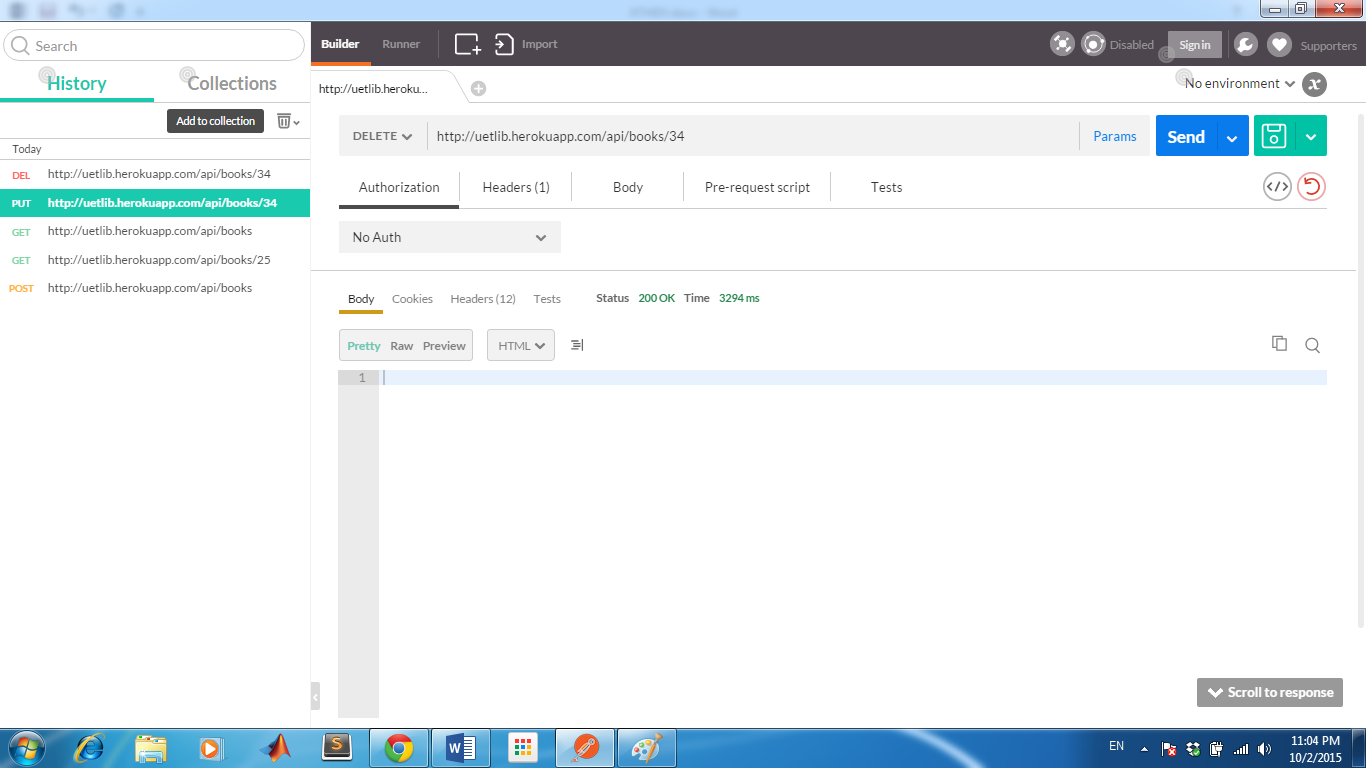
* Create a new book with method **POST**



* Update a book by id with method **PUT**



* Delete a book by Id with method **DELETE**



# Application

## Web browser (<http://uetlib.herokuapp.com/books> )

1. In web browser app, we provide function see all books for user and search function



When using search function, user can type name of books or apart name of book they want to find.



## Desktop application

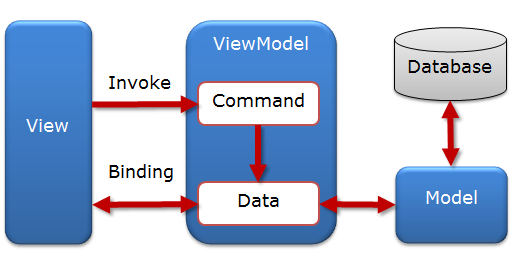
### Technology

In this project I choose Windows Presentation Foundation (WPF) to develop desktop application. Windows Presentation Foundation (or WPF) is a graphical subsystem for rendering user interfaces in Windows-based applications by Microsoft. WPF is one of the four main products in Microsoft's .NET Framework 3.0.

WPF employs XAML, an XML-based language, to define and link various interface elements. WPF applications can be deployed as standalone desktop programs or hosted as an embedded object in a website. WPF aims to unify a number of common user interface elements, such as 2D/3D rendering, fixed and adaptive documents, typography, vector graphics, runtime animation, and pre-rendered media. These elements can then be linked and manipulated based on various events, user interactions, and data bindings.

### Design Pattern

My desktop application use Model-View-ViewModel Design Pattern:



WPF was designed to make it easy to build applications using the MVVM pattern because of some feature:

* Data binding
* Data template
* Resource system

### User Interface Design

I use material design to design UI for desktop application.

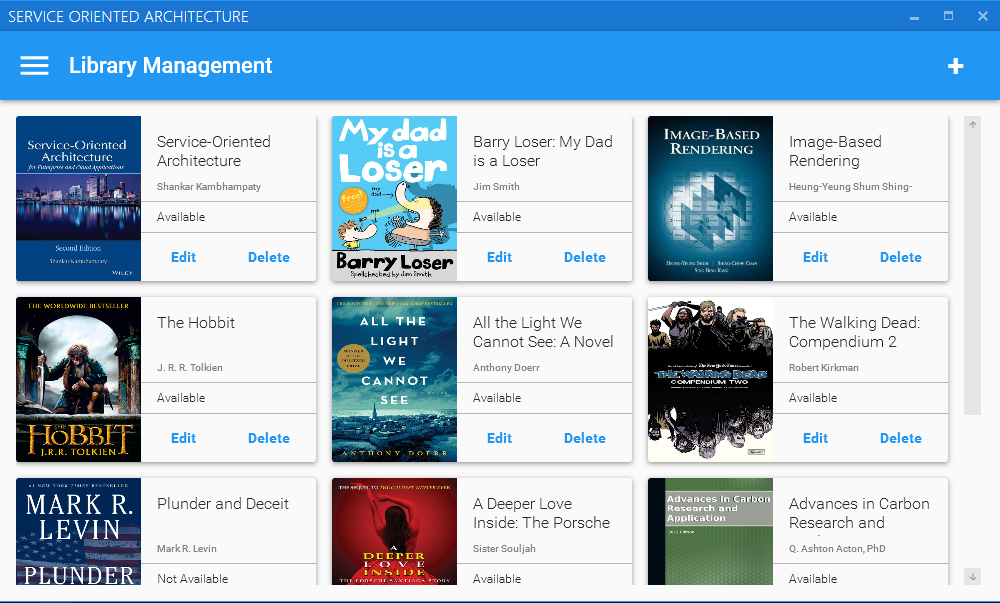
To let everything be quick and easy, I use Material Design In XAML Toolkit - An open source library to providing the Material Design palette, plenty of control themes, and new custom controls to bring this popular design language to your desktop applications.

The library’s repository on Github:

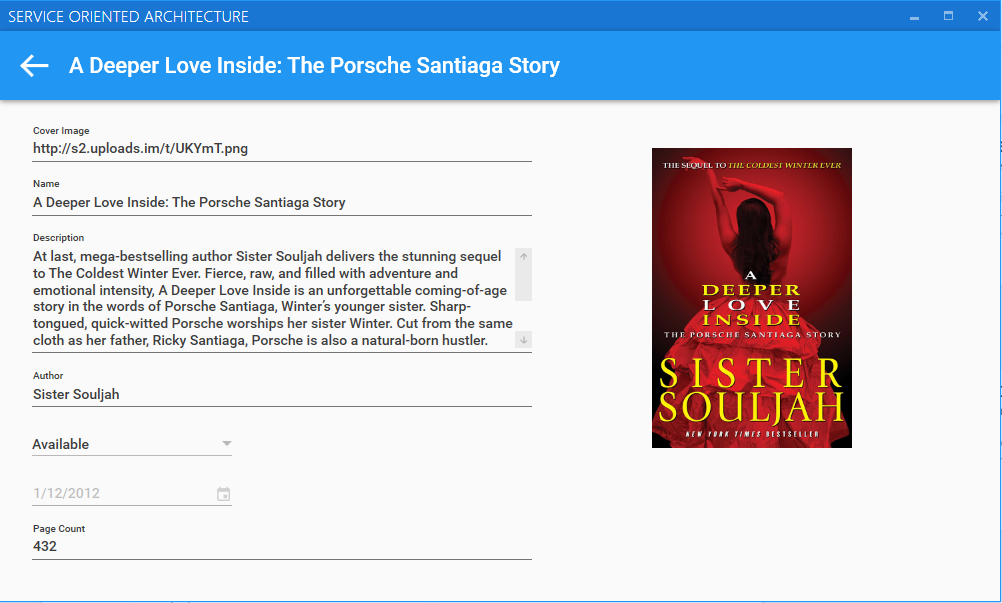
<https://github.com/ButchersBoy/MaterialDesignInXamlToolkit>

Several screenshot of the desktop application:

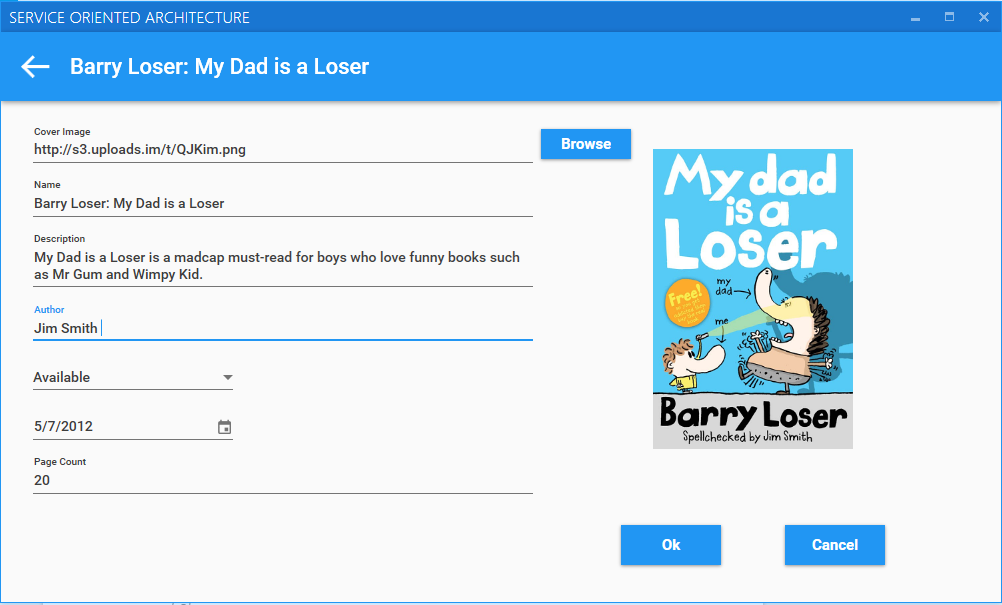
* Main screen of application will show all of book in library:
* User click to a book to view book details, click to “*plus*” button in top right corner to add a new book, click to “*delete*”, “*edit*” button on each book to delete, edit book.



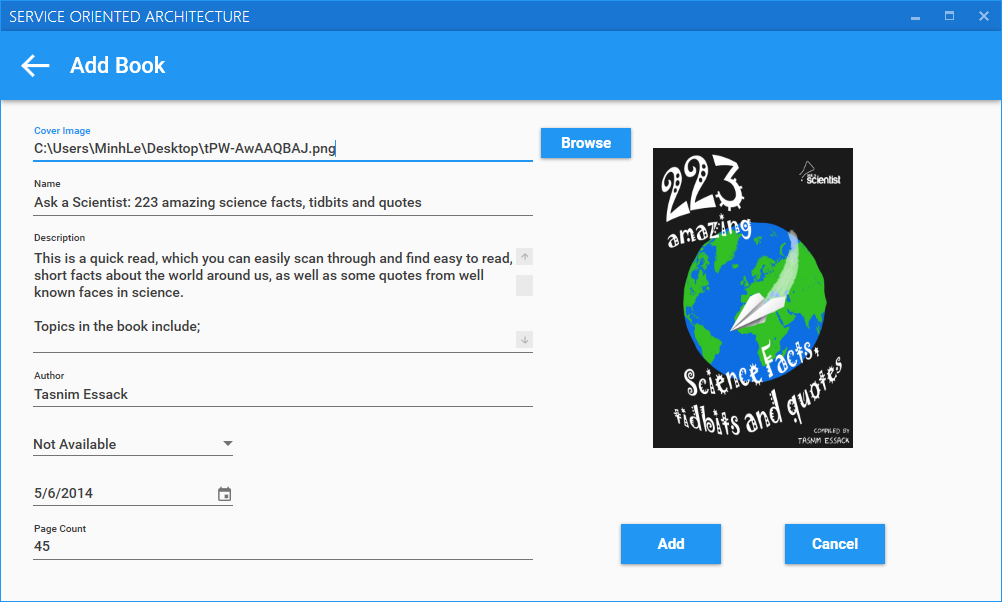
* View details of the book (view only):



* Edit book details



* Add a new book



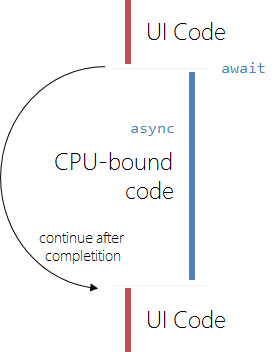
### Call API from server and get response

In .NET, the System.Net namespaces provide the **WebRequest** class to encapsulate a request for an Internet resource, and the **WebResponse** class to represent the data that is returned.

By using these objects, you can obtain a stream that represents the response for a particular request. When you have a stream, you can read the response just as you read from a local text file or from any other source.

When process request, I use asynchronous programming model for asynchronous access to Internet resources. It help main thread (UI thread) is always responsive.

Asynchronous model:



After receive response from server as JSON type, my desktop app will parse it and update the data model.

## Mobile application (Android application)

I also develop an Android application for student to view book in library.

### Technology

Android programming is based on Java programming language.   
In this project, I choose Android Studio to develop this app because  
Android Studio is officially supported by Google team.

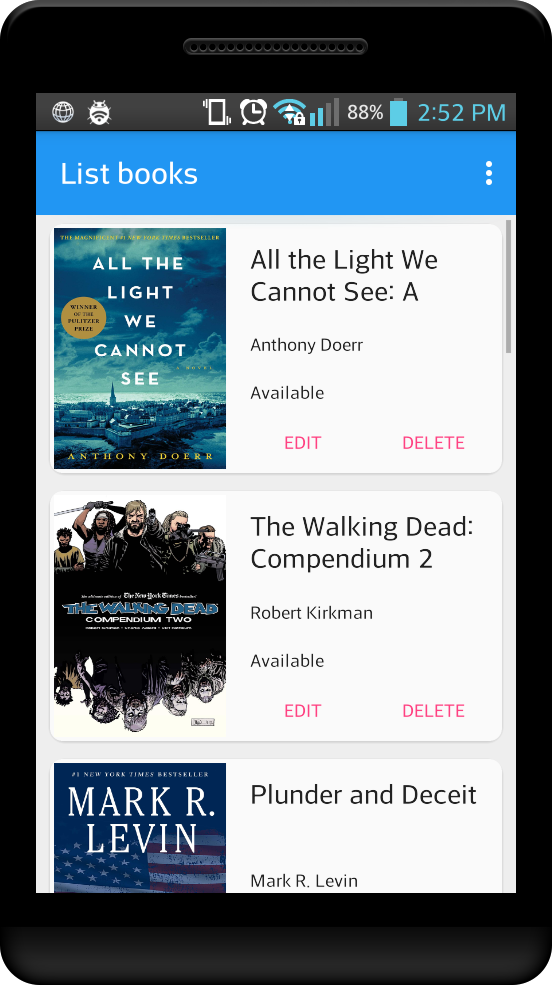
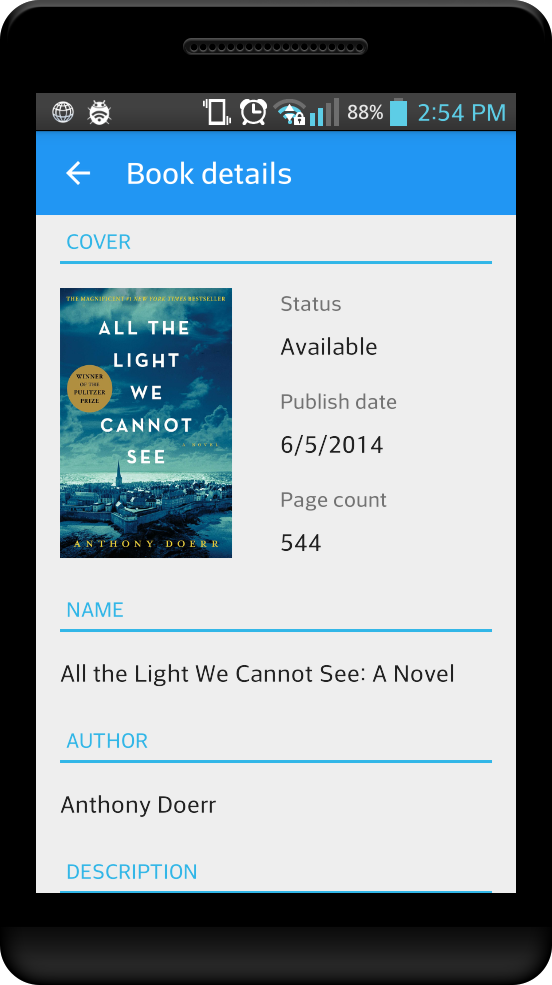
### User Interface Design

I also use Material Design for this app.

Fortunately, material design is invented by Google so it is easy to create an Android app with material design.

With latest Android Design Support Library almost important material  
design components will be brought to all Android 2.1 or higher devices.

Several screenshot of my desktop application:

### Call API from server and get response

Like in desktop app, request and response will be process asynchronously to make UI thread always responsive.

In android app I use Android Asynchronous Http Client library - an open source library to make asynchronous callback based Http client for Android built on top of Apache’s HttpClient libraries to send request and get callback from server.

The library’s repository on Github:

<https://github.com/loopj/android-async-http>

# References

[https://msdn.microsoft.com/en-us/magazine/dd419663.aspx#id0090016](https://msdn.microsoft.com/en-us/magazine/dd419663.aspx%23id0090016)

<https://en.wikipedia.org/wiki/Windows_Presentation_Foundation>

<http://materialdesigninxaml.net/>

<http://loopj.com/android-async-http/>